SuperPure: Efficient Purification of Localized and Distributed Adversarial Patches via Super-Resolution GAN Models

Hossein Khalili[1], Seongbin Park[1], Venkat Bollapragada[1], and Nader Sehatbakhsh[1]

[1]University of California, Los Angeles (UCLA)

Preprint. This work has been submitted to arXiv.

Abstract—As vision-based machine learning models are increasingly integrated into autonomous and cyber-physical systems, concerns about (physical) adversarial patch attacks are growing. While state-of-the-art defenses can achieve certified robustness with minimal impact on utility against highly-concentrated localized patch attacks, they fall short in two important areas: *(i)* they are vulnerable to low-noise *distributed* patches where perturbations are subtly dispersed to evade detection or masking, as shown recently by the DorPatch [1] attack; *(ii)* they are extremely time and resource-consuming, making them impractical for latency-sensitive applications.

To address these challenges, we propose *SuperPure*, a new defense strategy that combines pixel-wise adversarial masking and GAN-based super-resolution. It is robust against both distributed and localized patches while achieving low inference latency. Extensive evaluations using ImageNet and two standard classifiers (ResNet and EfficientNet) show that SuperPure: (i) improves robustness against localized patches by ¿20% and clean accuracy by 10%; (ii) achieves 58% robustness against distributed patch attacks (vs. 0% for PatchCleanser); (iii) reduces defense latency by 98%. SuperPure is robust to patch sizes and white-box attacks. Code is open-source.

I. INTRODUCTION

Deep learning models have achieved remarkable success in various computer vision tasks including image classification, object detection, and semantic segmentation [2], [3], [4]. However, they are highly vulnerable to adversarial attacks, which involve adding perturbations to input data in order to mislead models into making incorrect predictions [5], [6]. Among these, adversarial patches—large perturbations confined to a localized region—pose a significant and practical threat due to their effectiveness and ease of deployment [7], [8]. Specifically, they pose a serious threat to real-world vision applications such as autonomous driving and security systems, where attackers can physically print out and place a patch in the environment to manipulate the model's output [9], [10].

Various defense strategies have been proposed to counter adversarial patches [11], [12], [13], [14], [15], [16]. A common approach involves eliminating adversarial samples through various forms of masking and image transformations. Currently, the most popular and effective defense method is one developed by Xiang *et al.* called PatchCleanser [14]. This certified defense technique identifies and masks suspicious areas in an image, using randomized smoothing methods to provide theoretical guarantees. However, while PatchCleanser performs well against localized patches of a specific size, its effectiveness diminishes when it encounters advanced *distributed* attacks that spread perturbations to escape detection. Exploiting this vulnerability, Tang *et al.* introduced DorPatch [1], a dispersed and occlusion-robust adversarial patch attack that distributes perturbations across multiple regions. Importantly, the misclassified results from adversarially patched samples created by DorPatch can obtain certification from PatchCleanser, leading to a false sense of security in the guaranteed predictions. These findings highlight the urgent need for the development of effective defenses to counter such attacks.

Apart from concerns about *robustness* against distributed attacks, the *computation overhead* of current defense mechanisms has also been a growing concern. This is particularly important when considering latency-sensitive applications such as image classification for autonomous systems, where the solution has to be both robust and lightweight. Despite recent efforts to further improve the computational efficiency [15], the overhead remains quite high—on the order of *tens* of seconds per image for state-of-the-art methods [15], [14], as we will demonstrate in this paper.

This paper introduces *SuperPure*, a novel iterative defense mechanism designed specifically to counter adversarial patch attacks, addressing critical robustness and latency constraints. At its core, *SuperPure* iteratively masks adversarial patches by combining low-pass downsampling, GAN-based super-resolution upsampling, and precise pixel-level comparisons to identify and eliminate both localized and distributed adversarial signals.

At each iteration, *SuperPure* first downsamples the input image. This step acts as a low-pass filter, attenuating high-frequency adversarial perturbations while retaining essential image content [17], [18], [19]. The primary structure and semantics of the image remain largely unaffected due to inherent redundancy and spatial correlation typically observed in natural images [20]. The iterative non-linear GAN-based upsampling further disrupts adversarial features, progressively restoring clean image regions.

An important insight was that simple upsampling with a GAN would <u>not</u> be sufficient. Instead, we propose a novel method called ***GAN-guided pixel masking.*** Specifically, we
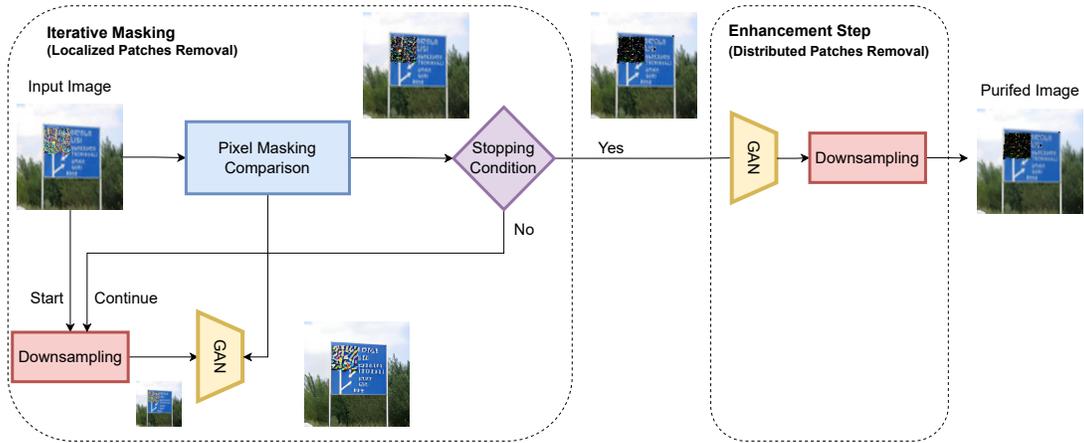
Fig. 1. SuperPure pipeline: at each iteration, we downsample, GAN-upsample, and mask high-error pixels. If newly masked pixels exceed the threshold, we repeat. At the end, a final upsample–downsample "enhancement" removes subtle perturbations.

first upsample the image using the GAN to introduce *non-linearity* to the processing pipeline, and then *compare* this newly generated image with the original image. The key insight is that, since the super-resolution model (GAN) is trained on natural (clean) images, *it affects the pixels related to the patches more than the clean pixels when creating a new image.* We then implement an additional non-linear step called pixel masking, in which the pixels of the newly generated image are compared with those of the original image on a pairwise basis. Any pixels that differ significantly from the original, based on a predefined threshold, are masked. This approach allows our method to eliminate pixels that are more influenced by the down-sampling and super-resolution processes. The important takeaway is that pixels associated with the patch are much more likely to be among those that differ.

The second purpose of GAN-guided pixel masking is to *improve clean accuracy*. Simple downsampling and upsampling alone can lead to significant information loss, which in turn reduces clean accuracy; hence, a more sophisticated scheme is used for upsampling.

This *down-up sampling plus masking* process continues until the percentage of newly masked pixels drops below a small threshold. We find that this method is more effective when changes are gradual. Additionally, to guard against less noticeable, distributed patches, we include an *enhancement step* that involves upsampling the image using super-resolution, followed by downsampling it back to its original size. Figure 1 illustrates our scheme.

To minimize computation overhead, *SuperPure* utilizes several techniques. Primarily, it employs a lightweight GAN-based super-resolution model for the upsampling step. It also utilizes an adaptive stopping condition. In Section IV, we outline the details.

Through extensive experiments on the ImageNet dataset [21] using state-of-the-art classifiers, Resnet and EfficientNet, we demonstrate that our method significantly enhances robustness against adversarial patch attacks of both singular and distributed varieties. We compare our method to the widely popular method, PatchCleanser [14], and the most recent defense mechanism in this domain, PAD [11]. Compared to state-of-the-art, *SuperPure* improves the robustness against conventional localized patches by more than 20%, on average, while also improving top-1 clean accuracy by almost 10%. *SuperPure* also achieves 58% robustness against distributed patch attacks while PatchCleanser [14] is completely vulnerable to this attack (0% robustness). Lastly, *SuperPure* also decreases the end-to-end latency by over 98%.

In summary, our work presents the following contributions:

1) We present *SuperPure*, a novel defense mechanism against white-box patch attacks of both singular and distributed varieties that is compatible with various image classifiers.
2) We introduce multiple techniques to reduce the computation complexity of our approach without significantly impacting the accuracy and/or robustness of the defense.
3) We evaluate *SuperPure* on standard datasets and classifiers and compare our method against state-of-the-art methods.
4) We provide ablation and sensitivity studies to provide more insights about our method and highlight its effectiveness.
5) We open-source our source code.

II. BACKGROUND

A. Adversarial Patch Attacks

Given a model M that produces an output $M(x)$ given a sample x, adversarial attacks involve finding a modified sample $\tilde{x}$ such that $M(\tilde{x}) \neq M(x)$ [5], [22], [23], [24], [25]. A common type of attack involves adding imperceptible perturbations throughout the whole image, designed to subtly distort the image without drawing human attention [26], [6], [27].

In contrast, **adversarial patch attacks** add conspicuous perturbations to a restricted area of the image called a patch [28], [8]. These patches are especially relevant in physical settings where they can be applied as stickers or printed patches [7], [29], [9]. Early papers assumed a white-box threat model, where adversaries have access to the internal details of the target classifier. In contrast, PatchAttack introduced a reinforcement learning-based method for generating adversarial patches in a black-box setting [30].

Unlike singular patches that are localized to one specific region, RP2 [9] uses a distributed adversarial patch, which is harder to locate and cover. More recently, researchers proposed Distributed and Occlusion-Robust Adversarial Patch (DorPatch), which uses group lasso on patch masks, image dropout, density regularization, and structural loss to produce a distributed and occlusion-robust patch for real-world deployment [1].

There have been adversarial patch attacks proposed in other domains such as object detection [10], [31], but in this paper, we focus on image classifiers as our target model.

B. Defense Mechanisms

Initial defenses against adversarial patches include Digital Watermarking [28], which involves masking unnaturally dense regions in the saliency map, and Local Gradient Smoothing [32], which regularizes gradients in the estimated noisy regions. However, Chiang *et al.* showed that these defenses can be bypassed using adaptive white-box attacks, and proposes the first certifiable defense against patch attacks by extending interval bound propagation (IBP) defenses [12]. Clipped Bag-Net (CBN) [33] uses clipped BagNet, a classification model with small receptive fields, for certified robustness. Levine and Feizi extends randomized smoothing robustness schemes by leveraging the constrained nature of patch attacks to derive larger, deterministic robustness certificates. However, these works are overshadowed by PatchGuard [13] in terms of performance.

PatchGuard utilizes small receptive fields in CNNs to prevent adversarial patches from influencing classification. However, this approach is less effective for larger models with wider receptive fields and requires significant architectural modifications, limiting its broader applicability. PatchCleanser [14] employs two rounds of pixel masking on the input image to neutralize the effect of adversarial patches. PatchCure is based on the same principles as PatchCleanser but is optimized for efficiency [15]. However, unlike PatchCleanser, PatchCure requires retraining the underlying classifiers to integrate its defense mechanisms, introducing additional computational overhead and limiting its practical deployment flexibility. Both PatchCleanser and PatchCure are vulnerable to disturbed patch attacks [1].

Many patch localization-based defenses have also been proposed. SentiNet utilizes techniques from model interpretability and object detection to detect potential attacks [34]. Jedi utilizes input entropy analysis to detect and remove adversarial patches [35]. PatchZero [36] detects adversarial pixels using a patch detector and repaints them with the mean color of the surrounding pixels. Similarly, PAD [11] leverages mutual information and recompression to locate and remove adversarial patches, effectively addressing challenges related to patch appearance, shape, size, location, and quantity without needing prior attack knowledge.

There have also been research done utilizing adversarial training [6], which is widely used to mitigate attacks based on adversarial samples, to mitigate patch attacks. These works [37], [38], [39] propose novel training schemes to boost robustness, but they require retraining and/or incur high computational overhead. In this work, we focus on preprocessing defenses that do not require retraining of the target model.

C. Image Super Resolution

Image Super-Resolution (SR) is a long-standing problem in computer vision, aiming to recover a high-resolution (HR) image from its low-resolution (LR) counterpart. Traditional interpolation methods, such as bicubic and bilinear interpolation, often fail to reconstruct fine details and high-frequency textures, resulting in blurred images.

With the rise of deep learning, Convolutional Neural Networks (CNNs) have become predominant in SR tasks. Dong et al. [40] introduced the Super-Resolution Convolutional Neural Network (SRCNN), pioneering the use of deep learning for SR. Following this, Kim et al. [41] proposed the Very Deep Super-Resolution (VDSR) network, utilizing a much deeper architecture and residual learning to enhance performance.

In addition to this, Generative adversarial networks (GANs) [14] have become a popular approach, often used as a loss function to push the results closer to the natural image manifold, improving perceptual quality. Ledig et al. [42] proposed the Super-Resolution Generative Adversarial Network (SR-GAN), which introduced an adversarial loss and a perceptual loss based on high-level feature mappings from a pre-trained VGG network. This approach enabled the generation of more photo-realistic images with sharper details. However initial methods often struggle to perform well on real-world images. This was due to the fact that standard blurs or filters often did not capture the complex intricacies of image degradation. Building upon SRGAN, Wang et al. [43] developed the Enhanced SRGAN (ESRGAN), which introduced the Residual-in-Residual Dense Block (RRDB) to facilitate training of deeper networks without degradation. ESRGAN also replaced the standard discriminator with a relativistic average discriminator to provide stronger supervision and better gradient flow, resulting in superior perceptual quality.

D. Super-Resolution Defenses

Recent works have utilized super-resolution (SR) to eliminate adversarial noise by projecting low-quality inputs back onto a manifold of natural images. Mustafa *et al.* [44] introduced an SR-based approach to remove subtle ℓ_p-bounded perturbations, restoring clean performance at modest overhead. Meanwhile, DiffPure [45] applies diffusion-based sampling to expunge mild adversarial artifacts, also yielding high-quality

outputs under small or dispersed corruptions. However, **both** SR- and diffusion-based defenses commonly assume lower-amplitude distortions; large, localized patch signals often remain partially intact—sometimes even sharpened—after SR. Consequently, these methods fail to robustly handle overt patch attacks without dedicated detection and masking strategies.

III. PROBLEM FORMULATION

In this section, we clearly define the objectives of the attacker and defender considered in this paper, focusing explicitly on two primary adversarial scenarios: *localized* and *distributed* patches.

A. Attack Objective

1) Attacker Goal: The adversary seeks to mislead a classifier by embedding adversarial patches into input images. Formally, consider an original clean image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, where H, W, and C represent its height, width, and number of channels, respectively. An adversarial patch introduces an additive perturbation $\boldsymbol{\delta} \in \mathbb{R}^{H \times W \times C}$, resulting in an adversarial image:

$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \boldsymbol{\delta}. \tag{1}$$

We specifically consider two distinct types of adversarial patches:

- **Localized patches:** Perturbations are concentrated within a single, contiguous region $\mathcal{P}_L \subseteq \{1, \ldots, H\} \times \{1, \ldots, W\}$.
- **Distributed patches:** Perturbations are divided across multiple small regions $\mathcal{P}_D = \{p_i\}_{i=1}^n$ spread throughout the image, specifically crafted to evade detection by distributing the adversarial signal, as introduced by DorPatch [1].

Given a classifier $\mathcal{F}$, original image $\mathbf{x}$, and true label y, the attacker's objective is to construct $\mathbf{x}_{\text{adv}}$ such that:

$$\mathcal{F}(\mathbf{x}_{\text{adv}}) \neq y. \tag{2}$$

2) Attacker Capabilities: We assume that the attacker generally has full access to the classifier's gradients and parameters, enabling effective optimization of both localized and distributed patches. Additionally, in stronger adaptive (white-box) settings explicitly evaluated in Section V, the attacker gains knowledge of SuperPure's super-resolution model parameters, increasing their capacity to craft robust patches. For PatchCleanser [14], direct gradient-based (white-box) attacks are infeasible due to its non-differentiable masking process; thus, we specifically evaluate PatchCleanser against distributed adversarial patches crafted using DorPatch. All patch attacks adhere to realistic size constraints (patch budgets) appropriate for physical deployment scenarios, such as printed adversarial stickers.

B. Defense Objective

The defender aims to neutralize adversarial patches—both localized and distributed—while preserving accurate classification on unaltered images. To achieve broad applicability, our proposed defense method, *SuperPure*, meets the following key criteria:

Clean Robustness. Given a clean image-label pair $(\mathbf{x}, y)$ and a classifier $\mathcal{F}$, the defended input maintains accurate classification:

$$\mathcal{F}(SuperPure(\mathbf{x})) = y.$$

Adversarial Robustness. Given an adversarial input $\mathbf{x}_{\text{adv}}$ such that $\mathcal{F}(\mathbf{x}_{\text{adv}}) \neq y$, the defended method restores the correct classification:

$$\mathcal{F}(SuperPure(\mathbf{x}_{\text{adv}})) = y.$$

Model Independence. *SuperPure* operates independently of the underlying classifier, requiring no modifications to classifier architecture or retraining.

Scalability and Efficiency. *SuperPure* executes efficiently, suitable for real-world deployment scenarios involving high-resolution images and strict resource constraints.

IV. METHOD

In this section, we propose *SuperPure*, a novel defense strategy against adversarial patch attacks that combines a downsampling and upsampling process with a pixel-by-pixel comparison to remove both singular and distributed adversarial patches. The details of our method are presented in Algorithm 1.

Briefly, our algorithm first downsamples an image, x_{adv}, by a factor of s (=4 in our setup). The downsampled image, x_{down} is then fed into an upsampling method to generate a new image, x_{up}. The upsampled image has the same size as the original (i.e., $|x_{up}| = |x_{adv}|$). While various upsampling/image generation mechanisms exist, our experiments reveal that a GAN-based super-resolution balances accuracy, robustness, and latency. It further introduces non-linearity that helps remove the patches while retaining the information about the original image. The new image, x_{up}, is compared, pixel-wise, with x_{adv} using a threshold, λ (see lines 17-22). In Section VI, we show how this threshold should be set. This process (downsampling, upsampling, and masking) repeats multiple times until the number of masked pixels becomes smaller than a threshold, ϵ. We also report how to find this threshold in Section VI. The final step involves first upsampling and then downsampling the image. No masking is applied during this step (lines 11-14). The purpose of this final step is to *(a)* enhance the image quality for improved accuracy and *(b)* eliminate low-noise distributed patches.

We explain the theory and details of the downsampling process in Section IV-A and then discuss the details of the super-resolution and masking algorithm in Section IV-B. In Section IV-C, we outline how the method is iterated and specify the stopping condition for these iterations, and in

4

Algorithm 1: Forward Method for *SuperPure*

Input: Input image $\mathbf{x}_{\text{adv}}$, max iteration count $\mathcal{K}$, masking threshold λ, stopping criterion ϵ, enhance flag $\mathcal{E}$, downsampling function DOWN, pretrained superresolution model G

Output: Processed image $\mathbf{x}$

1 **Procedure** *SuperPure* $(\mathbf{x}_{\text{adv}}, \mathcal{K}, \lambda, \epsilon, \mathcal{E})$:
2   **for** $k \leftarrow 0$ **to** $\mathcal{K}$ **do**
3         $\mathbf{x}_{\text{down}} \leftarrow$ DOWN $(\mathbf{x}_{\text{adv}}, 4)$;
4         $\mathbf{x}_{\text{up}} \leftarrow G(\mathbf{x}_{\text{down}}, 4)$;
5         $\mathbf{x}_{\text{adv}}, c \leftarrow$ GetDiff $(\mathbf{x}_{\text{adv}}, \mathbf{x}_{up}, \lambda)$;
6   **if** $c < \epsilon$ **then**
7   **break**;
8   **end**
9   **end**
10     $\mathbf{x} \leftarrow \mathbf{x}_{\text{adv}}$;
11   **if** $\mathcal{E}$ **then**
12         $\mathbf{x} \leftarrow G(\mathbf{x}, 2)$;
13         $\mathbf{x} \leftarrow$ DOWN $(\mathbf{x}, 2)$;
14   **end**
15   **return** $\mathbf{x}$;
16
17 **Procedure** GetDiff $(\mathbf{x}_{\text{adv}}, \mathbf{x}_{up}, \lambda)$:
18     $\mathbf{d} \leftarrow \mathcal{L}_2(\mathbf{x}_{\text{adv}}, \mathbf{x}_{\text{up}})$;
19     $\mathbf{m} \leftarrow \mathbf{d} > \lambda$;
20     $\mathbf{x}_{\text{adv}} \leftarrow \mathbf{x}_{\text{adv}} \odot \mathbf{m}$;
21     $\mathbf{c} \leftarrow \sum_{i,j} \mathbf{m}(i,j)$;
22   **return** $\mathbf{x}_{\text{adv}}, \mathbf{c}$;

Section IV-D, we describe the enhancement step. Lastly, we explain the optimizations proposed in our design to reduce the computation complexity in Section IV-E.

A. Downsampling Step

Downsampling an image reduces its spatial resolution, leading to the loss of high-frequency information; this disproportionally affects adversarial patches in comparison to the essential image content [17], [18]. This is due to adversarial patches being heavily reliant on precise, high-frequency perturbations [7]. For downsampling, we utilize *bilinear interpolation*, which replaces every $n \times n$ window by its average pixel value when downsampling by a factor of n [46]. This averaging process smooths out the detailed perturbations in the patch, causing the high-frequency information needed for the patch's effectiveness to be lost. In contrast, natural images typically display redundancy and correlation among neighboring pixels [47], [20]. Important features and structures are often replicated across the image, making essential information less susceptible to significant degradation.

Formally, for an image of size $H \times W$, downsampling can be modeled as a function $D_s : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{h \times w \times C}$, where s is the scaling factor ($s > 1$), $h = H/s$, and $w = W/s$. Applying downsampling to the adversarial image yields:

$$\mathbf{x}'_{\text{adv}} = D_s(\mathbf{x}_{\text{adv}}) = D_s(\mathbf{x} + \boldsymbol{\delta}) = D_s(\mathbf{x}) + D_s(\boldsymbol{\delta}). \quad (3)$$

Then, the energy of the adversarial patch after downsampling can be approximated as:

$$\|D_s(\boldsymbol{\delta})\|_2^2 \approx \frac{1}{s^2} \|\boldsymbol{\delta}\|_2^2, \quad (4)$$

indicating a substantial decrease in the perturbation's magnitude due to spatial averaging.

This theoretical insight aligns with empirical evidence. Pixels within adversarial patches consistently exhibit significantly higher reconstruction errors compared to non-patch pixels; in our experiments, patch regions showed approximately $8\times$ higher errors (mean squared error of 0.6054 vs. 0.0829, as shown in Figure 11 in Appendix). Additionally, prior work by Guo et al. [48] similarly demonstrated that downsampling (e.g., via JPEG compression) effectively reduces adversarial perturbation effectiveness while preserving high accuracy on clean images. Nonetheless, simple downsampling alone remains vulnerable to adaptive attacks, underscoring the necessity of our proposed iterative and non-linear GAN-based upsampling process.

B. Upsampling and Masking

While downsampling degrades adversarial patch regions more significantly than natural non-patch areas, our initial analysis showed that applying naive down and upsampling is not sufficient to fully remove distributed patches and stronger adaptive attacks. Instead, a more powerful transformation is needed. The key idea is to apply a transformation that ***disproportionally*** affects adversarial regions over benign regions.

Based on this insight, we propose utilizing a super-resolution model for upsampling. While there are various super-resolution models available, we have chosen Real-ESRGAN [49] as our primary model. Real-ESRGAN is a GAN-based model designed and trained to reconstruct high-quality images from low-resolution inputs; we use a pretrained model provided by the authors, with no further fine-tuning on our datasets.

There are two reasons why the GAN struggles to reconstruct patch regions in comparison to benign non-patch regions.

First, for a pixel $p_{i,j}$ within the adversarial patch, the surrounding pixels $p \in B((i,j), \tau)$ are typically uncorrelated with $p_{i,j}$ and are also heavily degraded, leaving the GAN G with insufficient information for precise reconstruction. On the other hand, natural non-patch regions have more structural coherence and redundancy, leaving enough information for reconstruction even after downsampling.

Second, since the GAN is trained on a dataset of natural images, it is optimized to generate outputs that align with the natural image distribution. However, adversarial patches deviate from this distribution, resulting in a greater pixel variation in these regions in comparison to the rest of the image. Let p_{i,j_a} denote an adversarial pixel and p_{i,j_c} a clean pixel inside an image π. This relationship can be expressed through the following inequality:

$$\mathbb{E}(|p_{i,j_a} - G(D_s(\pi))_{i,j}|) \gg \mathbb{E}(|p_{i,j_c} - G(D_s(\pi))_{i,j}|). \quad (5)$$
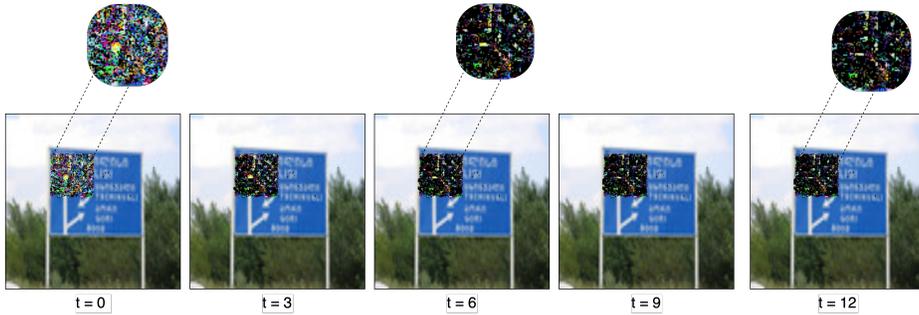
Fig. 2. Shows the masking process of SuperPure across multiple time steps.

This disparity results in *greater pixel-wise differences* between the original image and the reconstructed output in patch regions compared to non-patch regions. Therefore, by comparing the original adversarial image $\mathbf{x}_{\mathrm{adv}}$ to the upsampled image $\mathbf{x}_{\mathrm{up}}$ on a pixel-by-pixel basis, we can mask adversarial regions by identifying where the reconstruction error exceeds a pre-defined threshold λ. Specifically, a pixel-wise comparison computes the $\mathcal{L}_2$-distance between corresponding pixels in $\mathbf{x}_{\mathrm{adv}}$ and $\mathbf{x}_{\mathrm{up}}$. A binary mask $\mathbf{m}$ is then generated, where:

$$\mathbf{m}(i,j) = \begin{cases} 1, & \text{if } \|\mathbf{x}_{\mathrm{adv}}(i,j) - \mathbf{x}_{\mathrm{up}}(i,j)\|_2 > \lambda, \\ 0, & \text{otherwise.} \end{cases}$$

The binary mask $\mathbf{m}$ is then overlaid on the adversarial image $\mathbf{x}_{\mathrm{adv}}$, effectively masking the regions that exhibit high reconstruction error and are likely to be adversarial patches.

C. Iteration and Stopping Condition

The inequality from Equation (5) suggests that with an appropriate choice of λ we can effectively mask adversarial pixels while preserving clean ones. However, a single iteration may not suffice to identify most adversarial pixels beyond the threshold, as a GAN can utilize local context. As a result, we use multiple iterations and observe that as the number of iterations increases, our model converges and progressively masks out adversarial pixels. This is demonstrated empirically in Figure 3 as we can see the total masked pixels and total new pixels converge and steady as the number of iterations increases. The progressive masking can also be seen in Figure 2.

Based on the insight above, the processes presented in the previous two sections are repeated to ensure that the patch is as completely masked as possible. At each iteration, the adversarial image $\mathbf{x}_{\mathrm{adv}}$ is updated by overlaying the binary mask $\mathbf{m}$, and the new masked image becomes the input for the next iteration. This allows the adversarial regions to be gradually refined and more effectively suppressed with each cycle. As presented in Figure 2, with each iteration, a greater percentage of the patch regions in the image is masked. The majority of the masking is done in earlier iterations; the number of newly masked pixels steadily decreases until the stopping condition is met.
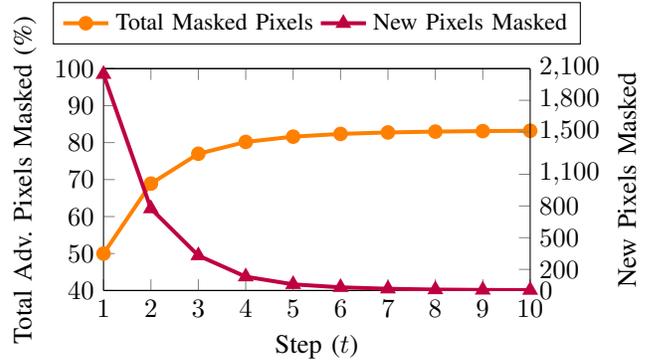

Fig. 3. Total and new adversarial pixels masked (ImageNet, Patch Size 64, Threshold=0.7)

The stopping condition is based on the fraction of pixels that have been newly masked at a given iteration. Specifically, the iteration continues until the percentage of the newly masked pixels at iteration t is below a threshold ϵ:

$$\frac{\sum_{i,j}^{W,H} \mathbf{m_t}(i,j)}{W \times H} < \epsilon,$$

where W and H are the width and height of the image. Once the stopping condition is met, perceptible adversarial patches are mostly eliminated.

D. Removing Small Distributed Perturbations

While the aforementioned iterative masking process is effective at removing perceptible adversarial patches, it struggles with smaller, distributed perturbations, such as those presented in Dorpatch [1]. These perturbations are subtler and often resemble natural variations within the image. Because our process relies on the assumption that the reconstruction of adversarial patches will differ significantly from that of non-patch natural image regions, this limitation of the masking process is expected.

A simple solution to mitigating these attacks involves upsampling the image via a GAN-based super-resolution model and then downsampling to the original size.

This upsampling step via super-resolution generates additional high-frequency details, while the subsequent downsampling removes that added information. The approach can be

compared to DiffPure [50], where random noise is added to an image before denoising it to remove any adversarial noise process. In our case, upsampling acts as the diffusion step, and downsampling serves as the denoising step, eliminating the small adversarial perturbations that are embedded within the image. Using Equation 5 and its assumptions, we observe that the generated pixel is more likely to resemble the natural distribution, excluding adversarial noise. The averaging process during downsampling further reduces this noise by smoothing out pixel-level variations, while the new clean pixel generated by the GAN helps to replace adversarial components. Together, these steps effectively reduce the overall noise, acting as a surrogate for the denoising step in diffusion models.

This process is performed after the initial iterative masking step, ensuring that any remaining subtle adversarial perturbations are addressed. We chose to upsample the image before downsampling (instead of vice versa) because starting with downsampling discards significant information in the image, leading to a loss of essential details and poor classifier performance. The final processed image can then be used for downstream tasks, ensuring that the adversarial manipulations no longer have a significant effect on model performance. We use *SuperPure+* as a shorthand for our method with this enhancement in subsequent sections.

E. Computation Overhead Optimizations

While our main goal is to improve the system's robustness against singular (localized) and distributed patch attacks, we've made a series of design decisions in order to reduce *SuperPure*'s computation overhead. In this part, we briefly discuss them and explain other alternatives.

Down vs. Upsampling. While *SuperPure* utilizes a downsampling step for patch removal, our initial analysis showed that a similar upsampling (using super-resolution) and then down-sampling could result in similar robustness. The main reason is that although the steps are swapped, the joint processing pipeline remains (relatively) the same hence similar theoretical analysis could be applied (i.e., super-resolution is harder for patch areas and down-sampling helps with filtering). Furthermore, as shown in Section IV-D, the up-down sampling option has an advantage over down-up since it can remove the distributed noises more effectively.

The key difference between the two options, however, relies on computation efficiency. In more detail, the up-down sequence increases the computation overhead because the more costly operation (super-resolution) needs to be performed on the larger image sizes (i.e., super-resolution on the original input size vs. super-resolution on the downsampled image). As a result, for the iterative masking part, we chose the down-up method instead of the up-down method to favor efficiency. However, to achieve enhancement, we add one final layer of up-down sampling. This way, accuracy-robustness-latency can be jointly optimized.

Super-Resolution Model. An important component in our design is the super-resolution model. Among various options, we opt for a GAN-based model since we found that it can achieve the right balance between robustness and complexity. The alternative option is using a more sophisticated model, e.g., a diffusion-based system [50]. While we expect that such a model would perform better (in terms of robustness and accuracy), it incurs orders of magnitude higher overhead (latency, memory, etc.).

Alternatively, the other extreme is using a much simpler upsampling strategy to further reduce the complexity. While we considered this, our initial analysis showed that simpler models are significantly more vulnerable to low-noise distributed attacks. Even worse, they are far more vulnerable to adaptive white-box attacks where an adversary creates patches that are resistant to up/downsampling. In Section VI, we study our model's robustness against white-box attacks and show that *SuperPure* retains decent robustness even in the presence of an adaptive attack.

Stop Condition. Another important factor in optimizing end-to-end latency is the stop condition. There is a tradeoff between the number of iterations and end-to-end latency. On one hand, more iterations are necessary to enhance robustness (see Figure 2). On the other hand, fewer iterations result in lower latency. We address this balance by implementing a dynamic stop condition method (see lines 6-7 in Algorithm 1) based on a user-defined parameter (ϵ). In Section VI, we examine the impact of different thresholds on robustness. Overall, our results demonstrate that *SuperPure* can achieve high robustness without requiring an excessively large number of iterations (fewer than ten on average), enabling it to attain both efficiency and robustness simultaneously.

V. RESULTS

In this section, we provide a comprehensive evaluation of *SuperPure*. We describe the experimental setup in Section V-A, followed by a presentation of the robustness analysis against various patch attacks in Section V-B. We also report the computation overhead results (latency and memory usage) in Section V-C.

Specifically, the goal of our analysis is to answer the following research questions:

- Q1: Compared to state-of-the-art, how robust *SuperPure* is against singular (localized) patch attacks?
- Q2: Is *SuperPure* robust against distributed patch attacks (as opposed to state-of-the-art)?
- Q3: Is *SuperPure* robust against white-box attacks?
- Q4: Compared to prior methods, what is the computation complexity of *SuperPure*?

In addition to answering these questions, we provide an extensive ablation study in Section VI to further analyze the important factors in jointly optimizing robustness-accuracy-latency.

A. Experimental Setup

In this section, we detail the experimental setup used to evaluate the effectiveness of our proposed method against adversarial patch attacks. We describe the datasets, classifiers,

Fig. 4. Quality results for *SuperPure* before (top) and after purification (down).

super-resolution setup, attack configurations, and evaluation metrics.

1) Datasets: Similar to prior work [14], we conduct our experiments using the ImageNet dataset [21], specifically utilizing a subset of the validation set to assess the performance of our method. From the validation set, we select five images per class, resulting in a total of 5,000 images across the 1,000 classes. This subset provides a diverse and representative sample for evaluating the robustness of our approach.

2) Classifiers: To demonstrate that our method is truly classifier-agnostic, we evaluate it on three *architecturally distinct* network families: EfficientNet-B0 [51], ResNet-152 v2 [52], and ViT-B/16 [4].

EfficientNet-B0 is part of the EfficientNet family, which utilizes compound scaling to balance network depth, width, and resolution. We use a PyTorch-provided checkpoint trained on ImageNet [53].

ResNet-152 v2 is a deep residual network with 152 layers, incorporating identity mappings to facilitate the training of very deep architectures. We use a PyTorch-provided checkpoint trained on ImageNet [53].

ViT-B/16 is a Vision Transformer model that tokenizes an input image into 16×16 patches and processes them via a Transformer-based architecture to capture global context. We use a PyTorch-provided checkpoint trained on ImageNet [53].

These three models are notably diverse in design: EfficientNet focuses on compound scaling, ResNet employs deep residual blocks, and ViT adopts a patch-based Transformer architecture. By testing on these fundamentally different paradigms, we thoroughly assess the robustness and generality of our proposed defense across a wide spectrum of network architectures.

3) SuperPure and Super-Resolution Model Setup: Our method leverages super-resolution techniques to mitigate the impact of adversarial patches. Specifically, we use the Real-ESRGAN models [49] for image upsampling, which serve to diminish the adversarial perturbations introduced by the attacks. We choose Real-ESRGAN due to *(a)* its minimal

domain overlap with ImageNet (it is not trained on ImageNet), *(b)* the availability of publicly released checkpoints for reproducibility, and *(c)* its GAN-based speed advantage over diffusion-based methods—crucial when it must be called multiple times in our pipeline.

We employ two versions of Real-ESRGAN; one that upsamples images by a factor of four, and another by a factor of two. The choice of these numbers is based on our preliminary analysis. We also considered other factors, but ultimately observed that four and two achieve the best balance between robustness and latency.

For both models, we use the checkpoints provided by the original authors, which were trained on the DIV2K [54], Flickr2K [55], and OutdoorSceneTraining (OST) dataset [56]. Note that ImageNet, our evaluation dataset, was <u>not</u> included in the Real-ESRGAN training set. We believe that fine-tuning on ImageNet could further improve our results. However, our following results show that *SuperPure* can achieve excellent performance even without fine-tuning.

Unless stated otherwise, for all our experiments, we set $\lambda = .7$ and $\epsilon = 4$ (pixels).

4) Adversarial Attacks: To evaluate the robustness of our method, we generate adversarial patches using the Masked Projected Gradient Descent (Masked PGD) method. Following the approach used in PatchGuard [13] and PatchCleanser [14], we use the following attack configuration:

- **Maximum Perturbation** (ϵ): Set to 1 (with pixel values normalized to the $[0, 1]$ range).
- **Number of Iterations**: 100 iterations.
- **Step Size**: 0.05 per iteration.
- **Random Start**: Each attack began from a random point within the allowed perturbation range.
- **Optimization Strategy**: Loss function evaluated every 10 iterations to select the sample that maximized the loss.
- **Patch Location**: Randomized for each image to simulate unpredictable attack scenarios.
- **Patch Sizes**: Patches of sizes 16×16, 32×32, 48×48, 64×64, and 96×96 pixels (approximately

Model	Defense Method	Patch Size					
		0 (no attack)	16×16	32×32	48×48	64×64	96×96
ViT	No Defense	74.84	38.02	4.32	00.50	0.16	0.00
	PatchCleanser [14]	72.10	54.33	44.21	35.3	30.74	20.72
	PAD [11]	44.76	46.58	47.04	46.36	45.62	41.64
	SuperPure	74.96	74.30	73.66	72.86	70.36	62.76
	SuperPure+	**82.98**	**80.70**	**77.82**	**77.52**	**74.66**	**65.9**
EfficientNet	No Defense	60.76	30.82	5.12	0.82	0.20	0.02
	PatchCleanser [14]	57.98	43.60	38.46	27.46	22.66	10.92
	PAD [11]	34.70	35.30	34.70	33.94	32.72	26.02
	SuperPure	61.10	59.76	58.34	**55.98**	**52.42**	**41.86**
	SuperPure+	**69.08**	**63.54**	**60.48**	54.12	46.72	28.22
ResNet	No Defense	71.70	45.10	24.52	14.28	4.38	0.10
	PatchCleanser [14]	68.98	56.72	51.64	41.28	33.82	19.66
	PAD [11]	48.19	50.10	50.28	49.38	45.04	43.30
	SuperPure	71.20	70.48	70.10	68.52	66.20	**58.18**
	SuperPure+	**79.86**	**76.74**	**76.30**	**74.20**	**70.64**	57.84

$0.5\%, 2\%, 4.6\%, 8.2\%$, and 18.4% of the average size of ImageNet [21] images respectively).

In addition to singular localized patches, we also consider distributed patches. Specifically, we create new patches using the method proposed by DorPatch [1]. This attack uses a patch budget of 12%, a density of 0.1%, and a maximum of $5,000$ iterations. It evaluates the ability of our method to handle sparse yet effective perturbations designed to evade detection.

5) Evaluation Metrics: We use clean accuracy (top-1) and robust accuracy as our primary evaluation metrics. Clean accuracy refers to the proportion of clean test images that our defended model correctly classifies. Robust accuracy is similarly defined as the proportion of adversarial test images accurately classified by our model. Additionally, we measure per-example inference time (s/img) to assess computational overhead.

We also report the defense performance of two state-of-the-art methods: PAD [11] and PatchCleanser [14], for comparison. We use the optimal defense settings stated in their respective papers; note that since PatchCleanser is dependent on patch size, we tested multiple configurations (window sizes) and chose the best results to report. Figure 4 presents the quality results.

6) Hardware Setup for Latency Measurement: All latency measurements are conducted using an NVIDIA RTX 4090 GPU with 24 GB of memory to ensure precise and consistent evaluation of computational performance. We use PyTorch version 1.12. Our code and results will be ***open-source***.

B. Robustness Results

In this subsection, we present robustness results for our defense against singular black-box and white-box patches of varying sizes. We also present results against distributed patches.

1) Single Patch Attacks: Table I compares the robustness of our defense method against PatchCleanser [14] and PAD [11]

for clean (patch size 0) and singular adversarial patch samples on EfficientNet and ResNet models.

We denote our approach without the last enhancement step as *SuperPure* and with enhancement as *SuperPure+*. As shown in Table I, both *SuperPure* and *SuperPure+* demonstrate significant robustness improvements over other defense methods under varying patch attack sizes.

On the smallest patches of size 16×16, *SuperPure+* achieves 63.54% and 76.74% Top-1 accuracy on EfficientNet and ResNet, respectively, compared to 30.82% and 45.10% for the baseline with no defense. For a patch size of 48×48, *SuperPure+* achieves 54.12% accuracy on EfficientNet and 74.20% on ResNet, while the highest-performing baseline defense (PAD) only reaches 33.94% and 49.38% on the respective models. Even with the largest patch size tested (96×96), *SuperPure+* achieves 28.22% Top-1 accuracy on EfficientNet and 57.84% on ResNet. This result is significantly higher than the no-defense baseline (close to 0% for both classifiers) and surpasses PatchCleanser and PAD, confirming our method's robustness against severe adversarial perturbations. Similar results can be observed for ViT, where *SuperPure* and *SuperPure+* consistently perform better compared to prior works.

An additional noteworthy aspect of our results is the *clean accuracy* achieved by both *SuperPure* and *SuperPure+*. On clean, unaltered images (patch size 0), *SuperPure* maintains strong Top-1 accuracy, while *SuperPure+* exhibits even better performance, achieving 82.98% on ViT, 69.08% on EfficientNet, and 79.86% on ResNet. These results underscore that our defense methods not only provide mitigations for attacks but also retain high accuracy for benign situations.

For *SuperPure+*, the clean accuracy is higher than the baseline configuration with no defense and no attack by around 8% for all models (74.84% for ViT, 60.76% for EfficientNet, and 71.7% for ResNet). Specifically for the ResNet classifier, *SuperPure+* achieves higher accuracy on adversarial samples

than the baseline no defense on clean images up to a patch size of 48×48, and shows comparable performance at 64×64. This indicates that our enhancement step contributes a dual benefit: **superior robustness to adversarial patches and improved accuracy on clean inputs**. The key reason for this is due to the super-resolution-based enhancement strategy described in Section IV-D. We further study the impact of enhancement in our ablation study (Section VI-B).

2) Distributed Patch Attacks: Unlike localized adversarial patches that occupy a single, contiguous region, *distributed* patches fragment perturbations into multiple sub-regions, making it harder for a single masking window to neutralize the entire patch. DorPatch [1] follows this principle by scattering its patch budget across many smaller pieces. Consequently, each piece can use a lower noise amplitude, which both reduces visibility and maintains high adversarial effectiveness.

Table II compares our defense with PatchCleanser and PAD against DorPatch. Even though DorPatch and standard patches share the *same overall budget*, distributing that budget means partial masking has less impact on the attack. This makes *certifiable* defenses like PatchCleanser—which relies on covering a single contiguous area—ineffective (robustness stays at 0% under DorPatch), whereas *SuperPure+* still defends successfully with 59% accuracy.

While *SuperPure* cannot purify distributed patches, the *enhancement* step in *SuperPure+* (see Section IV-D) remains effective at removing these scattered perturbations, allowing our defense to handle sophisticated multi-patch threats that circumvent single-window defenses like PatchCleanser.

Notably, DorPatch leverages *low noise amplitude* patches that subtly alter the image, rendering a purely threshold-based masking (i.e., *SuperPure–* without enhancement) ineffective: the per-pixel differences often lie below the threshold and thus remain unmasked. In contrast, *SuperPure+* adds an *enhancement* step to purify small, distributed perturbations, allowing our defense to handle DorPatch's scattered, low-amplitude noise when other single-window or simple threshold-based defenses fail.

3) Adaptive White-Box Attacks: We also evaluate our defense under an adaptive *white-box* threat model, where the adversary has complete access to both our purification network (including the super-resolution module and our method) and the target classifier. This means the attacker can compute gradients through every component of our defense to craft adversarial patches tailored explicitly to our defense mecha-

TABLE II
EFFECTIVENESS AGAINST DISTRIBUTED DORPATCH [1] ATTACK. THE ENHANCEMENT STEP IN *SuperPure+* MAKES OUR SYSTEM ROBUST.

Method	Clean Accuracy	DorPatch Robustness
No Defense	72%	0%
PatchCleanser [14]	69%	0%
PAD [11]	48%	39%
SuperPure	71%	0%
SuperPure+	**80%**	**59%**

TABLE III
ROBUST ACCURACY (%) UNDER WHITE-BOX ATTACKS ON RESNET.

Defense	Patch Size	
	48×48	64×64
Naïve Down&Up (white-box)	9.12	4.89
PatchCleanser [14] (non–white-box)	41.28	31.28
SuperPure/SuperPure+ (ours, white-box)	**60.38**	**51.52**

nism.

As shown in Table III, which compares the robust accuracy of three different defenses—*Naïve Down&Up*, *PatchCleanser*, and our method *SuperPure+*—against adversarial patches of size 48×48 and 64×64 on ResNet[1], a simple "Naïve Down&Up" defense completely fails when facing an adaptive attacker. This naive approach applies a fixed downsampling followed by upsampling in hopes of smoothing out adversarial noise. However, under a white-box setting where the attacker has the knowledge of this defense, it becomes trivial for the attacker to generate perturbations that survive such transformations. As a result, the robust accuracy drops sharply to only 9% and 5% for patch sizes of 48×48 and 64×64, respectively.

PatchCleanser, which is not differentiable and thus not directly applicable in a white-box setting, is evaluated here under its own original (black-box) setup. While it performs better than the naive method in that setting (41.28% and 31.28%), it still falls short compared to our method. In contrast, our proposed *SuperPure+* is evaluated in a fully adaptive white-box setting, where the attacker has complete access to the super-resolution model and can backpropagate through the entire pipeline. Despite this, our method maintains robust accuracy of 60.38% and 51.52%, significantly outperforming both the naive baseline and PatchCleanser—even though the latter operates under a more favorable (black-box) scenario. This strong robustness, even under white-box conditions, stems from the inherent nonlinearity and complexity of our defense pipeline. Unlike simple smoothing operations, our method first projects inputs into a more natural image manifold using a deep super-resolution network, then applies a masking and enhancement mechanism that further disrupts adversarial structures. Because the entire purification process is nonlinear and includes operations that do not preserve gradients in a predictable way, it becomes significantly harder for the attacker to generate perturbations that survive all these transformations. As a result, even with full access to our model and the ability to compute gradients through it, the adversary struggles to construct successful attacks. The resulting drop in robust accuracy compared to the black-box setting is relatively small, demonstrating that our method retains strong practical resilience—even in challenging adaptive white-box scenarios.

---

[1]We report results for two patch sizes on a single model due to the high computational cost of white-box experiments. However, based on prior findings, we believe these results are representative and likely transferable to other models and patch sizes.

TABLE IV
END-TO-END LATENCY.

Method	Time (s)
SuperPure/SuperPure+	**0.53 / 0.58**
PatchCleanser-Efficient/ PatchCleanser [14]	3.89 / 36.63
PAD	8.80

C. Computation Overhead

1) Latency: Another major benefit of *Super-Pure/SuperPure+* is its superior end-to-end latency compared to prior work. To highlight this, we compare *SuperPure/SuperPure+* to PatchCleanser and PAD. Specifically, we employ two different setups for Patch Cleanser. One setup used a larger masking window, resulting in fewer masks (16), which we refer to as *PatchCleanser-Efficient*. The other setup used a smaller masking window, increasing the number of masks (49) for greater robustness. These setups are evaluated on ResNet.

As shown in Table IV, our method *SuperPure+* takes 0.58 seconds per image, and *SuperPure* takes 0.53 seconds per image, both significantly faster than even Patch Cleanser Efficient (3.89 seconds). Additionally, our approach is substantially faster than regular Patch Cleanser (36.63 seconds) and PAD (8.8 seconds). This significant reduction in latency makes our approach **more practical** for real-world scenarios while still remaining effective against both localized and distributed physical patch attacks.

The key reason for this improvement over state-of-the-art methods lies in the design decisions detailed in Section IV-E. Specifically, compared to PatchCleanser, *SuperPure+* requires significantly fewer steps. Likewise, compared to PAD, our per-iteration analysis is much simpler, involving only a GAN-based super-resolution and straightforward downsampling.

2) GPU Memory Overhead: Table V provides the GPU memory overhead comparison for each method, excluding the classifier. Since PatchCleanser does not rely on any external model, its overhead is effectively zero. In contrast, both our method and PAD require external models, resulting in higher GPU overhead. This trade-off, however, is justified by the improved robustness and speed offered by our approach.

TABLE V
GPU MEMORY OVERHEAD

Method	GPU Memory Usage
PatchCleanser [14]	**0**
PAD [11]	5290M
SuperPure	1895M
SuperPure+	2087M

D. Comparison with PatchCURE

We also compare *SuperPure* against PatchCURE [15], a recent state-of-the-art extension of PatchCleanser that modifies model architectures (e.g., ViT-SRF) to improve *performance* and efficiency while preserving roughly the same robustness as PatchCleanser. Its pipeline first uses an SRF (small receptive field) sub-model to extract an intermediate feature map so that only part of the features is corrupted; it then applies a secure operation that typically requires multiple calls to a large receptive field (LRF) sub-model. Because these architectural changes require partial retraining, PatchCURE is not a simple plug-and-play solution and is thus omitted from our main comparisons (Section V-B). Instead, we conduct a targeted evaluation on ImageNet classification with a 32×32 patch, using the *pretrained ViT-SRF* model from PatchCURE. For our own defense, we rely on the same pretrained ViT model that underpins our broader evaluations, ensuring consistency across all tested approaches.

Method	Clean (%)	Robust (%)	Retrain?
PatchCURE [15]	72	41	Yes
SuperPure	80	75	No
SuperPure+	**83**	**79**	No

As shown in Table VI, our approaches *SuperPure* and *SuperPure+* substantially outperform PatchCURE in both clean accuracy and robustness. In particular, *SuperPure+* achieves an absolute gain of 38% in robustness and 11% in clean accuracy compared to PatchCURE. More importantly, PatchCURE still mandates partial retraining when modifying the classifier, increasing deployment complexity and overhead. By contrast, our approach is entirely plug-and-play, requiring no classifier modifications—thus highlighting the **practicality, efficiency, and scalability** of our proposed method for real-world applications.

TABLE VII
LATENCY COMPARISON ON A JETSON DEVICE FOR VIT-BASED SETUPS.

Method	Latency (s)	Repeated Classifier?
PatchCleanser [14]	>50	Yes
PatchCURE [15]	12	Partial
SuperPure	**0.67**	No (single pass)
SuperPure+	0.72	No (single pass)

Finally, Table VII compares the end-to-end latency on a Jetson device using ViT-based setups. PatchCleanser requires over 50 s, whereas PatchCURE reduces the total inference time to about 12 s. By contrast, our method needs only **0.67 s**—thanks to a single-pass purification strategy that eliminates repeated classifier calls. These results confirm that *SuperPure* exceeds PatchCURE not only in accuracy and robustness but also in latency-critical scenarios, all while maintaining a plug-and-play design with no modifications to existing classifiers.

11

VI. ABLATION STUDIES

A. Iterations to Convergence

Figure 5 demonstrates the relationship between patch size, Top-1 accuracy, and the average number of iterations to convergence for ResNet model in the contexts of *SuperPure* and *SuperPure+*.
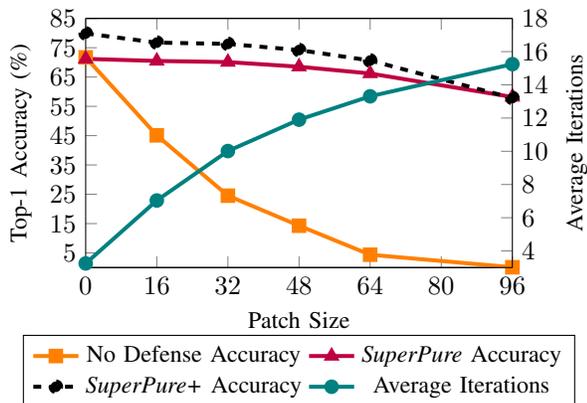

Fig. 5. Relationship between patch size, accuracy, and average number of iterations until convergence.

As seen in the figure, the average number of iterations needed for convergence increases as the patch size grows, suggesting that our method is able to **adapt to different patch sizes** for computational efficiency. In scenarios where there is no patch present, *SuperPure+* only requires around 3 iterations on average before stopping. As the patch size increases, the number of iterations rises accordingly, reflecting the greater complexity of defending against larger adversarial patches. Note that the number of iterations for EfficientNet and ResNet are slightly different because patches are generated specifically for each classifier.

B. Effect of Enhancement

An integral component of our algorithm includes the option to enable or disable a feature we refer to as "enhancement," i.e., the difference between *SuperPure* and *SuperPure+*. As described in Section IV-D, this feature involves a two-step process where the input image is first up-sampled to a higher resolution and then down-sampled back to its original dimensions. In Table I, we observe that enhancement is not only essential for defending against smaller, distributed patches but also beneficial for robustness for clean images and singular adversarial patches.

To better understand the role of enhancement, we conduct experiments where we apply this process, without iterative masking, to clean images. In addition to the ResNet and EfficientNet architectures, we evaluate three other classifiers: VGG-16 with batch normalization [57], WideResNet-50-2: [58], and ViT-B/16 [4]. The results reported in Table VIII reveal that top-1 accuracy increases by an average of approximately 10 percentage points. We can see in Figure 6 that enhancement improves the visual quality of images, with

Fig. 6. Clean images before (top) and after (bottom) enhancement.

clearer boundaries and more defined textures, which may aid the model in focusing on key features for classification.

TABLE VIII
THE IMPACT OF ENHANCEMENT ON TOP-1 CLEAN ACCURACY.

Model	Standard	Enhanced	Change
EfficientNet-B0[51]	60.76%	70.60%	+9.84%
WideResNet-50-2[58]	61.00%	74.46%	+13.46%
VGG-16 with BN[57]	47.12%	58.28%	+11.16%
ViT-B/16[4]	74.84%	84.82%	+9.98%
ResNet-152 V2[52]	71.70%	81.52%	+9.82%

C. Effect of Masking Threshold

Figure 7 illustrate the impact of the masking threshold, λ, on classifier accuracy and the average number of iterations until convergence. We observe that a low masking threshold leads to suboptimal accuracy, with the most effective range being around 0.75, although performance begins to plateau at approximately 0.6. The average number of iterations is notably higher at lower thresholds, as a lower threshold increases the likelihood of masking more pixels at each iteration. The lowest number of iterations occurs near a threshold of 0.55, but as the threshold increases beyond this point, the number of iterations rises, possibly because fewer pixels are masked per iteration. If the threshold is too high, the number of iterations drops, but accuracy also shows a slight decline.

D. Effect of Super Resolution

The primary reason for using a super-resolution model like Real-ESRGAN instead of a simple downsampling and upsampling operation is that GAN-based models aim to map the image distribution closer to that of natural images. In contrast, naive upsampling and downsampling merely perform basic averaging, which can be exploited by attackers. Specifically, an attacker can craft sufficiently smooth noise, so it remains unchanged after downsampling and upsampling. We demonstrate this in Figure 8. In this comparison, the threshold and setup remain consistent, with the only difference being that the naive method replaces the GAN-based model with simple upsampling. The results show that, unlike our approach, the naive method fails to mask the adversarial patch effectively.
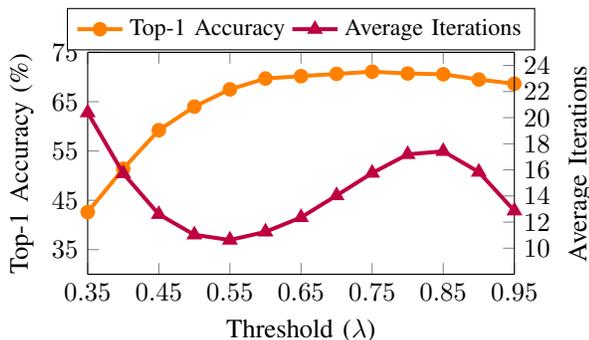
Fig. 7. Effect of changing threshold (λ) on top-1 accuracy and average iterations for ResNet and Patch Size= 64.
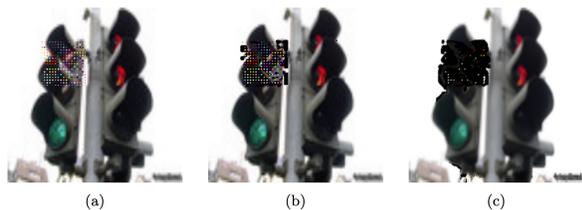

(a)   (b)   (c)

Fig. 8. (a) White-box attack with the smoothed adversarial patch. (b) Result after applying the naive defense, which fails because the adversarial patch is smoothed. (c) Output of our proposed model, successfully masking the adversarial patch.

E. Additional Results

We provide additional results for different experiments, including results on the COCO (object detection) dataset, distributed white-box patches, and alternative solutions in *Appendix A* (A1-A6).

VII. CONCLUSIONS

In this paper, we proposed a new model-agnostic defense method against both singular and distributed patch attacks. *SuperPure* utilizes discrepancies between the outputs of a reconstructed image using a super-resolution GAN and the original input to mask adversarial regions; for smaller, less perceptible patches, *SuperPure* includes an enhancement step (*SuperPure+*) to filter adversarial perturbations missed by the masking process. Through extensive experiments, we demonstrated the superior robustness of our method compared to prior work, showcasing its ability to defend effectively against a variety of patch-based adversarial attacks.

REFERENCES

[1] C. He, X. Ma, B. B. Zhu, Y. Zeng, H. Hu, X. Bai, H. Jin, and D. Zhang, "DorPatch: Distributed and occlusion-robust adversarial patch to evade certifiable defenses," in *Network and Distributed System Security Symposium, NDSS, 2024, San Diego, CA, USA, February 26 - March 1, 2024*, The Internet Society, 2024.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

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

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[7] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.

[8] D. Karmon, D. Zoran, and Y. Goldberg, "Lavan: Localized and visible adversarial noise," in *International Conference on Machine Learning*, pp. 2507–2515, 2018.

[9] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

[10] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen, "Dpatch: An adversarial patch attack on object detectors," *arXiv preprint arXiv:1806.02299*, 2018.

[11] L. Jing, R. Wang, W. Ren, X. Dong, and C. Zou, "Pad: Patch-agnostic defense against adversarial patch attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24472–24481, 2024.

[12] P.-y. Chiang, R. Ni, A. Abdelkader, C. Zhu, C. Studer, and T. Goldstein, "Certified defenses for adversarial patches," in *8th International Conference on Learning Representations (ICLR 2020)(virtual)*, International Conference on Learning Representations, 2020.

[13] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "{PatchGuard}: A provably robust defense against adversarial patches via small receptive fields and masking," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2237–2254, 2021.

[14] C. Xiang, S. Mahloujifar, and P. Mittal, "{PatchCleanser}: Certifiably robust defense against adversarial patches for any image classifier," in *31st USENIX Security Symposium (USENIX Security 22)*, pp. 2065–2082, 2022.

[15] C. Xiang, T. Wu, S. Dai, J. Petit, S. Jana, and P. Mittal, "{PatchCURE}: Improving certifiable robustness, model utility, and computation efficiency of adversarial patch defenses," in *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 3675–3692, 2024.

[16] C. Mao, Y. Zhu, N. Z. Gong, and X. Zhang, "Defending against adversarial attacks by randomized diversification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1313–1327, 2022.

[17] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed System Security Symposium (NDSS)*, Internet Society, 2018.

[18] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[19] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high-frequency functions in low-dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[20] A. Torralba and A. Oliva, "Statistics of natural image categories," *Network: Computation in Neural Systems*, vol. 14, no. 3, pp. 391–412, 2003.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, IEEE, 2009.

[22] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, Ieee, 2017.

[23] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine learning*, vol. 81, pp. 121–148, 2010.

[24] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, pp. 387–402, Springer, 2013.

[25] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*, pp. 372–387, IEEE, 2016.

[26] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *International Conference on Machine Learning*, pp. 2196–2205, PMLR, 2020.

[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[28] J. Hayes, "On visible adversarial perturbations & digital watermarking," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1597–1604, 2018.

[29] X. Wei, Y. Guo, and J. Yu, "Adversarial sticker: A stealthy attack method in the physical world," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 2711–2725, 2022.

[30] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. Yuille, "Patchattack: A black-box texture-based attack with reinforcement learning," in *European Conference on Computer Vision*, pp. 681–698, Springer, 2020.

[31] M. Lee and Z. Kolter, "On physical adversarial patches for object detection," *arXiv preprint arXiv:1906.11897*, 2019.

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

[33] Z. Zhang, B. Yuan, M. McCoyd, and D. Wagner, "Clipped bagnet: Defending against sticker attacks with clipped bag-of-features," in *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 55–61, IEEE, 2020.

[34] E. Chou, F. Tramer, and G. Pellegrino, "Sentinet: Detecting localized universal attacks against deep learning systems," in *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 48–54, IEEE, 2020.

[35] B. Tarchoun, A. Ben Khalifa, M. A. Mahjoub, N. Abu-Ghazaleh, and I. Alouani, "Jedi: Entropy-based localization and removal of adversarial patches," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4087–4095, 2023.

[36] K. Xu, Y. Xiao, Z. Zheng, K. Cai, and R. Nevatia, "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, 2023.

[37] S. Rao, D. Stutz, and B. Schiele, "Adversarial training against location-optimized adversarial patches," in *European conference on computer vision*, pp. 429–448, Springer, 2020.

[38] T. Wu, L. Tong, and Y. Vorobeychik, "Defending against physically realizable attacks on image classification," in *International Conference on Learning Representations*, 2020.

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

[40] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*, pp. 184–199, Springer, 2014.

[41] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1646–1654, 2016.

[42] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4681–4690, 2017.

[43] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European Conference on Computer Vision Workshops*, pp. 63–79, 2018.

[44] A. Mustafa, S. H. Khan, M. Hayat, J. Shen, and L. Shao, "Image super-resolution as a defense against adversarial attacks," *IEEE Transactions on Image Processing*, vol. 29, pp. 1711–1724, 2019.

[45] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar, "Diffusion models for adversarial purification," *arXiv preprint arXiv:2205.07460*, 2022.

[46] A. Youssef, "Analysis and comparison of various image downsampling and upsampling methods," in *Proceedings DCC'98 Data Compression Conference (Cat. No. 98TB100225)*, p. 583, IEEE, 1998.

[47] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[48] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, 2018.

[49] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-esrgan: Training real-world blind super-resolution with pure synthetic data," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1905–1914, 2021.

[50] W. Nie, B. Chen, A. Anandkumar, and J. Huang, "Diffusion models for adversarial purification," in *Advances in Neural Information Processing Systems*, 2022.

[51] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645, Springer, 2016.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[54] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 126–135, 2017.

[55] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 114–125, 2017.

[56] X. Wang, K. Yu, C. Dong, and C. C. Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 606–615, 2018.

[57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[58] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference 2016*, British Machine Vision Association, 2016.

[59] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

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

[61] Y. Chen, J. Zhao, and ..., "Dpatch: An adversarial patch attack on object detectors," in *CVPR*, 2022.

[62] "Adversarial robustness toolbox (art)," 2023. Available at: https://adversarial-robustness-toolbox.readthedocs.io/.

[63] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[64] J. Ho, C. Saharia, and T. Salimans, "Image super-resolution via iterative refinement (sr3)," *arXiv preprint arXiv:2104.07636*, 2021.

APPENDIX

A. COCO & DPatch Attack

We evaluate *SuperPure* on the **COCO** dataset [59] using **Faster R-CNN** [60] for object detection, under a **DPatch** threat [61]. DPatch stands for *"An Adversarial Patch Attack on Object Detectors"* and we generated the patch using the framework provided by the Adversarial Robustness Toolbox [62]. Specifically, the attack inserts a 96×96 adversarial region designed to disrupt detection performance.

1) Experimental Setup: We do not retrain or modify the detector; instead, we apply *SuperPure* to each adversarial image, then feed the purified output into the original Faster R-CNN. This underscores *SuperPure*'s *plug-and-play* capability, as no task-specific retraining is required.

2) Results and Observations: Table IX summarizes the micro-precision on randomly selected COCO pictures. Clean detection accuracy of **60%** plunges to **35%** under DPatch, but *SuperPure* restores it to **58%**, indicating robust generalization beyond classification tasks.

Model	Clean	Attack	After *SuperPure*
Faster R-CNN	60%	35%	58%

Figure 9 shows an example COCO image before and after purification. On the left (**a**), the original image with a 96×96 DPatch is shown. On the right (**b**), *SuperPure* effectively neutralizes the patch while minimally affecting the rest of the scene. These results confirm that *SuperPure* successfully purifies adversarial patches on COCO and remains applicable to diverse vision tasks.

(a)   (b)

Fig. 9. Visual results on COCO: (**a**) original adversarial image (DPatch), (**b**) purified by *SuperPure*. The 96×96 patch is successfully mitigated.

B. Distributed White-Box Patches vs. PatchCleanser

In this section, we explore *distributed* adversarial patches where multiple 32×32 regions are placed throughout the image. Crucially, the attacker has **white-box** knowledge of our *SuperPure* method, specifically crafting these patches to exploit *SuperPure*'s iterative masking. We also evaluate PatchCleanser [14] under the same multi-patch distribution for comparison, even though PatchCleanser itself does not operate in a white-box mode.

1) Experimental Setup: For each experiment, we increment the number of 32×32 patches scattered across the image. The total adversarial area thus becomes increasingly fragmented, posing a stronger challenge. While *SuperPure* faces a white-box attacker, PatchCleanser is tested as-is. This setup highlights the difference in how each defense copes with multiple small patches.

2) Results and Observations: Figure 10 plots the robust accuracy as the number of distributed patches increases. Despite the attacker's full knowledge of *SuperPure*, our method
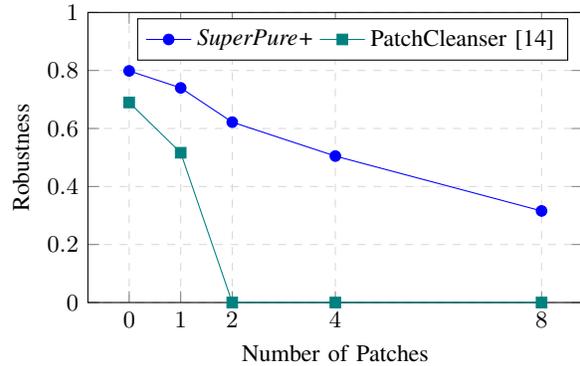

Fig. 10. Comparison of the impact of increasing the number of patches in a white-box attack between our method and Patch Cleanser. Each patch measures 32×32, with a (low) noise level of 8/255.

preserves high robustness. In contrast, **PatchCleanser** [14] degrades rapidly as more patches are introduced.

a) Discussion.: These experiments confirm:

- **Iterative Masking Under White-Box Attacks.** Even when the attacker tailors multiple patches with inside knowledge of *SuperPure*, its repeated non-linear reconstructions still hinder patch survival.
- **PatchCleanser Limitations.** Although PatchCleanser performs well against fewer/larger patches, it fails to maintain robustness when faced with many distributed patches.

Hence, *SuperPure* outperforms PatchCleanser in complex, high-fragmentation adversarial scenarios.

C. Evaluation on CIFAR-10 & CIFAR-100

We additionally tested *SuperPure* on the **CIFAR-10** and **CIFAR-100** datasets [63] to assess its generalization to smaller-resolution images. We employed a ResNet-18 [3] model initially pretrained on ImageNet [21] and then fine-tuned for each CIFAR dataset. Since we use a 32×32 adversarial patch, we *upscaled* each 32×32 CIFAR image to 256×256 so that the patch would not occupy the entire image, thus creating a realistic test scenario for our iterative defense.

1) Experimental Setup and Preliminary Results: Table X shows the accuracy on clean CIFAR images, the accuracy under the 32×32 patch attack, and the recovered accuracy after applying *SuperPure*. Despite the aggressive upscaling, *SuperPure* substantially mitigates adversarial damage, suggesting that its iterative down-up masking pipeline is *dataset-agnostic* and does not rely on large native resolutions.

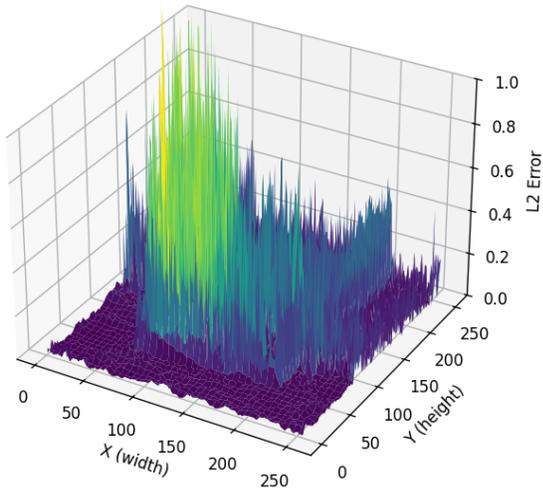| Dataset	Clean	Attack	After *SuperPure*
CIFAR-10	94.60%	2.38%	85.78%
CIFAR-100	80.09%	2.06%	62.26%

Fig. 11. Reconstruction error across adversarial patch regions versus non-patch regions, illustrating that patched pixels exhibit significantly higher mean squared error (MSE) post-GAN upsampling.

Patch Size	*SuperPure* Acc.	Reverse Acc.
0×0	71.20%	69.66%
16×16	70.48%	69.90%
32×32	70.10%	69.52%
48×48	68.52%	68.30%
64×64	66.20%	65.90%
96×96	58.18%	58.66%

differences are modest for all patch sizes, confirming that the similar mechanisms of the two pipelines (as outlined in Section IV-E) translate to similar performance in terms of robustness.

F. Reconstruction Error Visualization

In Figure 11, we illustrate an example image where the adversarial patch region shows distinctly higher reconstruction error than non-patch areas after GAN upsampling.

These preliminary findings highlight *SuperPure*'s resilience against patch-based adversarial threats, even for comparatively small datasets.

D. Alternative SR: Diffusion-Based Models

In the main paper, we selected Real-ESRGAN for super-resolution due to its relatively fast inference, which is critical for our *iterative* down-up cycles. However, recent diffusion-based SR approaches, such as **SR3** [64], can sometimes yield higher-quality reconstructions. We briefly experimented with SR3 to explore this trade-off.

1) Latency vs. Robustness Trade-off: Our tests show that while SR3 can improve image fidelity slightly, it is *significantly slower*. On a single image, SR3 may take *several seconds*, making multiple passes impractical. By contrast, Real-ESRGAN performs sufficiently fast to allow repeated down-up cycles. Moreover, we observed *only minor differences* in overall adversarial robustness between SR3 and Real-ESRGAN, reaffirming that *any sufficiently non-linear SR* method can disrupt patch artifacts effectively, as long as it shifts images closer to the *natural* manifold.

a) Conclusion.: If speed is not a concern, a diffusion-based SR might enhance the final visual quality slightly more. However, for our repeated masking design, a lightweight and relatively fast SR generator remains more suitable for real-time or large-scale deployment.

E. Effect of Rescaling Order

Table XI presents the effect of reversing the rescaling order (i.e., down-up vs. up-down) during the masking phase on accuracy, evaluated at various patch sizes. Specifically, the table compares the accuracy (*SuperPure* Acc.) of *SuperPure*, which downsamples the image before super-resolution in the masking phase, against a reversed approach that upsamples first (Reverse Acc.). The results show that the performance