Assist Is Just as Important as the Goal: Image Resurfacing to Aid Model's Robust Prediction

Abhijith Sharma
University of British Columbia
BC, Canada
sharma86@student.ubc.ca

Phil Munz
TrojAI Inc.
NB, Canada
phil.munz@troj.ai

Apurva Narayan
Western University
ON, Canada
apurva.narayan@uwo.ca

Abstract

Adversarial patches threaten visual AI models in the real world. The number of patches in a patch attack is variable and determines the attack's potency in a specific environment. Most existing defenses assume a single patch in the scene, and the multiple patch scenarios are shown to overcome them. This paper presents a model-agnostic defense against patch attacks based on total variation for image resurfacing (TVR). The TVR is an image-cleansing method that processes images to remove probable adversarial regions. TVR can be utilized solely or augmented with a defended model, providing multi-level security for robust prediction. TVR nullifies the influence of patches in a single image scan with no prior assumption on the number of patches in the scene. We validate TVR on the ImageNet-Patch benchmark dataset and with real-world physical objects, demonstrating its ability to mitigate patch attack.

1. Introduction

The vulnerabilities of CNNs against adversarial corruption are widely known [1]. Adversaries in the physical world can have a devastating impact, endangering human lives [2], [8], [27]. Physical corruption can be natural (like snow, dust, or blur) or artificially designed by malicious attackers (adversarial attacks). Natural noises are less threatening, and robustness against them is extensively explored [26], [32]. Brown et al.'s proposal of the adversarial patch [3] fueled the research on physical-world attacks. An adversarial patch is an overt, optimally formulated, and localized perturbation, printed as a poster in the scene [30], [38]. Although initial works only considered a single patch [14], [19], [18], multiple patches provide more freedom to the attacker to design a stronger adversary [31].

CNNs have made remarkable progress in making human-like predictions [22], [37], yet, their failure against adversarial patches concerns their real-world deployment.

Hence, the focus has been on defending CNN models in recent years. A CNN model that is either trained to be robust [25], [21] or augmented with a defensive technique [20], [40], will be able to protect itself from adversaries. Most existing defenses require access to the CNN's parameters directly or indirectly [38]. However, accessing model weights is sometimes infeasible due to privacy concerns and expensive model training limits the defense's applicability.

A typical patch attack involves the manipulation of the scene rather than the model itself. Hence, is there a way to eradicate corruption at the root level by cleansing unwanted and suspicious patterns from the image? As the title suggests:"assist is just as important as the goal" we propose image resurfacing to assist the goal of the model's robust prediction. This work acts as a first line of defense against adversarially patched images. It can be augmented with any CNN model effortlessly. The motivation is not to substitute the robust models but to implement multi-level security. Even in cases where designing robust models are complicated, this technique can even independently mitigate the influence of adversarial patches from the scene.

2. Related Work

Designing a defense for patch attacks is a location identification problem. The detected patch's location can be masked or inpainted to mitigate the adversarial influence. Initial defenses primarily utilize a saliency map to locate adversarial patches and mask them [11], [7], [23]. The method is simple, but the performance depends on the quality of the saliency map. Generating a salience map also requires access to the model's weights, which might be restricted sometimes. Moreover, detecting multi-patch attacks is a challenge due to multiple salient regions. The next set of defenses proposes a particular CNN architecture based on the small receptive field [39], [41]. A smaller field leads to the reduced global influence of a localized patch. The major drawback is the dependence on special robust architectures with relatively low clean accuracy.

Figure 1. Framework of the total-variation based outlier detection followed by GAN inpainting to carry out image resurfacing.

There have been proposals of defenses based on adversarial training, but they are unscalable due to the nuance of the patch attack [25], [9]. The methodology of adversarial training against digital attacks like PGD and FGSM cannot directly translate to patch attacks. On the other hand, certified defense is ideal as it provides a confidence guarantee along with their defending properties [6], [16], [40]. However, such defenses come with an additional cost of computational complexity and are slower than their counterparts. Also, no existing defense could formulate guarantees when multiple patches are in the scene. Certain architectures show inherent robustness against adversarial patches [5], [28], [17], yet their performance evaluation against multi-patch attacks is still in progress. Overall, there has been considerable work on single patch-based defenses, and defending against multi-patch attacks has been challenging.

3. Our Contribution

In this work, we utilize a popular image processing technique called total variation (TV) [4], [29] to perform image resurfacing, and call it total-variation based image resurfacing (TVR). The steps involved in the TVR is shown in Figure 1. The resurfaced image is close to its original form such that even a normally trained model can perform robust prediction to some extent. The TV score has been previously utilized in adversarial defenses to understand pixel's spatial complexity in an image [33], [10]. Figure 2 demonstrates how an adversarial image is processed and reconstructed to mitigate the adversarial patch under three instances.

The TVR is specifically designed to work against adversarial patches as it exploits the properties of patch attacks like localised and contiguous perturbation of pixels. But more importantly, the perturbation level of patch attacks are higher than that of digital adversarial attacks, which primar-

ily inspires the development of TVR. The TVR only works on the image with no need of model or task information, which extends it's applicability to numerous applications. The characteristics of TVR are summarised below:

- **Model Agnostic:** The performance of the TVR is independent of the CNN model architecture.
- **Patch Number Agnostic:** Can overcome multi-patch attack with no assumption on the number of patches.
- **Patch Location Agnostic:** The patches will be detected irrespective of their location in the scene.
- **Location detection:** TVR is capable of determining approximate location of patches in the scene.

4. TVR Formulation

This section discusses the formulation of total-variation based image resurfacing (TVR) method. The framework in Figure 1 illustrates image cleansing using TVR involving five broad steps as explained below:

Channel-wise Blocks: TVR calculates a channel-wise variation in pixel values, as an attack might not equally affect each channel. Averaging TV scores for three channels might lead to a loss of information. Hence, we perform a channel-wise inspection and later make image-level conclusions. The first step in TVR involves dividing the image into multiple equal-sized blocks as shown in Figure 1.

Converting an image into a block set is explained in Algorithm 1. The block set is a collection of equally sized blocks formed by dividing the image, as shown in Figure 1. We denote the block set as $\mathcal{B} \in \mathbb{R}^{3 \times \mathrm{nk} \times k \times k}$, where nk is the total number of blocks in the block-set, with each block being $k \times k$. The $\mathcal{B}$ stores channel-wise information. The input to the Algorithm is the image x and block-size k, and the output is the block-set $\mathcal{B}$.

Block-wise Total variation: This step calculates the block-wise total variation (TV) score. The TV is a metric to determine the complexity of an image to its spatial variation in pixel values. TV score has been extensively used in various applications in image processing literature. The TV of an RGB image $\boldsymbol{x} \in \mathbb{R}^{3 \times n \times n}$ is defined as follows:

$$\mathcal{TV}(\boldsymbol{x}) = \sum_{i,j \in \mathcal{N}} ||x_i - x_j||_p^q \qquad (1)$$

where $\mathcal{N}$ is the pixel neighborhood. Typically, $\mathcal{N}$ consists of the horizontally and vertically adjacent pixels, and $||.||_p^q$ is the l_p norm to the power of q. This work utilizes $\mathcal{TV}$ with l_2 norm and $q = 1$ (refer Line 3, Algorithm 2).

Figure 1 shows 3-D surface plots of the TV score over the image landscape (block set) for each channel. The neighborhood $\mathcal{N}$ corresponds to each block b in the block set $\mathcal{B}$. Hence, the number of data points forming the plot's surface is the total number of blocks nk that the image is divided into. The red mountain peaks translate to the highly probable region of adversarial perturbations from high TV score. Hence, the TV score based surface plots provides the approximate location of malicious patches in the scene.

Channel-wise Outlier Detection: This step calculates the outlier O using the simple formula $Q_3 + 1.5 \times (Q_3 - Q_1)$. The $Q1$ and $Q3$ is the first (25th percentile) and third quartile (75th percentile) of the TV distribution, respectively. Although outliers can be present on both ends of the spectrum, but the outliers on the higher end corresponds to the high total variation score which could have caused by patch attacks. Since the TV score will vary for each channel, so will the outliers. Hence, we calculate channel-wise outliers in this block (refer Lines (4-11), Algorithm 2).

Pixel Masking: The outliers from the previous step are

Algorithm 1: IMAGE_TO_BLOCK

Input: Image $\boldsymbol{x} \in \mathbb{R}^{3 \times n \times n}$, Block-size $k \times k$.
Output: Block-set $\mathcal{B} \in \mathbb{R}^{3 \times nk \times k \times k}$

1   $\text{nk} = (n \times n)/(k \times k)$ // number of blocks in image
2   $\mathcal{B} \in \mathbb{R}^{3 \times nk \times k \times k}$ // block-set dimensions
3   $\text{nrow} = (n/k)$ // number of blocks in each row
4   **for** $c \leftarrow 0, 1, 2$ **do**
5   **for** $b \leftarrow 0$ *to* nk **do**
6   **for** $i \leftarrow 0$ *to* k **do**
7   **for** $j \leftarrow 0$ *to* k **do**
8           $\mathcal{B}[c][b][i][j] = \boldsymbol{x}[c][\text{int}(b/\text{nrow}) \times k + i][(b\%\text{nrow}) \times k + j]$
9   **end**
10   **end**
11   **end**
12 **end**

Algorithm 2: TV-based Image Resurfacing (TVR)

Input: Adversarial image $\boldsymbol{x}'$, Cropped image $\boldsymbol{x_c}$,
       Generated image $\boldsymbol{x_g}$ ($\boldsymbol{x}', \boldsymbol{x_c}, \boldsymbol{x_g} \in \mathbb{R}^{3 \times n \times n}$),
       Block-size $k \times k$, Mask set $\mathcal{M} \in \mathbb{R}^{3 \times nk \times k \times k}$, ,
       Generator $\mathcal{G}(.)$, Percentile function $\mathcal{P}^{th}(.)$
Output: Reconstructed image $\boldsymbol{x_r} \in \mathbb{R}^{3 \times n \times n}$

1   $\text{nk} = (n \times n)/(k \times k)$ // total number of blocks
2   $\mathcal{B} = \text{IMAGE\_TO\_BLOCK}(\boldsymbol{x}', k)$ // create block-set
3   $\mathcal{TV}(x) = \sum_{i,j \in \mathcal{N}} ||x_i - x_j||_2$ // TV loss function
    for $c \leftarrow 0, 1, 2$ **do**
4   **for** $b \leftarrow 0$ *to* nk **do**
5       $tv[c][b] = \mathcal{TV}(\mathcal{B}[c][b])$ // TV loss
6   **end**
7     $Q_1 = \mathcal{P}^{th}(tv[c], 0.25)$ // first quartile
8     $Q_3 = \mathcal{P}^{th}(tv[c], 0.75)$ // third quartile
9     $O[c] = Q_3 + 1.5 \times (Q_3 - Q_1)$ // outliers
10 **end**
11 **for** $c \leftarrow 0, 1, 2$ **do**
12   **for** $b \leftarrow 0$ *to* nk **do**
13   **if** $O[c] > \mathcal{TV}(\mathcal{B}[c][b])$ **then**
14         $\mathcal{B}[c][b] \leftarrow 0$
15         $\mathcal{M}[c][b] \leftarrow 1$
16   **else**
17         $\mathcal{M}[c][b] \leftarrow 0$
18   **end**
19   **end**
20 **end**
21 $\boldsymbol{x_c} = \text{IMAGE\_TO\_BLOCK}^{-1}(\mathcal{B})$ // create image
22 $\mathbf{m} = \text{IMAGE\_TO\_BLOCK}^{-1}(\mathcal{M})$ // create mask
23 $\boldsymbol{x_g} = \mathcal{G}(\boldsymbol{x_c})$ // generated image using GAN
24 $\boldsymbol{x_r} = (1 - \mathbf{m}) \odot \boldsymbol{x_c} + \mathbf{m} \odot \boldsymbol{x_g}$ // final image

---

used to mask the pixels of the blocks b where the values are larger than the outlier, indicating the probability of patch presence. For pixels whose values are larger than the outlier, then the entire block's pixel values are assigned zero to form a cropped image x_c. For the final reconstructed image, a mask m, which is created out of mask set $\mathcal{M} \in \mathbb{R}^{3 \times nk \times k \times k}$ similar to that of image's block set. However, unlike the image, the mask is a binary tensor, where $m \in \{0, 1\}^{3 \times n \times n}$.

The pixel values of all blocks in the mask set are zero by default. The values are set to one for only those blocks in the mask set which correspond to the suspicious blocks (tagged by outlier detection) in the image's block set (refer Lines (13-24), Algorithm 2). The algorithm of converting block set into an image and mask set into a mask is inverse of Algorithm 1 (IMAGE_TO_BLOCK), which we call BLOCK_TO_IMAGE = IMAGE_TO_BLOCK^{-1}. Specifically, the operands on Line 8, Algorithm 1 would interchange. The input to IMAGE_TO_BLOCK^{-1} is the block set $\mathcal{B}$ and output is the RGB image $\boldsymbol{x}$.

Image Reconstruction: This step concerns reconstructing or inpainting the cropped image x_c using a generative adversarial network (GAN). From the previous step, we obtain a mask m and a cropped image x_c. The cropped image has masked regions as pixel values are assigned zero for the suspicious blocks. Although the previous step is sufficient to mitigate the influence of the patch, we lose some information due to masking, which degrades the prediction accuracy. The image inpainting helps to reduce the gap between natural and adversarial accuracy.

A crucial component of this step is the quality of images produced by generator. The generator $\mathcal{G}$ model is an autoencoder which tries to produce fake images to fool the discriminator $\mathcal{D}$. The discriminator is basically a CNN model followed by a Sigmoid function that finally gives a single scalar as output, whether the input image is real or fake. The loss function of the generator $\mathcal{G}$ consists of two parts:

Reconstruction Loss: The loss of recreating an image using generator $\mathcal{G}$. It assists in capturing the structure and coherence of the missing region based on its context. The reconstruction loss is a l_2 loss, which reduces the mean pixel wise error leading to a blurry image. Hence, it is not sufficient to use reconstruction loss. The loss is defined as:

$$\mathcal{L}_{reconstruct} = \mathbb{E}_{\hat{x} \sim P_c} \left[\frac{1}{N} \sum_{i=1}^{N} ||\mathcal{G}(\hat{x}^{(i)}) - \hat{x}^{(i)}||^2 \right] \quad (2)$$

where $\hat{x}$ is the image to be reconstructed (x_c in the Algorithm 2) and $\mathbb{P}_c$ is the distribution of all x_c.

Adversarial Loss: With the reconstruction loss we have blurry image and adversarial loss adds some realism to offset the blur. The goal of a generator in a GAN is to produce realistic images to fool the discriminator. The adversarial loss measures the extent to which generator is able to mislead the discriminator. This loss is defined as:

$$\mathcal{L}_{adver} = \max_{\mathcal{D}} \quad \mathbb{E}_{x \sim P_r} \left[\log(\mathcal{D}(x)) \right] + \\ \mathbb{E}_{\hat{x} \sim P_c} \left[\log(1 - \mathcal{D}(\mathcal{G}(\hat{x}))) \right] \quad (3)$$

where x is the real/original image and $\mathbb{P}_c$ is the distribution of real images. The adversarial loss is augmented with reconstruction loss for image inpainting. The augmentation is done using a hyper-parameter λ. Typically we use $\lambda_{adver} + \lambda_{reconstruct} = 1$. The adversarial network produces a realistic image. The total loss function is given:

$$\mathcal{L}_{total} = \lambda_{adver}\mathcal{L}_{adver} + \lambda_{reconstruct}\mathcal{L}_{reconstruct} \quad (4)$$

To summarise, the aim is to minimize the $\mathcal{L}_{total}$. Hence, the min-max optimization as a apart of $\mathcal{L}_{adver}$ ensures that the generated images look similar to real ones, and minimization of $\mathcal{L}_{reconstruct}$ assists generator in this process. The new image x_{new} is reconstructed with the x_c and x_g using

Figure 2. Illustration of TVR Demonstration with three examples.

the formula $(1 - \mathbf{m}) \odot \boldsymbol{x_c} + \mathbf{m} \odot \boldsymbol{x_g}$ as stated in Line 26, Algorithm 2. The $\odot$ is a Hadamard operator representing element-wise multiplication between the tensor operands.

5. Experiments and Results

This section examines TVR's defending ability over a set of experiments. We use two datasets, one is ImageNet-Patch benchmark dataset [24], which is a dataset made with 10 adversarially trained patches (Soap Dispenser (SD), Cornet (CN), Plate (P), Banana (B), Cup (C), Typewriter (TK), Electric Guitar (EG), Hair Spray (HS), Socks (S), Cellphone (CT)) over the ImageNet [15] dataset. The Image-Net Patch dataset only consists of single patches, and to evaluate our method against the most potent design of multi-patch attack [31] we use Imagenette dataset. The Imagenette is a 10 class subset of ImageNet with 10,000 images shared almost equally between 10 classes. The adversarial patch used with this dataset is self trained for the target class of 'Golf Ball'.

5.1. Experimental Setup

The patches in the ImageNet-Patch dataset are trained with ensemble training on AlexNet [15], ResNet18 [12], and SqueezeNet [13]. On the other hand, VGG16 [34], GoogleNet [35] and Inception v3 [36] is used for black-box testing. All models are off-the-shelf from PyTorch's

Torchvision library. For Imagenette, we used VGG16 and ResNet18 that are pre-trained from Torchvision and fine-tuned on Imagenette. The patches are trained in a white-box testing. All images are resized to 224×224 for consistency in the results. The patch size in the ImageNet-Patch dataset is approximately 5-6% of the image size. Imagenette's patch size varies between 4%, 8%, and 12% of the image size. The self-trained single and multiple patches over Imagenette involved patch training over 8,000 images from the train set. The patches trained over Imagenette are less robust than those in the ImageNet-Patch benchmark. However, they are sufficient to validate the multi-patch attack results. Training patches for the Imagenette dataset is easier, as it is possible to attack with less potent adversarial patches due to the simplicity of the dataset.

The inpainting in TVR is done using a pre-trained generator. We use a rudimentary custom defined GAN architecture that is easy to train and just enough to demonstrate our framework. However, using a better and deeper generator will improve the inpainting process. The input and output of the generator have exact dimensions $3 \times 224 \times 224$. With the mean-squared loss, we used PyTorch's `Adam` optimizer with a learning rate of 0.002 over 300 epochs for generator training over a subset of the ImageNet train-set. For the TVR, the block size is chosen as 28×28, which we found is the most appropriate size for this dataset, as discussed in Section 5.4. Quantitative experiments were carried out on the single Nvidia RTX A6000, and the resources available with Google Colab were computationally sufficient for visualizations and plots.

5.2. Analysis of TVR Against Adversarial Patches

We primarily use three evaluation metric: clean accuracy, adversarial accuracy and targeted success rate. We report all metrics for top-k $\mathcal{T}_k$ scenario, which is determined by checking if the original class y is within the k highest predicted class (based on CNN's probabilistic output). In our experiments, since ImageNet-Patch dataset has 1000 classes, it is reasonable to calculate Top-1 and Top-3. However, for Imagenette, we limit ourselves to only Top-1 as it is just a 10 class subset.

Performance on ImageNet-Patch Benchmark: The performance of TVR is averaged over 10 adversarial patches and is shown in Table 1. The 'Naive' model refers to the one without any defensive properties. The 'Target' is the targeted success rate of the attack. We also report the average for the ensemble models separately to understand how TVR performs in a white-box setting separately. Since the patch is trained on models involved in the white-box ensemble set, the accuracy will be lower as the patch will be more effective. We observe a 42% increase in top-1, a 34% increase in top-3, and a 25% increase in top-10 overall adversarial accuracy with TVR.

Table 1. Performance of the TVR on benchmark ImageNet-Patch Dataset. All values are reported in %.

	Model	Naive			TVR		
		Clean	Adversarial	Target	Clean	Adversarial	Target
TOP - 1	AlexNet	64.2	24.9	6.9	56.3	53.6	0
	ResNet18	78.1	47.8	13.1	76.7	68	0
	SqueezeNet	56.2	33.6	18.4	50.8	46.2	0
	VGG16	74.7	53.8	7.6	74.2	71.3	0
	GoogleNet	74.3	54.4	2.6	72.2	64.4	0
	Inception v3	72.1	40	6.2	72.1	56.9	0
	Ensemble	**66.1**	**35.4**	**12.8**	**60.7**	**55.9**	**0**
	Overall	**69.9**	**42.4**	**9.0**	**66.7**	**60.1**	**0**
TOP - 3	AlexNet	86.5	39.8	14.4	86.0	75.6	0.2
	ResNet18	96.5	69.1	25.3	96.3	91.1	0
	SqueezeNet	86.1	50.9	28.4	86.2	72	0.7
	VGG16	96.6	71.3	12.9	94.5	89.8	0.2
	GoogleNet	94.9	74.4	6.9	94.8	84.9	0
	Inception v3	92.0	62.7	13.6	88.1	81.8	0
	Ensemble	**89.7**	**53.3**	**22.7**	**89.3**	**79.6**	**0.3**
	Overall	**92.1**	**61.4**	**16.9**	**90.7**	**82.5**	**0.2**

The higher adversarial accuracy translates to superior robustness. With the TVR, we observe a marginal drop in clean accuracy for all the accuracy levels. It may be because of the false outlier predictions even in clean images due to the high total variation score in some regions in the scene. Moreover, the success rate of a targeted attack for the top-1 scenario with TVR is zero. Even for top-3, the success rate is extremely low compared to the Naive model. It demonstrates that TVR nullifies the influence of adversarial patches. Among the white-box models, the ResNet18 shows the highest level of robustness, with VGG16 being the most robust for the black-box setting.

Performance against Multi-Patch attack on Imagenette: For analyzing TVR on Imagenette, we use ResNet18 and VGG16, as shown in Table 2. The patches for ResNet18 and VGG16 are trained individually using LaVAN [14] methodology and attacked in a white box setting. We use three scenarios with Imagenette: original image, image with single patch, and image with multi-patch. Moreover, we assume that a single patch of size δ is evaluated against the multi-patch attack (of n patches) with the size of each patch being (δ/n) for fair comparison. In the scope of this experiment, we use four patches for the multi-patch attack.

Imagenette is a simple dataset, hence, ResNet18 and VGG16 could achieve 99% clean accuracy. For single patch attack, we see a drastic drop in adversarial accuracy of the Naive model from 4% to 8%, yet the TVR defended model is able to retain the accuracy. Even for a 12% patch, the TVR can mitigate the patch's effect to a large extent. For the multi-patch, we notice that 4% perturbation level is capable of reducing the accuracy to 7%. For 8% and 12% patch size, the accuracy comes down to 0%. Among ResNet18 and VGG16, we notice that VGG16 performs better with TVR, with accuracy above 95% for all the perturbation levels and both attack types (except one - 12% Multi-Patch attack).

5

Table 2. TVR's performance against Multi and Single patch attack on Imagenette for perturbations of 12%, 8% and 4%, respectively.

Model		Original		Single Patch		Multi Patch	
		Naive	TVR	Naive	TVR	Naive	TVR
4%	ResNet18	99.7	98.5	88.7	95.8	1.1	96.3
	VGG16	99.1	98.1	10.5	98.1	13.2	99.1
	Overall	**99.4**	**98.3**	**49.7**	**97.0**	**7.1**	**97.1**
8%	ResNet18	99.7	96.2	20.3	94.1	0	73.5
	VGG16	99.1	98.4	0	97.7	0	95.8
	Overall	**99.4**	**97.3**	**10.1**	**95.6**	**0**	**84.6**
12%	ResNet18	99.7	97.3	9.6	87.4	0	69.3
	VGG16	99.1	99.2	0	82.4	0	81.2
	Overall	**99.4**	**98.2**	**4.8**	**84.9**	**0**	**75.2**

Table 3. Comparison of existing defenses against TVR for single and multi-patch attack. The patch is of 'banana' class. The values are the number of correct classification out of 100 test samples.

	Defense Method	Clean	Single	Multi-2	Multi-3	Multi-4
AlexNet	Naive	**65**	31	4	2	0
	LGS [23]	61	**56**	41	18	17
	PatchCleanser [40]	56	53	30	8	2
	TVR (ours)	56	56	**40**	**27**	**20**
ResNet18	Naive	**78**	57	5	0	0
	LGS [23]	68	65	56	22	4
	PatchCleanser [40]	74	46	16	4	1
	TVR (ours)	76	**72**	**52**	**42**	**23**
SqueezeNet	Naive	**57**	18	0	0	0
	LGS [23]	52	**48**	30	9	4
	PatchCleanser [40]	54	20	4	3	0
	TVR (ours)	50	49	**42**	**22**	**12**
VGG16	Naive	**75**	39	4	4	0
	LGS [23]	73	64	42	24	11
	PatchCleanser [40]	75	44	17	6	2
	TVR (ours)	74	**66**	**46**	**43**	**24**
GoogleNet	Naive	**74**	54	29	20	10
	LGS [23]	73	54	46	37	20
	PatchCleanser [40]	72	53	30	8	2
	TVR (ours)	73	**62**	**58**	**42**	**25**
Inception	Naive	**76**	50	38	25	16
	LGS [23]	74	62	47	42	29
	PatchCleanser [40]	70	55	31	10	8
	TVR (ours)	72	**65**	**60**	**48**	**33**

Comparison with state-of-art defenses against Multi-Patch attack on ImageNet-Patch Benchmark: In this study, we compare TVR against two model-agnostic defenses: Localized Gradient Smoothing (LGS) [23] and PatchCleanser (PC) [40] along with a Naive (undefended) model for reference. The results are shown in Table 3. As the defenses are model-agnostic, we evaluate them for the six CNNs similar to Table 1. We consider single and multiple patches (with one, two, three, and four patches) placed randomly over the scene. The size of each patch is around 5% of the image area. The analysis is carried out over 100 randomly sampled images from the test set.

As evident from Table 3, even though PC is a certified defense, it does fail for multi-patch attacks. The PC is inherently designed for single patches and can be scaled to two patches by increasing the search complexity. However, TVR detects single and multiple patches in one image scan. The performance of LGS was similar to TVR for a single patch attack. However, TVR has a superior performance when it comes to multiple adversarial patches.

For a two-patch attack (Multi-2), we observe that TVR outperforms LGS by 6% and PC by 28% on average across six models. Similarly, for a three-patch attack (Multi-3), the TVR outperforms LGS and PC by 12% and 31% , respectively. For four patches, we observe that all defended models have low accuracy because the scene occlusion is high and scene reconstruction is complex. But using advanced GAN for image inpainting can improve the performance of TVR substantially. However, TVR still outperforms LGS and PC in four patch attack scenarios, which proves TVR has the best patch mitigation technique overall.

To summarize, Table 1, 2 and 3 showcases the ability of TVR to achieve robustness against single and multi-patch attacks on ImageNet dataset. We also did not notice a substantial difference between the white and black box settings. The primary reason is that the TVR works on the total-variation of pixel values, which has no connection with whether the pixels are trained in a white or black box fashion. Moreover, the TVR also shows superior performance against LGS and PC for multiple adversarial patches.

5.3. Visualization of TVR's Working

This section qualitatively analyzes how TVR works by visualizing 3-D surface plots as shown in Figure 3. Although, the TV scores and outliers are calculated channel-wise, for visualization purpose, the 3-D surface plots are made by averaging over three channels. The analysis involves testing the patch attack under three scenarios: a single patch attack, a multi-patch attack using two patches, and a multi-patch attack using four patches. For the first two scenarios, we use a patch with the target class of 'Cornet' from the ImageNet-Patch dataset. For the third scenario, we use a multi-patch trained with a 'Golf Ball' target class. The original image belongs to the class 'Parachute'.

In the experiment, we visualize the TV score over the image landscape as a surface plot. Hence the x and y axis of the plot is the image's length and width, which is 224 × 224. We display the TV score of each block against the z-axis. The surface plots consist of a continuous color scale from blue to red, with blue as the lowest score and red as the highest. Figure 3 shows the plots for four instances: original image, adversarial image, adversarial image with the outlier surface, and finally the reconstructed image.

In the surface plots, we can see red peaks in the region corresponding to patched areas in the original image. The light gray translucent surface represents the outlier of the TV score for each example. The regions in the surface that are higher than the outlier surface are masked or cropped

Figure 3. Surface plot of the TV score over the image landscape. The overall size of the patch is between 6-8% of the image's size.

Figure 4. Variation in accuracy with changing TVR's block size. Figure 4a and 4b show clean and adversarial top-1 accuracy. Figure 4c and 4d show natural and adversarial top-3 accuracy.

to zero, which is later reconstructed using a generator. We demonstrate that the TVR defends against multi-patch attacks in the same way it deals with single-patch attacks. The surface plot of a single patch-reconstructed image has the highest resemblances with the original one. Inpainting a cropped image formed post the mitigation of a multi-patch attack is relatively complex. But, we used a simple inpainting generator, and using a better generator will increase the similarity between the original image and the image recovered after a multi-patch attack.

5.4. Influence of TVR's Block Size on Accuracy

The block size is the most critical hyper-parameter affecting the performance of TVR. The outlier in TVR can vary drastically depending on the block size, affecting its defending capability. An appropriate block size depends on the size of the adversarial patch in the scene. The design of most defenses against adversarial attacks requires a conservative estimation of the attack's potency, and TVR is no exception. The attack's potency in the case of digital adversarial attacks like PGD and FGSM is referred to in terms of the noise magnitude. Whereas for patch attacks, patch size proportionally determines the strength of the attack.

In this experiment, we vary the block size from 7×7

to 112×112 by doubling the side length in each step. We evaluate the accuracy of all six models stated in Section 5.1 against the 'Cornet' adversarial patch (randomly chosen). Figures. As evident from Figure 4, block:7×7 has higher adversarial accuracy than the naive model,but the natural accuracy decreases considerably. Smaller-sized blocks seem to have a higher tendency for false positives for detecting patches, eventually masking some of the clean regions, which considerably lowers natural accuracy.

Block:14×14 and block:28×28 have the best performance as they lower the natural accuracy marginally but improve the adversarial accuracy to a large extent. On the other hand, block:112×112 have lower robustness because calculating the total variation score over large regions lowers the chance of detecting patch as outliers. Since top-1 and top-3 adversarial accuracy peaks at block:28×28, which we finally decided as our block size. However, please note that the block size for the best performance will vary based on the conservative assumption of the patch's size.

5.5. Influence of Image Inpainting on Accuracy

As discussed in Section 4, the total variation based outlier detection and masking leads to mitigation of the patch attack. Even though TVR nullifies the influence of the patch, we lose some information about the original image. The image inpainting helps to recreate close to an original image using a trained generator. This study evaluates the necessity of image inpainting. We test eleven scenarios (ten patch attacked, and one unattacked) on the ImageNet-Patch benchmark dataset and calculate adversarial accuracy based on images from the test set.

Target Class	Cellphone	Cornet	Guitar	Hair Spray	Soap Dispenser		Sock	Typewriter	Plate	Banana	Cup
Predicted Class	Cellphone	Cornet	Guitar	Joystick	Soap Dispenser		Sock	Patten	Burrito	Banana	Cup
Attacked											
Top-3	No	No	No	Yes	No		No	Yes	No	No	No
Predicted Class	Joystick	Joystick	Joystick	Joystick	Joystick		Buckle	Sandal	Sandal	Backpack	Modem
Defended											
Top-3	Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	No

Figure 5. Physical Experiments with Joystick and Sandal

We calculate the accuracy for three scenarios: naive model, TVR without image inpainting, and TVR with image inpainting. Figure 6 show top-3 accuracy highlighting that TVR with image inpainting outperforms the one without against all patches with an improvement of 16%. We also observe inpainting lowering the gap between natural (unattacked) and adversarial (patch attacked) accuracy.

6. Physical Demonstration of TVR

A patch attack is a physical world attack, which makes it crucial to test TVR with a real-world example. Validating defensive ability of TVR in the digital setup is insufficient, as the attack may lose potency during its translation from the digital to the physical environment. This study considers two object categories: Joystick and Sandal, as shown in Figure 5. These classes also belong to the ImageNet benchmark dataset. The objects are subjected to ten different patches from the ImageNet-Patch dataset. The patches are applied with random transformation and scaling in the range [0.7, 1]. The environmental factors like lighting, which may influence the prediction are maintained. This experiment follows the setup proposed in [24]. For this study, we did

Figure 6. Increment in top-3 accuracy post image inpainting in different scenarios. For patch target class labels, refer Section 5.

not perform image inpainting, as our goal is to demonstrate the TVR's ability to mitigate the effect of the patch.

We use a ResNet18 model for the prediction in all test cases. The predicted class shows the top-1 class predicted by the model. We also mention if the predicted class is within the top-3 prediction. As seen in Figure 5, we only have 2 out of 10 correct top-3 predictions for the naive model, whereas the correct prediction rose to 9 out of 10 with TVR. The results show that predicting the adversarial Joystick example is easier. One possible reason might be that Backpack is much closer to Sandal in the feature space (because the straps of the sandals look like the straps of the Backpack). However, there is no such class for the Joystick, making its prediction relatively easier.

This study can be summarised as follows: First, the size of patches used to attack is larger than that of a digital scenario. It is necessary because the attack efficacy drops while translating the patch from a digital to a physical scenario. Second, the TVR does not have to mask the entire patch to mitigate its malicious influence on the prediction. Instead, it only masks regions with higher TV scores than the outlier of the whole image. Nevertheless, TVR is able to assist the model to make correct predictions in the physical setting.

7. Conclusion

This work proposes a total-variation-based image resurfacing technique to mitigate the threat from adversarial patches. TVR is a model and task-agnostic technique that only depends on the scene. We validate the performance of TVR in both digital and physical environments. The TVR can cleanse single or multiple patches in a single scan of the image. It ensures mitigation of the attack as long as the TV score of perturbations is larger than the outlier threshold. TVR is a first step towards dealing with arbitrary multiple localized perturbations in the scene.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. Ieee Access, 6:14410–14430, 2018. 1

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In International conference on machine learning, pages 284–293. PMLR, 2018. 1

[3] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. arXiv preprint arXiv:1712.09665, 2017. 1

[4] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. Theoretical foundations and numerical methods for sparse recovery, 9(263-340):227, 2010. 2

[5] Zhaoyu Chen, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Wenqiang Zhang. Towards practical certifiable patch defense with vision transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15148–15158, 2022. 2

[6] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studer, and Tom Goldstein. Certified defenses for adversarial patches. In ICLR, 2020. 2

[7] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In 2020 IEEE Security and Privacy Workshops (SPW), pages 48–54. IEEE, 2020. 1

[8] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1625–1634, 2018. 1

[9] Thomas Gittings, Steve Schneider, and John Collomosse. Vax-a-net: Training-time defence against adversarial patch attacks. In ACCV, 2020. 2

[10] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117, 2017. 2

[11] Jamie Hayes. On visible adversarial perturbations & digital watermarking. In CVPR Workshops, pages 1597–1604, 2018. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 4

[13] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360, 2016. 4

[14] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In International Conference on Machine Learning, pages 2507–2515. PMLR, 2018. 1, 5

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017. 4

[16] Alexander Levine and Soheil Feizi. (De) randomized smoothing for certifiable defense against patch attacks. volume 33, pages 6465–6475, 2020. 2

[17] Junbo Li, Huan Zhang, and Cihang Xie. Vip: Unified certified detection and recovery for patch attack with vision transformers. In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV, pages 573–587. Springer, 2022. 2

[18] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pages 1028–1035, 2019. 1

[19] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. arXiv preprint arXiv:1806.02299, 2018. 1

[20] Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Xinyu Liu, and David Wagner. Minority reports defense: Defending against adversarial patches. In Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings, pages 564–582. Springer, 2020. 1

[21] Jan Hendrik Metzen, Nicole Finnie, and Robin Hutmacher. Meta adversarial training against universal patches. arXiv preprint arXiv:2101.11453, 2021. 1

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. nature, 518(7540):529–533, 2015. 1

[23] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1300–1307. IEEE, 2019. 1, 6

[24] Maura Pintor, Daniele Angioni, Angelo Sotgiu, Luca Demetrio, Ambra Demontis, Battista Biggio, and Fabio Roli. Imagenet-patch: A dataset for benchmarking machine learning robustness against adversarial patches. Pattern Recognition, 134:109064, 2023. 4, 8

[25] Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16, pages 429–448. Springer, 2020. 1, 2

[26] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. Advances in Neural Information Processing Systems, 34:29935–29948, 2021. 1

[27] Huali Ren, Teng Huang, and Hongyang Yan. Adversarial examples: attacks and defenses in the physical world.

International Journal of Machine Learning and Cybernetics, pages 1–12, 2021. 1

[28] Hadi Salman, Saachi Jain, Eric Wong, and Aleksander Madry. Certified patch robustness via smoothed vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15137–15147, 2022. 2

[29] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In Proceedings of the 2016 acm sigsac conference on computer and communications security, pages 1528–1540, 2016. 2

[30] Abhijith Sharma, Yijun Bian, Phil Munz, and Apurva Narayan. Adversarial patch attacks and defences in vision-based tasks: A survey. arXiv preprint arXiv:2206.08304, 2022. 1

[31] Abhijith Sharma, Yijun Bian, Vatsal Nanda, Phil Munz, and Apurva Narayan. Vulnerability of cnns against multi-patch attacks. In Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, pages 23–32, 2023. 1, 4

[32] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48, 2019. 1

[33] Abdul Jabbar Siddiqui and Azzedine Boukerche. A novel lightweight defense method against adversarial patches-based attacks on automated vehicle make and model recognition systems. Journal of Network and Systems Management, 29(4):41, 2021. 2

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 4

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015. 4

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016. 4

[37] Y Taigman, M Yang, M Ranzato, and L Wolf. Closing the gap to human-level performance in face verification. deepface. In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), volume 5, page 6. 1

[38] Hui Wei, Hao Tang, Xuemei Jia, Hanxun Yu, Zhubo Li, Zhixiang Wang, Shin'ichi Satoh, and Zheng Wang. Physical adversarial attack meets computer vision: A decade survey. arXiv preprint arXiv:2209.15179, 2022. 1

[39] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. Patchguard: Provable defense against adversarial patches using masks on small receptive fields. arXiv preprint arXiv:2005.10884, 2020. 1

[40] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. {PatchCleanser}: Certifiably robust defense against adversarial patches for any image classifier. In 31st USENIX Security Symposium (USENIX Security 22), pages 2065–2082, 2022. 1, 2, 6

[41] Chong Xiang and Prateek Mittal. Patchguard++: Efficient provable attack detection against adversarial patches. arXiv preprint arXiv:2104.12609, 2021. 1