Patch Replacement: A Transformation-based Method to Improve Robustness against Adversarial Attacks

Hanwei Zhang
Inria, CNRS, IRISA, Univ Rennes
East China Normal University

Yannis Avrithis
Inria, CNRS, IRISA, Univ Rennes

Teddy Furon
Inria, CNRS, IRISA, Univ Rennes

Laurent Amsaleg
Inria, CNRS, IRISA, Univ Rennes

ABSTRACT

Deep Neural Networks (DNNs) are robust against intra-class variability of images, pose variations and random noise, but vulnerable to imperceptible *adversarial* perturbations that are well-crafted precisely to mislead. While random noise even of relatively large magnitude can hardly affect predictions, adversarial perturbations of very small magnitude can make a classifier fail completely.

To enhance robustness, we introduce a new adversarial defense called *patch replacement*, which transforms both the input images and their intermediate features at early layers to make adversarial perturbations behave similarly to random noise. We decompose images/features into small patches and quantize them according to a codebook learned from legitimate training images. This maintains the semantic information of legitimate images, while removing as much as possible the effect of adversarial perturbations.

Experiments show that patch replacement improves robustness against both white-box and gray-box attacks, compared with other transformation-based defenses. It has a low computational cost since it does not need training or fine-tuning the network. Importantly, in the white-box scenario, it increases the robustness, while other transformation-based defenses do not.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Security and privacy** → **Intrusion detection systems**.

KEYWORDS

Adversarial robustness, transformation, product quantization

ACM Reference Format:
Hanwei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. 2021. Patch Replacement: A Transformation-based Method to Improve Robustness against Adversarial Attacks. In *Proceedings of the 1st International Workshop on Trustworthy AI for Multimedia Computing (Trustworthy AI '21), October 24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3475731.3484955

Trustworthy AI '21, October 24, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8674-6/21/10…$15.00
https://doi.org/10.1145/3475731.3484955

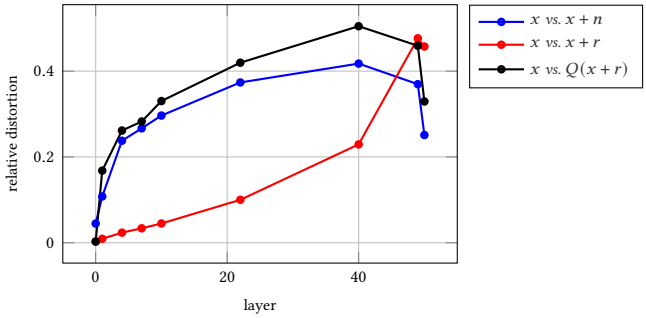


Figure 1: *Making adversarial perturbation behave like noise.* Relative distortion of random noise, adversarial perturbation and our patch replacement on input images (layer 0) *vs.* ResNet-50 layer, averaged over the ImageNet test set. Random noise follows a normal distribution and adversarial perturbations are generated by Decoupling Direction and Norm (DDN) attack [28]. x: original input image; n: random noise; r: adversarial perturbation; Q: our patch replacement quantizer.

1 INTRODUCTION

Adversarial perturbations are modifications of small magnitude on images, almost imperceptible to human eyes, which lead classifiers to make erroneous predictions [31]. They reveal the vulnerability of Deep Neural Networks (DNNs) and the potential danger in machine learning-based applications, for instance, traffic sign detection in self-driving cars.

Based on different assumptions on adversarial perturbations, different defenses are proposed to address the security problem and enhance the robustness of networks against adversarial attacks. Considering the lack of corresponding data, *adversarial training* [6, 19] includes adversarial images as part of training data. It improves the robustness against adversarial attacks, but decreases the accuracy of legitimate images and is very expensive to train.

Several works improve the architecture of networks, assuming that attackers take advantage of weaknesses in model design [3, 6, 11, 18, 19, 22, 25, 32]. Instead of using the image labels, distillation defenses [11, 22] train a small network on the probability vectors predicted by a large network. To reduce the sensitivity of networks to their input, one may constrain *e.g.* each layer to be a Lipschitz

function [7, 33]. However, these defenses are either expensive to train or not practical due to their mathematical hypotheses.

Considering adversarial perturbations as a special kind of noise is a simple and practical perspective. Motivated by this, a number of defenses detect the adversarial image and/or pre-process input images to remove the effect of the adversarial perturbation [8, 16, 17, 24, 36]. For instance, MagNet [20] uses auto-encoders to project the input image to the manifold of natural images. However, these approaches are complicated to train, attack-specific or vulnerable in white-box settings.

It is well-known that DNNs, such as AlexNet [14], Inception [30] and ResNet-50 [10], are robust to random noise and transformations such as cropping, reshaping, and rotation but vulnerable to adversarial perturbations, even when the are of much smaller magnitude than random noise. By considering adversarial perturbations as a special kind of noise, a fundamental question arises:

> *What is the difference between random noise and adversarial perturbations?*

To answer this question, we consider images with random noise and adversarial perturbations as inputs to a ResNet-50 classifier. For adversarial perturbations, we use the DDN attack [28], which is known for its very low distortion. The network generalizes to random noise despite an accuracy drop of around 3%, but is completely deceived by adversarial perturbations. We measure the magnitude of the noise or perturbations relative to intermediate features and observe its evolution through network layers. In particular, given input x and noise/perturbation r, we measure the relative distortion in layer L as $\|f_L(x+r) - f_L(x)\| / \|f_L(x)\|$, where $f_L(\cdot)$ is the part of network from the input to layer L and layer 0 is the input.

As shown in Figure 1, the relative distortion of random noise increases fast at the beginning because max pooling amplifies the distortion by taking the maximum values locally, and decreases at the end because average pooling weakens the effect of noise. By contrast, the relative distortion of adversarial perturbations increases slowly at the beginning and faster at the end.

In image space, random noise is of much larger relative distortion than adversarial perturbations (0.044 *vs.* 0.003). However, the situation is reversed at the logit layer (0.251 *vs.* 0.457). We speculate that adversarial perturbations at the logit layer focus on a particular class, resulting in misclassification. We confirm this by measuring the entropy of the output distribution as 0.76 on clean inputs, 0.73 on images with random noise but 0.69 on adversarial images.

Based on these observations, we propose to transform inputs and features such that adversarial perturbations behave more like random noise. To do so, we split them into patches, both spatially and over channel dimensions, and replace them with nearest neighbors in a codebook learned from clean training data. This introduces quantization noise that reduces the effect of adversarial perturbations, while having little effect on legitimate images. Codebook learning is independent of network training and patch replacement is efficient at inference time.

We find that it is most effective to apply patch replacement to the early layers of the network, including the input image. By doing so, we achieve indeed a behavior similar to random noise, even when the input is adversarial, as shown in Figure 1. This improves significantly the robustness against adversarial attacks, while slightly reducing the accuracy on legitimate images.

Contributions Our contributions can be summarized as follows:

- we introduce a relatively simple and efficient defense, without network training or using any attack, which can be easily adapted to networks of different architecture;
- we investigate experimentally the impact of patch replacement in different layers and the influence of codebook quality;
- we apply patch replacement on images as well as features at different layers, finding a good trade-off between accuracy and robustness;
- we outperform other transformation-based defenses in both gray-box and white-box settings; and
- we achieve better accuracy than adversarial training and even better robustness under attacks of low distortion.

The rest of the paper is organized as follows. Section 2 briefly recaps defenses against adversarial perturbations. Section 3 presents our patch replacement defense, and section 4 provides experimental analysis and comparisons. Conclusions are drawn in section 5.

2 RELATED WORK

Existing defenses are either *reactive* [8, 16, 17, 24, 36], *i.e.* adding an extra element to detect or remove adversarial perturbation, or *proactive* [3, 6, 11, 18, 19, 22, 25, 32], *i.e.* making the network intrinsically robust against adversarial attacks. Reactive methods are easy to compute and adapt to different networks but vulnerable in white-box settings, while proactive methods are more robust but expensive and hard to integrate into a new model since they train the network, either from scratch or by fine-tuning.

Transformation-based defenses [8, 24, 29, 34] are reactive defenses that attempt to reform adversarial examples while not changing their semantics. Basic transformations, such as cropping, rescaling, bit-depth reduction, jpeg compression, total variance minimization, and image quilting, succeed in removing adversarial effects to some extent [8]. Inspired by this result, feature squeezing [36] detects adversarial perturbations by comparing features of given inputs and their filtered versions. However, these defenses fail to defend against strong attacks [2].

To make the transformation-based defenses more robust, *pixel deflection* [23] redistributes pixels according to a robust activation map generated by *Class Activation Maps (CAM)* [39] and softens the introduced noise and adversarial perturbations by a subsequent wavelet-based denoising operation. Inspired by pixel deflection, *CIIDefence* [9] reconstructs the small and carefully selected image areas that are most influential to the current prediction according to the class activation map obtained for multiple top-ranking class labels. These works achieve good performance in gray-box settings but are still relatively vulnerable in white-box settings.

Another class of transformation-based methods attempts to map inputs to a latent space, such that legitimate images and their adversarial versions share the same representations [13, 21, 27, 29]. These defenses are normally more expensive than other transformation-based methods, but require less computation cost than proactive methods. For instance, *Divide, Denoise and Defend (D3)* [21] encodes

the input according to multiple sparse dictionaries for different sparsity levels. It divides the input into multiple patches and denoises each one with sparse reconstruction. It builds a set of dictionaries greedily by selecting important and diverse patches.

Adversarial perturbations are amplified through the layers of a network and introduce noise in otherwise flat areas of their features. Based on this observation, a new architecture design, *i.e. denoising block* [35], reduces their effect by feature denoising. *BlurNet* [26] proposes to remove high frequencies via a depthwise convolution layer of standard blur kernels after the first layer and is effective against the Robust Physical Perturbations (RP_2) [5] attack. Asadi et al. [1] propose a method based on whitening coloring transform to diminish the misrepresentation of any desirable layer caused by adversaries. These works indicate the importance of intermediate features in augmenting robustness.

Inspired by these approaches, we introduce *patch replacement*, a transformation-based method that removes the adversarial effect from both input images and features. Patch replacement shares a similar principle with D3 [21]. However, to our knowledge, it is the first transformation-based defense on both images and intermediate features, achieving a good trade-off between accuracy and robustness. Also, as a variant of matching pursuit, D3 is expensive. Our approach is more efficient, both at learning (by k-means) and at inference (by directly quantizing). Since the code of D3 is not published, we cannot compare to it experimentally.

3 METHODOLOGY

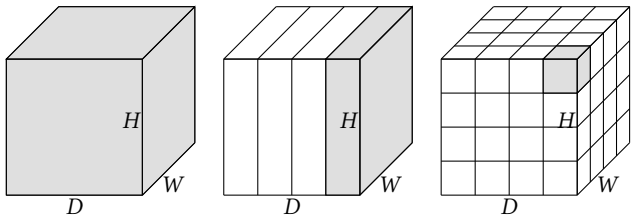
Patch replacement is a transformation-based defense against adversarial perturbations. It reduces the adversarial effect not only from images but also from feature maps, *i.e.* intermediate representations of convolutional, pooling, and fully connected layers. We first decompose inputs and feature maps into patches and replace them with their nearest neighbor according to a codebook learned on training data. To understand the approach, we first discuss preliminary concepts of *features*, *slices*, and *patches*. We then explain how we build the codebook and introduce a number of replacement strategies to limit the loss of information incurred by quantization.

3.1 Preliminaries

A Convolutional Neural Network (CNN) processes images into a sequence of *feature maps* obtained by learnable convolutional layers. As shown in Figure 2(a), a feature map has depth D, the number of filter channels, as well as height H and width W, which depend on the input size and the stride and padding of convolutional layers. We denote the feature map of layer L by tensor $F := f_L(x) \in \mathbb{R}^{W \times H \times D}$, where x is the input image, f_L denotes the part of network from the input to layer L and layer 0 is the input ($f_L(x) = x$).

To combat the curse of dimensionality, we first decompose features into *slices*. That is, a feature is represented as a concatenation of slices over filter channels, *i.e.* $F = [F_1, F_2, \cdots, F_m]$ where F_k denotes a slice. As shown in Figure 2(b), slice $F_k := F^{(k-1)d+1:kd} \in \mathbb{R}^{W \times H \times d}$ contains channels $(k-1)d+1$ to kd of feature map F, where d is the depth of the slice and $d \times m = D$.

Each slice F_k is then decomposed into sub-tensors with same depth but smaller width and height over spacial locations. Each sub-tensor is called a *patch* P_{ijk} where i, j denote the horizontal and vertical location and k denotes the slice. More precisely, patch $P_{ijk} := F_k(i-a : i+a, j-a : j+a)$ is a sub-tensor of size $(2a+1) \times (2a+1) \times d$, centered at location (i, j) of slice F_k.



(a) Feature map (b) Slice (c) Patch

Figure 2: (a) Feature map with width W, height H and depth D. (b) We decompose the feature map into slices along depth (over channels). (c) For each slice, we decompose the feature map into patches over spatial locations.

Patches can be sampled densely or sparsely. In the sparse case, patch centers are sampled on a spatial grid with cell size c. When $c = (2a+1)$, there is no overlapping among patches, as shown in Figure 2(c); while for $c < (2a+1)$, patches overlap.

3.2 Codebook

To be able to replace patches, we learn a *codebook* from patches of training data. For each slice F_k, we learn a quantizer q_k, whose objective is to quantize patches obtained from this particular slice. Each quantizer q_k has its own codebook C_k with K codewords, learned by K-means. At inference, we decompose the feature map of a test image into slices and patches and use the corresponding quantizer to find their nearest neighbors in the training data and replace them.

The set of codebooks C_k for all slices F_k can be seen as a codebook $C := C_1 \times \cdots \times C_m$ according to *product quantization* (PQ) [12]. PQ allows a codebook size that is exponential in the number m of slices, while both training and inference are linear in m. The number K of centroids per slice and the depth d of slices control the quality of the codebook C. To maintain classification accuracy of legitimate images, we need a fine codebook; whereas, to remove the effects of adversarial perturbations, we need a coarse codebook. To handle this trade-off, we introduce replacement strategies as follows.

3.3 Replacement Strategies

When the codebook is coarse, quantization incurs a significant loss of information. To limit the loss, we introduce a number of *quantization strategies*. Those are functions of a given quantizer that are continuous in a given parameter. For the sake of simplicity, we denote a general patch (in any spatial location or slice) as P and a general quantizer (in any slice) as q in this subsection. The nearest patch of P is then denoted as $q(P)$.

Plain strategy As shown in Figure 3(a), the baseline strategy refers to directly replacing patches by their nearest codewords, *i.e.* $P' = q(P)$.

L_2 strategy As shown in Figure 3(b), we limit quantization of P within the L_2 ball of radius $\epsilon > 0$ centered at P. If the nearest

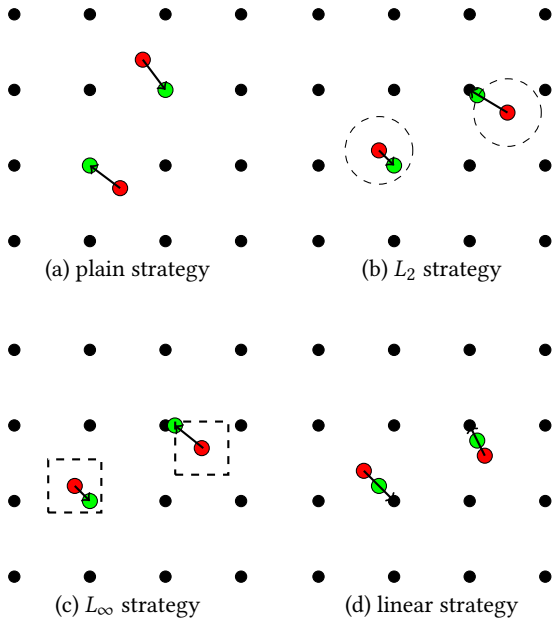


Figure 3: Visualization of the four replacement strategies. Black: codewords; red: original patches; green: replaced patches. In (b), dashed circles indicate L_2 balls with radius ϵ. In (c), the dashed boxes indicate L_∞ balls with radius ϵ.

codeword is inside this L_2 ball, we replace P with $q(P)$; otherwise, we project $q(P)$ on the ball:

$$q^{L_2}(P) := \begin{cases} q(P) & \|q(P) - P\| < \epsilon \\ P + \epsilon \mathsf{n}(q(P) - P) & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathsf{n}(P) := P/\|P\|$ for any patch P.

L_∞ strategy As shown in Figure 3(c), we limit quantization of P within the L_∞ ball of radius $\epsilon > 0$ centered at P. If the nearest codeword is inside this L_∞ ball, we replace P with $q(P)$, otherwise we project $q(P)$ on the ball (clip element-wise):

$$q^{L_\infty}(P) := P + \mathrm{clip}_{[-\epsilon,\epsilon]}(q(P) - P). \tag{2}$$

Linear strategy As shown in Figure 3(d), we use a linear interpolation between the original patch P and its nearest codeword $q(P)$:

$$q^{\mathrm{lin}}(P) := P + \lambda(q(P) - P), \tag{3}$$

where $\lambda \in [0, 1]$.

L_2 and L_∞ strategies limit the distortion within ϵ but according to different norms. When $\epsilon = 0$, the patch P is not replaced. When ϵ is large enough, both strategies are equivalent to the plain strategy.

Instead of setting a limit on the distortion, the linear strategy interpolates between the original patch P and its nearest codeword $q(P)$ with interpolation factor λ. When $\lambda = 0$, the patch is not replaced. When $\lambda = 1$, the linear strategy is equivalent to the plain strategy.

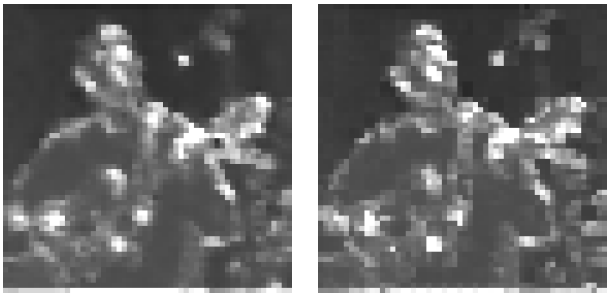

(a) with overlap (b) without overlap

Figure 4: Reconstruction of first layer features with overlap ($c < 2a + 1$) and without overlap ($c = 2a + 1$).

3.4 Reconstruction

After quantizing the patches P into P' according to a replacement strategy, we reconstruct a feature map F' from these patches. For every slice, we concatenate all patches P' over spatial locations, applying linear interpolation if patches are overlapping. We then concatenate all slices over channels, *i.e.* $F' = [F'_1, F'_2, \cdots, F'_m]$. As a whole, we denote the patch replacement operation, including quantization and reconstruction, as $F' = Q(F)$.

We then feed the reconstructed feature map F' through the remaining part of the network to obtain a prediction for the original input. Since the reconstructed feature F' consists of the nearest patches in the learned codebook, F' preserves the semantic information of the original input, while reducing the adversarial effect.

When we reconstruct features, the stride c determines whether patches are overlapping or not. The example of Figure 4 shows that with overlap ($c < 2a + 1$), the reconstructed features $\mathcal{F}'$ are smoother than without ($c = 2a+1$). We prefer the smoother features since it works slightly better in all experiments.

3.5 Multi-layer patch replacement

Depending on the size of images and feature maps, codebooks with similar parameters on different layers have different qualities. To further investigate the trade-off between accuracy and robustness, we propose to apply patch replacement on multiple layers. We first apply patch replacement on a chosen layer, then reconstruct the new feature map, feed it to the network and apply patch replacement on another layer.

On one hand, patch replacement on multiple layers provides more space to search for the optimal trade-off, allowing to progressively remove the adversarial effect at multiple layers. On the other hand, it increases the complexity as a defense, so that it is more difficult to be attacked.

4 EXPERIMENTS

We evaluate our method *patch replacement* (PR) and compare it to existing defenses under the gray-box setting and white-box setting.

4.1 Experimental setup

Dataset We use the ImageNet [4] dataset. We randomly sample 50,000 images (50 per class) from the training set to learn a codebook and 1,000 images (one per class) from the validation set for testing.

Networks All experiments are carried out on PyTorch[1]. We use the pre-trained ResNet-50 [10] from PyTorch-Torchvision models[2], whose accuracy is 75.7% on the test set. As robust network, we take ResNet-50 as pre-trained by adversarial training[3] on adversarial examples generated by Projected Gradient Descent (PGD) attack with L_∞ upper bound $\epsilon = 8$ [19].

Attacks For the gray-box setting, we employ the *target success* attack DDN [28] with 20 iterations, which achieves a 0.999 success rate on the test set against ResNet-50, and the *target distortion* attacks Fast Gradient Sign Method (FGSM) [6], PGD [19], and Basic Iterative Method (BIM) [15] with 20 iterations for PGD and BIM. The implementation of DDN [28] is from its authors[4] and the implementation of FGSM, PGD, and BIM is from foolbox[5].

For the white-box setting, we use *Backward Pass Differentiable Approximation (BPDA)* [2][6], a smart attack for transformation-based defenses, using 20 iterations. It is also a target distortion attack that is equivalent to PGD when there is no defense.

The distortion bound ϵ for target distortion attacks is discussed with evaluation metrics below.

Competitors We compare patch replacement to other defense methods, including adversarial training and other transformation-based defenses. For *adversarial training*, we use ResNet-50 pre-trained with PGD [19] as discussed above. As transformation-based defenses, we use bit-depth reduction to 3 bits and 5 bits, denoted as bit3 and bit5 [8]; median smoothing filter with a kernel size of two and three, denoted as ms2 and ms3 [36]; *pixel deflection* [23] with CAM as a robust activation map[7].

Evaluation metrics In ablation, we evaluate patch replacement by accuracy on both legitimate images (*original accuracy*) and adversarial images (*adversarial accuracy*). At testing, we also use success rate and distortion. For accuracy and distortion, higher is better; for success rate, lower is better.

Given a test set of N' images, we only consider its subset X of N images that are classified correctly without attack. The accuracy of the classifier on legitimate images is thus N/N'. Let X_{suc} be the subset of X with $N_{\text{suc}} := |X_{\text{suc}}|$ where the attack succeeds and let $D(\mathbf{x}) := \|\mathbf{x} - \mathbf{y}\|$ be the distortion for image $\mathbf{x} \in X_{\text{suc}}$, where $\mathbf{y}$ is the closest adversarial example the attack succeeds to forge. The global statistics are the *success probability* (*rate*) P_{suc} and *conditional average distortion* $\overline{D}$

$$P_{\text{suc}} := \frac{N_{\text{suc}}}{N}, \quad \overline{D} := \frac{1}{N_{\text{suc}}} \sum_{\mathbf{x} \in X_{\text{suc}}} D(\mathbf{x}). \tag{4}$$

[1]We use PyTorch1.4.0-py3.7 with CUDA 10.0.130.
[2]https://github.com/Cadene/pretrained-models.pytorch
[3]https://github.com/MadryLab/robustness
[4]https://github.com/jeromerony/fast_adversarial
[5]https://github.com/bethgelab/foolbox
[6]https://github.com/Annonymous-repos/attacks-in-pytorch
[7]https://github.com/iamaaditya/pixel-deflection

LAYER	d	D	K
image	1	3	392, 785, 3927, 6284, 7855, 11783, 15711, 19639, 26185, 31422, 39278, 58917, 78557, 117835, 157114, 235671
layer 1	4	64	39278, 78557, 117836, 157114, 235671, 314228, 785568
	8	64	39278, 78557, 117836, 157114, 261856, 392784, 785568
layer 4	2	256	78557
	4	256	78557, 785568
	8	256	78557
layer 7	2	256	78557
	4	256	78557
	8	256	78557, 785568
layer 10	2	256	78557, 785568
	4	256	78557, 785568
	8	256	78557, 785568

Table 1: The list of the number of clusters K and slice depth d for different layers tested for codebooks. Depth D is the number of filter channels on the given layer.

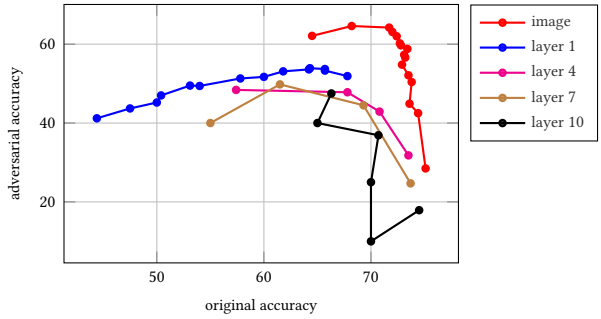


Figure 5: Effect of codebook quality, as controlled by K (number of clusters) and d (slice depth), applying patch replacement on different layers independently, using the plain strategy. We plot the accuracy in the same curve for varying K, d per layer, ranking the points according to original accuracy.

Here, $\overline{D}$ is conditioned on success. Indeed, distortion makes no sense for a failure. For target success attacks, the attack is only performed once per image, which yields success/failure and a distortion result. For target distortion attacks, we use a fixed distortion bound $\epsilon = 0.03$ when measuring adversarial accuracy and a set of values[8] for ϵ when measuring success rate and distortion. In the latter case, we increase ϵ until the attack succeeds, yielding a distortion measurement, or the attack fails for all ϵ. We also use *operating characteristics* [37, 38], which provide a more detailed picture of this process.

4.2 Ablation study: single layer

Codebook quality When applying patch replacement to a single layer, we evaluate the effect of the quality of the codebook and the layer where we apply patch replacement.

[8]$\epsilon \in \{0.0005, 0.001, 0.003, 0.005, 0.01, 0.03, 0.05, 0.08, 0.1\}$

The quantizers of all slices are trained independently. The quality of the codebook depends on the number of clusters K for K-means and the depth d of a slice. A larger K or a smaller d results in a finer codebook, preserving more information but doing little against an attack. We vary K from 392 to 785, 568 and d from 1 to 8. A compete list of values for K and d we have explored in different layers is given in Table 1.

In Figure 5, points on the left have a coarse codebook (small K / large d) and points on the bottom right have a fine codebook (large K / small d). A coarse codebook improves the adversarial accuracy but drops the original accuracy. By contrast, a fine codebook maintains original accuracy but is of little help for adversarial accuracy.

We further observe from Figure 5 that the deeper layer we choose, the worse performance we get overall (trade-off between original and adversarial accuracy). This can be attributed to the fact that the effect of the adversarial perturbation is amplified in deeper layers, as shown in Figure 1. It is thus best to apply the defense as early as possible. The best single-layer setting is in the top right corner of Figure 5, *i.e.* $K = 3927, d = 1$ on images (layer 0). This gives original accuracy 71.7% (4% loss compared with baseline) and adversarial accuracy 64.2% (64.1% gain compared with baseline). For the first layer, the best setting is $K = 235671, d = 4$, giving original accuracy 65.7% and adversarial accuracy 53.7%.

Effect through network layers By selecting the best setting, *i.e.* $K = 3927, d = 1$ on images, we study the effect or random noise and adversarial perturbation with and without patch replacement through the network layers. Figure 6 is an extension of Figure 1 and it confirms that our patch replacement changes the behavior of the adversarial perturbation through the network and makes it similar to random noise: the quantized original input $Q(x)$ and the quantized adversarial input $Q(x + r)$ (green and black line, respectively) are near identical and similar to random noise (blue line) when compared to the input x and give small distortion when compared to each other (brown line).

Replacement strategy When applying patch replacement to either the image or the first network layer and fixing the codebook quality, we evaluate the effect of the replacement strategy. We choose $K = 3927, d = 1$ in the image layer and $K = 235671, d = 4$ in the first layer, as discussed in the previous paragraph.

We control the replacement strategies by parameters ϵ or λ. We let $\epsilon, \lambda \in \{0, 0.1, \ldots, 0.9, 1\}$ for L_2 and linear strategies. For L_∞ strategy, we let $\epsilon \in \{0, 0.01, \ldots, 0.09, 0.1\}$ on the image layer and $\epsilon \in \{0, 0.01, \ldots, 0.19, 0.2\}$ on the first layer.

In Figure 7, points in the bottom right corner correspond to $\epsilon = 0$ or $\lambda = 0$. In this case, any replacement strategy other than plain strategy is equivalent to the original network (no defense). Points in the top left corner correspond to the maximum value of ϵ for L_2 strategy ($\epsilon = 1$) and L_∞ ($\epsilon = 0.1$), and $\lambda = 1$ for the linear strategy. These cases are equivalent to the plain strategy.

As shown in Figure 7, the behavior of all strategies is similar and there is no clear winner. In general, points in the top right corner are more interesting. These points correspond to strategies improving the original accuracy comparing to the plain strategy, while maintaining the adversarial accuracy or even improving it slightly. We highlight three particular settings:

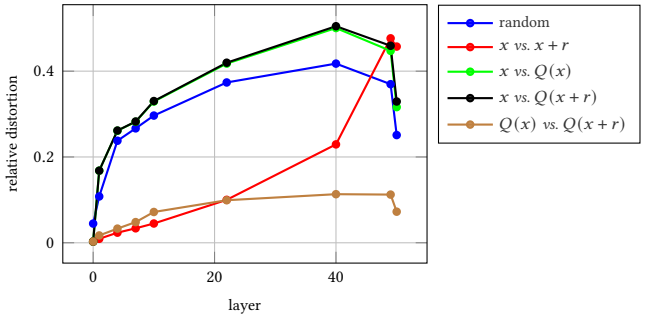
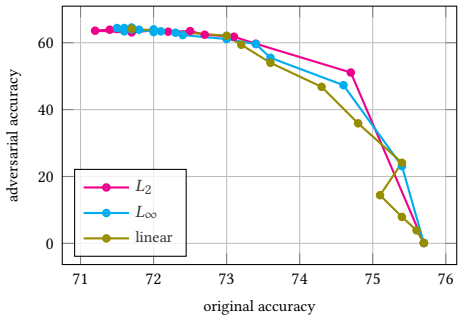
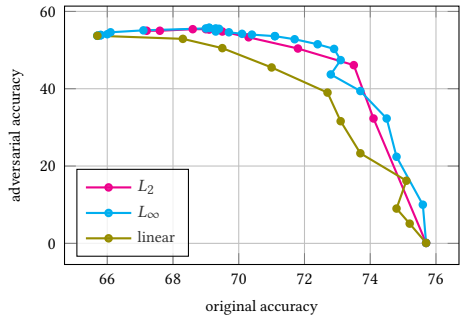


Figure 6: Relative distortion of random noise, adversarial perturbation and our patch replacement on input images (layer 0) using the plain strategy *vs.* ResNet-50 layer, averaged over the ImageNet test set. Random noise follows a normal distribution and adversarial perturbations are generated by DDN [28]. x: original input image; n: random noise; r: adversarial perturbation; Q: our patch replacement quantizer.



(a) image layer



(b) layer 1

Figure 7: Effect of replacement strategies for varying parameters ϵ, λ, applying patch replacement independently on (a) image layer with $K = 3927, d = 1$; and (b) first layer with $K = 235671, d = 4$.

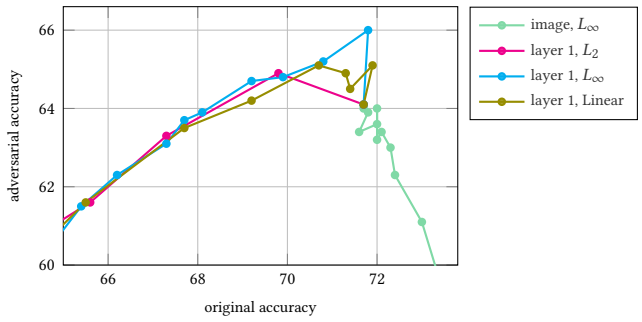


Figure 8: Based on the best codebook and strategy on the image layer, we add patch replacement on layer 1 with the best codebook, and find the best strategy for this combination.

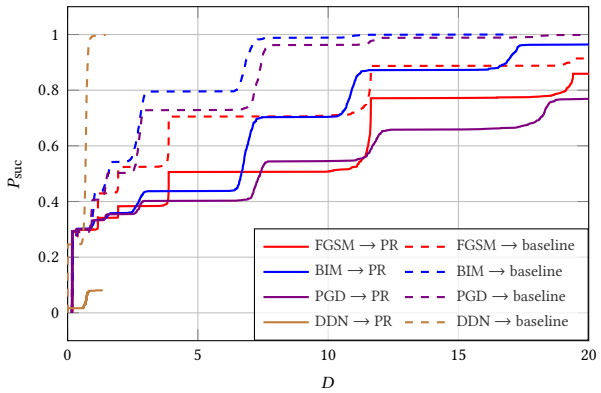


Figure 9: Operating characteristics of gray-box attacks against ResNet-50, with patch replacement (PR) and without (baseline).

- Image layer, L_∞ strategy with $\epsilon = 0.1$: original accuracy 72.0% (0.3% gain compared to plain strategy), adversarial accuracy 64.0% (0.2% loss compared to plain strategy).
- Image layer, L_∞ strategy with $\epsilon = 0.17$: original accuracy 71.7% (zero loss), adversarial accuracy 64.5% (0.3% gain).
- Layer 1, L_∞ strategy with $\epsilon = 0.17$: original accuracy 69.3% (3.6% gain), adversarial accuracy 55.6% (1.9% gain).

By comparing Figure 7(a) with Figure 7(b), it can be seen that replacement strategies in the first layer improve the performance more than on images. This is possibly due to the fact that the codebook chosen from Figure 5 for images is finer than the codebook chosen for the first layer.

4.3 Ablation study: multi-layer

From single layer experiments, we know that

- patch replacement on images works with a fine codebook, giving high original and adversarial accuracy;
- patch replacement on features of the first layer works with a coarse codebook; and
- replacement strategies help patch replacement in the first layer most, increasing original accuracy while maintaining or slightly improving adversarial accuracy.

To benefit from both fine and coarse codebooks, we investigate applying patch replacement on both the image layer and the first layer. We take the setting of patch replacement on images with the plain strategy, *i.e.* $K = 3927, d = 1$, losing 4% original accuracy but gaining 64.1% adversarial accuracy. We then apply patch replacement in the first layer, where we find that the best codebook with the plain strategy is $K = 235671, d = 4$. To reduce the cost, we use the same set of codebooks learned independently per layer, as in the previous experiments.

With codebooks being fixed in both the image layer and the first layer, we optimize the replacement strategy in the first layer. In Figure 8, the bottom right point of the plots corresponding to layer 1, is obtained with $\epsilon/\lambda = 0$, such that patch replacement only applies to images. This is the same as the top left point of the image layer plot, which corresponds to the plain strategy for that layer. We observe that replacement strategies on layer 1 improve the adversarial accuracy while losing little or no original accuracy. Thus, the combination of patch replacement on both layers achieves a more interesting operating point in terms of both accuracies. We select the L_∞ strategy with $\epsilon = 0.01$, giving original accuracy 71.8% and adversarial accuracy 66.0%.

It is possible to apply patch replacement to more layers, fixing the settings of the first layers before optimizing the next. However, this process is expensive and increases the inference cost and required space for codebooks, while the improvement in performance is small. We thus apply patch replacement only to images and layer 1.

4.4 Comparisons

We investigate two kinds of attacks. *Gray-box attacks* have full knowledge of the original network but no knowledge of the defense. *White-box attacks* are aware of both the network and the transformation-based defense.

Gray-box attacks As shown in Table 2, adversarial training with PGD performs poorly on legitimate images (45.9%). It is vulnerable against DDN attack, but it defends well against FGSM, BIM, and PGD. Transformation-based defenses, like bit5 and ms2, have slightly higher original accuracy than patch replacement but significantly lower adversarial accuracy. Comparing to bit5 and ms2, patch replacement gains around 20% on adversarial images obtained by BIM, PGD and DDN, while it loses around 3% on legitimate images. Comparing to pixel deflection, patch replacement loses around 1.5% on legitimate images but gains around 8% on adversarial images of DDN and more on BIM and PGD.

Our defense as well as the other transformation-based defenses perform better on adversarial images with small distortion, DDN in particular. For adversarial images with large distortion, patch replacement still outperforms other transformation-based defenses. Focusing on success rate and distortion, all attacks need greater distortion for the same success rate close to 1, but the required distortion is greatest under patch replacement.

Figure 9 provides an detailed view of the performance of patch replacement against gray-box attacks using operating characteristics [37, 38]. The plots with patch replacement are all below the plot

Method	Ori	FGSM [6]			BIM [15]			PGD [19]			DDN [28]			BPDA [2]		
	Acc	Acc	P_{suc}	$\overline{D}$	Acc	P_{suc}	$\overline{D}$	Acc	P_{suc}	$\overline{D}$	Acc	P_{suc}	$\overline{D}$	Acc	P_{suc}	$\overline{D}$
Baseline	75.7	12.1	0.95	5.13	1.20	1.00	2.50	3.80	1.00	3.12	0.10	1.00	0.53	3.80	1.00	3.12
Patch replacement (ours)	71.8	23.2	0.92	**7.98**	30.1	0.99	**6.10**	**46.4**	0.83	**7.21**	**66.0**	**0.08**	**0.57**	**48.3**	**0.89**	**12.89**
Adv. training [19]	45.9	**44.3**	**0.69**	6.25	**44.1**	**0.67**	5.07	44.3	**0.65**	3.81	19.0	0.58	0.32	–	–	–
Bit3 [8]	64.7	17.8	0.94	6.66	17.5	1.00	4.53	32.9	0.98	6.30	55.1	0.15	0.49	0.9	1.00	1.00
Bit5 [8]	**74.9**	12.2	0.95	5.42	1.70	1.00	2.93	6.50	1.00	3.81	18.9	0.75	0.53	1.9	1.00	1.00
Ms2 [36]	74.2	21.2	0.93	7.71	13.1	0.99	4.57	26.5	0.92	5.62	47.9	0.33	0.51	3.3	1.00	1.76
Ms3 [36]	71.8	21.0	0.95	7.66	17.8	0.98	4.76	34.2	0.84	5.45	55.6	0.23	0.53	1.6	1.00	1.25
Pixel deflection [23]	73.2	16.4	0.94	6.97	13.4	1.00	4.57	31.0	1.00	6.70	58.6	0.20	0.53	1.3	1.00	0.89

Table 2: Original accuracy, adversarial accuracy, success rate (P_{suc}) and average distortion ($\overline{D}$) for combinations of defenses (adversarial training and transformation-based) and attacks, including gray-box (FGSM, BIM, PGD, DDN) and white-box (BPDA).

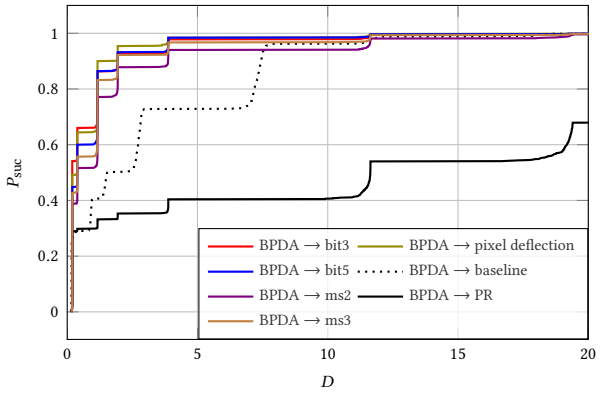


Figure 10: Operating characteristics of white-box attack BPDA [2] against different defenses or the original network (baseline). PR: patch replacement. P_{suc}: success rate; D: distortion.

without, meaning that attacks either need more distortion for the same success rate, or have less success rate for the same distortion. For example, when fixing $D = 5$, all attacks achieve success rate 0.7 or more on the original network but roughly 0.5 or less against patch replacement. Conversely, when fixing $P_{\text{suc}} = 0.5$, a distortion around 2 is enough for all attacks to succeed against the original network, but against patch replacement, a distortion of more than 7 is needed.

White-box attacks We use *BPDA* [2] smart attack, which includes the transformation-based defense in the forward pass but replaces it with the identity function in the backward. We apply BPDA on patch replacement as well as the other transformation-based defenses but not adversarial training, as this would be equivalent to PGD.

As shown in the rightmost part of Table 2, for competing defenses, BPDA has near zero adversarial accuracy, success rate of 1 and small distortion, less than 2 on average. By contrast, for patch replacement, it has adversarial accuracy 48.3, success rate 0.89 and average distortion of more than 12. Patch replacement is thus an effective defense against this smart attack, while competing defenses are not.

Figure 10 elaborates on the previous results by using operating characteristics [37, 38]. Patch replacement not only outperforms all other transformation-based defenses by a large margin, but it also improves over the baseline of BPDA against the original network, while all other defenses are actually outperformed by the baseline. This means that in the white-box setting, patch replacement makes the network more robust even though the attacker is aware of the defense, while all other transformation-based defenses fail by making the network easier to attack.

5 CONCLUSION

Motivated by the fact that networks are robust against random noise but vulnerable to adversarial perturbations, we have introduced *patch replacement*, a transformation-based defense, which succeeds in making adversarial perturbations behave similarly to random noise and defending against adversarial attacks with at low training and inference cost.

We have found it most effective to apply patch replacement in the early stages of the network, in particular, to input images and the first layer. This indicates that the effect of the perturbation is amplified in the deeper layers and is thus harder to remove. Applying patch replacement to both input images and the first layer improves the trade-off between accuracy and robustness.

Patch replacement is more effective than other transformation-based defenses. Against a gray-box attack with lower distortion (DDN [28]) and a white-box attack (BPDA [2]), it is even more effective than adversarial training, which is notoriously more expensive to train. In the case of the white-box attack, all other transformation-based defenses fail, making the network easier to attack.

Acknowledgements Experiments were performed using HPC resources of GENCI–IDRIS[9] (Grant 2019-AD011011287). This work is supported by ANR chaire IAD SAIDA (Grant ANR-20-CHIA-0011-01).

[9] http://www.genci.fr/?lang=en

REFERENCES

[1] Nader Asadi, AmirMohammad Sarfi, Mehrdad Hosseinzadeh, Sahba Tahsini, and Mahdi Eftekhari. 2019. Diminishing the effect of adversarial perturbations via refining feature representation. *arXiv preprint arXiv:1907.01023* (2019).

[2] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420* (2018).

[3] Hao-Yun Chen, Jhao-Hong Liang, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. 2019. Improving adversarial robustness via guided complement entropy. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 4881–4889.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 248–255.

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

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[7] Shixiang Gu and Luca Rigazio. 2014. Towards Deep Neural Network Architectures Robust to Adversarial Examples. arXiv:1412.5068 [cs.LG]

[8] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2017. Countering Adversarial Images using Input Transformations. *arXiv preprint arXiv:1711.00117* (2017).

[9] Puneet Gupta and Esa Rahtu. 2019. Ciidefence: Defeating adversarial attacks by fusing class-specific image inpainting and image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 6708–6717.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 630–645.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[12] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE Trans. PAMI)* 33, 1 (2010), 117–128.

[13] Vishaal Munusamy Kabilan, Brandon Morris, Hoang-Phuong Nguyen, and Anh Nguyen. 2021. Vectordefense: Vectorization as a defense to adversarial examples. In *Soft Computing for Biomedical Applications and Related Topics*. Springer, 19–35.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the International conference on Neural Information Processing Systems (NeurIPS)*. 1097–1105.

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[16] Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, Wenchang Shi, and Xiaofeng Wang. 2018. Detecting adversarial image examples in deep neural networks with adaptive noise reduction. *IEEE Transactions on Dependable and Secure Computing* (2018).

[17] Jiajun Lu, Theerasit Issaranon, and David Forsyth. 2017. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 446–454.

[18] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. 2015. A unified gradient regularization family for adversarial examples. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. IEEE, 301–309.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[20] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the Conference on Computer and Communications Security (SIGSAC)*. ACM, 135–147.

[21] Seyed-Mohsen Moosavi-Dezfooli, Ashish Shrivastava, and Oncel Tuzel. 2018. Divide, denoise, and defend against adversarial attacks. *arXiv preprint arXiv:1802.06806* (2018).

[22] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*. IEEE.

[23] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. 2018. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 8571–8580.

[24] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. 2018. Protecting JPEG images against adversarial attacks. In *Proceedings of the Data Compression Conference*. IEEE, 137–146.

[25] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. 2019. Barrage of random transforms for adversarially robust defense. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 6528–6537.

[26] Ravi S Raju and Mikko Lipasti. 2020. Blurnet: Defense by filtering the feature maps. In *Proceedings of the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 38–46.

[27] Thiyagarajan Ramanathan, Abinaya Manimaran, Suya You, and CC Jay Kuo. 2019. Robustness of saak transform against adversarial attacks. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2531–2535.

[28] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. 2019. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 4322–4330.

[29] Bo Sun, Nian-Hsuan Tsai, Fangchen Liu, Ronald Yu, and Hao Su. 2019. Adversarial Defense by Stratified Convolutional Sparse Coding. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

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

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[32] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. 2017. Ensemble Adversarial Training: Attacks and Defenses. *arXiv preprint arXiv:1705.07204* (2017).

[33] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. 2018. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Proceedings of the International conference on Neural Information Processing Systems (NeurIPS)*. 6541–6550.

[34] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2017. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991* (2017).

[35] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. 2019. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 501–509.

[36] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[37] Hanwei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. 2020. Smooth adversarial examples. *EURASIP Journal on Information Security* 2020, 1 (2020), 1–12.

[38] Hanwei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. 2020. Walking on the edge: Fast, low-distortion adversarial examples. *IEEE Transactions on Information Forensics and Security (IEEE Trans. TIFS)* 16 (2020), 701–713.

[39] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 2921–2929.