Neural Processing Letters (2025) 57:55
https://doi.org/10.1007/s11063-025-11775-5

PATCHOUT: Adversarial Patch Detection and Localization using Semantic Consistency

Dominic Simon[1] · **Sumit Jha**[2] · **Rickard Ewetz**[1]

Accepted: 20 May 2025 / Published online: 1 June 2025
© The Author(s) 2025

Abstract
Computer vision systems are actively deployed in safety-critical applications such as autonomous vehicles. Real-world adversarial patches are capable of compromising the artificial intelligence (AI) systems with catastrophic outcomes. Existing defenses against patch attacks are based on identifying neurons, features, or gradients of high intensity. However, these defenses are vulnerable to weaker attacks that have less obvious attack signatures. In this paper, we propose the PATCHOUT framework that detects and locates adversarial patches using semantic consistency. Within patch detection, the key insight is that the top class predictions for an entity are semantically consistent for benign images, whereas they are inconsistent for attacked images. Within patch localization, it is observed that patches are semantically consistent with a coarse grained segmentation of the image. This allows the PATCHOUT framework to detect and remove adversarial patches using a class consistency checker as well as image segmentation, attribution analysis, and image restoration techniques. The experimental evaluation demonstrates that PATCHOUT can detect a broad range of adversarial patches with over 90% accuracy. The framework achieves 20% higher accuracy than other defenses. The framework is also evaluated against unseen attacks and adaptive attacks, reducing the success rate of adaptive attacks from 56% to 24%.

Keywords Adversarial Machine Learning · Adversarial Patch · Computer Vision · Artificial Intelligence

✉ Dominic Simon
dominic.simon@ufl.edu

Sumit Jha
sjha@fiu.edu

Rickard Ewetz
rewetz@ufl.edu

[1] Electrical and Computer Engineering, University of Florida, 1889 Museum Rd, Gainesville 32611, Florida, USA

[2] Knight Foundation School of Computing and Information Sciences, Florida International University, 11200 SW 8th St, Miami 33199, Florida, USA

1 Introduction

Computer vision systems are broadly being deployed within cyber-physical systems for applications in manufacturing, surveillance, and autonomous driving. An emerging security risk for computer vision systems is the susceptibility to adversarial attacks, which can cause detrimental misclassifications [1]. While norm constrained adversarial attacks pose a serious threat to copyright infringement within the digital domain, the attacks are difficult to deploy in the real-word [2]. Adversarial patches restrict these perturbations to specific regions, but have no bounds on the perturbation values. This allows the attacks to be easily realized in the physical world using stickers, paper cutouts, and t-shirt prints [3, 4]. Several different methods of generating adversarial patches have been proposed [5–7], with real-world demonstrations.

The threat of adversarial patches has spurred research efforts into safeguarding computer vision systems [2, 8–16]. Adversarial patch attacks have been detected by identifying pixels with high intensity, odd colors, or high entropy. Other approaches include detecting high intensity neurons or irregular attributions [2, 9, 12]. A limitation of these defenses is that attacks of reduced strength have less obvious attack signatures [9], which provides a straightforward avenue for circumventing detection. Another challenge is that after an attack has been detected, a pixel map or heat-map of abnormalities are required to be distilled into a patch location. The distillation process has been performed using smoothing [8, 10], patch size and shape priors [14, 15], and autoencoders [17]. Unfortunately, smoothing is not very accurate and patch priors and autoencoders requires knowledge of the attack or training data [9, 15, 17].

In this paper, we propose the PATCHOUT framework that detects and localizes adversarial patches based on semantic consistency. The patch detection approach is based on the observation that the top output predictions from an image classifier are typically of related, or semantically consistent, for benign input images. In contrast, image classifiers are often conflicted between widely disparate classes for attacked images. In the PATCHOUT framework, patch attacks are detected by analyzing the output class consistency using a lightweight neural network. The patch localization approach is based on the observation that patches are semantically consistent with a coarse grained segmentation of the input image. This transforms the patch localization problem into a segment selection problem, which is solved using attribution analysis in the proposed framework.

The main innovations of this paper are as follows:

(a) (b)

Fig. 1 (**a**) An image attacked by the Generative Dynamic Patch. (**b**) The attacked image defended by PATCHOUT

- **Patch Detection using Semantic Consistency:** We propose a novel concept of detecting patch attacks based on the class consistency of the model output predictions. The proposed approach is orthogonal to techniques of detecting high intensity pixels, gradients or features as in previous studies. The proposed patch detector is lightweight and is able to separate benign and attacked images with 90% accuracy.
- **Patch Localization using Semantic Consistency:** The recent advancement of pre-trained zero-shot image segmentation models provides new opportunities for patch localization. The proposed approach utilizes coarse image segments as patch priors, which circumvents the need for knowledge of the patch size or shape [15], or training data [17].
- **Experimental Evaluation using Unseen and Adaptive Attacks:** On the ImageNet [18] dataset, the PATCHOUT framework demonstrates as much as a 20% higher accuracy compared with state-of-the-art defenses against adversarial patches. We also evaluate the framework on adaptive attacks and show that while the framework can be bypassed, it is at a great detriment to the overall attack success rate to do so, decreasing the success rate to as low as 28%.

The remainder of the paper is organized, as follows: background is provided in Section 2. The motivation for the use of semantic consistency is outlined in Section 3. The methodology of the PATCHOUT framework is given in Section 4. The experimental evaluation with adaptive and unseen attacks are described in Section 5. The limitations of PATCHOUT are discussed in Section 6. The paper is concluded in Section 7.

2 Background

2.1 Adversarial Patches

Adversarial patches are attacks on computer vision models that aim to cause a model to purposefully output incorrect predictions. Patches take the form of sticker-like objects that are placed in images, videos, and physical settings. The patches we focus on are generated under a white-box scenario, where patches have full access to the model they are trying to attack. There are four main descriptors used for patches: per-image, universal, targeted, and untargeted. Per-image patches are generated for a specific location on a single image while universal patches are able to be applied anywhere on any image from the target dataset. Per-image patches are considered stronger attackers than universal patches since the generality of universal patches makes them easier to detect [15]. Targeted patches attempt to maximize the likelihood of a specific non-ground truth prediction while untargeted attacks push model predictions in any direction other than the ground truth.

Generating patches can be formulated as an optimization problem. For example, a Local and Visible Adversarial Noise (LaVAN) [5] patch p is generated by repeatedly adding noise n to a target image x at a restricted location l such that the probability $\mathbb{P}$ of predicting the ground truth classification y_{gt} is as unlikely as possible for the model F on noised image $\hat{x}$. Optimization of the patch occurs through existing processes such as Projected Gradient Descent (PGD) [19]. The attack designs the patch by solving the optimization problem below:

$$p = \underset{y_{gt}}{\operatorname{argmin}}[\mathbb{P}(F(\hat{x}) = y_{gt}|n, x, l)]. \tag{1}$$

The starting points and optimization processes of patches vary widely. The original adversarial patch [3] treated the patch pixels as weights to be optimized, performing gradient

descent to update the patch. The Texture-based Patch Attack [6] generates a texture that represents the target class and updates the texture to be adversarial through a PGD-like process based on the texture's features. The Generative Dynamic Patch Attack [7] uses an encoder-decoder pair to generate a patch.

This work is mainly concerned with patches for classification networks, but patches have been created to attack object detectors [20–23] and vision transformers as well. The Naturalistic Patch [4] is an object detection patch that is generated by continual updates from a generative adversarial network. The patch proposed in [24] generates a patch for vision transformers by applying the adversarial patch to the most influential image patch and updating the patch using an attention-based loss with PGD.

2.2 Defenses Against Patches

A common defense approach has been to search the model in use to find abnormal gradients or activation values. These defenses rely on the observation that patches cause large activation values to occur in order for their effects to propagate to the output layer. Scalecert [2] detects patches by tracing the top neurons from the initial layers of a network back to the input. Patchguard [12] uses custom networks with small receptive fields that can bypass the effects of patches as well as employing a neuron masking algorithm that masks neurons potentially affected by patches. Local Gradients Smoothing [8] adds the normalized image gradients back into the image in order to cover the patch pixels. These defenses can be bypassed by weaker patches that sacrifice attack success in order to cause smaller activation values.

Another approach has been to mask parts of the input image. Typically these defenses use multiple masks and get model predictions for each of the mask locations in order to generate a consensus on which predictions are correct. PatchCleanser [14] splits the input image into a grid and masks the image at each grid position. Each masked image is passed to the model in order generate a set of predictions from which the majority prediction is likely the true image class. Object Seeker [25] takes a similar approach for object detection patches, but it masks out rows and columns rather than image patches. PatchCleanser and ObjectSeeker are both examples of certified defenses. Certified defenses attempt to set a lower bound on the defense accuracy against attacked images. Another example of a certified defense is Derandomized Smoothing [10]. While certified defenses are useful in guaranteeing a minimum accuracy across many patches, this accuracy tends to be lower than desired. Additionally, the masking approach many certified defenses use incurs a high run time overhead due to many model calls.

Recently, defenses have begun to more heavily rely on the help of outside networks. Segment and Complete [15] uses a specially trained image segmenter to determine an approximate patch location before using a masking procedure to determine the exact patch location. Jedi [17] builds a heatmap of the patch location based on image entropy. The patch location is determined by thresholding the heatmap and is finalized using an autoencoder. Some defenses against adversarial examples [26–28] have begun employing diffusion models, with patch defenses following suit. DIFFender [29] observes that diffusion models are unable to significantly alter the patch in images that have undergone the forward process [30], so they create a patch mask by taking the difference of the original image and the image after going through both the forward and reverse diffusion process. Networks trained with patch data can be limiting as the defender is not always guaranteed access to a patch before defending against it. Diffusion-based defenses have proven effective against prior patches, but are now being targeted and broken with diffusion-based attacks [31].

Fig. 2 (**a**) A benign image and the corresponding top-5 output classes. The class predictions are consistent with each other. (**b**) An attacked image and the corresponding top-5 output predictions. The class predictions are semantically inconsistent with each other. (**c**) A segmentation of the image in (b). The patch is semantically consistent with an image segment. The PATCHOUT framework leverages the semantic consistency of classes and segments to detect and locate adversarial patches

3 Motivation of Semantic Consistency

In this section, we describe how adversarial patches can be detected and localized using semantic consistency.

Detection of Adversarial Patches using Semantic Consistency: Detecting the occurrence of a patch attack is perhaps the most challenging problem within defenses against patch attacks. We propose to detect patches by examining the semantic consistency of the model output predictions, which is illustrated in Figure 2. The figure shows both a benign image of a paddlewheel boat and a paddlewheel boat that has been attacked with a patch attack of class tusker. When examining the top-5 model predictions of the benign image in Figure 2(a), we observe that the class predictions paddlewheel, fireboat, drilling platform, lifeboat, and dock are somewhat consistent. In contrast, the classes are widely disparate for the attacked image in Figure 2(b). The first three predictions are of elephants and the last two are boat related. Hence, there exists an opportunity to design a patch detector based on the class consistency of the image classifier. *A human can by inspection understand that it is not reasonable that a classifier is confused between if the object in an image is a boat or elephant!*

Semantic consistency: The concept of semantic consistency can be quantified using Jensen-Shannon distance, a metric commonly used to determine the similarity between two probability distributions. Jensen-Shannon distance d, a symmetrized version of Kullback-Leibler divergence that has an output bounded between [0, 1], is described in the following equation:

$$d = \sqrt{\frac{D_{KL}(p||m)}{2} + \frac{D_{KL}(q||m)}{2}} \tag{2}$$

where $D_{KL}()$ is the Kullback-Leibler divergence between two distributions, p or q and their element-wise average $m = \frac{p+q}{2}$. An experiment analyzing the Jensen-Shannon distance is provided in Table 1. The table shows the Jensen-Shannon distance between benign output logits (Benign) from images of the same class, LaVAN-attacked image logits and benign image logits from an image sharing the same class as the patch, and logits from different classes (Different). Each row is the average distance between five comparison images and 1300 images with the same classes from Imagenet's [18] training set. There is a clear split between the divergence of benign outputs from images of the same class, outputs with patch classes that share the same class as a benign output, and completely unrelated classes. However, the values in this table come from images that were handpicked to show a clear difference across output types. Other images and classes tested showed the same split between output types,

Table 1 Jensen-Shannon Divergence between benign and attacked logits

Output Type	Distance
Benign	0.35
Adv	0.50
Different	0.82

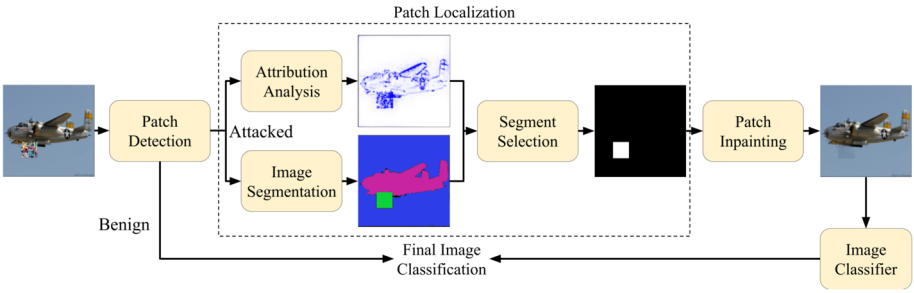

Fig. 3 The PATCHOUT framework. A lightweight detector determines whether an image is benign or attacked. If attacked, patch localization is performed using attribution analysis and image segmentation. Finally, the patch location is inpainted before being passed to the image classifier

but across varying values. This approach would require manual intervention to be applied on a larger scale. Instead of manually encoding what classes should be considered consistent, we propose to train to learn the concept of class consistency using a lightweight neural classifier.

Patch Localization using Semantic Consistency: Once a patch attack has been detected, the secondary problem relates to localizing the the patch for removal. We propose to localize adversarial patches by utilizing the semantic consistency with respect to a coarse image segmentation, which is shown in Figure 2. A coarse grained segmentation of Figure 2(b) is shown in Figure 2(c). The segmentation is obtained using the pre-trained zero-shot segment anything model (SAM) [32]. *It can be observed that the patch is clearly labeled with a unique, individual segment, meaning it is semantically consistent with a segment.* This observation transforms the problem of patch localization into a segment selection problem. We propose to utilize advancements [33, 34] within attribution analysis to identify the segment corresponding to the patch. This semantic based approach eliminates the need for any prior knowledge of the patch type, size, or shape (within the localization step).

4 Methodology of PATCHOUT

In this section, we present the methodology of the PATCHOUT framework. The flow of the framework is illustrated in Figure 3. The defense consists of a patch detection step, a patch localization step, and an inpainting step. The first objective is to detect if the input image contains a patch using a lightweight neural network. The details are provided in Section 4.1. If no patch is detected, the output class of the image classifier is directly returned. Otherwise, the second step is to locate the patch using attribution analysis and image segmentation, which is outlined in Section 4.2. The last step is to remove the patch and restore the image using inpainting. The inpainting is described in Section 4.3. The restored image is fed to the image classifier to determine the output class.

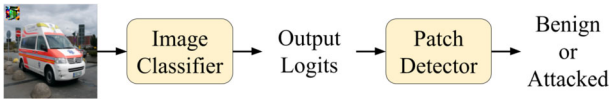

Fig. 4 The input to the patch detector is the output logits of the image classifier. The patch detector determines if the image is benign or attacked based on the learned semantic consistency of the classes

4.1 Patch Detection

In this section, we outline the proposed patch detection approach. The flow of the patch detection is shown in Figure 4. The patch detector is a lightweight neural network. The input to the detector is the logit for each of the output classes of the image classifier the defended image will be passed to. The output is a binary result that states if the image does or does not contain a patch.

The patch detector is a feed-forward neural network with five layers for the patch detector. The five-layer network consists of two fully connected layers, followed by a max pooling layer, followed by two fully connected layers. Each fully connected layer is followed by ReLU activation functions. A separate detector network is trained for each combination of patch and classification network. This allows the detectors to more accurately learn the semantic intricacies of each combination, providing a better patch detection accuracy. An experiment on the transferability of these networks to unseen patches is provided in Section 5.4.1. Training data for each network consists of 10,000 images that were successfully misclassified using an even split of the expected sizes of the patch to be detected. In practice, these are the patch sizes used in Section 5. Next, both the benign and the attacked training images are fed to an image classifier to capture the output logits for all the classes. Then, the five-layer patch detector is trained with supervised learning using stochastic gradient descent, with the benign logits labeled benign and the attacked logits labeled attacked. The detector loss L_{DT} is the cross-entropy loss L_{CE} between the prediction y_{DT} of the detector D on the training logits T and the ground truth class y_{GTT}:

$$L_{DT} = L_{CE}(y_{DT}, y_{GTT}), \tag{3}$$

where $y_{DT} = D(T)$. Each detector used an optimizer momentum of 0.9, but the number of epochs, batch size, and learning rate used to train each detector can be found in Table 2 along with a discussion on how the parameters were chosen in Section 5. Evaluations on patch detectors with additional layers are provided in Section H.

4.2 Patch Localization

The patch localization step consists of three sub-steps: (i) image segmentation, (ii) attribution analysis, and (iii) segment selection. The overall flow is shown in Figure 5. The framework first obtains candidate segments using image segmentation. Next, attribution analysis is used to identify the input features that have the strongest contribution to the model output decision. The final step is to use the attribution map of input features to score the candidate segments. Next, the candidate segment with the highest score is selected to be the patch mask.

Image segmentation: The input to image segmentation is an image I with a detected patch. An image segmentation method is used to obtain n segments, as follows:

$$\text{Segmentation_method}(I) = [S_1, \ldots, S_n], \tag{4}$$

Table 2 PATCHOUT patch detector parameters used during training

Classifier / Patch	Number of Epochs	Batch Size	Learning Rate
VGG19 LaVAN	15	128	1e-4
VGG19 GDPA	5	128	1e-4
VGG19 AP	5	128	1e-3
VGG19 TPA	15	128	1e-3
ResNet50 LaVAN	15	128	1e-4
ResNet50 GDPA	5	256	1e-3
ResNet50 AP	5	128	1e-3
ResNet50 TPA	10	128	1e-3
MobileNet LaVAN	5	128	1e-3
MobileNet GDPA	5	256	1e-3
MobileNet AP	5	128	1e-3
MobileNet TPA	10	64	1e-3

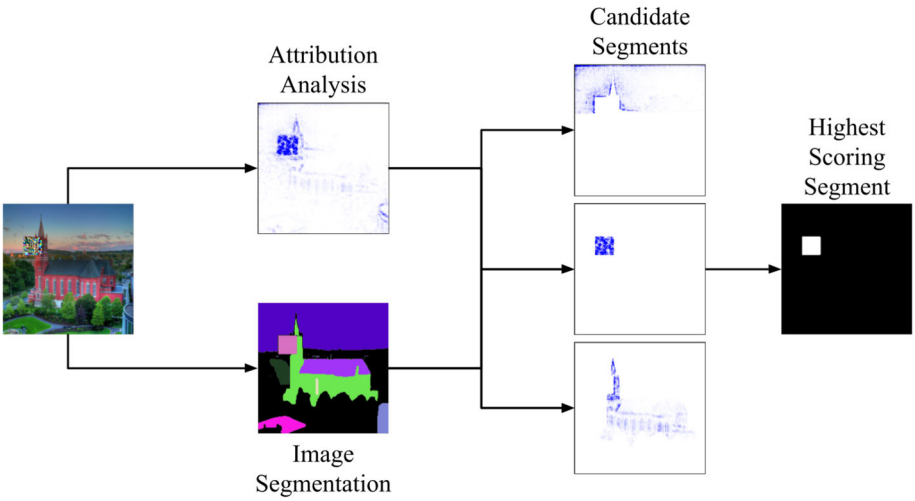

Fig. 5 PATCHOUT localization. Combining attributions and segments produces segmented attributions. The attribution image with the highest average absolute pixel values is used as the patch mask

where S_1 to S_n are binary masks with the same height and width as the input image I. Within each mask, a 1 indicates that the pixel belongs to the segment and a 0 indicates that the pixel does not belong to the segment.

We leverage the pre-trained foundational image segmentation model SAM [32] in the proposed framework. SAM is composed of a Vision Transformer [35]-based image encoder, a CLIP [36]-based prompt encoder, and a transformer [37]-based mask decoder. The image encoder takes an image as input and creates an image embedding. The prompt encoder either encodes a full image mask, or takes bounding boxes, targeted masks, or points as input and generates embeddings for them. The mask decoder combines the image and prompt embeddings to generate the output segmentation masks. There are also parameters, such as confidence and stability thresholds, that can be set that affect the generated segments.

SAM was trained using 11 million images with 1 billion segmentation masks. We find that configuring SAM to use coarse segments results in patches of various types being partitioned into an independent segment. We show examples of segments with different granularity in Section B. This allows the semantic consistency between segments and patches to be used to localize patches. In particular, it transforms the patch localization problem into a segment selection problem. The output of the image segmentation step is a set of candidate segments for the patch.

Attribution analysis: Attributions algorithms compute the contribution of each input feature to the output decision [38, 39]. The highest magnitude attributions are given to the pixels that contribute to the output class the most. If a patch is introduced to cause misclassification, then the highest magnitude attributions must at least partially point to the region with the patch. Moreover, some recent attributions methods satisfy completeness theorems that guarantee that attributions are proportional to the model predictions. Compared with using the ad hoc approaches based on high intensity neurons or features [2, 12], the use of attribution methods provides a mathematically sound approach to locating adversarial patches. The attributions A with respect to an image I, image classification model F, and top output class i are computed, as follows:

$$A = \text{Attribution_Method}(I, F, i) \tag{5}$$

where the attributions are gray scaled by default to avoid having independent attributions for separate color channels. The attribution map has the same height and width as the input image I.

In our implementation, we use guided backpropagation [33] to compute the attributions. This approach works by backpropagating through a model to the input, with respect to a given output logit previously predicted by the same model for a given image. During backpropagation, negative gradients are zeroed out after nonlinearities. Allowing the negative gradients to backpropagate reduces the magnitude of activation values in the previous layer that correspond to important pixels, which compounds until the final visualization, making it more difficult to distinguish important regions from noise. Zeroing the negative gradients stops the previous layer activation values corresponding to important pixels in the input from being reduced, resulting in a much clearer visualization [33]. Guided backpropagation was selected because it produces the clearest attributions for attacked images. We provide results justifying the selection of the attribution method in Section E. SmoothGrad [34] is also used in conjunction with guided backpropagation to produce higher quality attributions. SmoothGrad adds Gaussian noise with standard deviation σ to a batch of copies of the input image and applies the selected attribution method to each of the noised images. The resulting batch of attributions is then averaged to produce a single clearer attribution of the image. PATCHOUT uses a batch size of 25 and the σ value used is discussed further in Section K.

Segment Selection: The final step of the patch localization is to select one of the candidate segments to be the patch mask. The selection is guided by the input feature attributions. We score each of the segments by their average absolute attribution. Next, the patch mask is determined to be the candidate segment with the highest score. Attributions on the target object have been shown to have both large positive and negative scores [33]. Additional experiments on the attributions of the patch and non-patch locations are provided in Section D. The score of a segment S_n and corresponding attribution map A_n is mathematically computed, as follows:

$$score(S_n) = \frac{sum(S_n \circ |A_n|)}{count(S_n)}, \tag{6}$$

where $\circ$ denotes the Hadamard product operator, $sum(S_n \circ |A_n|)$ denotes the element-wise summation of the resulting matrix, and $count(S_n)$ denotes the number of pixels belonging to the segment S_n.

The final part of the defense is to restore the image through inpainting. Inpainting is applied to the image based on the location of the patch in the patch mask.

4.3 Patch Inpainting

Restoring an image with a known patch location is equivalent to an inpainting problem. Previous works [9, 17, 40] have used inpainting on recovered patch locations. Blacking out the patch location is a viable strategy since it removes the patches' influence, but inpainting can help improve accuracy by replacing patch pixels with pixel values that are within the image's original distribution of values. The trade-off is that inpainting is significantly slower than blacking out the patch. We explore the effects of blacking out and inpainting patch locations more in Section C.

PATCHOUT uses a generative inpainting network called DeepGIN [41]. The network is composed of two generators and two discriminators. The generators are split into a coarse generator that creates a rough outline of the area that needs to be inpainted and a refining generator that adds all of the details to the inpainted location. The discriminators are only used during training to help the detail refining generator to better learn to generate details. DeepGIN inpaints locations in an image based on the masks passed to it. PATCHOUT passes the patch mask found during patch localization to DeepGIN in order to inpaint the patch location.

5 Experimental Evaluation

In this section, we evaluate the effectiveness of the proposed PATCHOUT framework. The framework is implemented in Pytorch and the experiments have been performed locally on an NVIDIA A4000 with 6 CPU cores as well as two servers containing 4 NVIDIA A40s each and 32 CPU cores each. Linux is the main operating system used. All model training and experiments used ImageNet [18]. The patch detector was trained using attacked images generated from the ImageNet training set. The PyTorch implementation of VGG19 [42], Resnet50 [43], and MobileNet V3 Large [44] are the neural networks used in evaluation and have a 72%, 76%, and 74% accuracy respectively.

For the Local and Visible Adversarial Noise (LaVAN) patch [5] and Generative Dynamic Patch Attack (GDPA) [7], we generated patches of size 1%, 2%, and 3% of the image. For the Adversarial Patch (AP) [3] and Texture-based Patch Attack (TPA) [6], we generated patches of size 10%, 15%, and 20% of the image. The patches were allowed to be placed anywhere in the image for all four patches. We evaluate on sets of 3000 images for all experiments in Sections 5.1, 5.2, 5.3 and 250 images in Section 5.4. Each image included in the experimental sets is correctly classified by an unmodified classifier when benign. Each attacked image contains a patch that successfully causes the classifier to predict an incorrect class. No unsuccessfully attacked images are included in our experimental sets. Each patch size for each patch has its own set of 3000 images. We use the implementation of the LaVAN patch in [45], generative dynamic patch attack in [46], adversarial patch attack in [47], and texture-based patch attack in [48].

We list the hyperparameters used in the generation of each patch. LaVAN patches were trained for 250 iterations with a step size of 0.05. Patch generators for the generative dynamic patch were trained with $\alpha = 1$, $\beta = 300$, a batch size of 32, and a learning rate of 0.0005 for 30 epochs. Adversarial patches were generated using a batch size of 64 and a learning rate of 0.1 for 1000 iterations. All transformations were set to their minimum possible values. Both the generative dynamic patch and adversarial patch were trained on images from the Imagenet training set. The texture-based patches were provided by the authors' Github, but the reinforcement learning patch placement used a batch size of 500 for 50 steps.

We evaluate the effectiveness of a patch defense technique in terms of restored classification accuracy, which is the classification accuracy after a defense has been deployed. For clarity, we report the restored classification accuracy for both benign and attacked inputs separately.

We compare the PATCHOUT framework with state-of-the-art approaches such as PatchCleanser [14], Derandomized Smoothing (DRS) [10], JPEG compression (JPEG) [49], Local Gradients Smoothing (LGS) [8], and Segment and Complete (SAC) [15]. The PatchCleanser implementation was taken directly from the author's publicly available Github repository [50]. The Derandomized Smoothing implementation was taken from the repository in [45], which is the implementation of Derandomized Smoothing used in [12]. The JPEG Compression implementation was from the Adversarial Robustness Toolbox [47]. The Local Gradients Smoothing implementation was taken from the Github repository in [51]. The Segment and Complete implementation was taken directly from the author's publicly available Github repository [52].

We list the hyperparameters used for all approaches. For PatchCleanser, the number of masks was set to 20. For DRS, block ablations with side lengths of 5 were used and the confidence threshold used was 0.3. For JPEG, the output image quality was set to 50. For LGS, the threshold was set to 0.2, the smoothing factor to 2.3, the block side lengths to 15, and the allowed overlap to 5×5. For SAC, 16 filters were used for the segmenter and estimated patch sizes of 25, 50, 75, 100, and 125 were used. For PATCHOUT, the SmoothGrad standard deviation parameter was set to 0.15 and the SAM minimum confidence threshold was set to 0.99.

Table 2 provides the training parameters for each of the patch detectors used by PATCHOUT in Section 5. Detectors that end on a loss around 0.1 during training provide the best accuracy without overfitting or underfitting. The parameter values provided in the table were used to have the patch detectors achieve losses between 0.05 and 0.15. Additionally, the detectors are prone to getting stuck in local minima during training, which has also affected the chosen learning rates and batch sizes in order to avoid that issue.

5.1 Evaluation of Patch Detection

In this section, we evaluate the effectiveness of the proposed patch detection networks. Experiments on larger detector networks are provided in Section H. The accuracy of the patch detector for benign and attacked images is shown in Table 3. The benign accuracy and the attacked accuracy is defined to be the percentage of benign images and attacked images that are correctly detected as benign and attacked, respectively. Each row in the table corresponds to the classification network a detector was trained on while each patch represents the patch the detector was trained to detect. The VGG19 row under the LaVAN patch of sizes 1%, 2%, and 3% represents a detector that was trained to detect LaVAN patches aimed at breaking VGG19. For comparison, the accuracy of Jensen-Shannon distance as a detection method is

Table 3 PATCHOUT patch detector and Jensen-Shannon Distance accuracy (%) on different patch and classification network combinations

	Patch Type	Local and Visible Adversarial Noise [5]						Generative Dynamic Patch [7]					
	Patch Size	1% pixels		2% pixels		3% pixels		1% pixels		2% pixels		3% pixels	
	Image Type	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv
VGG19 [42]	JS Distance	66.0	58.2	66.9	42.4	67.0	35.1	66.7	0.2	66.8	14.8	66.7	0.2
	Detector	97.7	96.4	97.5	97.1	97.4	96.2	99.8	99.7	99.7	100.0	99.7	100.0
ResNet50 [43]	JS Distance	60.0	79.6	64.8	72.8	68.7	67.8	55.7	29.1	68.6	3.4	69.0	1.5
	Detector	97.8	92.8	97.5	97.6	96.7	98.4	99.5	98.3	99.7	99.9	99.4	100.0
MobileNet [44]	JS Distance	67.7	98.4	72.5	99.6	74.7	100.0	55.8	70.9	67.9	68.7	68.7	71.8
	Detector	97.5	90.5	97.7	98	97.9	99.2	96.4	88.9	96.6	99.5	96.7	99.9
	Patch Type	Adversarial Patch [3]						Texture-based Patch Attack [53]					
	Patch Size	1% pixels		2% pixels		3% pixels		1% pixels		2% pixels		3% pixels	
	Image Type	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv
VGG19 [42]	JS Distance	70.7	18.0	70.9	14.7	70.3	14.8	70.3	39.5	70.7	26.0	70.9	16.4
	Detector	98.3	99.2	98.3	99.3	98.4	99.2	94.7	90.2	94.5	93.2	94.4	94.6
ResNet50 [43]	JS Distance	73.0	28.3	74.6	29.3	74.6	39.2	74.3	56.5	74.5	49.1	75.2	48.4
	Detector	97.8	99.1	97.3	99.2	97.3	99.1	92.2	94.4	91.6	97.1	91.5	98.3
MobileNet [44]	JS Distance	77.9	46.0	79.8	35.3	80.6	33.0	77.8	69.5	78.7	62.9	79.9	60.9
	Detector	99.5	96.7	99.3	99.2	99.2	99.3	94.4	94.1	93.9	97.3	93.7	98.2

also provided in Table 3, where images with distances less than or equal to 0.425 are detected as benign and images with distances greater than 0.425 are detected as attacked. This value is the mean of the benign and attacked distances from Table 1. The logits of the image being detected are compared against the logits of an image of the same class from the Imagenet training set.

The detectors achieve a benign accuracy above 96% for LaVAN, GDPA, and AP and above 91% for TPA. The detectors consistently achieve an adversarial accuracy above 90%, with the only adversarial accuracy below that reported for the MobileNet GDPA detector on a 1% sized patch at 88.9%. The most difficult patch to detect is TPA, which could be due to the use of class textures as starting points rather than noise, leading to greater semantic consistency generated by the patch. Additionally, there does not seem to be any clear difference between the accuracy of detectors trained on the same patches but different models. In contrast, detecting images through the Jensen-Shannon distance is much less accurate than using a detector network. The benign accuracy falls between 65% and 80%, but the adversarial accuracy varies widely between 0% and 100%. The adversarial accuracy tends to decrease as the patch size increases, pointing to a smaller gap in the average distances between benign and attacked outputs and an increase in semantic consistency as the patch size increases.

We conclude that class consistency through patch detector networks is an effective technique for detecting patches within image classification scenarios. Ideally, we would have wanted to compare with the patch detector ScaleCert [2]. However, their code is not available publicly, so it is not easy to make a fair comparison. Nevertheless, the accuracy of the patch detection reported in ScaleCert is broadly in the 50% to 60% range for LaVAN-like patches. In contrast, our reported accuracy is over 95% for the same type of patches.

To gain a deeper understanding of the patch detector, we analyze the cases when the detector fails. We cannot observe any discernible reasons for the misclassification of benign images. On the other hand, when analyzing attacked images that have been misclassified as benign, we found that the patch class tends to be similar to the image class. For example, an image of a hammerhead shark had a tiger shark patch applied to it, causing the detection network to mistakenly report the image as benign. However, misclassifications between related classes may be less dangerous than between unrelated class. While it would be ideal to detect attacks of any class type, the detector substantially narrows the attack surface of the classifier.

5.2 Evaluation of Patch Localization

In this section, we evaluate the effectiveness of patch localization. The evaluation is performed using the intersection over union (IOU) metric. IOU is normally used in object detection to determine how well an object detector can detect the location of objects as compared to their ground truths. It measures the overlap of two objects A and B and can be formally defined as:

$$IOU = \frac{A \bigcap B}{A \bigcup B}. \tag{7}$$

Values closer to 0 represent smaller areas of overlap and values closer to 1 represent larger areas of overlap. We show the IOU results with respect to the detected patch locations and the actual patch locations in Table 4.

The results show that PATCHOUT is able to accurately determine the patch location, with IOU scores for LaVAN and GDPA between 0.80 and 0.99 and IOU scores for AP and

Table 4 Evaluation of PATCHOUT patch localization using intersection over union between the true patch location and PATCHOUT's detected patch location

Patch Type	LaVAN			GDPA		
Patch Size	1%	2%	3%	1%	2%	3%
VGG19	0.96	0.99	0.99	0.92	0.97	0.99
ResNet50	0.95	0.98	0.99	0.92	0.93	0.98
MobileNet	0.82	0.88	0.90	0.86	0.94	0.95
Patch Type	AP			TPA		
Patch Size	10%	15%	20%	10%	15%	20%
VGG19	0.77	0.77	0.77	0.78	0.78	0.78
ResNet50	0.77	0.77	0.77	0.79	0.76	0.72
MobileNet	0.76	0.76	0.76	0.83	0.84	0.78

TPA between 0.76 and 0.84. This also shows that LaVAN and GDPA are able to be located more accurately than AP and TPA. LaVAN and GDPA are more noise-based whereas AP and TPA tend to mimic the textures and classes of the images, which may make attributions less distinguishable. Similarly, the number of high magnitude attributions might not scale the same as the size of the patch, making the patch segments less distinguishable from other segments during localization.

5.3 Evaluation of PATCHOUT Framework

In this section, we evaluate the complete flow of the PATCHOUT framework. PATCHOUT and other defenses are evaluated on LaVAN and GDPA in Table 5 and AP and TPA in Table 6. The columns labeled 'benign' are benign images and the columns labeled 'adv' are attacked images. Bolded numbers indicate the best results for that column. Evaluations on a larger number of patch sizes and patches with irregular shapes are provided in Section G and Section F, respectively.

Table 5 shows that PATCHOUT achieves a 99.2% average benign accuracy on both LaVAN and GDPA across all three classification models, which is a higher average accuracy than the patch detector's accuracy. This improvement occurs because benign images still contain much information about their ground truth class, so even if a benign image is misclassified as attacked and a small portion of the image is removed, a classifier can still classify the images correctly. PatchCleanser achieves an average accuracy of 94.2% and LGS achieves a slightly lower benign accuracy at about 90.3% across both patches and all classifiers. JPEG and DRS perform the worst on benign images, with maximum benign accuracies of 83.5% and 85.1%, respectively. SAC is the only defense able to perform better than PATCHOUT, achieving only 1% to 3% better benign accuracies. However, PATCHOUT achieves much higher recovered accuracies, outperforming the next highest defense by an average of 3.4% on LaVAN and 5.7% on GDPA. The only instances of a defense performing better than PATCHOUT on recovered accuracy across both Tables 5 & 6 occur on the 1% LaVAN patch with MobileNet, and the 1% GDPA across all classifiers. Additional discussions on the results of PATCHOUT in Table 5 are provided in Section I.

Table 6 shows that it is more difficult to defend against AP and TPA, which is understandable given that it is more difficult to locate both patches. Defenses have similar benign accuracies to those shown in Table 5 for AP and TPA. PATCHOUT achieves the highest recovered accuracy across both AP and TPA, outperforming the next highest defenses by

Table 5 Defense accuracy (%) on Local and Visible Adversarial Noise and Generative Dynamic Patch Attack

Patch Type	Local and Visible Adversarial Noise [5]						Generative Dynamic Patch [7]					
Patch Size	1% pixels		2% pixels		3% pixels		1% pixels		2% pixels		3% pixels	
Image Type	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv
Model	VGG19 [42]											
JPEG [49]	78.1	23.8	78.5	8.7	78.9	4.2	78.8	17.3	78.7	0.4	78.9	0.1
LGS [8]	92.0	90.1	91.9	88.3	92.1	85.5	92.1	93.5	92.1	91.9	92.1	90.7
PatchCleanser [14]	94.4	61.1	94.1	54.3	94.2	48.3	94.4	65.6	94.2	58.5	94.3	53.6
DRS [10]	48.7	4.5	50.0	1.9	50.6	0.6	50.5	6.7	50.6	1.7	50.7	0.2
SAC [15]	**100.0**	90.4	**100.0**	79.5	**99.9**	72.9	99.9	**95.2**	99.9	88.5	99.9	75.9
PATCHOUT	99.1	**91.6**	99.1	**94.1**	99.0	**92.6**	**100.0**	94.3	**100.0**	**98.1**	**100.0**	**97.8**
Model	ResNet50 [43]											
JPEG [49]	74.2	48.3	79.3	50.6	82.1	52.1	68.9	47.9	82.5	37.9	83.5	43.3
LGS [8]	88.4	83.2	92.5	86.7	93.2	87.9	86.2	79.6	93.6	84.7	94.0	90.9
PatchCleanser [14]	91.4	42.6	93.5	45.0	94.2	44.6	89.7	36.8	94.9	56.2	95.0	51.6
DRS [10]	48.2	16.5	54.5	17.3	59.3	18.7	41.1	29.3	59.8	32.7	59.3	18.7
SAC [15]	**100.0**	84.7	**100.0**	80.5	**100.0**	79.2	**100.0**	90.4	**100.0**	82.3	**100.0**	86.6
PATCHOUT	98.8	**86.7**	99.0	**93.9**	98.7	**94.6**	99.9	**93.1**	99.9	**94.6**	99.9	**98.6**
Model	MobileNet V3 Large [44]											
JPEG [49]	70.4	43.7	76.9	46.1	81.3	45	58.1	31.6	70.9	32.8	75.5	23.7
LGS [8]	87.5	76.8	91.7	81.6	92.7	80.9	75.9	58.6	87.4	76.8	90.3	80.3
PatchCleanser [14]	95.1	67.8	96.0	73.4	96.7	72.7	91.6	45.9	95.0	60.5	96.0	63.4
DRS [10]	76.4	38.1	83.1	43.9	85.1	41.2	61.1	27.8	75.8	34.7	80.0	27.8
SAC [15]	**100.0**	**84.9**	**100.0**	81.6	**100.0**	78.6	**100.0**	**85.5**	**100.0**	77.4	**100.0**	76.7
PATCHOUT	98.8	75.0	99.0	**85.3**	99.2	**87.4**	98.3	76.6	98.6	**91.9**	98.3	**92.3**

Table 6 Defense accuracy (%) on Adversarial Patch and Texture-based Patch Attack

Patch Type	Adversarial Patch [3]						Texture-based Patch [6]					
Patch Size	10% pixels		15% pixels		20% pixels		10% pixels		15% pixels		20% pixels	
Image Type	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv
Model	VGG19 [42]											
JPEG [49]	79.1	1.2	79.1	0.1	79.3	0.1	78.5	17.8	79.1	13.6	79.1	9.2
LGS [8]	92.3	55.2	92.2	47.4	92.2	38.8	91.8	32.1	92.3	23.1	92.3	16.1
PatchCleanser [14]	91.8	25.0	87.1	17.4	79.2	14.2	91.3	21.3	86.9	21.3	78.9	20.4
DRS [10]	49.6	0.8	50.5	0.2	50.6	0.2	49.0	8.1	49.5	5.3	49.8	3.3
SAC [15]	**99.9**	65.0	**99.9**	57.6	**99.9**	9.3	**99.9**	45.8	**99.9**	34.6	**99.9**	8.6
PATCHOUT	99.2	**87.8**	99.3	**81.3**	99.4	**72.5**	97.7	61.4	97.5	**56.8**	97.5	**49.0**
Model	ResNet50 [43]											
JPEG [49]	82.2	14.1	84.3	6.3	84.6	5.5	84.0	23.4	84.9	19.0	85.1	14.2
LGS [8]	92.7	59.9	93.8	53.0	94.0	47.0	93.8	33.5	94.0	25.3	94.2	21.3
PatchCleanser [14]	91.5	20.9	88.2	20.6	81.2	25.7	93.1	20.8	88.5	22.2	81.7	23.7
DRS [10]	56.5	9.1	59.2	6.5	61.8	5.2	59.9	11.7	62.3	8.7	63.0	5.2
SAC [15]	**100.0**	60.4	**100.0**	58.9	**100.0**	18.1	**100.0**	45.4	**100.0**	35.0	**100.0**	11.9
PATCHOUT	99.2	**86.2**	99.1	**79.6**	99.2	**74.5**	96.7	**60.1**	96.6	**55.0**	96.5	**46.0**
Model	MobileNet V3 Large [44]											
JPEG [49]	79.2	7.8	81.0	3.0	83.0	2.0	79.1	10.9	80.9	8.9	81.7	6.4
LGS [8]	91.7	43.0	92.9	39.8	93.4	33.7	91.6	20.2	92.9	16.1	93.1	11.4
PatchCleanser [14]	95.2	47.5	93.9	42.0	91.6	36.3	95.5	41.9	93.6	39.0	91.4	38.1
DRS [10]	83.8	6.9	85.4	5.3	86.3	2.7	84.9	7.9	85.1	5.8	86.0	4.5
SAC [15]	**100.0**	56.7	**100.0**	58.9	**100.0**	14.4	**100.0**	42.7	**100.0**	32.4	**100.0**	10.9
PATCHOUT	99.8	**82.5**	99.7	**80.8**	99.6	**74.5**	98.5	**59.8**	98.2	**58.8**	98.1	**50.2**

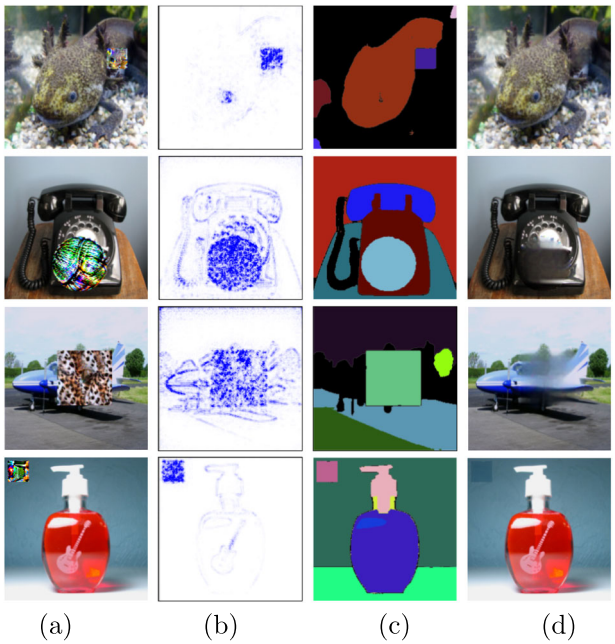

Fig. 6 (**a**) The attacked image. (**b**) The attributions of the image. (**c**) The segments of the image. (**d**) The image after localization and inpainting

27.9% and 19.9% respectively. SAC and LGS have diffculty with AP and TPA, seeing a more substantial drop in recovered accuracy than PATCHOUT. SAC achieves an average recovered accuracy of 44.3% on AP and 29.7% on TPA while LGS achieves an average recovered accuracy of 46.4% on AP and 22.1% on TPA. PatchCleanser, JPEG, and DRS all perform poorly on AP and TPA. PatchCleanser attains an average recovered accuracy of 27.7% on AP and 27.6% on TPA. JPEG attains an average recovered accuracy of 6.1% on AP and 13.7% on TPA. DRS attains an average recovered accuracy of 4.1% on AP and 6.7% on TPA.

We qualitatively show the performance of the PATCHOUT framework using four examples in Figure 6. The columns show the attacked images, the image attributions, the image segments, and the resulting restored images. The examples show that the patches are accurately located using the attribution analysis and image segmentation techniques. Combining the two approaches allows for patches to be easily localized. As seen in the second column, patches tend to have the highest magnitude attributions, which makes them easy to separate out from the other segments shown in the third column. Even in cases where there are high attributions that are not on the patch, such as in the third row, the patch location is still able to be recovered. Next, inpainting relatively accurately restores the original input image. Clearly, it is a difficult task to restore the telephone when a large portion of the telephone is covered by a patch. A possible solution to this problem would be to use an inpainting algorithm that is trained to be aware of the classes that exist in the ImageNet dataset. Additional visualizations of the PATCHOUT framework are provided in Section M.

Table 7 Evaluation of a LaVAN-trained detector across unseen patches

Patch Size	1% pixels		2% pixels		3% pixels	
Patch Type	benign	adv	benign	adv	benign	adv
LaVAN	97.8	92.8	97.5	97.6	96.7	98.4
GDPA	96.9	82.4	96.4	97.8	96.0	99.7
Patch Size	10% pixels		15% pixels		20% pixels	
Patch Type	benign	adv	benign	adv	benign	adv
AP	97.1	82.6	96.9	83.4	96.3	76.2
TPA	96.6	21.4	96.3	21.2	96.0	21.3

5.4 Evaluation of Unseen and Adaptive Attacks

In this section, we evaluate the robustness of PATCHOUT to unseen and adaptive attacks. The performance against unseen attacks is analyzed in Section 5.4.1 and the susceptibility against adaptive attacks is studied in Section 5.4.

5.4.1 Evaluation Against Unseen Attacks

Detectors will not always operate under ideal circumstances where the patch they were trained on is the one they are trying to detect. Table 7 evaluates a ResNet50 attacking LaVAN-trained detector on its intended patch, LaVAN, as well as the unseen patches GDPA, AP, and TPA. As expected, the benign accuracies are about the same since the images are from the same dataset. While the recovered accuracies for GDPA and AP are degraded, the detector is far from broken, still achieving a recovered accuracy on those patches that is better than other defenses compared against in Table 6. The LaVAN-trained detector's performance is severely degraded on TPA. The expected semantics of the TPA logits are likely significantly different than those of the other patches because TPA is generated using image textures rather than starting from noise like LaVAN, GDPA, and AP.

5.4.2 Evaluation Against Adaptive Attacks

In this section, we evaluate the robustness of PATCHOUT to adaptive attacks. We evaluate one adaptive attack that targets the detection step and one adaptive attack that targets the localization step. The detection attack attempts to bypass the patch detector. The localization attack is an attribution based attack that aims to distort the attributions such that an incorrect segment is labeled as the patch. Both of the adaptive attacks are based on the LaVAN patch. We first outline how the LaVAN patch is generated. Next, we explain how the LaVAN attack is modified to create the adaptive attacks.

The base LaVAN patch is generated by iteratively taking the gradients with respect to a loss, multiplying the gradients by some step size, and adding the reduced gradients to a patch location. The loss L_{LaVAN} is as follows:

$$L_{LaVAN} = L_{CE}(\hat{y}, y) - L_{CE}(\hat{y}, y_{gt}), \tag{8}$$

where $\hat{y} = F(\hat{x})$ and $y = F(x)$. The loss L_{LaVAN} contains two terms. The first term is the cross-entropy loss L_{CE} between the attacked logits $\hat{y}$ of the current iteration of the attacked image $\hat{x}$ on a model F and the benign logits y of the benign image x on F. The second term

is the L_{CE} between $\hat{y}$ and the ground truth class y_{gt} of x. L_{LaVAN} is meant to minimize the probability of F to predict y_{gt}. Next, the gradient ∇_L of L_{LaVAN} with respect to $\hat{x}$ is found:

$$\nabla_L = \frac{dL_{LaVAN}}{d\hat{x}}. \tag{9}$$

Finally, ∇_L is multiplied by a step size $step_size$ and added in to the predetermined location of the patch in $\hat{x}$:

$$\hat{x} = \hat{x} + step_size \times \nabla_L. \tag{10}$$

The process in Equations 8, 9, 10 is repeated for some number n of iterations, where n is a hyperparameter. Both of the adaptive attacks are constructed by combining the L_{LaVAN} loss with an additional loss term while keeping the remainder of the patch generation process the same.

The Adaptive Detector based Attack: The loss for the detector based adaptive attack L_D is formulated as follows:

$$L_D = L_{LaVAN} + L_{CE}(y_D, y_{benign}) \times \epsilon, \tag{11}$$

where $y_D = D(\hat{y})$. The term L_{CE} is the cross-entropy loss between the detector logits y_D of $\hat{y}$ on the PATCHOUT detector D and the ground truth representation of a benign image y_{benign}. This is minimizing the likelihood of D predicting $\hat{y}$ to come from an attacked image. Practically, the patch also tries to force the class logit magnitudes to be arranged in such a way that D considers the classes to be semantically consistent, and therefore the image benign. The term is also multiplied by a scaling factor ϵ such that L_{LaVAN} and L_{CE} are the same order of magnitude. The new loss term is significantly smaller than L_{LaVAN}, so it had a negligible impact on the patch, which was then unable to bypass the detector. By scaling up the new loss term to have the same order of magnitude as L_{LaVAN}, it now has an impact on the generation of the patch, which is now able to bypass the PATCHOUT detector.

The Attribution-based Adaptive Attack: The attribution loss L_A follows a similar scheme:

$$L_A = L_{LaVAN} + L_{CE}(A_{patch}, 0.0001) \times \epsilon, \tag{12}$$

where the new loss term is the L_{CE} between the attributions of the patch A_{patch} and a matrix of the same size containing the value 0.0001. This new term attempts to minimize the attribution magnitudes of the patch since the PATCHOUT localization step relies on the patch having a higher average magnitude than other segments. The attributions used in A_{patch} are the same that are used in the PATCHOUT localization step. We also generate two other adaptive attack examples against JPEG and LGS. For the attack against JPEG, we apply JPEG compression with a quality rate of 50 to the patch location in $\hat{x}$ in Equation 8 at each iteration. LGS relies on large image gradients to nullify the patch, so we apply Gaussian smoothing to the patch location in $\hat{x}$ in Equation 8 at each iteration to create a patch with the smallest possible image gradients.

The results of the defenses against the base LaVAN patch and the two adaptive attacks are evaluated in Table 8. Each set of 1%, 2%, and 3% image patches contain 250 successfully attacked images on ResNet50.

As expected, adaptive attacks are able to severely degrade defense accuracies. JPEG's average accuracy is halved, dropping from 50.3% on the base LaVAN patch to 24.0% on the adaptive LaVAN patch. LGS also sees a substantial drop in average accuracy, from 85.9% to 11.6%, meaning the defense is broken. PATCHOUT is resistant to an attack that

Table 8 Defense accuracies on the base LaVAN patch compared to an adaptive LaVAN patch

Attack Type	1% image	2% image	3% image
LaVAN on JPEG	48.3	50.6	52.1
JPEG patch on JPEG	29.6	25.6	16.8
LaVAN on LGS	83.2	86.7	87.9
Gaussian Smoothing patch on LGS	14.0	9.2	11.6
LaVAN on PATCHOUT	86.7	93.9	94.6
Detector-focused patch on PATCHOUT	0.4	0.0	0.0
Attribution-focused patch on PATCHOUT	89.6	94.0	91.6

Table 9 Comparison of PATCHOUT detector and PATCHOUT localization accuracies on the baseline and detector-focused LaVAN patch

PATCHOUT Step	Detector (%)			Localization (%)		
Patch Size	1% image	2% image	3% image	1% image	2% image	3% image
Baseline Patch	93.6	98.0	97.6	92.8	95.6	98.0
Adaptive Patch	0.8	0.0	0.0	85.2	86.8	95.6

targets the patch localization step, achieving a 91.7% average accuracy on the attribution-focused patch. The attribution-focused patch is likely unsuccessful due to the nature of attributions. Decreasing attribution values correspond to the classifier viewing those pixels as less important to the final image classification. If a patch has low magnitude attributions, the classifier is not assigning much weight to the patch's influence in the final classification of the image. Reducing patch attributions renders the patch ineffective.

PATCHOUT's detector is fully broken by the detector-focused patch. It is difficult for networks to be robust against fully white-box attacks because the attacker is able to incorporate the network's loss in the creation of an attack. The detector-focused adaptive attack uses the detector's loss to cause the detector to determine attacked images to be benign. This patch training objective causes misclassification in such a way that the top classes are semantically consistent with an incorrect top-1 class. For example, the top-1 class might be shark, but the patch forces the top-5 classes to be plane, kite, rocket, moon, and bird, which are all related to flight. We have also observed the detector-focused patch moving the top-1 class to an adjacent class, such as from tiger shark to whale shark. While this does bypass the detector, it is likely less harmful than an attack that changes a person to a car. It is also important to note that the detector is the part of the defense being bypassed rather than the patch localization step for this adaptive attack.

We show the difference between detector and localization accuracy on the base and detector-focused LaVAN patches in Table 9. The results show that the detector is able to be fully bypassed, but patch localization recovered accuracy is only slightly degraded, meaning the patch localization approach works well as long as the image is detected as an attacked image. Currently, the detector works well against baseline attacks, but needs to be strengthened so that it is robust against adaptive attacks.

Since adaptive attacks should be able to break their target defense, it is important for the defender to make it as difficult as possible for the adaptive attacker to successfully create an attack. We show the success rate of the base and detector-focused LaVAN patch when trying to generate 250 successfully attacked images in Table 10. We define the success rate as the

Table 10 Attack success rate of the base and detector-focused LaVAN patch when trying to generate 250 successfully attacked images

Patch Size	1% pixels patch	2% pixels patch	3% pixels patch
Baseline Patch	33.6	58.5	75.9
Adaptive Patch	9.7	22.7	40.3

number of images where the patch successfully changes the image class divided by the total number of images attacked. The base LaVAN patch success rates are 33.6%, 58.5%, and 75.9% for the 1%, 2%, and 3% patches respectively. The success rates drop when the attack is made adaptive. The worst success rate drop is at the 2% patch, which drops from 58.5% to 22.7%. Across all patch sizes, the success rate drops by an average of 31.8%. Clearly, creating a patch that targets the PATCHOUT detector causes the patch to become less reliable.

6 Limitations

In this section, we discuss the limitations of Patchout. These limitations include PATCHOUT being limited to the classification task, the PATCHOUT detector's susceptibility to adaptive attacks, PATCHOUT's localization focusing on finding a single patch, and PATCHOUT's computation intensity.

some recent defenses [54–56] have made efforts to be broadly applicable to different computer vision tasks because adversarial patches have become successful at attacking more domains. However, PATCHOUT is specifically designed to defend against patch attacks on classification models, so it is unable to reliably defend against patch attacks in other scenarios. Since PATCHOUT is unable to reliable defend models that have tasks other than classification, its range of use is limited. To have PATCHOUT work in a scenario such as object detection, both the detector and localization step would need to be modified. Similarly, converting between other scenarios, such as from object detection to image segmentation, would require task-specific changes to be implemented. The detector would need to be trained on some task-specific model outputs. The attributions used during patch localization would either need to be replaced with a task-specific attribution method or some other pixel scoring approach. The interaction between the attribution and segments from SAM would also need to be modified to account for having multiple model outputs, such as bounding boxes or segments, rather than just one ground truth class.

The detector is susceptible to adaptive attacks, as shown in Section 5.4. The detector is difficult to fortify against adaptive attacks because it is a neural network itself. Adaptive attacks are white-box attacks, which are very difficult for networks to be robust against because attackers have full access to the model weights, gradients, and loss. Having access to these components allows an attacker to easily disrupt the network by using the above components as terms to optimize over during the attack generation. Like any other network, the patch can use the detector loss during patch generation to attack the detector. The detector loss can be added in to the original patch loss, allowing the patch to jointly minimize the likelihood of the classification model predicting the ground truth class of the image and minimize the likelihood of the detector to predict the attacked image as attacked.

Commonly, only one patch is used in patch attack evaluations, so PATCHOUT is designed to find a single patch within an image. Malicious actors are not limited to using only a single patch at a time, meaning this a limiting factor of PATCHOUT. Adding additional patches

reduces PATCHOUT's ability to successfully defend an image. Experiments on images with multiple patches are provided in Section L. Altering the patch localization step to gather the segments with the highest attributions and performing an iterative masking of the segments to determine how each segment affects the model output could serve as a way to overcome this limitation.

The PATCHOUT framework is computationally expensive. A model call is required for the initial patch detection and more model calls are required during patch localization due to the combination of Guided Backpropagation and SmoothGrad. SAM is used for segmenting images, which results in more time and resources used. The inpainting is performed by neural network, again resulting in more time and resources used. The multitude of model calls within PATCHOUT cause it to be relatively slow and resource intensive, making PATCHOUT unsuitable for real-time applications and devices that do not have sufficient hardware available. Evaluations on the runtime of different defenses is provided in Section J.

7 Conclusion & Future Works

In this paper, we propose a defense against adversarial patches for image classifiers called PATCHOUT. The framework detects patch attacks by analyzing the consistency of output classes of the image classifier. We train a lightweight five-layer neural network to analyze the consistency, which achieves a patch attack detection accuracy of over 95%. We view the concept of analyzing the class consistency as a promising orthogonal approach to methods in previous works that focus on high intensity features or neurons. When an attack is detected, we leverage techniques from attribution analysis, image segmentation, and image restoration to locate and remove the patch. The experimental evaluation demonstrates that PATCHOUT improves the recovered classification accuracy with respect to state-of-the-art defenses by 20%. We also demonstrate PATCHOUT's ability on unseen patches and adaptive attacks. Future works should explore other how semantic consistency can be further used to defend against patches.

Appendix A Overview

This appendix covers twelve additional sections relating to the PATCHOUT framework. Examples of images segmented at different granularities and explanations as to why we choose to use coarse segment granularity is discussed in Section B. The effects of blacking out and inpainting the patch location is discussed in Section C. Where high intensity attributions tend to be located on an attacked image is analyzed in Section D. A discussion around why Guided Backpropagation [33] was chosen over other attribution methods is presented in Section E. PATCHOUT is evaluated on irregularly shaped patches in Section F. PATCHOUT's recovered accuracy is evaluated across more patch sizes in Section G. PATCHOUT detectors with additional layers are evaluated in Section H. PATCHOUT's behaviour on small patches and benign images is discussed in Section I. The average runtimes for each defense are presented in Section J. Ablation studies for two parameters in PATCHOUT's localization step are presented in Section K. PATCHOUT's effectiveness on attacked images with multiple patches is evaluated in Section L. Figures showing the intermediate steps of PATCHOUT across multiple images and patches are displayed in Section M. All experiments in the appendices are run using ResNet50 as the classifier.

Appendix B Segmentation Granularity Analysis

In this section, we evaluate the choice of the segment granularity setting in the Segment Anything Model (SAM) [32]. There is a hyper parameter that controls the confidence threshold of segments in the SAM model which affects the granularity of the segments. With respect to this parameter, the average total number of segments in an image and the average number of segments that overlap with the patch is shown in Figure 7 and Figure 8. The results are obtained from 2000 images at every confidence level. The results for the LaVAN patch [5] and generative dynamic patch [7] are from 2% of pixels patches while the results for the adversarial patch [3] and texture-based patch [6] are from 15% of pixels patches.

Both figures show that SAM is generally confident in the segments it creates since the total number of segments and the segments overlapping the patch don't decrease much until about the 0.8 threshold value. However, for the purpose of patch localization, patches of any type are consistently segmented to a single segment when the confidence threshold is set close to 1.0. Therefore, we use this coarse setting within the PATCHOUT framework. It also allows PATCHOUT to better avoid any complications that arise when multiple segments overlap the patch. In the evaluations in the main paper, we use a 0.99 confidence threshold to allow for any minor confidence fluctuations.

We also provide a qualitative analysis of different confidence thresholds in Figure 9. We show the input image at five different confidence thresholds, starting from 0.8 and ending at

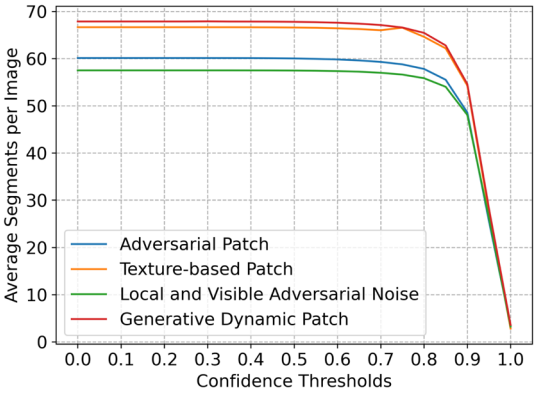

Fig. 7 The average number of segments SAM creates for an image with respect to different confidence thresholds

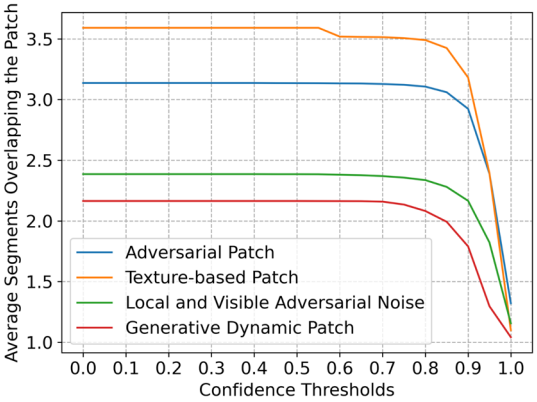

Fig. 8 The average number of segments generated by SAM that overlap with the patch location with respect to different confidence thresholds

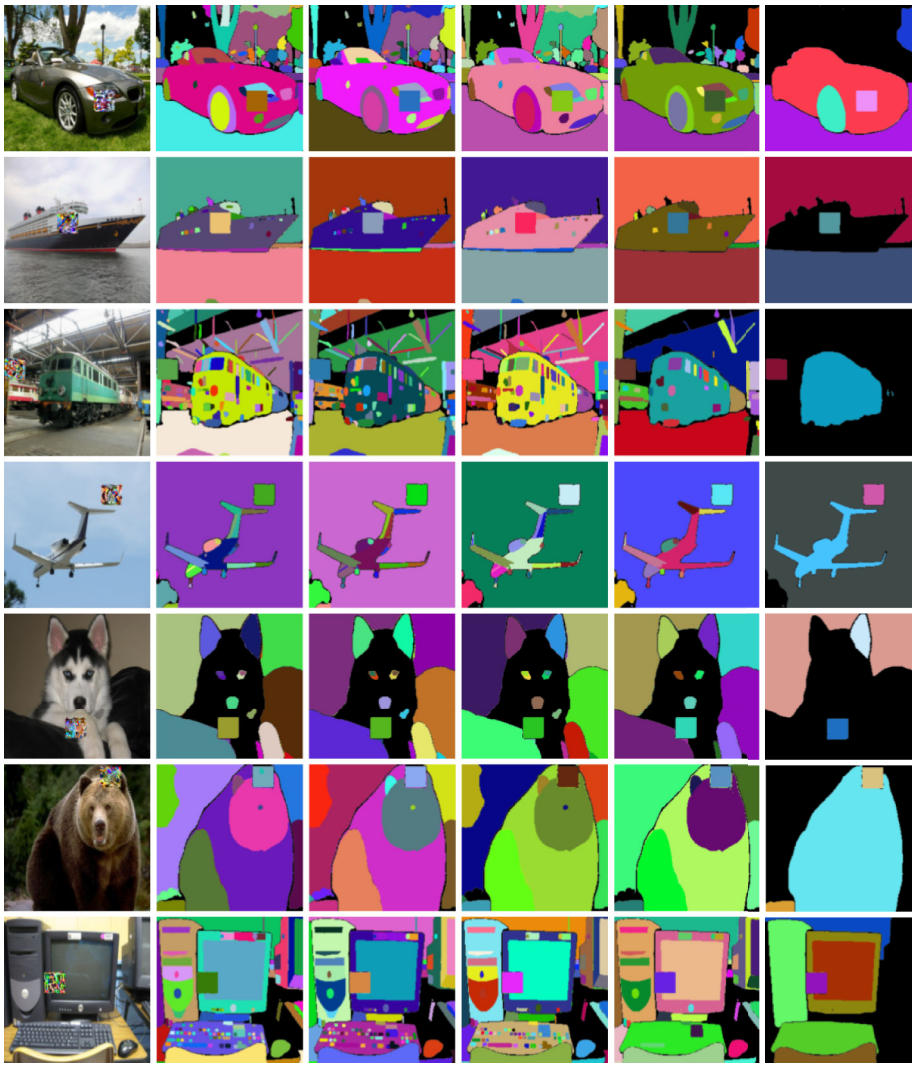

Fig. 9 (a) The input image at confidence thresholds of (b) 0.80, (c) 0.85, (d) 0.90, (e) 0.95, and (f) 1.0

1.0. We start at 0.8 since the mask composition variance is negligible for any lower confidence thresholds. The finer-grained segments gradually disappear between thresholds 0.8 and 0.95. For example, the train in the third row has many small details on the front of the train as well as the windows on the front and side at the 0.8 threshold, but by the 0.95 threshold, most of the front segments have disappeared and the number of window segments has also been reduced. Across all images, the biggest jump in segments comes between the 0.95 and 1.0 thresholds. At this point, only the patch and a few large segments remain, which again is why we decided to use a very high confidence threshold value.

Table 11 Recovered accuracy when the patch location is perfectly found across multiple patches using the blackout and inpainting removal approaches

Patch Type	Local and Visible Adversarial Noise			Adversarial Patch		
Patch Size	1% pixels	2% pixels	3% pixels	10% pixels	15% pixels	20% pixels
Blackout	92.6	91.6	91.4	72.8	77.4	69.2
Inpainting	98.2	94.0	94.4	87.4	87.8	80.4

Appendix C Patch Removal Approach

The most straightforward approach to removing a patch once it has been located is to fill-in the location with black pixels. The runtime overhead of blacking out the patch is negligible with respect to the image classification. While this allows an image's true class to be restored quickly, it does not recover the part of the image that was erased by the patch. Inpainting attempts to restore the pixel information lost by the patch, which assists in restoring the original class of the image. The main limitation of inpainting is that it is significantly slower than blacking out pixels.

The recovered accuracy when blacking out and inpainting the exact patch location for 500 LaVAN patches and adversarial patches of multiple sizes is shown in Table 11. Across all six patch sizes, inpainting outperforms blacking out. This is likely because inpainting fills the patch location with pixel values that fall within the unmodified image's original distribution of values, making it easier for the classifier to determine what the image is since these pixel values can be used to further determine the image's true class. Blacking out the pixels removes those pixels for consideration, which can be detrimental when the patch overlaps with the object representing the image class. There's also a greater difference in recovered accuracy occurring at the larger patch sizes. Another interesting trend is that patches become more difficult to defend against across both restoration approaches as the patches become bigger because they overwrite so many pixels.

Appendix D Attribution Value Density

In the main text, we mentioned that we expect the highest magnitude attributions to at least partially point towards the patch location. In Tables 12 & 13, we evaluate this claim by analyzing the average absolute attribution value of the patch and the rest of the image, denoted as background, across all four evaluation patches of mixed patch sizes. We use Guided Backpropagation and 2000 images for each patch size in the evaluation.

Table 12 Comparison of absolute attribution values between the patch and the rest of the image on small patches

Patch Type	Local and Visible Adversarial Noise			Generative Dynamic Patch		
Patch Size	1% pixels	2% pixels	3% pixels	1% pixels	2% pixels	3% pixels
Patch	0.0117	0.0110	0.0104	0.0170	0.0079	0.0123
Background	0.0001	0.0003	0.0004	0.0002	0.0002	0.0004

Table 13 Comparison of absolute attribution values between the patch and the rest of the image on large patches

Patch Type	Adversarial Patch			Texture-based Patch		
Patch Size	10% pixels	15% pixels	20% pixels	10% pixels	15% pixels	20% pixels
Patch	0.0065	0.0054	0.0050	0.0100	0.0084	0.0079
Background	0.0007	0.0009	0.0012	0.0010	0.0015	0.0020

Across both of the small patches in Table 12, the patch attributions are one to two orders of magnitude greater than the rest of the image. Similarly, the patch attributions for large patches, shown in Table 13, are always larger than the attributions in the rest of the image. However, the large patches have closer average absolute attribution values to the rest of the image than the smaller patches. The patch attributions tend to be within one order of magnitude of the rest of the image. As stated in the main text when discussing patch localization, this is likely due to the number of high magnitude patch attributions not scaling at the same rate as the patch size, leading to a drop in the average absolute attribution values.

Another important point to bring up is that we are attacking images with only a single patch per image, which is why we are able to use SAM [32] to find the patch location. This approach will be ineffective if multiple patches are introduced into a single image. However, Tables 12 & 13 show that it is possible to create a threshold to separate the patches from the rest of the image since the distance between patch and background average attribution values is so high. The threshold approach would be a useful way to use attributions to defend against multiple patches in one image.

Appendix E Attribution Analysis Methods

We evaluate the use of different attribution methods in the PATCHOUT framework by using PATCHOUT with Integrated Gradients (IG) [57], GradientShap (GS) [58], Guided GradCAM (GGC) [59], and Guided Backpropagation (GBP) [33]. All attribution implementations are taken from [60]. PATCHOUT with different attribution methods is evaluated on the LaVAN patch with 250 images per patch size. All evaluations are performed on attacked images since the accuracy for benign images should not be significantly affected by the choice of attribution method as it is more of a problem for the patch detector. Table 14 shows the results of this evaluation.

Integrated Gradients and GradientShap perform the worst, getting maximum accuracies of 72% and 65%, respectively. Guided Backpropagation and Guided GradCAM perform the best, with Guided Backpropagation achieving slight increases over Guided GradCAM.

Table 14 Comparison of different attribution methods used in PATCHOUT defending against the LaVAN patch

Patch Type	Local and Visible Adversarial Noise		
Patch Size	1% pixels	2% pixels	3% pixels
IG	71.6	69.2	68.8
GS	64.4	62.8	60.0
GGC	92.0	95.6	94.8
GBP	93.6	96.0	96.0

The one advantage Guided Backpropagation has over Guided GradCAM is that it requires minimal preprocessing to use. Guided GradCAM needs to be given a model layer to analyze, with the attributions generated by Guided GradCAM heavily relying on the layer that was chosen. For the evaluations in Table 14, we searched the model in use, Resnet50 [43], for the optimal layer and used it. Having Guided GradCAM in PATCHOUT rather than Guided Backpropagation as the attribution method used during attribution analysis would make the framework less out-of-the-box since additional analysis is needed to choose the appropriate layer, which is why we decided to use Guided Backpropagation.

Appendix F Evaluation on Irregularly Shaped Patches

In all of our evaluations, the patches we've used have been squares and circles. We believe our results will generalize to other common shapes, such as triangles and hexagons. However, it is possible for a patch to take on the shape of irregular polygons. We evaluate PATCHOUT on irregularly shaped patches that have between 3 and 15 sides. PATCHOUT is evaluated on 1000 images per number of sides. Each patch is randomly assigned a regularity factor that determines how irregular the patch will look. The results of this evaluation are shown

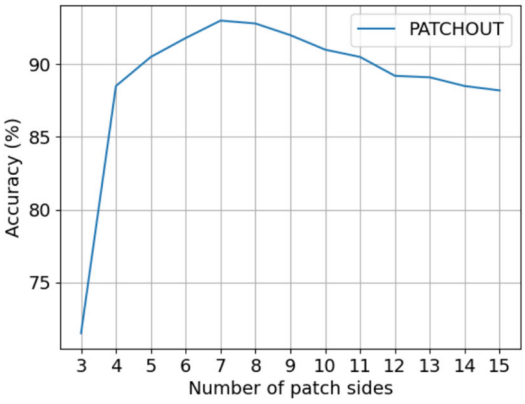

Fig. 10 The accuracy of PATCHOUT on Local and Visible Adversarial Noise as the number of sides increases and the regularity of the shape is varied

(a) (b)

Fig. 11 Examples of irregularly shaped patches with (a) 4 and (b) 10 sides

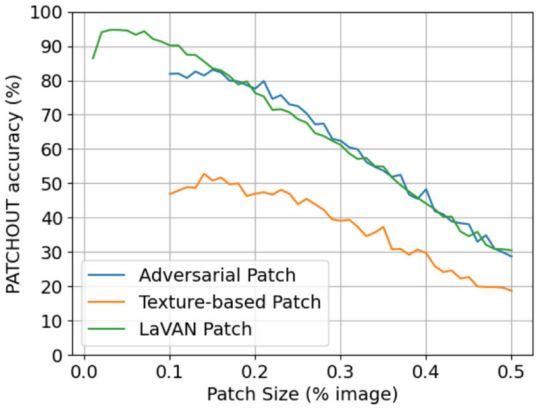

Fig. 12 PATCHOUT defense success rate evaluated across patches sized 1% to 50% of the image on 1% increments

in Figure 10. Examples of the patch shapes are shown in Figure 11. PATCHOUT suffers a significant performance drop on triangular patches, but otherwise performs well across the other shapes. On the irregular square patches, PATCHOUT achieves somewhere between an 88% and 89% accuracy, which is at most a 6% drop from the regularly shaped square patches. Additionally, irregularly shaped patches with between 5 and 8 sides perform on par with the square patches presented in the main evaluation. As the number of sides increase from 9 onward, there is a slight drop in accuracy, but overall, changing the shape or regularity of the patch does not seem to significantly degrade the performance of PATCHOUT.

Appendix G Evaluation Across Patch Sizes

Previously, we only reported the results of PATCHOUT and other defenses on specific patch sizes. We now evaluate the defense success rate of PATCHOUT on the LaVAN patch [5], Adversarial Patch [3], and Texture-based patch [6] across patch sizes from 1% of the image to 50% of the image with 1% increments. PATCHOUT was evaluated over 1000 successfully attacked images per increment. The results are shown in Figure 12. We omit the Generative Dynamic patch [7] due to the high runtime required to generate the patch. We start the Adversarial Patch and Texture-based patch sizes at 10% since the attack success rate below that is negligible for both patches. For all three patches, the defense success rate decreases as the patch size increases. This is expected since larger patches remove more image information than smaller patches. Even if a defense successfully detects and localizes the patch, the exact image values that occupied the patch space cannot be recovered. Larger amounts of unrecoverable image data will naturally lead to lower defense success rates, even when restoration techniques such as inpainting are applied. For reasonably sized patches, between 1% and 20% of the image, PATCHOUT has a high defense success rate.

Appendix H Larger Detectors on Unseen Patches

The PATCHOUT patch detector is a network consisting of 5 layers: 2 linear layers, followed by a max pooling layer, followed by 2 more linear layers. Each linear layer is followed by a ReLU activation function. The detector architecture was originally chosen for its speed and simplicity. However, it is possible that larger detectors could produce higher success

Table 15 The accuracy (%) of patch detectors with additional layers evaluated on all patches

Patch Type	Local and Visible Adversarial Noise						Generative Dynamic Patch Attack					
Patch Size	1% pixels		2% pixels		3% pixels		1% pixels		2% pixels		3% pixels	
Image Type	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv
ResNet50 LaVAN 8	99.0	96.7	99.0	99.5	99.1	99.5	98.7	80.8	98.7	80.7	98.7	97.4
ResNet50 LaVAN 11	98.1	96.7	97.9	99.3	97.8	99.7	97.5	87.0	97.5	92.5	97.5	99.6
ResNet50 AP 8	98.3	66.2	98.3	73.5	98.1	72.0	97.8	49.0	97.7	96.3	97.4	89.3
ResNet50 AP 11	99.3	54.5	99.2	63.2	99.2	63.1	99.1	37.1	99.1	90.9	99.1	88.7
MobileNet LaVAN 8	97.2	90.5	96.3	97.9	96.5	99.0	95.8	63.4	96.0	90.2	96.4	97.9
MobileNet LaVAN 11	99.1	87.3	99.0	97.5	99.2	98.9	99.1	47.6	98.9	78.4	99.2	94.8
MobileNet AP 8	98.2	51.1	98.3	71.8	98.1	72.5	98.2	23.4	98.2	64.4	98.6	94.2
MobileNet AP 11	99.2	43.0	99.1	65.4	98.9	68.5	99.2	15.7	99.1	47.9	99.2	90.2
Patch Type	Adversarial Patch						Texture-based Patch Attack					
Patch Size	10% pixels		15% pixels		20% pixels		10% pixels		15% pixels		20% pixels	
Image Type	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv
ResNet50 LaVAN 8	98.9	75.4	98.9	78.4	98.8	70.4	98.7	21.0	98.7	21.4	98.7	22.5
ResNet50 LaVAN 11	98.0	81.5	97.7	80.5	97.7	72.8	97.6	21.4	97.5	22.0	97.5	22.4
ResNet50 AP 8	98.5	98.5	97.8	97.9	97.8	97.7	97.8	17.0	97.8	22.1	97.7	22.9
ResNet50 AP 11	99.6	96.7	99.3	96.3	99.3	96.5	99.2	10.6	99.1	13.5	99.2	15.2
MobileNet LaVAN 8	97.2	85.7	96.8	78.2	96.6	72.6	96.6	20.1	96.5	16.7	96.3	13.6
MobileNet LaVAN 11	99.2	74.2	99.2	67.4	99.1	61.4	99.1	10.7	99.1	9.2	99.1	8.0
MobileNet AP 8	98.7	98.1	98.3	99.4	98.0	99.5	98.0	30.4	98.0	29.9	97.9	26.0
MobileNet AP 11	99.3	97.4	99.2	99.1	99.0	99.3	99.1	23.4	99.0	21.6	98.9	18.4

rates due to their ability to learn more complex functions, so we test two larger detector architectures. We also test the larger detectors on unseen patches to see if transferability improves.

One key characteristic of the detector is that it is small and lightweight, so to keep in line with the original drive behind the detectors, we create two larger detectors that contain only 3 and 6 additional layers. The additional 3 layers consist of a max pooling layer followed by two linear layers added to the original final layer of the detector. The additional 6-layer network follows the same scheme, adding those 3 layers to the end of the model twice.

These larger architectures are trained using the same approach as the original detectors and tested on the same images used in Section 5.1. The detectors are tested on all four patches, but VGG19 is omitted for brevity. The results of these experiments are provided in Table 15. Each row corresponds to a different detector, with ResNet50 LaVAN 8 referring to the 8-layer detector trained on LaVAN patches attacking ResNet50.

Comparing the results to those in Table 3 and Table 7, the detector accuracies do improve a few percent for the patches they were trained on over the original 5-layer detector. However, there is no general trend for the unseen patches. The large detector accuracies on unseen images tend to greatly fluctuate, likely as a result of where the training ended rather than the size of the detectors.

Appendix I Discussions on PATCHOUT Behaviour

We provide a more indepth analysis on two behaviours of PATCHOUT. The first is PATCHOUT's increase in recovered accuracy as patches increase in size from 1% to 3% of an image. The second is a discussion of the cases in which PATCHOUT's localization fails on benign images.

Defenses tend to have increased difficulty defending images as the size of patches increases. Contrary to this common belief, PATCHOUT performs better as the patch size increases in Table 5. Our reasoning is that smaller patches are more difficult to generate, meaning it is more difficult to get the classifier to focus on the patch class as the class of the image rather than the ground truth class. It's possible some images attacked by the smaller patches are not truly taking on the patch class, but are instead taking on the class of a different object within the image because the patch is able to at least deflect away from the ground truth class. If a segment containing the deflected object is generated, then that object will have a higher average attribution magnitude than the patch, leading PATCHOUT to remove the wrong segment. Another possibility is that very high magnitude attributions are more sparse as a result of the difficulty in creating small patches, which could lead to a lower average attribution magnitude than other segments generated from the image. This again would lead PATCHOUT to choose the wrong segment because it is looking for the segment with the highest average attribution magnitude. As the patch size increases, it becomes easier to attack an image, so these issues would occur less often, leading to a higher defense success rate.

PATCHOUT localization is typically harmless on benign images, as shown in Table 5 and Table 6. Cases where PATCHOUT localization disrupts a benign image occur when a portion or all of the focus of the image is removed. This can occur when the Segment Anything Model (SAM) only locates one segment which is removed by default. This can also occur when enough of the image focus is segmented, which should have the largest attributions since the benign images used are initially correctly classified during the experimental

dataset generation. Since this segment has the highest average attributions, it will be removed, disrupting the classification of the image.

Appendix J Defense Runtime Evaluation

the average runtimes of all defenses over 500 images for each patch and model are provided in Table 16. JPEG and LGS defend using image properties rather than network properties, allowing them to bypass the need for model calls. This allows them to keep their runtimes in the thousandths of seconds. SAC is also relatively lightweight, as the only model call it has is that of its patch segmenter. SAC takes an average of 0.05 seconds to process benign images and 0.02 seconds to defend attacked images. DRS accrues multiple model calls as it checks the model's top-1 class of multiple ablated images, taking 2 seconds for both benign and attacked images.

PATCHOUT requires multiple model calls from multiple models, which slows down the defense. We present runtimes for PATCHOUT when the patch location is blacked out and inpainted. The inpainting network used in PATCHOUT was chosen for its ability to reconstruct sections of images, but other inpainting networks and methods exist that are faster, and the option to blackout the patch rather than inpaint exists. Both approaches are included to give an upper and lower bound for PATCHOUT's speed. PATCHOUT without inpainting achieves an average speed of 0.04 and 2.0 seconds on benign and attacked images respectively. PATCHOUT with inpainting runs an average speed of 0.1 and 5.6 seconds on benign and attacked images respectively. That is an average 3.6 second difference between PATCHOUT with and without inpainting. It is also not surprising that PATCHOUT processes benign images much quicker because the patch detector exists to avoid running PATCHOUT localization on benign images unnecessarily. PatchCleanser is the slowest defense, taking as long as 19.8 seconds on average to process attacked images. This is unsurprising as PatchCleanser makes multiple model calls with its multi-step masking procedure that scales with number of masks used.

Appendix K PATCHOUT Parameter Ablation Studies

In this section, we perform recovered accuracy ablation studies on two parameters that can be set in PATCHOUT. The first is a standard deviation parameter for use with SmoothGrad [34] during the attribution analysis step of patch localization. The second is the segment confidence threshold used by SAM during the image segmentation step of patch localization.

SmoothGrad uses Gaussian smoothing with the standard deviation σ as a parameter. The standard deviation used by PATCHOUT in SmoothGrad is the following:

$$\sigma = (max(I) - min(I)) \times \alpha, \tag{K1}$$

where $max()$ and $min()$ denote the maximum and minimum pixel values of the image I, and α is the parameter modifying the size of the standard deviation used. PATCHOUT's recovered accuracy is evaluated over different α values starting at 0 and ending at 1 on 0.05 increments in Figure 13. PATCHOUT is tested on 200 images successfully attacked by a LaVAN patch at each increment. The recovered accuracy does not fluctuate much as α changes. Based on Figure 13, the ideal range for α is between 0.05 and 0.5. This range provides for the highest

Table 16 Runtime (s) of defenses on Local and Visible Adversarial Noise and Adversarial Patch

Patch Type	Local and Visible Adversarial Noise						Adversarial Patch					
Patch Size	1% image		2% image		3% image		10% image		15% image		20% image	
Patch Type	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv	benign	adv
Model	VGG19											
JPEG	$4e^{-3}$	$4e^{-3}$	$3e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$5e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$5e^{-3}$
LGS	$7e^{-4}$	$1e^{-3}$	$7e^{-4}$	$1e^{-3}$	$8e^{-4}$	$1e^{-3}$	$6e^{-4}$	$9e^{-4}$	$6e^{-4}$	$9e^{-4}$	$6e^{-4}$	$9e^{-4}$
PC	2.3	5.4	3.6	7.5	3.0	7.4	4.1	10.7	4.6	12.9	3.5	8.9
DRS	1.5	1.5	1.5	1.5	1.4	1.4	1.1	1.1	1.1	1.1	1.1	1.1
SAC	$4e^{-2}$	$2e^{-2}$	$3e^{-2}$	$1e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$1e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$3e^{-2}$
PO (blackout)	$4e^{-2}$	1.8	$3e^{-2}$	1.9	$4e^{-2}$	1.8	$3e^{-2}$	2.0	$3e^{-2}$	2.0	$3e^{-2}$	2.0
PO (inpaint)	0.1	4.9	0.1	5.0	0.1	5.0	0.1	6.8	0.1	6.0	0.1	5.8
Model	ResNet50											
JPEG	$3e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$5e^{-3}$
LGS	$6e^{-4}$	$9e^{-4}$	$6e^{-4}$	$9e^{-4}$	$7e^{-4}$	$1e^{-3}$	$6e^{-4}$	$9e^{-4}$	$6e^{-4}$	$9e^{-4}$	$6e^{-4}$	$8e^{-4}$
PC	4.8	9.2	5.2	13.1	4.3	12.1	6.4	14.2	6.7	15.7	4.6	11.0
DRS	2.1	2.1	2.2	2.2	2.1	2.1	1.2	1.2	1.2	1.2	1.2	1.2
SAC	$5e^{-2}$	$3e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$1e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$4e^{-2}$
PO (blackout)	$4e^{-2}$	1.8	$4e^{-2}$	2.0	0.1	2.0	$4e^{-2}$	2.1	$5e^{-2}$	2.1	$4e^{-2}$	2.1
PO (inpaint)	0.1	5.1	0.1	5.5	$4e^{-2}$	7.7	0.1	5.9	0.1	5.6	0.1	5.2
Model	MobileNet V3 Large											
JPEG	$4e^{-3}$	$4e^{-3}$	$3e^{-3}$	$3e^{-3}$	$4e^{-3}$	$5e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$	$4e^{-3}$
LGS	$6e^{-4}$	$1e^{-3}$	$6e^{-4}$	$8e^{-4}$	$7e^{-4}$	$1e^{-3}$	$6e^{-4}$	$6e^{-4}$	$7e^{-4}$	$9e^{-4}$	$8e^{-4}$	$1e^{-3}$
PC	3.5	8.7	3.7	11.2	3.4	10.6	5.2	14.3	8.2	19.8	5.5	12.4
DRS	2.7	2.7	4.6	4.4	5.4	5.4	1.4	1.4	1.9	1.9	2.5	2.5
SAC	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$1e^{-2}$	$5e^{-2}$	$2e^{-2}$	$5e^{-2}$	$4e^{-2}$
PO (blackout)	0.1	1.8	$4e^{-2}$	1.9	$4e^{-2}$	2.0	$1e^{-2}$	2.1	$2e^{-2}$	2.1	$2e^{-2}$	2.1
PO (inpaint)	0.2	5.8	0.1	5.1	0.1	5.3	$3e^{-2}$	5.2	$4e^{-2}$	5.2	$4e^{-2}$	5.1

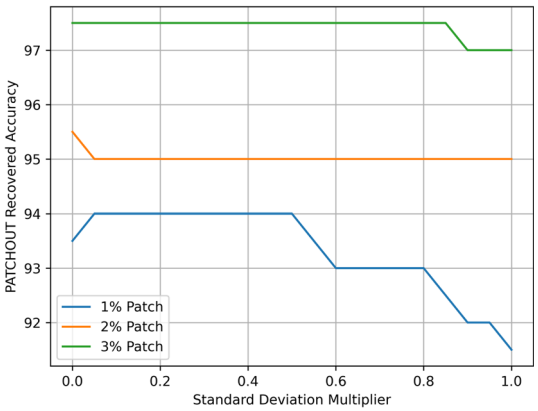

Fig. 13 PATCHOUT defense success rate evaluated across standard deviation values used with SmoothGrad, starting at 0 and ending at 1 on 0.05 increments

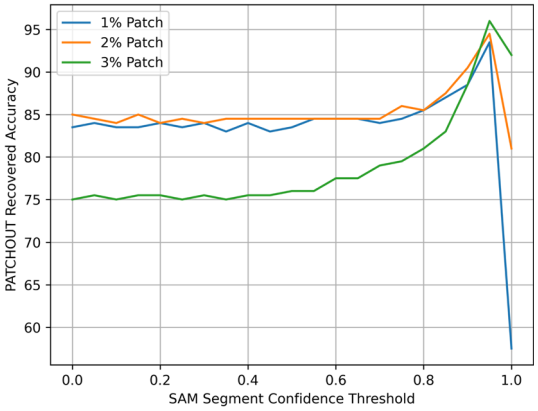

Fig. 14 PATCHOUT defense success rate evaluated across segment confidence thresholds used with SAM, starting at 0 and ending at 1 on 0.05 increments

recovered accuracies on the 1% and 3% patches while only causing a 0.5% drop from the highest recovered accuracy on the 2% patch. An α of 0.15 is used in PATCHOUT.

Experiments on the segment confidence threshold used by SAM were provided previously in Section B. An additional experiment on the segment confidence threshold is shown in Figure 14. PATCHOUT's recovered accuracy is evaluated over different segment threshold values starting at 0 and ending at 1 on 0.05 increments in Figure 14. PATCHOUT is tested on 200 images successfully attacked by a LaVAN patch at each increment. Recovered accuracies stay static until the 0.7 threshold for the 1% and 2% patches and the 0.4 threshold for the 3% patch. The recovered accuracy peaks at the 0.95 threshold for PATCHOUT on each of the patch sizes. Patches typically stand out against the images they are placed in, making it relatively easy for segmenters to confidently segment them. Using higher confidence thresholds eliminates many non-patch segments, leaving less room for error when determining the patch segment.

Appendix L PATCHOUT on Multi-Patched Images

All evaluations in Section 5 are performed on attacked images that contain a single patch. That is the main method of evaluation for defenses. However, real-world scenarios are not

Table 17 PATCHOUT accuracy (%) on attacked images containing 2, 3, and 4 patches

Patch Type	Local and Visible Adversarial Noise		
Patch Size	1% image	2% image	3% image
PATCHOUT on 2 Patches	67.6	48.8	35.6
PATCHOUT on 3 Patches	40.8	32.8	46.8
PATCHOUT on 4 Patches	46.8	29.6	22.0

limited to using only a single patch at a time. Malicious actors can use as many patches as necessary to overload the models they are targeting. We provide an evaluation of PATCHOUT on images containing 2, 3, and 4 patches in Table 17. There are 250 attacked images per patch size. The patches were generated and added to the image one at a time rather than all at once.

PATCHOUT's recovered accuracy generally decreases as the patch size increases and the number of patches increases. PATCHOUT's recovered accuracy is also much lower in Table 17 than in Table 5. This is to be expected since PATCHOUT is designed to only find one patch at a time. The patch used in this experiment was LaVAN, which is a per-image patch. Its likely that removing the initially placed patch would remove the adversarialness of of the subsequent patches, allowing PATCHOUT to effectively restore the image classification. Another test of PATCHOUT's ability on multiple patches at once would be to use multiple patches that are generated independent of any singular image so that removing one patch would not harm the adversarialness of another patch.

Appendix M Qualitative Results

The intermediate steps and results of the PATCHOUT framework against the LaVAN patch, generative patch, adversarial patch, and texture-based patch are respectively shown in Figures 15, 16, 17, & 18. Each figure shows the original image, attacked image, results of attribution analysis and image segmentation, the patch location found during the patch localization step, and the final inpainted image. It is interesting to note that the larger patches tend to have more high magnitude attributions spread throughout the image. Even then, the patches still tend to have the highest magnitude attributions of the whole image.

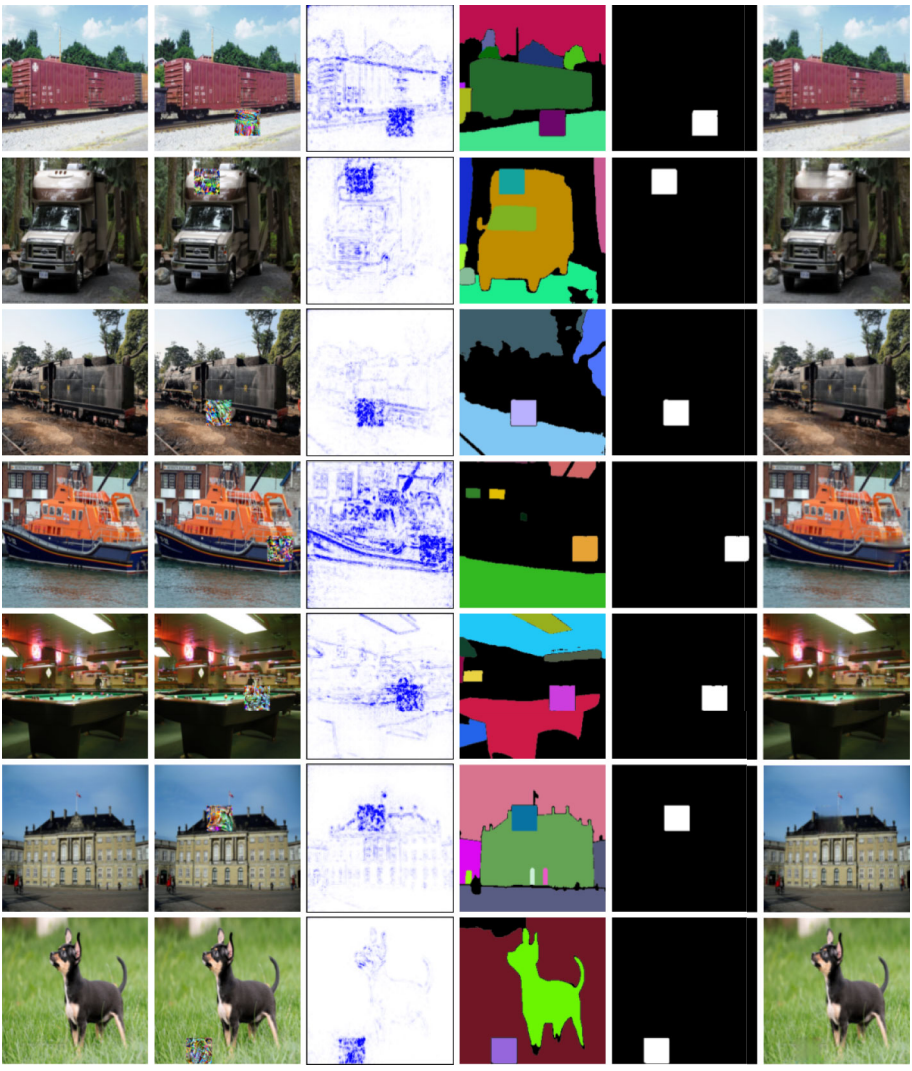

Fig. 15 The full process of defending against Local and Visible Adversarial Noise. (a) A benign image has a patch applied to it to become an (b) attacked image. PATCHOUT defends the image by using (c) attribution analysis and (d) image segmentation to find the (e) patch location. Finally, the patch location is inpainted, resulting in the (f) restored image

Fig. 16 The full process of defending against a Generative Dynamic Patch Attack. (a) A benign image has a patch applied to it to become an (b) attacked image. PATCHOUT defends the image by using (c) attribution analysis and (d) image segmentation to find the (e) patch location. Finally, the patch location is inpainted, resulting in the (f) restored image

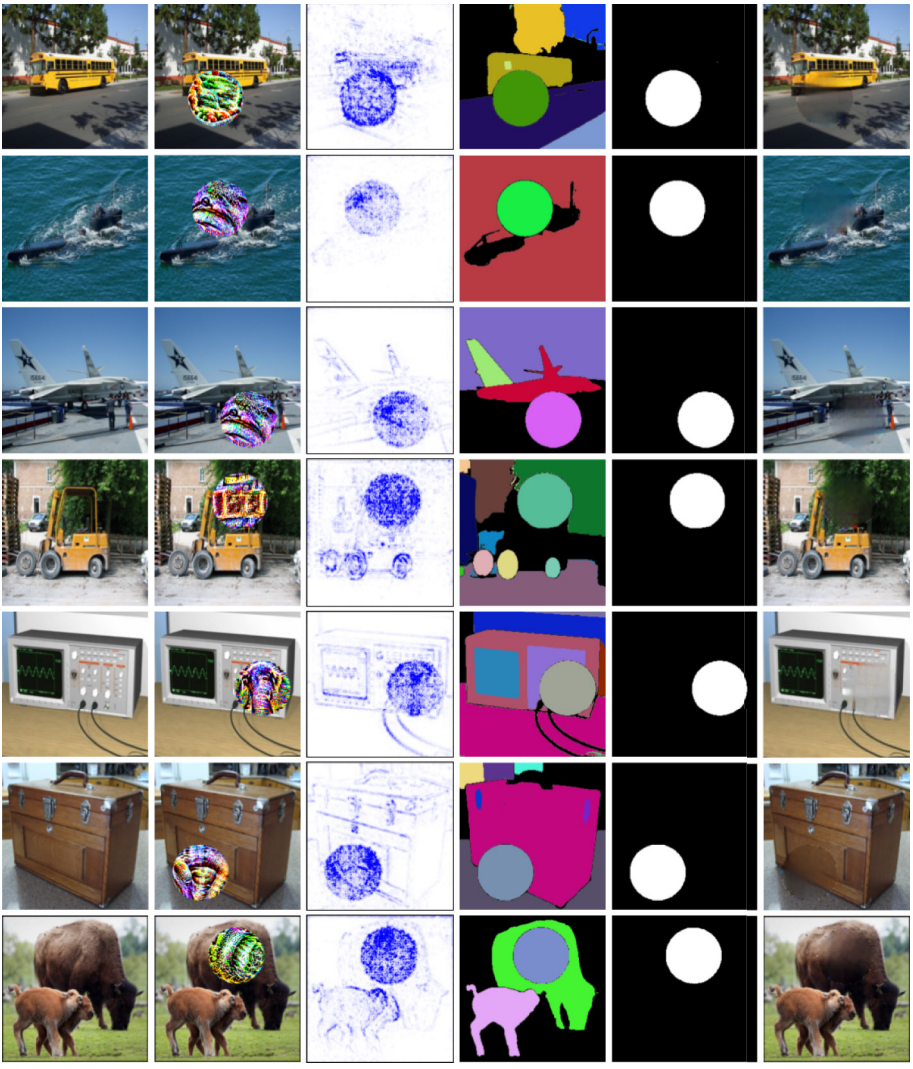

Fig. 17 The full process of defending against an Adversarial Patch. (a) A benign image has a patch applied to it to become an (b) attacked image. PATCHOUT defends the image by using (c) attribution analysis and (d) image segmentation to find the (e) patch location. Finally, the patch location is inpainted, resulting in the (f) restored image

Fig. 18 The full process of defending against a Texture-based Patch Attack. (a) A benign image has a patch applied to it to become an (b) attacked image. PATCHOUT defends the image by using (c) attribution analysis and (d) image segmentation to find the (e) patch location. Finally, the patch location is inpainted, resulting in the (f) restored image

Acknowledgements This work was supported in part by Defense Advanced Research Projects Agency (DARPA) grant #FA87502320501 and #HR00112420004.

Author Contributions Dominic Simon wrote all of the code, conducted the experiments, and wrote the paper. Sumit Jha and Rickard Ewetz advised on the direction of the research as well as wrote and edited the paper.

Funding This work was supported in part by Defense Advanced Research Projects Agency (DARPA) grant #FA87502320501 and #HR00112420004.

Data Availability Not applicable.

Declarations

Conflicts of Interest See funding.

Ethical Approval and Consent to Participate Not applicable.

Consent for Publication All authors consent.

Materials Availability Not available.

Code Availability Code will be made available upon publication.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing properties of neural networks. arXiv arXiv:1312.6199. https://doi.org/10.48550/ARXIV.1312.6199
2. Han H, Xu K, Hu X, Chen X, Liang L, Du Z, Guo Q, Wang Y, Chen Y (2021) Scalecert: Scalable certified defense against adversarial patches with sparse superficial layers. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Vaughan JW (eds) Advances in Neural Information Processing Systems, vol 34. Curran Associates, Inc United Kingdom, pp 28169–28181 (https://proceedings.neurips.cc/paper/2021/file/ed519c02f134f2cdd836cba387b6a3c8-Paper.pdf)
3. Brown TB, Mané D, Roy A, Abadi M, Gilmer J (2017) Adversarial Patch. arXiv. https://doi.org/10.48550/ARXIV.1712.09665 . arXiv:1712.09665
4. Hu Y-C-T, Kung B-H, Tan DS, Chen J-C, Hua K-L, Cheng W-H (2021) Naturalistic physical adversarial patch for object detectors. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)
5. Karmon D, Zoran D, Goldberg Y (2018) LaVAN: Localized and visible adversarial noise. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 2507–2515. PMLR. https://proceedings.mlr.press/v80/karmon18a.html
6. Yang C, Kortylewski A, Xie C, Cao Y, Yuille AL (2020) Patchattack: A black-box texture-based attack with reinforcement learning. CoRR arXiv:2004.05682
7. Li X, Ji S (2021) Generative dynamic patch attack. British Machine Vision Conference (BMVC)
8. Naseer M, Khan S, Porikli F (2019) Local gradients smoothing: Defense against localized adversarial attacks. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1300–1307. https://doi.org/10.1109/WACV.2019.00143
9. Hayes J (2018) On visible adversarial perturbations & digital watermarking. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1678–16787. https://doi.org/10.1109/CVPRW.2018.00210
10. Levine A, Feizi S (2020) (de)randomized smoothing for certifiable defense against patch attacks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 6465–6475. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/47ce0875420b2dbacfc5535f94e68433-Paper.pdf

11. McCoyd M, Park W, Chen S, Shah N, Roggenkemper R, Hwang M, Liu JX, Wagner D (2020) Minority reports defense: Defending against adversarial patches. In: Zhou J, Conti M, Ahmed CM, Au MH, Batina L, Li Z, Lin J, Losiouk E, Luo B, Majumdar S, Meng W, Ochoa M, Picek S, Portokalidis G, Wang C, Zhang K (eds) Applied Cryptography and Network Security Workshops. Springer, Cham, pp 564–582
12. Xiang C, Bhagoji AN, Sehwag V, Mittal P (2021) PatchGuard: A provably robust defense against adversarial patches via small receptive fields and masking. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 2237–2254. USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/xiang
13. Yu C, Chen J, Xue Y, Liu Y, Wan W, Bao J, Ma H (2021) Defending against universal adversarial patches by clipping feature norms. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 16414–16422. https://doi.org/10.1109/ICCV48922.2021.01612
14. Xiang C, Mahloujifar S, Mittal P (2022) PatchCleanser: Certifiably robust defense against adversarial patches for any image classifier. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 2065–2082. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity22/presentation/xiang
15. Liu J, Levine A, Lau CP, Chellappa R, Feizi S (2022) Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14953–14962. https://doi.org/10.1109/CVPR52688.2022.01455
16. Chou E, Tramèr F, Pellegrino G (2020) Sentinet: Detecting localized universal attacks against deep learning systems. In: 2020 IEEE Security and Privacy Workshops (SPW), pp. 48–54. https://doi.org/10.1109/SPW50608.2020.00025
17. Tarchoun B, Ben Khalifa A, Mahjoub MA, Abu-Ghazaleh N, Alouani I (2023) Jedi: Entropy-based localization and removal of adversarial patches. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4087–4095
18. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet Large Scale Visual Recognition Challenge. Inter. J. Comput. Vis. (IJCV) 115(3):211–252. https://doi.org/10.1007/s11263-015-0816-y
19. Liu Z, Han J (2022) Projected gradient descent algorithms for solving nonlinear inverse problems with generative priors. In: Raedt LD (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, pp. 3271–3277. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2022/454 . Main Track
20. Liu X, Yang H, Liu Z, Song L, Li H, Chen Y (2019) DPatch: An Adversarial Patch Attack on Object Detectors
21. Lee M, Kolter Z (2019) On Physical Adversarial Patches for Object Detection
22. Zhao Y, Yan H, Wei X (2020) Object Hider: Adversarial Patch Attack Against Object Detectors
23. Wu Z, Lim S-N, Davis L, Goldstein T (2020) Making an Invisibility Cloak: Real World Adversarial Attacks on Object Detectors
24. Fu Y, Zhang S, Wu S, Wan C, Lin Y (2022) Patch-fool: Are vision transformers always robust against adversarial perturbations? In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, OpenReview.net. https://openreview.net/forum?id=28ib9tf6zhr
25. Xiang C, Valtchanov A, Mahloujifar S, Mittal P (2023) Objectseeker: Certifiably robust object detection against patch hiding attacks via patch-agnostic masking. In: 2023 IEEE Symposium on Security and Privacy (SP), pp. 1329–1347. https://doi.org/10.1109/SP46215.2023.10179319
26. Nie W, Guo B, Huang Y, Xiao C, Vahdat A, Anandkumar A (2022) Diffusion models for adversarial purification. In: Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G, Sabato S (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 16805–16827. PMLR. https://proceedings.mlr.press/v162/nie22a.html
27. Wang J, Lyu Z, Lin D, Dai B, Fu H (2022) Guided Diffusion Model for Adversarial Purification
28. Xiao C, Chen Z, Jin K, Wang J, Nie W, Liu M, Anandkumar A, Li B, Song D (2022) DensePure: Understanding Diffusion Models towards Adversarial Robustness
29. Kang C, Dong Y, Wang Z, Ruan S, Chen Y, Su H, Wei X (2023) DIFFender: Diffusion-Based Adversarial Defense against Patch Attacks
30. Ho J, Jain A, Abbeel P (2020) Denoising diffusion probabilistic models. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H (eds) Advances in Neural Information Processing Systems, vol 33. Curran Associates, Inc., United Kingdom, pp 6840–6851 (https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf)
31. Kang M, Song D, Li B (2023) Diffattack: Evasion attacks against diffusion-based adversarial purification. Thirty-seventh Conf. Neural Inf. Process. Syst. 36:73919–73942 (https://openreview.net/forum?id=QB7ot7p6j7)

32. Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, Gustafson L, Xiao T, Whitehead S, Berg AC, Lo, W-Y, Dollar P, Girshick R (2023) Segment anything. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4015–4026
33. Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2014) Striving for Simplicity: The All Convolutional Net. https://doi.org/10.48550/ARXIV.1412.6806.arXiv arXiv:1412.6806
34. Smilkov D, Thorat N, Kim B, Viégas F, Wattenberg M (2017) SmoothGrad: removing noise by adding noise. arXiv:abs/1706.03825
35. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N (2021) An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations. https://openreview.net/forum?id=YicbFdNTTy
36. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J et al (2021) Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp. 8748–8763. PMLR
37. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser LU, Polosukhin I (2017) Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in Neural Information Processing Systems. Curran Associates, Inc, United Kingdom
38. Pan D, Li X, Zhu D (2021) Explaining deep neural network models with adversarial gradient integration. In: Zhou Z-H (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, pp. 2876–2883. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2021/396 . Main Track
39. Walker C, Jha S, Chen K, Ewetz R (2024) Integrated decision gradients: Compute your attributions where the model makes its decision. Proc. AAAI Conf. Artif. Intell. 38(6):5289–5297. https://doi.org/10.1609/aaai.v38i6.28336
40. Chen Z, Dash P, Pattabiraman K (2021) Turning your strength against you: Detecting and mitigating robust and universal adversarial patch attacks. arXiv preprint arXiv:2108.05075
41. Li C-T, Siu W-C, Liu Z-S, Wang L-W, Lun DP-K (2020) Deepgin: Deep generative inpainting network for extreme image inpainting. In: Bartoli A, Fusiello A (eds) Computer Vision - ECCV 2020 Workshops. Springer, Cham, pp 5–22
42. Simonyan K, Zisserman A (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556
43. He K, Zhang X, Ren S, Sun J (2015) Deep Residual Learning for Image Recognition. arXiv. https://doi.org/10.48550/ARXIV.1512.03385 . arXiv:1512.03385
44. Howard A, Sandler M, Chu G, Chen L-C, Chen B, Tan M, Wang W, Zhu Y, Pang R, Vasudevan V, Le QV, Adam H (2019) Searching for MobileNetV3. arXiv:1905.02244
45. Xiang C (2021) Derandomized Smoothing. GitHub. https://github.com/inspire-group/PatchGuard
46. Li X (2021) Generative Dynamic Patch Attack. GitHub. https://github.com/lxuniverse/gdpa
47. Nicolae M-I, Sinn M, Tran MN, Buesser B, Rawat A, Wistuba M, Zantedeschi V, Baracaldo N, Chen B, Ludwig H, Molloy I, Edwards B (2018) Adversarial robustness toolbox v1.2.0. CoRR arXiv:1807.01069
48. Yang C (2020) Texture-based Patch Attack. GitHub. https://github.com/Chenglin-Yang/PatchAttack
49. Das N, Shanbhogue M, Chen S, Hohman F, Chen L, Kounavis ME, Chau DH (2017) Keeping the bad guys out: Protecting and vaccinating deep learning with JPEG compression. CoRR arXiv:1705.02900
50. InspireGroup: PatchCleanser. GitHub (2022). https://github.com/inspire-group/PatchCleanser
51. Local Gradients Smoothing. GitHub (2020). https://github.com/metallurk/local-_gradients_smoothing
52. Liu J (2022) Segment and Complete. GitHub. https://github.com/joellliu/Segment-AndComplete
53. Yang C, Kortylewski A, Xie C, Cao Y, Yuille A (2020) Patchattack: A black-box texture-based attack with reinforcement learning. In: Vedaldi A, Bischof H, Brox T, Frahm J-M (eds) Computer Vision - ECCV 2020. Springer, Cham, pp 681–698
54. Rossolini G, Nesti F, Brau F, Biondi A, Buttazzo G (2023) Defending from physically-realizable adversarial attacks through internal over-activation analysis. Proc. AAAI Conf. Artif. Intell. 37:15064–15072
55. Rossolini G, Biondi A, Buttazzo G (2024) Attention-based real-time defenses for physical adversarial attacks in vision applications. In: 2024 ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS), pp. 23–32. IEEE
56. Xu K, Xiao Y, Zheng Z, Cai K, Nevatia R (2023) Patchzero: Defending against adversarial patch attacks by detecting and zeroing the patch. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 4632–4641
57. Sundararajan M, Taly A, Yan Q (2017) Axiomatic attribution for deep networks. In: Precup D, Teh YW (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine

Learning Research, vol. 70, pp. 3319–3328. PMLR. https://proceedings.mlr.press/v70/sundararajan17a.html
58. Lundberg SM, Lee S-I (2017) A unified approach to interpreting model predictions. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in Neural Information Processing Systems. Curran Associates, Inc, United Kingdom (https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf)
59. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D (2017) Grad-cam: Visual explanations from deep networks via gradient-based localization. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 618–626. https://doi.org/10.1109/ICCV.2017.74
60. Kokhlikyan N, Miglani V, Martin M, Wang E, Alsallakh B, Reynolds J, Melnikov A, Kliushkina N, Araya C, Yan S, Reblitz-Richardson O (2020) Captum: A unified and generic model interpretability library for PyTorch

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.