Sparse-PGD: A Unified Framework for Sparse Adversarial Perturbations Generation

Xuyang Zhong, Chen Liu*
City University of Hong Kong, Hong Kong, China
xuyang.zhong@my.cityu.edu.hk, chen.liu@cityu.edu.hk

Abstract

This work studies sparse adversarial perturbations, including both unstructured and structured ones. We propose a framework based on a white-box PGD-like attack method named Sparse-PGD to effectively and efficiently generate such perturbations. Furthermore, we combine Sparse-PGD with a black-box attack to comprehensively and more reliably evaluate the models' robustness against unstructured and structured sparse adversarial perturbations. Moreover, the efficiency of Sparse-PGD enables us to conduct adversarial training to build robust models against various sparse perturbations. Extensive experiments demonstrate that our proposed attack algorithm exhibits strong performance in different scenarios. More importantly, compared with other robust models, our adversarially trained model demonstrates state-of-the-art robustness against various sparse attacks.

1 Introduction

Deep learning has been developing tremendously fast in the last decade. However, it is shown vulnerable to adversarial attacks: imperceivable adversarial perturbations [1, 2] could change the prediction of a model without altering the input's semantic content, which poses great challenges in safety-critical systems. Among different kinds of adversarial perturbations, the ones bounded by l_∞ or l_2 norms are mostly well-studied [3, 4, 5] and benchmarked [6], because these adversarial budgets, i.e., the sets of all allowable perturbations, are convex, which facilitates theoretical analyses and algorithm design. By contrast, we study *sparse perturbations* in this work, including both *unstructured* and *structured* ones which are bounded by l_0 norm and group l_0 norm, respectively. These perturbations are quite common in physical scenarios, including broken pixels in LED screens to fool object detection models and adversarial stickers on road signs to make an auto-driving system fail [7, 8, 9, 10, 11].

However, constructing such adversarial perturbations is challenging as the corresponding adversarial budget is non-convex. Therefore, gradient-based methods, such as projected gradient descent (PGD) [4], usually cannot efficiently obtain a strong adversarial perturbation. In this regard, existing methods to generate sparse perturbations [12, 13, 14, 15, 16, 17] either cannot control the l_0 norm or the group l_0 norm of perturbations or have prohibitively high computational complexity, which makes them inapplicable for adversarial training to obtain robust models against sparse

*Correspondence author.

perturbations. The perturbations bounded by l_1 norm are the closest scenario to l_0 bounded perturbations among convex adversarial budgets defined by an l_p norm. Nevertheless, adversarial training in this case [18, 19] still suffers from issues such as slow convergence and instability. Jiang et al. [20] demonstrates that these issues arise from non-sparse perturbations bounded by l_1 norm. In other words, l_1 adversarial budget guarantees still cannot the sparsity of the perturbations. Furthermore, existing works investigating structured sparse perturbations [16, 17, 21, 22, 23] only support generating a single adversarial patch, lacking flexibility and generality. Thus, it is necessary but challenging to develop a unified framework for both unstructured and structured sparse adversarial perturbations.

In this work, we propose a white-box attack named Sparse-PGD (sPGD) to effectively and efficiently generate unstructured and structured sparse perturbations. For unstructured sparse perturbations, we decompose the perturbation $\boldsymbol{\delta}$ as the product of a magnitude tensor $\boldsymbol{p}$ and a binary sparse mask $\boldsymbol{m}$: $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}$, where $\boldsymbol{p}$ and $\boldsymbol{m}$ determine the magnitudes and the locations of perturbed features, respectively. Although $\boldsymbol{p}$ can be updated by PGD-like methods, it is challenging to directly optimize the binary mask $\boldsymbol{m}$ in the discrete space. We thereby introduce an alternative continuous variable $\widetilde{\boldsymbol{m}}$ to approximate $\boldsymbol{m}$ and update $\widetilde{\boldsymbol{m}}$ by gradient-based methods, $\widetilde{\boldsymbol{m}}$ is then transformed to $\boldsymbol{m}$ by projection to the discrete space. To further boost the performance, we propose the unprojected gradient of $\boldsymbol{p}$ and random reinitialization mechanism. For structured sparse perturbations, we study group l_0 norm and its approximated version based on the group norm proposed in [24]. The structured sparse perturbation bounded by approximated group l_0 norm can be thus decomposed as $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m} = \boldsymbol{p} \odot \min\left(\mathrm{TConv}(\boldsymbol{v}, \boldsymbol{k}), 1\right)$, where the binary group mask $\boldsymbol{v}$ determines the positions of the groups to be perturbed and the customized binary kernel $\boldsymbol{k}$ determines the pattern of groups. That is to say, we decompose the structured sparse perturbations into their positions and the patterns. Furthermore, we leverage transposed convolution (TConv) and clipping operations to map the positions of the groups $\boldsymbol{v}$ to the positions of perturbed features $\boldsymbol{m}$. Similar to the unstructured cases, we introduce the continuous $\widetilde{\boldsymbol{v}}$ and update it by gradient-based methods. Ultimately, we manage to transform the problem of optimizing the structured sparse mask $\boldsymbol{m}$ into the problem of optimizing the group mask $\boldsymbol{v}$ bounded by l_0 norm, which can be resolved in the framework of sPGD. The whole pipeline is illustrated in Figure 2. On top of sPGD, we propose Sparse-AutoAttack (sAA), which is the ensemble of the white-box sPGD and another black-box sparse attack, for a more comprehensive and reliable evaluation against both unstructured and structured sparse perturbations. Through extensive experiments, we show that our method exhibits better performance than other attacks.

More importantly, we explore adversarial training to obtain robust models against sparse attacks. In this context, the attack method will be called in each mini-batch update, so it should be both effective and efficient. Compared with existing methods, our proposed sPGD performs much better when using a small number of iterations, making it feasible for adversarial training and its variants [25]. Empirically, models adversarially trained by sPGD demonstrate the strongest robustness against various sparse attacks.

We summarize the contributions of this paper as follows:

1. We propose an effective and efficient white-box attack algorithm named Sparse-PGD (sPGD) which can be utilized to generate both unstructured and structured sparseadversarial perturbations.

2. sPGD achieves the best performance among white-box sparse attacks. We then combine it with a black-box sparse attack to construct Sparse-AutoAttack (sAA) for more comprehensive

robustness evaluation against sparse adversarial perturbations.

3. sPGD achieves much better performance in the regime of limited iterations, it is then adopted for adversarial training. Extensive experiments demonstrate that models adversarially trained by sPGD have significantly stronger robustness against various sparse attacks.

Preliminaries: We use image classification as an example, although the proposed methods are applicable to any classification model. Under l_p bounded perturbations, the robust learning aims to solve the following min-max optimization problem.

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \max_{\boldsymbol{\delta}_i} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}_i + \boldsymbol{\delta}_i), \\ &\text{s.t. } ||\boldsymbol{\delta}_i||_p \leq \epsilon,\ 0 \leq \boldsymbol{x}_i + \boldsymbol{\delta}_i \leq 1. \end{aligned} \tag{1}$$

where $\boldsymbol{\theta}$ denotes the parameters of the model and $\mathcal{L}$ is the loss objective function. $\boldsymbol{x}_i \in \mathbb{R}^{h \times w \times c}$ is the input image where h, w and c represent the height, width, and number of channels, respectively. $\boldsymbol{\delta}_i$ has the same shape as $\boldsymbol{x}_i$ and represents the perturbation. The perturbations are constrained by its l_p norm and the bounding box. In this regard, we use the term *adversarial budget* to represent the set of all allowable perturbations. Adversarial attacks focus on the inner maximization problem of (1) and aim to find the optimal adversarial perturbation, while adversarial training focuses on the outer minimization problem of (1) and aims to find a robust model parameterized by $\boldsymbol{\theta}$. Due to the high dimensionality and non-convexity of the loss function when training a deep neural network, [26] has proven that solving the problem (1) is at least NP-complete.

We consider the pixel sparsity for image inputs in this work, which is more meaningful than feature sparsity and consistent with existing works [13, 17]. That is, a pixel is considered perturbed if *any* of its channel is perturbed, and sparse perturbation means few pixels are perturbed.

2 Related Works

Non-sparse Attacks: The pioneering work [1] finds the adversarial perturbations to fool image classifiers and proposes a method to minimize the l_2 norm of such perturbations. To more efficiently generate adversarial perturbations, the fast gradient sign method (FGSM) [3] generates l_∞ perturbation in one step, but its performance is significantly surpassed by the multi-step variants [27]. Projected Gradient Descent (PGD) [4] further boosts the attack performance by using iterative updating and random initialization. Specifically, each iteration of PGD updates the adversarial perturbation $\boldsymbol{\delta}$ by:

$$\boldsymbol{\delta} \longleftarrow \Pi_{\mathcal{S}}(\boldsymbol{\delta} + \alpha \cdot s(\nabla_{\boldsymbol{\delta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{\delta}))) \tag{2}$$

where $\mathcal{S}$ is the adversarial budget, α is the step size, $s : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{h \times w \times c}$ selects the steepest ascent direction based on the gradient of the loss $\mathcal{L}$ with respect to the perturbation. Inspired by the first-order Taylor expansion, Madry et al. [4] derives the steepest ascent direction for l_2 bounded and l_∞ bounded perturbations to efficiently find strong adversarial examples; SLIDE [18] and l_1-APGD [19] use k-coordinate ascent to construct l_1 bounded perturbations, which is shown to suffer from the slow convergence [20].

Besides the attacks that have access to the gradient of the input (i.e., white-box attacks), there are black-box attacks that do not have access to model parameters, including the ones based on

gradient estimation through finite differences [28, 29, 30, 31, 32] and the ones based on evolutionary strategies or random search [33, 34]. To improve the query efficiency of these attacks, [35, 36, 37, 38] generate adversarial perturbation at the corners of the adversarial budget.

To more reliably evaluate the robustness, Croce and Hein [39] proposes AutoAttack (AA) which consists of an ensemble of several attack methods, including both black-box and white-box attacks. Croce and Hein [19] extends AA to the case of l_1 bounded perturbations and proposes AutoAttack-l_1 (AA-l_1). Although the l_1 bounded perturbations are usually sparse, Jiang et al. [20] demonstrates that AA-l_1 is able to find non-sparse perturbations that cannot be found by SLIDE to fool the models. That is to say, l_1 bounded adversarial perturbations are not guaranteed to be sparse. We should study perturbations bounded by l_0 norm.

Sparse Attacks: For perturbations bounded by l_0 norm, directly adopting vanilla PGD as in Eq. (2) leads to suboptimal performance due to the non-convexity nature of the adversarial budget: PGD_0 [13], which updates the perturbation by gradient ascent and project it back to the adversarial budget, turns out very likely to trap in the local maxima. Different from PGD_0, CW L0 [40] projects the perturbation onto the feasible set based on the absolute product of gradient and perturbation and adopts a mechanism similar to CW L2 [40] to update the perturbation. SparseFool [12] and GreedyFool [15] also generate sparse perturbations, but they do not strictly restrict the l_0 norm of perturbations. If we project their generated perturbations to the desired l_0 ball, their performance will drastically drop. Sparse Adversarial and Interpretable Attack Framework (SAIF) [41] is similar to our method in that SAIF also decomposes the l_0 perturbation into a magnitude tensor and sparsity mask, but it uses the Frank-Wolfe algorithm [42] to separately update them. SAIF turns out to get trapped in local minima and shows poor performance on adversarially trained models. Besides white-box attacks, there are black-box attacks to generate sparse adversarial perturbations, including CornerSearch [13] and Sparse-RS [17]. However, these black-box attacks usually require thousands of queries to find an adversarial example, making it difficult to scale up to large datasets. In addition to the unstructured adversarial perturbations mentioned above, there are several works discussing structured sparse perturbations, including universal adversarial patch for all data [21, 22] and some image-specific patches [16, 17, 23]. However, these works only support generating a single adversarial patch, which lacks of flexibility and generality.

Adversarial Training: Despite the difficulty in obtaining robust deep neural network, adversarial training [4, 43, 44, 45, 46, 47, 48, 49] stands out as a reliable and popular approach to do so [39, 50]. It generates adversarial examples first and then uses them to optimize model parameters. Despite effective, adversarial training is time-consuming due to multi-step attacks. Shafahi et al. [51], Zhang et al. [52], Wong et al. [53], Sriramanan et al. [54] use weaker but faster one-step attacks to reduce the overhead, but they may suffer from catastrophic overfitting [55]: the model overfits to these weak attacks during training instead of achieving true robustness to various attacks. Kim et al. [56], Andriushchenko and Flammarion [57], Golgooni et al. [58], de Jorge et al. [59] try to overcome catastrophic overfitting while maintaining efficiency.

Compared with l_∞ and l_2 bounded perturbations, adversarial training against l_1 bounded perturbations is shown to be even more time-consuming to achieve the optimal performance [19]. In the case of l_0 bounded perturbations, PGD_0 [13] is adopted for adversarial training. However, models trained by PGD_0 exhibit poor robustness against strong sparse attacks. In this work, we propose an effective and efficient sparse attack that enables us to train a model that is more robust against various sparse attacks than existing methods.

3 Unstructured Sparse Adversarial Attack

In this section, we introduce Sparse-PGD (sPGD) for generating unstructured sparse perturbations. Its extension that generates structured sparse perturbations is introduced in Sec. 4. Similar to AutoAttack [19, 39], we further combine sPGD with a black-box attack to construct sparse-AutoAttack (sAA) for more comprehensive and reliable robustness evaluation.

3.1 Sparse-PGD (sPGD)

Inspired by SAIF [41], we decompose the sparse perturbation $\boldsymbol{\delta}$ into a magnitude tensor $\boldsymbol{p} \in \mathbb{R}^{h\times w\times c}$ and a sparsity mask $\boldsymbol{m} \in \{0,1\}^{h\times w\times 1}$, i.e., $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}$. Therefore, the attacker aims to maximize the following loss objective function:

$$\max_{\|\boldsymbol{\delta}\|_0 \leq \epsilon, 0\leq \boldsymbol{x}+\boldsymbol{\delta}\leq 1} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}+\boldsymbol{\delta}) = \max_{\boldsymbol{p}\in\mathcal{S}_{\boldsymbol{p}}, \boldsymbol{m}\in\mathcal{S}_{\boldsymbol{m}}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}+\boldsymbol{p}\odot\boldsymbol{m}). \tag{3}$$

The feasible sets for $\boldsymbol{p}$ and $\boldsymbol{m}$ are $\mathcal{S}_{\boldsymbol{p}} = \{\boldsymbol{p} \in \mathbb{R}^{h\times w\times c} | 0 \leq \boldsymbol{x}+\boldsymbol{p} \leq 1\}$ and $\mathcal{S}_{\boldsymbol{m}} = \{\boldsymbol{m} \in \{0,1\}^{h\times w\times 1} | \|\boldsymbol{m}\|_0 \leq \epsilon\}$, respectively. Similar to PGD, sPGD iteratively updates $\boldsymbol{p}$ and $\boldsymbol{m}$ until finding a successful adversarial example or reaching the maximum iteration number.

Update Magnitude Tensor $\boldsymbol{p}$: The magnitude tensor $\boldsymbol{p}$ is only constrained by the input domain. In the case of images, the input is bounded between 0 and 1. Note that the constraints on $\boldsymbol{p}$ are elementwise and similar to those of l_∞ bounded perturbations. Therefore, instead of greedy or random search [13, 17], we utilize PGD in the l_∞ case, i.e., use the sign of the gradients, to optimize $\boldsymbol{p}$ as demonstrated by Eq. (4) below, with α being the step size.

$$\boldsymbol{p} \longleftarrow \Pi_{\mathcal{S}_{\boldsymbol{p}}} \left(\boldsymbol{p} + \alpha \cdot \texttt{sign}(\nabla_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}+\boldsymbol{p}\odot\boldsymbol{m}))\right), \tag{4}$$

Update Sparsity Mask $\boldsymbol{m}$: The sparsity mask $\boldsymbol{m}$ is binary and constrained by its l_0 norm. Directly optimizing the discrete variable $\boldsymbol{m}$ is challenging, so we update its continuous alternative $\widetilde{\boldsymbol{m}} \in \mathbb{R}^{h\times w\times 1}$ and project $\widetilde{\boldsymbol{m}}$ to the feasible set $\mathcal{S}_{\boldsymbol{m}}$ to obtain $\boldsymbol{m}$ before multiplying it with the magnitude tensor $\boldsymbol{p}$ to obtain the sparse perturbation $\boldsymbol{\delta}$. Specifically, $\widetilde{\boldsymbol{m}}$ is updated by gradient ascent. Projecting $\widetilde{\boldsymbol{m}}$ to the feasible set $\mathcal{S}_{\boldsymbol{m}}$ is to set the ϵ-largest elements in $\widetilde{\boldsymbol{m}}$ to 1 and the rest to 0. In addition, we adopt the sigmoid function to normalize the elements of $\widetilde{\boldsymbol{m}}$ before projection.

Mathematically, the update rules for $\widetilde{\boldsymbol{m}}$ and $\boldsymbol{m}$ are demonstrated as follows:

$$\widetilde{\boldsymbol{m}} \longleftarrow \widetilde{\boldsymbol{m}} + \beta \cdot \nabla_{\widetilde{\boldsymbol{m}}} \mathcal{L} / \|\nabla_{\widetilde{\boldsymbol{m}}} \mathcal{L}\|_2, \tag{5}$$

$$\boldsymbol{m} \longleftarrow \Pi_{\mathcal{S}_{\boldsymbol{m}}}(\sigma(\widetilde{\boldsymbol{m}})) \tag{6}$$

where β is the step size for updating the sparsity mask's continuous alternative $\widetilde{\boldsymbol{m}}$, $\sigma(\cdot)$ denotes the sigmoid function. Furthermore, to prevent the magnitude of $\widetilde{\boldsymbol{m}}$ from becoming explosively large, we do not update $\widetilde{\boldsymbol{m}}$ when $||\nabla_{\widetilde{\boldsymbol{m}}}\mathcal{L}||_2 < \gamma$, which indicates that $\widetilde{\boldsymbol{m}}$ is located in the saturation zone of sigmoid function. The gradient $\nabla_{\widetilde{\boldsymbol{m}}}\mathcal{L}$ is calculated at the point $\boldsymbol{\delta} = \boldsymbol{p} \odot \Pi_{\mathcal{S}_{\boldsymbol{m}}}(\sigma(\widetilde{\boldsymbol{m}}))$, where the loss function is not always differentiable. We demonstrate how to estimate the update direction in the next part.

Backward Function: Based on Eq. (3), we can calculate the gradient of the magnitude tensor $\boldsymbol{p}$ by $\nabla_{\boldsymbol{p}}\mathcal{L} = \nabla_{\boldsymbol{\delta}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}+\boldsymbol{\delta}) \odot \boldsymbol{m}$ and use $g_{\boldsymbol{p}}$ to represent this gradient for notation simplicity. At most, ϵ non-zero elements are in the mask $\boldsymbol{m}$, so $g_{\boldsymbol{p}}$ is sparse and has at most ϵ non-zero elements. That is to say, we update at most ϵ elements of the magnitude tensor $\boldsymbol{p}$ based on the gradient $g_{\boldsymbol{p}}$.

Algorithm 1 Sparse-PGD for l_0 Bounded Perturbations

Input: Clean image: $\boldsymbol{x} \in [0,1]^{h \times w \times c}$; Model parameters: $\boldsymbol{\theta}$; Max iteration number: T; Tolerance: t; l_0 budget: ϵ; Step size: α, β; Small constant: $\gamma = 2 \times 10^{-8}$
Random initialize $\boldsymbol{p}$ and $\widetilde{\boldsymbol{m}}$
$\boldsymbol{m} = \Pi_{\mathcal{S}_{\boldsymbol{m}}}(\sigma(\widetilde{\boldsymbol{m}}))$
for $i = 0, 1, ..., T-1$ **do**
  Calculate the loss $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{p} \odot \boldsymbol{m})$
  if unprojected **then**
    $g_{\boldsymbol{p}} = \nabla_{\boldsymbol{\delta}} \mathcal{L} \odot \sigma(\widetilde{\boldsymbol{m}})$ $\quad \{\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}\}$
  else
    $g_{\boldsymbol{p}} = \nabla_{\boldsymbol{\delta}} \mathcal{L} \odot \boldsymbol{m}$
  end if
  $g_{\widetilde{\boldsymbol{m}}} = \nabla_{\boldsymbol{\delta}} \mathcal{L} \odot \boldsymbol{p} \odot \sigma'(\widetilde{\boldsymbol{m}})$
  $\boldsymbol{p} = \Pi_{\mathcal{S}_{\boldsymbol{p}}}(\boldsymbol{p} + \alpha \cdot \texttt{sign}(g_{\boldsymbol{p}}))$
  $\boldsymbol{d} = g_{\widetilde{\boldsymbol{m}}} / \|g_{\widetilde{\boldsymbol{m}}}\|_2$ **if** $||g_{\widetilde{\boldsymbol{m}}}||_2 \geq \gamma$ **else** 0
  $\boldsymbol{m}_{old}$, $\widetilde{\boldsymbol{m}} = \boldsymbol{m}$, $\widetilde{\boldsymbol{m}} + \beta \cdot \boldsymbol{d}$
  $\boldsymbol{m} = \Pi_{\mathcal{S}_{\boldsymbol{m}}}(\sigma(\widetilde{\boldsymbol{m}}))$
  if attack succeeds **then**
    break
  end if
  if $||\boldsymbol{m} - \boldsymbol{m}_{old}||_0 \leq 0$ for t consecutive iters **then**
    Random initialize $\widetilde{\boldsymbol{m}}$
  end if
end for
Output: Perturbation: $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}$

Like coordinate descent, this may result in suboptimal performance since most elements of $\boldsymbol{p}$ are unchanged in each iterative update. To tackle this problem, we discard the projection to the binary set $\mathcal{S}_{\boldsymbol{m}}$ when calculating the gradient and use the *unprojected gradient* $\widetilde{g}_{\boldsymbol{p}}$ to update $\boldsymbol{p}$. Based on Eq. (6), we have $\widetilde{g}_{\boldsymbol{p}} = \nabla_{\boldsymbol{\delta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{\delta}) \odot \sigma(\widetilde{\boldsymbol{m}})$. The idea of the unprojected gradient is inspired by training pruned neural networks and lottery ticket hypothesis [60, 61, 62, 63]. All these methods train importance scores to prune the model parameters but update the importance scores based on the whole network instead of the pruned sub-network to prevent the sparse update, which leads to suboptimal performance.

In practice, the performance of using $g_{\boldsymbol{p}}$ and $\widetilde{g}_{\boldsymbol{p}}$ to optimize $\boldsymbol{p}$ is complementary. The sparse gradient $g_{\boldsymbol{p}}$ is consistent with the forward propagation and is thus better at exploitation. By contrast, the unprojected gradient $\widetilde{g}_{\boldsymbol{p}}$ updates the $\boldsymbol{p}$ by a dense tensor and is thus better at exploration. In view of this, we set up an ensemble of attacks with both gradients to balance exploration and exploitation.

When calculating the gradient of the continuous alternative $\widetilde{\boldsymbol{m}}$, we have $\frac{\partial \mathcal{L}}{\partial \widetilde{\boldsymbol{m}}} = \frac{\partial \mathcal{L}(\theta, \boldsymbol{x}+\boldsymbol{\delta})}{\partial \boldsymbol{\delta}} \odot \boldsymbol{p} \odot \frac{\partial \Pi_{\mathcal{S}_{\boldsymbol{m}}}(\sigma(\widetilde{\boldsymbol{m}}))}{\partial \widetilde{\boldsymbol{m}}}$. Since the projection to the set $\mathcal{S}_{\boldsymbol{m}}$ is not always differentiable, we discard the projection operator and use the approximation $\frac{\partial \Pi_{\mathcal{S}_{\boldsymbol{m}}}(\sigma(\widetilde{\boldsymbol{m}}))}{\partial \widetilde{\boldsymbol{m}}} \simeq \sigma'(\widetilde{\boldsymbol{m}})$ to calculate the gradient.

Random Reinitialization: Due to the projection to the set $\mathcal{S}_{\boldsymbol{m}}$ in Eq. (6), the sparsity mask $\boldsymbol{m}$ changes only when the relative magnitude ordering of the continuous alternative $\widetilde{\boldsymbol{m}}$ changes. In

other words, slight changes in $\widetilde{\boldsymbol{m}}$ usually mean no change in $\boldsymbol{m}$. As a result, $\boldsymbol{m}$ usually gets trapped in a local maximum. To solve this problem, we propose a random reinitialization mechanism. Specifically, when the attack fails, i.e., the model still gives the correct prediction, and the current sparsity mask $\boldsymbol{m}$ remains unchanged for three consecutive iterations, the continuous alternative $\widetilde{\boldsymbol{m}}$ will be randomly reinitialized for better exploration.

To summarize, we provide the pseudo-code of sparse PGD (sPGD) in Algorithm 1. SAIF [41] also decomposes the perturbation $\boldsymbol{\delta}$ into a magnitude tensor $\boldsymbol{p}$ and a mask $\boldsymbol{m}$, but uses a different update rule: it uses Frank-Wolfe to update both $\boldsymbol{p}$ and $\boldsymbol{m}$. By contrast, we introduce the continuous alternative $\widetilde{\boldsymbol{m}}$ of $\boldsymbol{m}$ and use gradient ascent to update $\boldsymbol{p}$ and $\widetilde{\boldsymbol{m}}$. Moreover, we include unprojected gradient and random reinitialization techniques in Algorithm 1 to further enhance the performance.

3.2 Sparse-AutoAttack (sAA)

AutoAttack (AA) [39] is an ensemble of four diverse attacks for a standardized parameters-free and reliable evaluation of robustness against l_∞ and l_2 attacks. Croce and Hein [19] extends AutoAttack to l_1 bounded perturbations. In this work, we propose sparse-AutoAttack (sAA), which is also a parameter-free ensemble of both black-box and white-box attacks for comprehensive robustness evaluation against l_0 bounded perturbations. It can be used in a plug-and-play manner. However, different from the l_∞, l_2 and l_1 cases, the adaptive step size, momentum and difference of logits ratio (DLR) loss function do not improve the performance in the l_0 case, so they are not adopted in sAA. In addition, compared with targeted attacks, sPGD turns out stronger when using a larger query budget in the untargeted settings given the same total number of back-propagations. As a result, we only include the untargeted sPGD with cross-entropy loss and constant step sizes in sAA. Specifically, we run sPGD twice for two different backward functions: one denoted as $\text{sPGD}_{\text{proj}}$ uses the sparse gradient $g_{\boldsymbol{p}}$, and the other denoted as $\text{sPGD}_{\text{unproj}}$ uses the unprojected gradient $\widetilde{g}_{\boldsymbol{p}}$ as described in Section 3.1. As for the black-box attack, we adopt the strong black-box attack Sparse-RS [17], which can generate l_0 bounded perturbations. We run each version of sPGD and Sparse-RS for 10000 iterations, respectively. We use cascade evaluation to improve the efficiency. Concretely, suppose we find a successful adversarial perturbation by one attack for one instance. Then, we will consider the model non-robust in this instance and the same instance will not be further evaluated by other attacks. Based on the efficiency and the attack success rate, the attacks in sAA are sorted in the order of $\textbf{sPGD}_{\text{unproj}}$, $\textbf{sPGD}_{\text{proj}}$ and **Sparse-RS**.

4 Structured Sparse Adversarial Attack

In this section, we extend Sparse-PGD (sPGD) and Sparse-AutoAttack (sAA) to generate structured sparse perturbations.

4.1 Formulation of Structured Sparsity

Given an input $\boldsymbol{x} \in \mathbb{R}^d$, we can partition its d features into several groups that may overlap and define structured sparsity based on them. These groups can represent pixels of a row, a column, a patch, or a particular pattern. Without the loss of generality, we use $\{1, 2, ..., d\}$ as the indices of d features of the input and N set of indices $\mathcal{G} = \{G_j\}_{j=1}^N$ to represent the groups. For any perturbation, we define its *group* l_0 *norm based on groups* $\mathcal{G}$ as *the minimal number* of groups

needed to cover its non-zero components. Mathematically, Bach [24] proposed a convex envelope of group l_0 norm as follows:

$$\Omega(\boldsymbol{x}) = \sum_{k=1}^{d} |x_{\pi_k}| \left[F(\{\pi_1, ..., \pi_k\}) - F(\{\pi_1, ..., \pi_{k-1}\})\right]. \tag{7}$$

where F is a submodular function defined on the subsets of $V = \{1, 2, ..., d\}$ and $\{\pi_i\}_{i=1}^{d}$ is a permutation of $\{1, 2, ..., d\}$. More specifically, $F(A)$ indicates the minimal number of groups from $\mathcal{G}$ to cover the set A and $\{\pi_i\}_{i=1}^{d}$ indicates the components of $|\boldsymbol{x}|$ in the decreasing order, i.e., $|x_{\pi_1}| \geq ... \geq |x_{\pi_d}| \geq 0$. Compared with the convex envelope Ω, the group l_0 norm Ω_0 is defined as follows:

$$\begin{aligned} \Omega_0(\boldsymbol{x}) &= \sum_{k=1}^{d} \mathbb{1}(|x_{\pi_k}|) \left[F(\{\pi_1, ..., \pi_k\}) - F(\{\pi_1, ..., \pi_{k-1}\})\right] \\ &= F(\{\pi_1, ..., \pi_{d'}\}) \quad \text{where } x_{\pi_{d'}} \neq 0 \text{ and } x_{\pi_{d'+1}} = 0. \end{aligned} \tag{8}$$

Similar to the difference between l_1 norm and l_0 norm, we replace $|x_{\pi_k}|$ with $\mathbb{1}(|x_{\pi_k}|)$ in Eq. (8) to indicate the number of groups with non-zero entities, where $\mathbb{1}$ is an indicator function. The second equality in Eq. (8) is based on the decreasing order of $\{|x_{\pi_i}|\}_{i=1}^{d}$ and is a more straightforward definition of the group l_0 norm. Like the l_0 norm, the adversarial budgets based on the group l_0 norm are not convex, either.

In general, it is difficult to decide the minimum number of groups to cover non-zero elements, i.e., to calculate the function F. Therefore, it is challenging to directly apply sPGD to the group l_0 norm constraints. To address this issue, we can approximate Ω_0 by a tight surrogate that facilitates the optimization by sPGD. In this regard, we introduce a binary group mask $\boldsymbol{v} = [v_1, ..., v_N] \in \{0, 1\}^N$ to indicate whether a specific group is chosen to be perturbed. Mathematically, $\forall i$, if $\boldsymbol{x}_i \neq 0$, then $\exists j \in \{1, 2, ..., N\}$, $v_j = 1$ and $i \in G_j$. In this context, we propose the following approximation of the group l_0 norm as the surrogate:

$$\Omega_0'(\boldsymbol{x}, \boldsymbol{v}) = \sum_{i=1}^{N} \mathbb{1}(\|\boldsymbol{x}_{G_i}\|_p) \cdot v_i, \tag{9}$$

where $\boldsymbol{x}_{G_i}$ is the subvector of $\boldsymbol{x}$ on the indices in G_i, and $p \in [0, +\infty]$. Compared with Ω_0 in Eq. (8), Ω_0' in Eq. (9) does not require the number of groups in $\mathcal{G}$ to cover non-zero elements in $\boldsymbol{x}$ to be minimum. Instead, Ω_0' calculates the number of perturbed groups in $\mathcal{G}$ selected by the binary vector $\boldsymbol{v}$. In addition, when $\mathbb{1}(\|\boldsymbol{x}_{G_i}\|_p) = 1$ for all $i \in \{j \mid v_j = 1\}$, the approximated group l_0 norm $\Omega_0'(\boldsymbol{x}, \boldsymbol{v})$ is equivalent to the l_0 norm of the group mask $\boldsymbol{v}$, i.e., $\Omega_0'(\boldsymbol{x}, \boldsymbol{v}) = \|\boldsymbol{v}\|_0$. In summary, we have the following theorem:

Theorem 4.1. *Given an input $\boldsymbol{x} \in \mathbb{R}^d$, a set of groups $\mathcal{G} = \{G_j\}_{j=1}^{N}$ and any vector $\boldsymbol{v}$ satisfying the constraint in the definition of Ω_0', then we have $\Omega_0(\boldsymbol{x}) \leq \Omega_0'(\boldsymbol{x}, \boldsymbol{v}) \leq \|\boldsymbol{v}\|_0$.*

Proof. The first inequality is based on the minimum optimality condition of the function F in Eq. (8). Since $\Omega_0(\boldsymbol{x})$ indicates the minimal number of groups needed to cover the indices of non-zero elements in $\boldsymbol{x}$, we have at least the same number of terms in Eq. (9) where $\boldsymbol{x}_{G_i} \neq 0$ and $v_i = 1$. Therefore, we have $\Omega_0(\boldsymbol{x}) \leq \Omega_0'(\boldsymbol{x})$. For the second inequality, $\boldsymbol{v}$ is a binary vector, i.e., $\forall i$, $\boldsymbol{v}_i \in \{0, 1\}$, so $\Omega_0'(\boldsymbol{x}, \boldsymbol{v}) \leq \|\boldsymbol{v}\|_0$ is clear.

□

Theorem 4.1 indicates that the number of perturbed groups determined by $\boldsymbol{v}$ can be larger than the minimal number of groups to cover the perturbed features $\boldsymbol{x}$, i.e., $\Omega_0'(\boldsymbol{x}, \boldsymbol{v})$ can be larger than $\Omega_0(\boldsymbol{x})$. The gap between $\Omega_0'(\boldsymbol{x}, \boldsymbol{v})$ and $\Omega_0(\boldsymbol{x})$ stem from the potential overlap among groups in $\mathcal{G}$. However, we demonstrate that $\Omega_0'(\boldsymbol{\delta}, \boldsymbol{v})$ is quite close to $\Omega_0(\boldsymbol{\delta})$ in practice. To corroborate this, we calculate the ratio between the group l_0 norm and approximated group l_0 norm, i.e., $\Omega_0(\boldsymbol{\delta})/\Omega_0'(\boldsymbol{\delta}, \boldsymbol{v})$ of both random and adversarial perturbations under different sparsity level ϵ defined by $\Omega_0'(\boldsymbol{\delta}, \boldsymbol{v}) \leq \epsilon$. As illustrated in Figure 1, when applying 3×3 patch perturbations, we can observe that the ratio $\Omega_0(\boldsymbol{\delta})/\Omega_0'(\boldsymbol{\delta}, \boldsymbol{v})$ decreases as ϵ increases. Specifically, when ϵ is smaller than 5, the ratios of both random and adversarial perturbations are larger than 0.95. Notably, the l_0 norm of 5 non-overlapping 3×3 patches reaches 45, which is very large for a sparse perturbation used in practice, so we can conclude based on the simulation results that Ω_0' is a good approximation of Ω_0 in the context of the structured sparse attack.

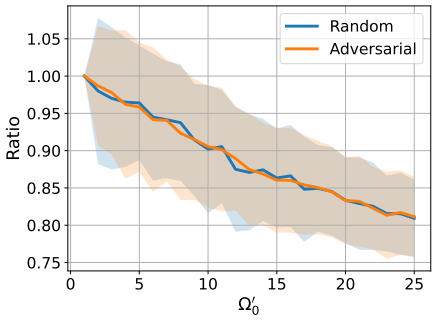


Figure 1: The ratio between the group l_0 norm and the approximated group l_0 norm. The x-axis is the approximated group l_0 norm, and the y-axis is the ratio. We plot the ratios of random and adversarial 3×3 patch perturbations, where the adversarial perturbations are generated on vanilla ResNet-18 trained on CIFAR-10. The solid line and shadow denote the mean value and standard deviation, respectively.

4.2 sPGD for Structured Sparse Perturbations

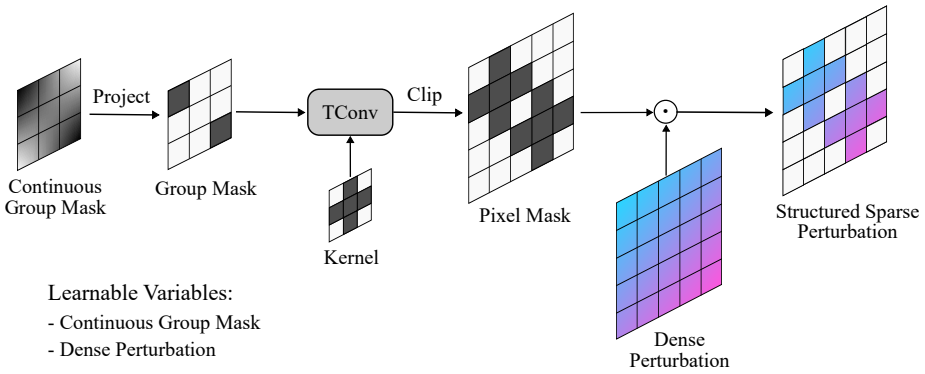


Figure 2: Pipeline of sPGD for structured sparse perturbations. The continuous group mask $\widetilde{\boldsymbol{v}}$ is first projected to get the binary group mask $\boldsymbol{v}$ to ensure $\|\boldsymbol{v}\|_0 \leq \epsilon$, which is similar to Eq. (6). Given the kernel $\boldsymbol{k} \in \{0,1\}^{r\times r}$ with a customized pattern, we can transform $\boldsymbol{v}$ to the pixel mask $\boldsymbol{m}$ using transposed convolution and clipping (see Eq. (10)). Finally, we element-wisely multiply $\boldsymbol{m}$ with the dense perturbation $\boldsymbol{p}$ to obtain the structured sparse perturbation $\boldsymbol{\delta}$. Note that the continuous group mask $\widetilde{\boldsymbol{v}}$ and the dense perturbation $\boldsymbol{p}$ are learnable.

Based on the approximated group l_0 norm Ω_0' proposed in Eq. (9), we extend Sparse-PGD (sPGD) to generate structured sparse perturbations. Similarly, we decompose the perturbation $\boldsymbol{\delta}$ into a magnitude tensor $\boldsymbol{p}$ and a binary pixel mask $\boldsymbol{m}$. Like unstructured cases, the magnitude tensor $\boldsymbol{p}$ is updated using l_∞ PGD, as in Eq. (4).

Based on the definition of structured sparsity and Ω_0' defined in Eq. (9), if a specific group G_i is chosen to be perturbed, then we let $v_i = 1$, otherwise v_i will be set 0. In this regard, finding the optimal $\boldsymbol{m}$ s.t. $\Omega_0'(\boldsymbol{m}, \boldsymbol{v}) \leq \epsilon$ is equivalent to finding the optimal group mask $\boldsymbol{v}$ s.t. $\|\boldsymbol{v}\|_0 \leq \epsilon$, which can be resolved by sPGD. Similar to the unstructured cases, we optimize the continuous alternative $\widetilde{\boldsymbol{v}}$ rather than optimizing a binary $\boldsymbol{v}$ directly. In the following, we will elaborate on the mapping from $\boldsymbol{v}$ to $\boldsymbol{m}$ using different examples.

Column / Row: If we aim to perturb at most ϵ columns in an image $\boldsymbol{x} \in [0,1]^{h \times w \times c}$. Since we can partition $\boldsymbol{x}$ into w groups (i.e., columns), $\boldsymbol{v}$ has the size of $1 \times w$. Thus, $\boldsymbol{m}$ can obtained through expanding $\boldsymbol{v}$ from $1 \times w$ to $h \times w$. As for row, the only difference is that the size of $\boldsymbol{v}$ is $h \times 1$. Note that, there is no overlap between any two columns or rows, so $\Omega_0'(\boldsymbol{m}, \boldsymbol{v}) \equiv \Omega_0(\boldsymbol{m})$ in this case.

Patch / Any Pattern: Without loss of generality, we assume that the perturbations are contained to be localized in $r \times r$ patches. To avoid wasting budget due to the potential overlap at the border or corner of images, we let $\boldsymbol{v} \in \{0,1\}^{(h-r+1)\times(w-r+1)}$, and we can derive the mapping operation from $\boldsymbol{v}$ to $\boldsymbol{m}$ as follows:

$$\boldsymbol{m} = \min\left(\mathrm{TConv}(\boldsymbol{v}, \boldsymbol{k}, s), 1\right), \tag{10}$$

where TConv is the transposed convolution operation, $\boldsymbol{k} = \mathbf{1}^{r \times r}$ is the kernel with all-one entities, $s = 1$ denotes the stride. The clipping operation is to ensure the output value assigned to $\boldsymbol{m}$ is binary. Moreover, we can customize the kernel $\boldsymbol{k} \in \{0,1\}^{r \times r}$ in Eq. (10) to make the perturbations localized in groups with any desired patterns, such as hearts, stars, letters and so on. Notably, when $\boldsymbol{k} = [1]$, i.e., a 1×1 matrix with a sole value 1, the resulting perturbations are degraded to unstructured sparse ones; when $\boldsymbol{k} = \mathbf{1}^{h \times 1}$ or $\mathbf{1}^{1 \times w}$, specific columns or rows are perturbed. The pipeline of sPGD for structured sparse perturbations in the general case is illustrated in Figure 2.

In addition to the forward mapping operation defined in Eq. (10), we provide its backward function as follows:

$$\boldsymbol{g}_{\boldsymbol{v}} = \mathrm{Conv}(\boldsymbol{g}_{\boldsymbol{m}}, \boldsymbol{k}, s), \tag{11}$$

where Conv is the convolution operation, and $\boldsymbol{g}_{\boldsymbol{m}}$ is the gradient of $\boldsymbol{m}$. We neglect the clipping operation when calculating the gradient of $\boldsymbol{v}$ since it is not always differentiable.

The extension of sPGD for structured sparse perturbations offers a unified framework to generate sparse adversarial perturbations. We provide its pseudo-code in Algorithm 2.

4.3 sAA for Structured Sparse Perturbations

Similar to Section 3.2, we also combine $\mathrm{sPGD}_{\mathrm{unproj}}$, $\mathrm{sPGD}_{\mathrm{proj}}$ and Sparse-RS to build Sparse-AutoAttack (sAA) for reliable evaluation of robustness against structured sparse perturbations. We extended the original version of Sparse-RS so that it can generate structured sparse perturbations with any customized patterns. Specifically, we just apply a pattern mask to the patches generated by Sparse-RS.

5 Adversarial Training

In addition to adversarial attacks, we explore adversarial training to build robust models against sparse perturbations. In the framework of adversarial training, the attack is used to generate

Algorithm 2 Sparse-PGD for Structured Sparse Perturbations

Input: Clean image: $\boldsymbol{x} \in [0,1]^{h \times w \times c}$; Model parameters: $\boldsymbol{\theta}$; Customized pattern kernel: $\boldsymbol{k} \in \{0,1\}^{r \times r}$; Stride: $s = 1$; Max iteration number: T; Tolerance: t; group l_0 budget: ϵ; Step size: α, β; Small constant: $\gamma = 2 \times 10^{-8}$
Random initialize $\boldsymbol{p}$ and $\widetilde{\boldsymbol{v}}$
$\boldsymbol{v} = \Pi_{\mathcal{S}_{\boldsymbol{v}}}(\sigma(\widetilde{\boldsymbol{v}}))$ where $\mathcal{S}_{\boldsymbol{v}} = \{\boldsymbol{v} | \|\boldsymbol{v}\|_0 \leq \epsilon\}$.
$\boldsymbol{m} = \min\left(\text{TConv}(\boldsymbol{v}, \boldsymbol{k}, s), 1\right)$
for $i = 0, 1, ..., T-1$ **do**
  Calculate the loss $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{p} \odot \boldsymbol{m})$
  if unprojected **then**
    $g_{\boldsymbol{p}} = \nabla_{\boldsymbol{\delta}} \mathcal{L} \odot \min\left(\text{TConv}(\sigma(\widetilde{\boldsymbol{v}}), \boldsymbol{k}, s), 1\right)$
  else
    $g_{\boldsymbol{p}} = \nabla_{\boldsymbol{\delta}} \mathcal{L} \odot \boldsymbol{m}$
  end if
  $g_{\widetilde{\boldsymbol{v}}} = \text{Conv}(\nabla_{\boldsymbol{m}} \mathcal{L}, \boldsymbol{k}, s) \odot \sigma'(\widetilde{\boldsymbol{v}})$
  $\boldsymbol{p} = \Pi_{\mathcal{S}_{\boldsymbol{p}}}(\boldsymbol{p} + \alpha \cdot \texttt{sign}(g_{\boldsymbol{p}}))$
  $\boldsymbol{d} = g_{\widetilde{\boldsymbol{v}}} / \|g_{\widetilde{\boldsymbol{v}}}\|_2$ **if** $||g_{\widetilde{\boldsymbol{v}}}||_2 \geq \gamma$ **else** 0
  $\boldsymbol{v}_{old}$, $\widetilde{\boldsymbol{v}} = \boldsymbol{v}$, $\widetilde{\boldsymbol{v}} + \beta \cdot \boldsymbol{d}$
  $\boldsymbol{v} = \Pi_{\mathcal{S}_{\boldsymbol{v}}}(\sigma(\widetilde{\boldsymbol{v}}))$
  $\boldsymbol{m} = \min\left(\text{TConv}(\boldsymbol{v}, \boldsymbol{k}, s), 1\right)$
  if attack succeeds **then**
    break
  end if
  if $||\boldsymbol{v} - \boldsymbol{v}_{old}||_0 \leq 0$ for t consecutive iters **then**
    Random initialize $\widetilde{\boldsymbol{v}}$
  end if
end for
Output: Perturbation: $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}$

adversarial perturbation in each training iteration, so the attack algorithm should not be too computationally expensive. In this regard, we run the untargeted sPGD (Algorithm 1) for 20 iterations to generate sparse adversarial perturbations during training. We incorporate sPGD in the framework of vanilla adversarial training [4] and TRADES [25] and name corresponding methods **sAT** and **sTRADES**, respectively. Note that we use sAT and sTRADES as two examples of applying sPGD to adversarial training since sPGD can be incorporated into any other adversarial training variant as well. To accommodate the scenario of adversarial training, we make the following modifications to sPGD.

Random Backward Function: Since the sparse gradient and the unprojected gradient as described in Section 3.1 induce different exploration-exploitation trade-offs, we randomly select one of them to generate adversarial perturbations for each mini-batch when using sPGD to generate adversarial perturbations. Compared with mixing these two backward functions together, as in sAA, random backward function does not introduce computational overhead.

Multi-ϵ Strategy: Inspired by l_1-APGD [19] and Fast-EG-l_1 [20], multi-ϵ strategy is adopted to strengthen the robustness of model. That is, we use a larger sparsity threshold, i.e., ϵ in Algorithm 1, in the training phase than in the test phase.

Higher Tolerance for Reinitialization: The default tolerance for reinitialization in sPGD is 3 iterations, which introduces strong stochasticity. However, in the realm of adversarial training, we have a limited number of iterations. As a result, the attacker should focus more on the exploitation ability to ensure the strength of the generated adversarial perturbations. While stochasticity introduced by frequent reinitialization hurts exploitation, we find a higher tolerance for reinitialization improves the performance. In practice, we set the tolerance to 10 iterations in adversarial training.

6 Experiments

In this section, we conduct extensive experiments to compare our attack methods with baselines in evaluating the robustness of various models against l_0 bounded and structured sparse perturbations. Besides the effectiveness with an abundant query budget, we also study the efficiency of our methods. Our results demonstrate that our sPGD performs best among white-box attacks. With limited iterations, sPGD achieves significantly better performance than existing methods. Therefore, sAA, consisting of the best white-box and black-box attacks, has the best attack success rate. sPGD, due to its efficiency, is utilized for adversarial training to obtain the best robust models against l_0 bounded and structured sparse adversarial perturbations. To further demonstrate the efficacy of our method, we evaluate the transferability of sPGD. The results show that sPGD has a high transfer success rate, making it applicable in practical scenarios. The adversarial examples generated by our methods are presented in Sec. 6.7. In addition, we conduct ablation studies for analysis in C. Implementation details are deferred to Appendix A.

6.1 Robustness against Unstructured Sparse Perturbations

First, we compare our proposed sPGD, including $\text{sPGD}_{\text{proj}}$ (sPGD_{p}) and $\text{sPGD}_{\text{unproj}}$ (sPGD_{u}) as defined in Section 3.2, and sAA with existing white-box and black-box attacks that generate l_0 bounded sparse perturbations. We evaluate different attack methods based on the models trained on CIFAR-10 [64] and report the robust accuracy with $\epsilon = 20$ on the whole test set in Table 1. Additionally, the results on ImageNet100 [65] and a real-world traffic sign dataset GTSRB [66] are reported in Table 2. Note that the image sizes in GTSRB vary from 15×15 to 250×250. For convenience, we resize them to 224×224 and use the same model architecture as in ImageNet-100. Furthermore, only the training set of GTSRB has annotations, we manually split the original training set into a test set containing 1000 instances and a new training set containing the rest data. In Appendix B, we report more results on CIFAR-10 with $\epsilon = 10$, $\epsilon = 15$, and the results of models trained on CIFAR-100 [64] in Table 7, 8, 9, respectively, to demonstrate the efficacy of our methods.

Models: We select various models to comprehensively evaluate their robustness against l_0 bounded perturbations. As a baseline, we train a ResNet-18 (RN-18) [67] model on clean inputs. For adversarially trained models, we select competitive models that are publicly available, including those trained against l_∞, l_2 and l_1 bounded perturbations. For the l_∞ case, we include adversarial training with the generated data (GD) [46], the proxy distributions (PORT) [44], the decoupled KL divergence loss (DKL) [48] and strong diffusion models (DM) [49]. For the l_2 case, we include adversarial training with the proxy distributions (PORT) [44], strong diffusion models (DM) [49], helper examples (HAT) [47] and strong data augmentations (FDA) [45]. The l_1 case is less explored in the literature, so we only include l_1-APGD adversarial training [19] and the efficient Fast-EG-l_1 [20] for comparison. The network architecture used in these baselines is either ResNet-18 (RN-18),

Table 1: Robust accuracy of various models on different sparse attacks, where the sparsity level $\epsilon = 20$. The models are trained on **CIFAR-10**. Note that we report results of Sparse-RS (RS) with fine-tuned hyperparameters, which outperforms its original version in Croce et al. [17]. CornerSearch (CS) is evaluated on 1000 samples due to its high computational complexity.

Model	Network	Clean	Black		White					**sAA**
			CS	RS	SF	PGD_0	SAIF	**sPGD$_p$**	**sPGD$_u$**	
Vanilla	RN-18	93.9	1.2	0.0	17.5	0.4	3.2	0.0	0.0	**0.0**
l_∞-adv. trained, $\epsilon = 8/255$										
GD	PRN-18	87.4	26.7	6.1	52.6	25.2	40.4	9.0	15.6	**5.3**
PORT	RN-18	84.6	27.8	8.5	54.5	21.4	42.7	9.1	14.6	**6.7**
DKL	WRN-28	92.2	33.1	7.0	54.0	29.3	41.1	9.9	15.8	**6.1**
DM	WRN-28	92.4	32.6	6.7	49.4	26.9	38.5	9.9	15.1	**5.9**
l_2-adv. trained, $\epsilon = 0.5$										
HAT	PRN-18	90.6	34.5	12.7	56.3	22.5	49.5	9.1	8.5	**7.2**
PORT	RN-18	89.8	30.4	10.5	55.0	17.2	48.0	6.3	5.8	**4.9**
DM	WRN-28	95.2	43.3	14.9	59.2	31.8	59.6	13.5	12.0	**10.2**
FDA	WRN-28	91.8	43.8	18.8	64.2	25.5	57.3	15.8	19.2	**14.1**
l_1-adv. trained, $\epsilon = 12$										
l_1-APGD	PRN-18	80.7	32.3	25.0	65.4	39.8	55.6	17.9	18.8	**16.9**
Fast-EG-l_1	PRN-18	76.2	35.0	24.6	60.8	37.1	50.0	18.1	18.6	**16.8**
l_0-adv. trained, $\epsilon = 20$										
PGD_0-A	PRN-18	77.5	16.5	2.9	62.8	56.0	47.9	9.9	21.6	**2.4**
PGD_0-T	PRN-18	90.0	24.1	4.9	85.1	61.1	67.9	27.3	37.9	**4.5**
sAT	PRN-18	84.5	52.1	36.2	81.2	78.0	76.6	75.9	75.3	**36.2**
sTRADES	PRN-18	89.8	69.9	61.8	88.3	86.1	84.9	84.6	81.7	**61.7**

PreActResNet-18 (PRN-18) [68] or WideResNet-28-10 (WRN-28) [69]. For the l_0 case, we evaluate PGD_0 [13] in vanilla adversarial training (PGD_0-A) and TRADES (PGD_0-T) using the same hyper-parameter settings as in Croce and Hein [13]. Since other white-box sparse attacks present trivial performance in adversarial training, we do not include their results. Finally, we use our proposed sPGD in vanilla adversarial training (sAT) and TRADES (sTRADES) to obtain PRN-18 models to compare with these baselines.

Attacks: We compare our methods with various existing black-box and white-box attacks that generate l_0 bounded perturbations. The black-box attacks include CornerSearch (CS) [13] and Sparse-RS (RS) [17]. The white-box attacks include SparseFool (SF) [12], PGD_0 [13] and Sparse Adversarial and Interpretable Attack Framework (SAIF) [41]. The implementation details of each attack are deferred to Appendix A. Specifically, to exploit the strength of these attacks in reasonable running time, we run all these attacks for either 10000 iterations or the number of iterations where their performances converge. Note that, the number of iterations for all these attacks are no smaller than their default settings. In addition, we report the results of RS with fine-tuned hyperparameters, which outperforms its default settings in [17]. Finally, we report the robust accuracy under CS attack based on only 1000 random test instances due to its prohibitively high computational complexity.

Based on the results in Table 1 and Table 2, we can find that SF attack, PGD_0 attack and SAIF attack perform significantly worse than our methods for all the models studied. That is, our proposed sPGD always performs the best among white-box attacks. Among black-box attacks, CS

Table 2: Robust accuracy of various models on different sparse attacks. Our sAT model is trained with $\epsilon = 1200$. **(a)** Results on **ImageNet-100** [65]. **(b)** Results on **GTSRB** [66]. Note that the results of Sparse-RS (RS) with tuned hyperparameters are reported. All models are RN-34, and are evaluated on 500 samples. The results of SparseFool (SF) and PGD_0 are included due to their poor performance, and CornerSearch (CS) is not evaluated here due to its high computational complexity, i.e. nearly 1 week on one GPU for each run.

(a) **ImageNet-100,** $\epsilon = 200$

Model	Clean	Black RS	White SAIF	White **sPGD$_p$**	White **sPGD$_u$**	**sAA**
Vanilla	83.0	0.2	0.6	0.2	0.4	**0.0**
l_0-adv. trained, $\epsilon = 200$						
PGD_0	76.0	6.8	11.0	1.8	18.8	**1.8**
sAT	86.2	61.4	69.0	78.0	77.8	**61.2**

(b) **GTSRB,** $\epsilon = 600$

Model	Clean	Black RS	White SAIF	White **sPGD$_p$**	White **sPGD$_u$**	**sAA**
Vanilla	99.9	18.0	9.5	**0.3**	**0.3**	**0.3**
l_0-adv. trained, $\epsilon = 600$						
PGD_0	99.8	37.6	13.8	**0.0**	**0.0**	**0.0**
sAT	99.8	88.4	96.2	88.6	96.2	**85.4**

attack can achieve competitive performance, but it runs dozens of times longer than our method does. Therefore, we focus on comparing our method with RS attack. For l_1 and l_2 models, our proposed sPGD significantly outperforms RS attack. By contrast, RS attack outperforms sPGD for l_∞ and l_0 models. This gradient masking phenomenon is, in fact, prevalent across sparse attacks. Given sufficient iterations, RS outperforms all other existing white-box attacks for l_∞ and l_0 models. Nevertheless, among white-box attacks, sPGD exhibits the least susceptibility to gradient masking and has the best performance. The occurrence of gradient masking in the context of l_0 bounded perturbations can be attributed to the non-convex nature of adversarial budgets. In practice, the perturbation updates often significantly deviate from the direction of the gradients because of the projection to the non-convex set. Similar to AA in the l_1, l_2 and l_∞ cases, sAA consists of both white-box and black-box attacks for comprehensive robustness evaluation. It achieves the best performance in all cases in Table 1 and Table 2 by a considerable margin.

In the case of l_0 adversarial training, the models are adversarially trained against sparse attacks. However, Figure 3 illustrates that the performance of RS attack drastically deteriorates with limited iterations (e.g., smaller than 100), so RS is not suitable for adversarial training where we need to generate strong adversarial perturbations in limited number iterations. Empirical evidence suggests that employing RS with 20 iterations for adversarial training, i.e., the same number of iterations as in other methods, yields trivial performance, so it is not included in Table 1 or Table 2 for comparison. In addition, models trained by PGD_0-A and PGD_0-T, which generate l_0 bounded perturbations, exhibit poor robustness to various attack methods, especially sAA. By contrast, the models trained by sAT and sTRADES show the strongest robustness, indicated by the comprehensive sAA method and all other attack methods. Compared with sAT, sTRADES achieves better performance in both robustness and accuracy. Finally, models trained with l_1 bounded perturbations are the most robust ones among existing non-l_0 training methods. It could be attributed to the fact that l_1 norm is the tightest convex relaxation of l_0 norm [70]. From a qualitative perspective, l_1 attacks also generate relatively sparse perturbations [20], which makes the corresponding model robust to sparse perturbations to some degree.

Our results indicate sPGD and RS can complement each other. Therefore, sAA, an AutoAttack-style attack that ensembles both attacks achieves the state-of-the-art performance on all models. It is designed to have a similar computational complexity to AutoAttack in l_∞, l_2 and l_1 cases.

6.2 Robustness against Structured Sparse Perturbations

Table 3: Robust accuracy of various models on different structured sparse attacks. **(a)** Results on **CIFAR-10**, all models are PRN-18. **(b)** Results on **ImageNet-100**, the test set contains 500 samples, all models are RN-34. Note that the evaluated group sparsity levels are approximately equivalent to the l_0 sparsity levels used in Table 1 and 2, and the Sparse-RS (RS) used here is the proposed extended version.

(a) **CIFAR-10**

Model	Clean	Black	White			
		RS	LOAP	**sPGD$_p$**	**sPGD$_u$**	**sAA**
row, $\epsilon = 1$						
Vanilla	93.9	22.5	-	1.0	1.3	**1.0**
sTRADES-l_0	89.8	83.1	-	54.6	65.0	**53.9**
sTRADES-row	89.3	85.9	-	77.3	83.7	**77.3**
3×3 patch, $\epsilon = 2$						
Vanilla	93.9	6.7	4.5	1.9	6.6	**1.7**
sTRADES-l_0	89.8	65.5	60.0	46.9	67.5	**46.7**
sTRADES-p3x3	87.9	77.5	81.4	72.8	81.7	**72.3**
5×5 patch, $\epsilon = 1$						
Vanilla	93.9	7.5	2.3	1.7	3.2	**1.7**
sTRADES-l_0	89.8	56.4	35.0	26.7	44.3	**26.7**
sTRADES-p5x5	89.5	72.9	73.1	56.0	73.3	**55.9**

(b) **ImageNet-100**

Model	Clean	Black	White-Box			
		RS	LOAP	**sPGD$_p$**	**sPGD$_u$**	**sAA**
row, $\epsilon = 1$						
Vanilla	83.0	54.8	-	5.2	8.0	**5.2**
sAT-l_0	86.2	78.6	-	65.8	73.8	**65.8**
sAT-row	81.0	78.2	-	76.6	78.4	**76.6**
10×10 patch, $\epsilon = 2$						
Vanilla	83.0	13.0	4.6	4.6	6.4	**4.0**
sAT-l_0	86.2	27.8	10.8	10.6	33.0	**10.0**
sAT-p10x10	81.6	64.0	62.8	35.8	73.2	**35.8**
14×14 patch, $\epsilon = 1$						
Vanilla	83.0	7.5	2.3	1.7	12.0	**1.7**
sAT-l_0	86.2	28.3	**10.2**	15.4	34.8	14.6
sAT-p14x14	80.0	57.2	73.0	41.2	71.2	**39.6**

Apart from unstructured sparse perturbations, we evaluate the effectiveness of our method in Algorithm 2 in generating structured sparse perturbations in this subsection. Given the paucity of available methods for comparison, we focus on comparing our method with the extended Sparse-RS (RS) and LOAP [16], which is a white-box approach generating adversarial patches, on CIFAR-10 and ImageNet-100. For a comprehensive evaluation, we include the results of different types of structured sparse perturbations, e.g., row and patches with different group sparsity level and different sizes. It should be noted that the unstructured l_0 norms of the structured perturbations studied here are approximately the same as the perturbations in Table 1 and 2. In addition, we evaluate the attacks on vanilla models, models trained with unstructured l_0 bounded perturbations, and those trained with the specific structured sparse perturbations.

The results in Table 3 indicate that the proposed sAA also achieves the state-of-the-art performance in generating structured sparse perturbations in almost all cases. Furthermore, the models trained with our methods exhibit the strongest robustness against structured sparse perturbations. As anticipated, models trained on specific structured sparse adversarial examples significantly outperform those trained on unstructured sparse adversarial examples. This highlights the inherent limitation of adversarially trained models in maintaining robustness against unseen types of perturbations during training.

6.3 Adversarial Watermarks

Although only a few pixels are perturbed in sparse attacks, such perturbations are still perceptible due to their unconstrained magnitude. In this subsection, we additionally constrain the l_∞ norm of magnitude $\boldsymbol{p}$, meaning that the feasible set for vp is rewritten as $\mathcal{S}_{\boldsymbol{p}} = \{\boldsymbol{p} \in \mathbb{R}^{h \times w \times c} | \|\boldsymbol{p}\|_\infty \leq \epsilon_\infty,\ 0 \leq \boldsymbol{x} + \boldsymbol{p} \leq 1\}$. Combining with structured sparsity constraint, we can generate alleged *adversarial watermarks* with our method.

Table 4: Robust accuracy of vanilla model (RN-34) against adversarial watermarks with different l_∞ adversarial budgets. The experiment is conducted on ImageNet-100.

(a) 18×18 circle

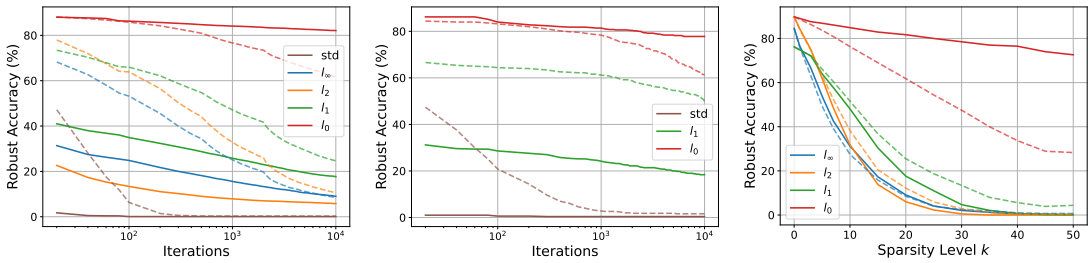
ϵ_∞	8/255	16/255	32/255	64/255
RS	59.0	44.6	32.4	26.4
sPGD$_{\text{p}}$	47.2	25.4	16.6	9.0
sPGD$_{\text{u}}$	56.4	39.6	25.8	16.8
sAA	**44.2**	**24.0**	**15.6**	**8.8**

(b) 60×60 letter "A"

ϵ_∞	8/255	16/255	32/255	64/255
RS	14.2	12.8	9.2	5.8
sPGD$_{\text{p}}$	3.8	2.6	2.4	2.2
sPGD$_{\text{u}}$	7.4	4.8	2.8	2.0
sAA	**2.0**	**0.8**	**0.8**	**0.6**

To evaluate the performance of our method in generating adversarial watermarks, we conduct experiments with different patterns and different $_\infty$ adversarial budgets. From Table 4, we can find that when the size of the pattern is large, like 60×60 letter "A" with 1747 non-zero elements, our attack can still achieve decent performance under different l_∞ adversarial budgets. In contrast, when we adopt a 18×18 circle pattern with 208 non-zero elements, the attack success rate abruptly declines as the l_∞ adversarial budget decreases. Nevertheless, from the aspect of attack success rate, our approaches still outperform Sparse-RS by a large margin.

6.4 Comparison under Different Iteration Numbers and Different Sparsity Levels



(a) CIFAR-10, $\epsilon = 20$ (b) ImageNet-100, $\epsilon = 200$ (c) CIFAR-10, $t = 10000$

Figure 3: Comparison between sPGD and RS attack under different iteration numbers and different sparsity levels. **(a)** Different iteration number comparison on CIFAR-10, $\epsilon = 20$. ResNet18 (std), PORT (l_∞ and l_2) [44], l_1-APGD (l_1) [19] and sTRADES (l_0) are evaluated. **(b)** Different iteration number comparison on ImageNet-100, $\epsilon = 200$. ResNet34 (std), Fast-EG-l_1 (l_1) [20] and sAT (l_0) are evaluated. In (a) and (b), the total iteration number ranges from 20 and 10000. For better visualization, the x-axis is in the log scale. **(c)** Different sparsity comparison on CIFAR-10. The evaluated models are the same as those in (a). The ϵ ranges from 0 and 50. The number of total iterations is set to 10000. **Note that the results of sPGD and RS attack are shown in solid lines and dotted lines, respectively.**

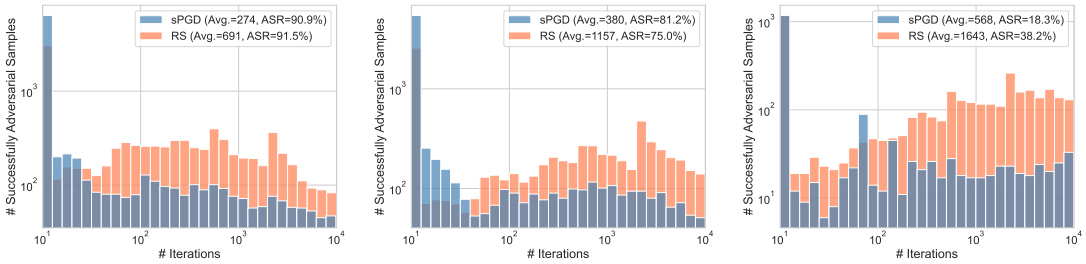
In this subsection, we further compare our method sPGD, which is a white-box attack, with RS attack, the strongest black-box attack in the previous section. Specifically, we compare them under various iteration numbers on CIFAR-10 and ImageNet-100, which have different resolutions. In addition, we also compare sPGD and RS under different sparsity levels on CIFAR-10.

As illustrated in Figure 3 (a) and (b), sTRADES has better performance than other robust models by a large margin in all iterations of both sPGD and RS attacks, which is consistent with

the results in Table 1, 2 and 9. For vanilla and other robust models, although the performances of both sPGD and RS attack get improved with more iterations, sPGD outperforms RS attack by a large margin when the iteration number is small (e.g. $<$ 1000 iterations), which makes it feasible for adversarial training. Similar to other black-box attacks, the performance of RS attack drastically deteriorates when the query budget is limited. In addition, our proposed gradient-based sPGD significantly outperforms RS on ImageNet-100, where the search space is much larger than that on CIFAR-10, i.e., higher image resolution and higher sparsity level ϵ. This suggests that our approach is scalable and shows higher efficiency on high-resolution images. Furthermore, although the performance of RS does not converge even when the iteration number reaches 10000, a larger query budget will make it computationally impractical. Following the setting in Croce et al. [17], we do not consider a larger query budget in our experiments, either.

Furthermore, we can observe from Figure 3 (c) that RS attack shows slightly better performance only on the l_∞ model and when ϵ is small. The search space for the perturbed features is relatively small when ϵ is small, which facilitates heuristic black-box search methods like RS attack. As ϵ increases, sPGD outperforms RS attack in all cases until both attacks achieve almost 100% attack success rate.

6.5 Efficiency of sPGD



(a) l_∞ PORT [44] (b) l_1-APGD [19] (c) sTRADES

Figure 4: Distribution of the iteration numbers needed by sPGD (blue) and RS (orange) to successfully generate adversarial samples. The results are obtained from different models: **(a)** l_∞ PORT [44], **(b)** l_1-APGD [19] and **(c)** our sPGD. The average iteration numbers (Avg.) and attack success rate (ASR), i.e., 1−Robust Acc., are reported in the legend. For better visualization, we clip the minimum iteration number to 10 and show the x- and y-axis in log scale.

To further showcase the efficiency of our approach, we first compare the distributions of the number of iterations needed by sPGD and RS to successfully generate adversarial samples. Figure 4 illustrates the distribution of the iteration numbers needed by sPGD and RS to successfully generate adversarial samples. For l_∞ and l_1 robust models, our proposed sPGD consumes distinctly fewer iteration numbers to successfully generate an adversarial sample than RS, the strongest black-box attack in Table 1, while maintaining a high attack success rate. Similar to the observations in Figure 3, the model

Table 5: Runtime of different attacks on 1000 test instances with batch size 500. The sparsity level $\epsilon = 20$. The evaluated model is sTRADES. The model is trained on **CIFAR-10**. The experiments are implemented on NVIDIA Tesla V100.

Attack	CS	RS	SF	PGD_0
Runtime	604 min	59 min	92 min	750 min
Attack	**SAIF**	**$sPGD_p$**	**$sPGD_u$**	**sAA**
Runtime	122 min	**42 min**	45 min	148 min

trained by sTRADES suffers from gradient masking. However, RS still requires a large query budget to successfully generate an adversarial sample. This further demonstrates the efficiency of sPGD, which makes adversarial training feasible.

Additionally, we compare the runtime of sPGD with other attacks in the same configuration as in Table 1. As shown in Table 5, the proposed sPGD shows the highest efficiency among various attacks. Although sAA consumes more time (approximately 2× sPGD + RS), it can provide a reliable evaluation against l_0 bounded perturbation.

6.6 Transferability of Adversarial Perturbations

To evaluate the transferability of our attack across different models and architectures, we generate adversarial perturbations based on one model and report the attack success rate (ASR) on another model. As shown in Table 6, sPGD exhibits better transferability than the most competitive baseline, Sparse-RS (RS) [17]. This further demonstrates the effectiveness of our method and its potential application in more practical scenarios.

Table 6: Transferability of RS and sPGD between vanilla VGG11 (V) and RN-18 (R) on CIFAR-10 with $\epsilon = 20$. Attack success rate (ASR) is reported. The perturbations are generated on the source model (left), and are evaluated on the target model (right). Note that α and β of sPGD are set to 0.75.

	V→V	V→R	R→R	R→V
RS	53.9	28.4	33.0	37.4
sPGD_p	58.0	**43.9**	50.8	48.0
sPGD_u	**64.9**	40.0	**52.7**	**56.7**

6.7 Visualization

We show some adversarial examples with different types of sparse perturbations in Figure 5, including l_0 bounded perturbations, row-wise perturbations, star-like perturbations, and adversarial watermarks of letter "A". The attack is sPGD, and the model is vanilla ResNet-34 trained on ImageNet-100. More adversarial examples are shown in Appendix D.

7 Conclusion

In this paper, we propose an effective and efficient white-box attack named sPGD to generate both unstructured and structured perturbations. sPGD obtains the state-of-the-art performance among white-box attacks. Based on this, we combine it with black-box attacks for more comprehensive and reliable evaluation of robustness against sparse perturbations. Our proposed sPGD is particularly effective in the realm of limited iteration numbers. Due to its efficiency, we incorporate sPGD into the framework of adversarial training to obtain robust models against sparse perturbations. The models trained with our method demonstrate the best robust accuracy.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *Computer Science*, 2013.

[2] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. 2016.

(a) l_0, $\epsilon = 200$ (b) l_0, $\epsilon = 200$ (c) row, $\epsilon = 1$ (d) row, $\epsilon = 1$

(e) star, 25×25, $\epsilon = 1$ (f) star, 25×25, $\epsilon = 1$ (g) "A", 60×60, $\epsilon_\infty = 32/255$ (h) "A", 60×60, $\epsilon_\infty = 32/255$

Figure 5: Visualization of different sparse adversarial examples in ImageNet-100. The model is vanilla ResNet-34. The predictions before (left) and after (right) attack are (a) cock→hen, (b) triceratops→spotted salamander, (c) great white shark→tench, (d) loggerhead→leatherback turtle, (e) black and gold garden spider→garden spider, (f) lorikeet→macaw, (g) bee eater→coucal, and (h) banded gecko→alligator lizard. **Note that the red squares in (g) and (h) are just for highlighting the perturbation position and not part of perturbations.**

[3] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *Computer Science*, 2014.

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

[5] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. 2019.

[6] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[7] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[8] Naveed Akhtar and Ajmal S. Mian. Threat of adversarial attacks on deep learning in computer

vision: A survey. *IEEE Access*, 6:14410–14430, 2018. URL https://api.semanticscholar.org/CorpusID:3536399.

[9] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17:151 – 178, 2019. URL https://api.semanticscholar.org/CorpusID:202660800.

[10] Ryan Feng, Neal Mangaokar, Jiefeng Chen, Earlence Fernandes, Somesh Jha, and Atul Prakash. Graphite: Generating automatic physical examples for machine-learning attacks on computer vision systems. In *2022 IEEE 7th European symposium on security and privacy (EuroS&P)*, pages 664–683. IEEE, 2022.

[11] Xingxing Wei, Yao Huang, Yitong Sun, and Jie Yu. Unified adversarial patch for cross-modal attacks in the physical world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4445–4454, 2023.

[12] Apostolos Modas, Seyed Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: a few pixels make a big difference. 2018.

[13] Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4724–4732, 2019.

[14] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[15] Xiaoyi Dong, Dongdong Chen, Jianmin Bao, Chuan Qin, Lu Yuan, Weiming Zhang, Nenghai Yu, and Dong Chen. Greedyfool: Distortion-aware sparse adversarial attack. 2020.

[16] Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *European conference on computer vision*, pages 429–448. Springer, 2020.

[17] Francesco Croce, Maksym Andriushchenko, Naman D Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6437–6445, 2022.

[18] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019.

[19] Francesco Croce and Matthias Hein. Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers. In *International Conference on Machine Learning*, pages 2201–2211. PMLR, 2021.

[20] Yulun Jiang, Chen Liu, Zhichao Huang, Mathieu Salzmann, and Sabine Süsstrunk. Towards stable and efficient adversarial training against l_1 bounded adversarial attacks. In *International Conference on Machine Learning*. PMLR, 2023.

[21] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *NeurIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.

[22] Danny Karmon, Daniel Zoran, and Yoav Goldberg. LaVAN: Localized and visible adversarial noise. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2507–2515. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/karmon18a.html.

[23] Chenglin Yang, Adam Kortylewski, Cihang Xie, Yinzhi Cao, and Alan Yuille. Patchattack: A black-box texture-based attack with reinforcement learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 681–698, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58574-7.

[24] Francis Bach. Structured sparsity-inducing norms through submodular functions. *Advances in Neural Information Processing Systems*, 23, 2010.

[25] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. *ArXiv*, abs/1901.08573, 2019. URL https://api.semanticscholar.org/CorpusID:59222747.

[26] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane S. Boning, Inderjit S. Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *ArXiv*, abs/1804.09699, 2018. URL https://api.semanticscholar.org/CorpusID:13750928.

[27] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BJm4T4Kgx.

[28] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Xiaodong Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *European Conference on Computer Vision*, 2018. URL https://api.semanticscholar.org/CorpusID:52951839.

[29] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:5046541.

[30] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *ArXiv*, abs/1807.07978, 2018. URL https://api.semanticscholar.org/CorpusID:49907212.

[31] Chun-Chen Tu, Pai-Shun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:44079102.

[32] Jonathan Uesato, Brendan O'Donoghue, Aäron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *ArXiv*, abs/1802.05666, 2018. URL https://api.semanticscholar.org/CorpusID:3639844.

[33] Moustafa Farid Alzantot, Yash Sharma, Supriyo Chakraborty, and Mani B. Srivastava. Genattack: practical black-box attacks with gradient-free optimization. *Proceedings of the Genetic*

and Evolutionary Computation Conference, 2018. URL https://api.semanticscholar.org/CorpusID:44166696.

[34] Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q. Weinberger. Simple black-box adversarial attacks. *ArXiv*, abs/1905.07121, 2019. URL https://api.semanticscholar.org/CorpusID:86541092.

[35] Abdullah Al-Dujaili and Una-May O'Reilly. There are no bit parts for sign bits in black-box attacks. *ArXiv*, abs/1902.06894, 2019. URL https://api.semanticscholar.org/CorpusID:67749599.

[36] Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. *ArXiv*, abs/1905.06635, 2019. URL https://api.semanticscholar.org/CorpusID:155100229.

[37] Laurent Meunier, Jamal Atif, and Olivier Teytaud. Yet another but more efficient black-box adversarial attack: tiling and evolution strategies. *ArXiv*, abs/1910.02244, 2019. URL https://api.semanticscholar.org/CorpusID:203837562.

[38] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. *ArXiv*, abs/1912.00049, 2019. URL https://api.semanticscholar.org/CorpusID:208527215.

[39] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[40] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[41] Tooba Imtiaz, Morgan Kohler, Jared Miller, Zifeng Wang, Mario Sznaier, Octavia Camps, and Jennifer Dy. Saif: Sparse adversarial and interpretable attack framework. *arXiv preprint arXiv:2212.07495*, 2022.

[42] Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[43] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. *ArXiv*, abs/1907.02044, 2019. URL https://api.semanticscholar.org/CorpusID:195791557.

[44] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? *arXiv preprint arXiv:2104.09425*, 2021.

[45] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy A. Mann. Fixing data augmentation to improve adversarial robustness. *CoRR*, abs/2103.01946, 2021.

[46] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A. Mann. Improving robustness using generated data. *CoRR*, abs/2110.09468, 2021.

[47] Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021. URL https://openreview.net/forum?id=BuD2LmNaU3a.

[48] Jiequan Cui, Zhuotao Tian, Zhisheng Zhong, Xiaojuan Qi, Bei Yu, and Hanwang Zhang. Decoupled kullback-leibler divergence loss. *ArXiv*, abs/2305.13948, 2023. URL https://api.semanticscholar.org/CorpusID:258841423.

[49] Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. Better diffusion models further improve adversarial training. *ArXiv*, abs/2302.04638, 2023. URL https://api.semanticscholar.org/CorpusID:256697167.

[50] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:3310672.

[51] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Neural Information Processing Systems*, 2019. URL https://api.semanticscholar.org/CorpusID:139102395.

[52] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Neural Information Processing Systems*, 2019. URL https://api.semanticscholar.org/CorpusID:146120969.

[53] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. *ArXiv*, abs/2001.03994, 2020. URL https://api.semanticscholar.org/CorpusID:210164926.

[54] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, and R. Venkatesh Babu. Towards efficient and effective adversarial training. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:245261076.

[55] Peilin Kang and Seyed-Mohsen Moosavi-Dezfooli. Understanding catastrophic overfitting in adversarial training. *ArXiv*, abs/2105.02942, 2021. URL https://api.semanticscholar.org/CorpusID:234093560.

[56] Hoki Kim, Woojin Lee, and Jaewook Lee. Understanding catastrophic overfitting in single-step adversarial training. In *AAAI Conference on Artificial Intelligence*, 2020. URL https://api.semanticscholar.org/CorpusID:222133879.

[57] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *ArXiv*, abs/2007.02617, 2020. URL https://api.semanticscholar.org/CorpusID:220363591.

[58] Zeinab Golgooni, Mehrdad Saberi, Masih Eskandar, and Mohammad Hossein Rohban. Zerograd : Mitigating and explaining catastrophic overfitting in fgsm adversarial training. *ArXiv*, abs/2103.15476, 2021. URL https://api.semanticscholar.org/CorpusID:232404666.

[59] Pau de Jorge, Adel Bibi, Riccardo Volpi, Amartya Sanyal, Philip H. S. Torr, Grégory Rogez, and Puneet Kumar Dokania. Make some noise: Reliable and efficient single-step adversarial training. *ArXiv*, abs/2202.01181, 2022. URL https://api.semanticscholar.org/CorpusID:246473010.

[60] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[61] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11893–11902, 2020.

[62] Yonggan Fu, Qixuan Yu, Yang Zhang, Shang Wu, Xu Ouyang, David Cox, and Yingyan Lin. Drawing robust scratch tickets: Subnetworks with inborn robustness are found within randomly initialized networks. *Advances in Neural Information Processing Systems*, 34:13059–13072, 2021.

[63] Chen Liu, Ziqi Zhao, Sabine Süsstrunk, and Mathieu Salzmann. Robust binary models by pruning randomly-initialized networks. *Advances in Neural Information Processing Systems*, 35:492–506, 2022.

[64] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[65] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[66] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32: 323–332, 2012.

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

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian 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*, pages 630–645. Springer, 2016.

[69] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[70] Thomas Bittar, Jean-Philippe Chancelier, and Michel De Lara. Best convex lower approximations of the l 0 pseudonorm on unit balls. 2021. URL https://api.semanticscholar.org/CorpusID:235254019.

[71] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. *Advances in neural information processing systems*, 13, 2000.

A Implementation Details

In experiments, we mainly focus on the cases of the sparsity of perturbations $\epsilon = 10, 15$ and 20, where $\epsilon = ||\sum_{i=1}^{c} \boldsymbol{\delta}^{(i)}||_0$ or $||\boldsymbol{m}||_0$, $\boldsymbol{\delta}^{(i)} \in \mathbb{R}^{h\times w}$ is the i-th channel of perturbation $\boldsymbol{\delta} \in \mathbb{R}^{h\times w\times c}$, and $\boldsymbol{m} \in \mathbb{R}^{h\times w\times 1}$ is the sparsity mask in the decomposition of $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}$, $\boldsymbol{p} \in \mathbb{R}^{h\times w\times c}$ denotes the magnitude of perturbations.

To exploit the strength of these attacks in reasonable running time, we run all these attacks for either 10000 iterations or the number of iterations where their performances converge. More details are elaborated below.

CornerSearch [13]: For CornerSearch, we set the hyperparameters as following: $N = 100, N_{iter} = 3000$, where N is the sample size of the one-pixel perturbations, N_{iter} is the number of queries. For bot CIFAR-10 and CIFAR-100 datasets, we evaluate the robust accuracy on 1000 test instances due to its prohibitively high computational complexity.

Sparse-RS [17]: For Sparse-RS, we set $\alpha_{\text{init}} = 0.8$, which controls the set of pixels changed in each iteration. Cross-entropy loss is adopted. Following the default setting in [17], we report the results of untargeted attacks with the maximum queries up to 10000.

SparseFool [12]: We apply SparseFool following the official implementation and use the default value of the sparsity parameter $\lambda = 3$. The maximum iterations per sample is set to 3000. Finally, the perturbation generated by SparseFool is projected to the l_0 ball to satisfy the adversarial budget.

PGD$_0$ [13]: For PGD$_0$, we include both untargeted attack and targeted attacks on the top-9 incorrect classes with the highest confidence scores. We set the step size to $\eta = 120000/255$. Contrary to the default setting, the iteration numbers of each attack increase from 20 to 300. Besides, 5 restarts are adopted to boost the performance further.

SAIF [41]: Similar to PGD$_0$, we apply both untargeted attack and targeted attacks on the top-9 incorrect classes with 300 iterations per attack, however, the query budget is only 100 iterations in the original paper [41]. We adopt the same l_∞ norm constraint for the magnitude tensor p as in sPGD.

Sparse-PGD (sPGD): Cross-entropy loss is adopted as the loss function of both untargeted and targeted versions of our method. The small constant γ to avoid numerical error is set to 2×10^{-8}. The number of iterations T is 10000 for all datasets to ensure fair comparison among attacks in Table 1. For generating l_0 bounded perturbations, the step size for magnitude $\boldsymbol{p}$ is set $\alpha = 0.25\times\epsilon_\infty$; the step size for continuous mask $\widetilde{\boldsymbol{m}}$ is set $\beta = 0.25 \times \sqrt{h \times w}$, where h and w are the height and width of the input image $\boldsymbol{x} \in \mathbb{R}^{h\times w\times c}$, respectively; the tolerance for reinitialization t is set to 3. For generating structured sparse perturbations, we let $\alpha = 0.0125 \times \epsilon_\infty$, $\beta = 0.0125 \times \sqrt{h \times w}$ and $t = 50$.

Sparse-AutoAttack (sAA): It is a cascade ensemble of five different attacks, i.e., **a)** untargeted sPGD with unprojected gradient (sPGD$_{\text{u}}$), **b)** untargeted sPGD with sparse gradient (sPGD$_{\text{p}}$), and **c)** untargeted Sparse-RS. The hyper-parameters of sPGD are the same as those listed in the last paragraph.

Adversarial Training: sPGD is adopted as the attack during the training phase, the number of iterations is 20, and the backward function is randomly selected from the two different backward functions for each batch. For sTRADES, we only compute the TRADES loss when training, and generating adversarial examples is based on cross-entropy loss. We use PreactResNet18 [68] with softplus activation [71] for experiments on CIFAR-10 and CIFAR-100 [64], and ResNet34 [67] for experiments on ImageNet100 [65] and GTSRB [66]. We train the model for 100 epochs on CIFAR-10 and CIFAR-100, for 40 epochs on ImageNet100, and for 20 epochs on GTSRB. The training batch

size is 128 on CIFAR-10 and CIFAR-100, and 32 on ImageNet100 and GTSRB. The optimizer is SGD with a momentum factor of 0.9 and weight decay factor of 5×10^{-4}. The learning rate is initialized to 0.05 and is divided by a factor of 10 at the $\frac{1}{4}$ and the $\frac{3}{4}$ of the total epochs. The tolerance for reinitialization is set to 10.

B More Results of Section 6.1

In this subsection, we present the robust accuracy on CIFAR-10 with the sparsity level $\epsilon = 10$ and $\epsilon = 15$, as well as those on CIFAR-100 with $\epsilon = 10$ in Table 7, 8 and 9, respectively. The observations with different sparsity levels and on different datasets are consistent with those in Table 1 and 2, which indicates the effectiveness of our method.

Table 7: Robust accuracy of various models on different sparse attacks, where the sparsity level $\epsilon = 10$. The models are trained on **CIFAR-10** [64]. Note that we report results of Sparse-RS (RS) with tuned hyperparameters, which outperforms its original version in [17]. CornerSearch (CS) is evaluated on 1000 samples due to its high computational complexity.

Model	Network	Clean	Black		White					**sAA**
			CS	RS	SF	PGD_0	SAIF	**sPGD$_p$**	**sPGD$_u$**	
Vanilla	RN-18	93.9	3.2	0.5	40.6	11.5	31.8	0.5	7.7	**0.5**
l_∞-adv. trained, $\epsilon = 8/255$										
GD	PRN-18	87.4	36.8	24.5	69.9	50.3	63.0	31.0	37.3	**23.2**
PORT	RN-18	84.6	36.7	27.5	70.7	46.1	62.6	31.0	36.2	**25.0**
DKL	WRN-28	92.2	40.9	25.0	71.9	54.2	64.6	32.6	38.8	**23.7**
DM	WRN-28	92.4	38.7	23.7	68.7	52.7	62.5	31.2	37.4	**22.6**
l_2-adv. trained, $\epsilon = 0.5$										
HAT	PRN-18	90.6	47.3	40.4	74.6	53.5	71.4	37.3	36.7	**34.5**
PORT	RN-18	89.8	46.8	37.7	74.2	50.4	70.9	33.7	33.0	**30.6**
DM	WRN-28	95.2	57.8	47.9	78.3	65.5	80.9	47.1	48.2	**43.1**
FDA	WRN-28	91.8	55.0	49.4	79.6	58.6	77.5	46.7	49.0	**43.8**
l_1-adv. trained, $\epsilon = 12$										
l_1-APGD	PRN-18	80.7	51.4	51.1	74.3	60.7	68.1	47.2	47.4	**45.9**
Fast-EG-l_1	PRN-18	76.2	49.7	48.0	69.7	56.7	63.2	44.7	45.0	**43.2**
l_0-adv. trained, $\epsilon = 10$										
PGD_0-A	PRN-18	85.8	20.7	16.1	77.1	66.1	68.7	33.5	36.2	**15.1**
PGD_0-T	PRN-18	90.6	22.1	14.0	85.2	72.1	76.6	37.9	44.6	**13.9**
sAT	PRN-18	86.4	61.0	57.4	84.1	82.0	81.1	78.2	77.6	**57.4**
sTRADES	PRN-18	89.8	74.7	71.6	88.8	87.7	86.9	85.9	84.5	**71.6**

C Ablation Studies

We conduct ablation studies in this section. We focus on CIFAR10 and the sparsity level $\epsilon = 20$. Unless specified, we use the same configurations as in Table 1.

sPGD: We first validate the effectiveness of each component in sPGD. The result is reported in Table 10. We observe that naively

Table 10: Ablation study of each component in $sPGD_{proj}$ in terms of robust accuracy. The model is Fast-EG-l_1 trained on CIFAR-10.

Ablations	Robust Acc.
Baseline (PGD_0 w/o restart)	49.4
+ Decomposition: $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}$	58.0 (+8.6)
+ Continuous mask $\widetilde{\boldsymbol{m}}$	33.9 (-15.5)
+ Random reinitialization	**18.1 (-31.3)**

Table 8: Robust accuracy of various models on different sparse attacks, where the sparsity level $\epsilon = 15$. The models are trained on **CIFAR-10** [64]. Note that we report results of Sparse-RS (RS) with tuned hyperparameters, which outperforms its original version in [17]. CornerSearch (CS) is evaluated on 1000 samples due to its high computational complexity.

Model	Network	Clean	Black		White					**sAA**
			CS	RS	SF	PGD_0	SAIF	$\textbf{sPGD}_\text{p}$	$\textbf{sPGD}_\text{u}$	
Vanilla	RN-18	93.9	1.6	0.0	25.3	2.1	12.0	0.0	0.0	**0.0**
l_∞-adv. trained, $\epsilon = 8/255$										
GD	PRN-18	87.4	30.5	12.2	61.1	36.0	51.3	17.1	24.3	**11.3**
PORT	RN-18	84.6	30.8	15.2	62.1	31.4	52.1	17.4	23.0	**13.0**
DKL	WRN-28	92.2	35.3	13.2	62.5	41.2	52.3	18.4	24.9	**12.1**
DM	WRN-28	92.4	34.8	12.6	57.9	38.5	49.4	17.9	24.0	**11.6**
l_2-adv. trained, $\epsilon = 0.5$										
HAT	PRN-18	90.6	38.9	23.5	65.3	35.4	60.2	19.0	18.6	**16.6**
PORT	RN-18	89.8	36.8	20.6	64.3	30.6	59.7	16.0	15.5	**13.8**
DM	WRN-28	95.2	48.5	27.7	68.2	47.5	70.9	25.0	26.7	**22.1**
FDA	WRN-28	91.8	47.8	31.1	71.8	40.1	68.2	28.0	31.4	**25.5**
l_1-adv. trained, $\epsilon = 12$										
l_1-APGD	PRN-18	80.7	41.3	36.5	70.3	50.5	62.3	30.4	31.3	**29.0**
Fast-EG-l_1	PRN-18	76.2	40.7	34.8	64.9	46.7	56.9	29.6	30.1	**28.0**
l_0-adv. trained, $\epsilon = 15$										
PGD_0-A	PRN-18	83.7	17.5	6.1	73.7	62.9	60.5	19.4	27.5	**5.6**
PGD_0-T	PRN-18	90.5	19.5	7.2	85.5	63.6	69.8	31.4	41.2	**7.1**
sAT	PRN-18	80.9	46.0	37.6	77.1	74.1	72.3	71.2	70.3	**37.6**
sTRADES	PRN-18	90.3	71.7	63.7	89.5	88.1	86.5	85.9	83.8	**63.7**

decomposing the perturbation $\boldsymbol{\delta}$ by $\boldsymbol{\delta} = \boldsymbol{p} \odot \boldsymbol{m}$ and updating them separately can deteriorate the performance. By contrast, the performance significantly improves when we update the mask $\boldsymbol{m}$ by its continuous alternative $\widetilde{\boldsymbol{m}}$ and l_0 ball projection. This indicates that introducing $\widetilde{\boldsymbol{m}}$ greatly mitigates the challenges in optimizing discrete variables. Moreover, the results in Table 10 indicate the performance can be further improved by the random reinitialization mechanism, which encourages exploration and avoids trapping in a local optimum.

In addition, we compare the performance when we use different step sizes for the magnitude tensor $\boldsymbol{p}$ and the sparsity mask $\boldsymbol{m}$. As shown in Table 11 and 12, the robust accuracy does not vary significantly with different step sizes. It indicates the satisfying robustness of our method to different hyperparameter choices. In practice, We set α and β to 0.25 and $0.25 \times \sqrt{hw}$, respectively. Note that h and w denote the height and width of the image, respectively, which are both 32 in CIFAR-10.

Ultimately, we compare the performance of sPGD with different tolerance iterations for reinitialization. As shown in Table 13, the performance of our method remains virtually unchanged, which showcases that our approach is robust to different choices of tolerance for reinitialization.

Adversarial training: We conduct preliminary exploration on adversarial training against sparse perturbations, since sPGD can be incorporated into any adversarial training variant. Ta-

Table 9: Robust accuracy of various models on different sparse attacks, where the sparsity level $\epsilon = 10$. The models are trained on **CIFAR-100** [64]. Note that we report results of Sparse-RS (RS) with tuned hyperparameters, which outperforms its original version in [17]. CornerSearch (CS) is evaluated on 1000 samples due to its high computational complexity.

Model	Network	Clean	Black		White					**sAA**
			CS	RS	SF	PGD_0	SAIF	$\textbf{sPGD}_\text{p}$	$\textbf{sPGD}_\text{u}$	
Vanilla	RN-18	74.3	1.6	0.3	20.1	1.9	9.0	0.1	0.9	**0.1**
l_∞-adv. trained, $\epsilon = 8/255$										
HAT	PRN-18	61.5	12.6	9.3	39.1	19.1	26.8	11.6	14.2	**8.5**
FDA	PRN-18	56.9	16.3	12.3	42.2	23.0	30.7	14.9	17.8	**11.6**
DKL	WRN-28	73.8	12.4	6.3	44.9	20.9	26.5	10.5	14.0	**6.1**
DM	WRN-28	72.6	14.0	8.2	46.2	23.4	29.8	12.7	15.8	**8.0**
l_1-adv. trained, $\epsilon = 6$										
l_1-APGD	PRN-18	63.2	22.7	22.1	47.7	33.0	43.5	19.7	20.3	**18.5**
Fast-EG-l_1	PRN-18	59.4	21.5	21.0	44.8	30.6	39.5	18.9	18.6	**17.3**
l_0-adv. trained, $\epsilon = 10$										
PGD_0-A	PRN-18	66.1	9.3	7.1	57.9	29.9	39.5	13.9	20.4	**6.5**
PGD_0-T	PRN-18	70.7	14.8	10.5	63.5	46.3	51.7	24.5	28.6	**10.2**
sAT	PRN-18	67.0	44.3	41.6	65.9	61.6	60.9	56.8	58.0	**41.6**
sTRADES	PRN-18	70.9	52.8	50.3	69.2	67.2	65.2	64.0	63.7	**50.2**

ble 1, 2, 7, 8 and 9 study sAT and sTRADES, while sTRADES outperforms sAT in all cases. In addition, the training of sAT is relatively unstable in practice, so we focus on sTRADES for ablation studies in this section. We leave the design of sPGD-adapted adversarial training variants to further improve model robustness as a future work.

Table 14 demonstrates the performance when we use different backward functions. The policies include always using the sparse gradient (Proj.), always using the unprojected gradient (Unproj.), alternatively using both backward functions every 5 epochs (Alter.) and randomly selecting backward functions (Rand.). The results indicate that randomly selecting backward functions has the best performance. In addition, Table 15 demonstrates the robust accuracy of models trained by sTRADES with different multi-ϵ strategies. The results indicate that multi-ϵ strategy helps boost the performance. The best robust accuracy is obtained when the adversarial budget for training is 6 times larger than that for test. Furthermore, we also study the impact of different tolerances during adversarial training in Table 16. The results show that higher tolerance during adversarial training benefits the robustness of the model, and the performance reaches its best when the tolerance is set to 10.

Structured Sparse Perturbations: Since the optimal hyperparameters for structured sparse perturbations differ from those for unstructured l_0 bounded perturbations, we conduct further ablation studies in the structured sparse setting.

The results in Table 17, 18 and 19 suggest that the step size α, β and the tolerance t for generating structured sparse perturbations should be smaller than those for unstructured cases. This can be attributed to the requirement for more exploitable attacks when generating structured sparse perturbations, similar to conventional l_∞ and l_2 attacks [39].

The multi-ϵ strategy has been proven effective in l_0 adversarial training. However, in the context of structured sparse attack, particularly in the patch setting, we can increase either the sparsity

level ϵ or the patch size to increase the corresponding l_0 adversarial budget. Therefore, we compare two different strategies here, i.e., multi-ϵ strategy and multi-size strategy. Specifically, the multi-ϵ strategy increases the sparsity level ϵ while maintaining the patch size unchanged during training, but the multi-size strategy only increases the patch size to make the l_0 sparsity level of generated perturbations similar to that in the multi-ϵ strategy. The results reported in Table 20 indicate that the multi-ϵ strategy provides better performance.

D More Visualization

More adversarial examples are shown in Figure 6. Additionally, we showcase some l_0 bounded adversarial examples generated on our sAT model in Figure 7. Compared to Figure 5 (a)-(b) and Figure 6 (a)-(b), the perturbations generated on the adversarially model are mostly located in the foreground of images. It is consistent with the intuition that the foreground of an image contains most of the semantic information [41].

Table 11: Robust accuracy at different step sizes α for magnitude p. The evaluated attack is $\text{sPGD}_{\text{proj}}$. The model is Fast-EG-$l_1$ trained on CIFAR-10.

α	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	1
Acc.	19.6	18.8	**18.1**	18.4	18.7	19.0

Table 12: Robust accuracy at different step sizes β for mask m. The evaluated attack is $\text{sPGD}_{\text{proj}}$. The model is Fast-EG-$l_1$ trained on CIFAR-10. Note that h and w are both 32 in CIFAR-10.

$\beta/\sqrt{hw}$	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	1
Acc.	20.0	19.3	**18.1**	18.4	19.4	21.0

Table 13: Robust accuracy at different tolerance for reinitialization t during attacking. The evaluated attack is $\text{sPGD}_{\text{proj}}$. The model is Fast-EG-$l_1$ trained on CIFAR-10.

t	1	3	5	7	10
Acc.	18.1	**18.1**	18.5	18.5	18.5

Table 14: Ablation study on different policies during adversarial training. The model is PRN18 trained by sTRADES with $6 \times k$. The robust accuracy is obtained through sAA.

Policy	Proj.	Unproj.	Alter.	Rand.
Acc.	41.5	39.4	51.5	**61.7**

Table 15: Ablation study on multi-ϵ strategy during adversarial training. The model is PRN18 trained by sTRADES with random policy. The robust accuracy is obtained through sAA.

ϵ	1×	2×	4×	6×	8×	10×
Acc.	34.0	39.7	54.5	**61.7**	60.2	55.5

Table 16: Ablation study on tolerance for reinitialization t during adversarial training. The model is PRN18 trained by sTRADES with $6 \times k$ and tolerance $t = 3$. The robust accuracy is obtained through sAA.

t	3	10	20
Acc.	51.7	**61.7**	60.0

Table 17: Robust accuracy at different step sizes α for magnitude p. The evaluated attack is $\text{sPGD}_{\text{proj}}$ generating 5×5 patch ($\epsilon = 1$). The model is sTRADES-$l_0$trained on CIFAR-10.

α	0.01	0.0125	0.05	0.1	0.15	0.25
Acc.	28.1	**26.7**	33.8	41.5	47.9	58.1

Table 18: Robust accuracy at different step sizes β for mask m. The evaluated attack is $\text{sPGD}_{\text{proj}}$ generating 5×5 patch ($\epsilon = 1$). The model is sTRADES-l_0 trained on CIFAR-10. Note that h and w are both 32 in CIFAR-10.

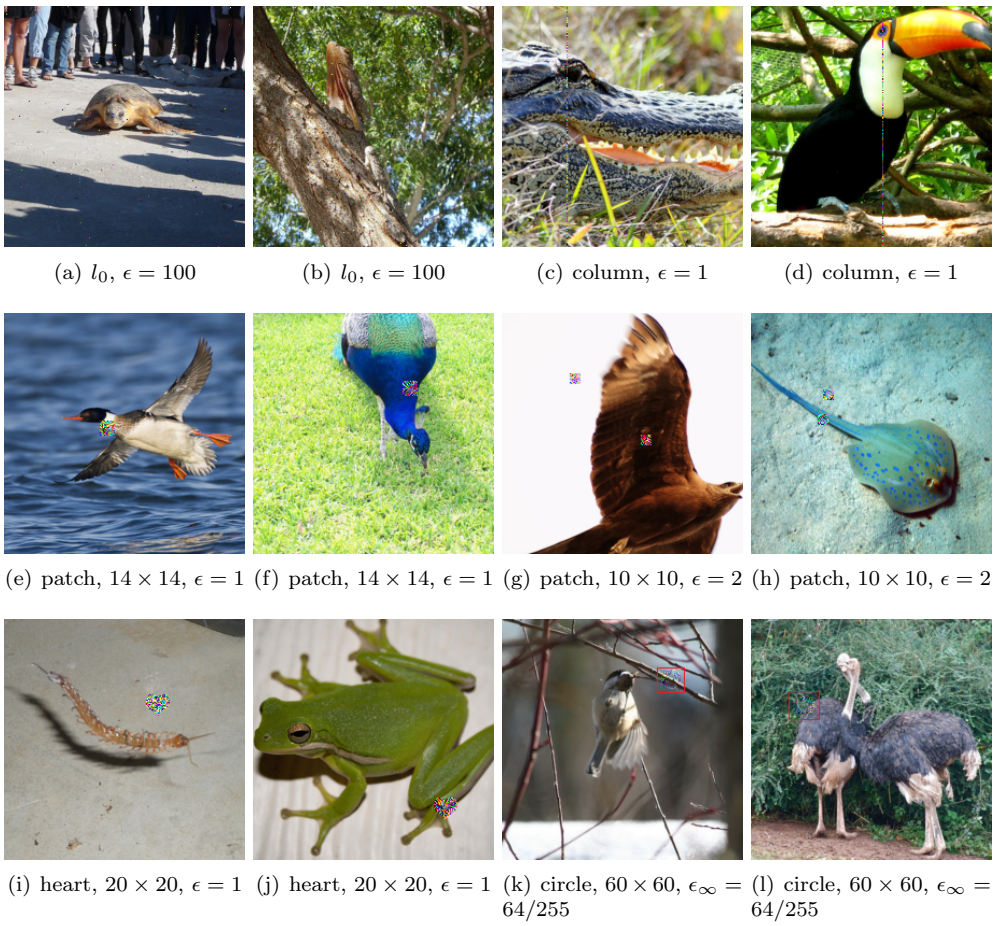
$\beta/\sqrt{hw}$	0.01	0.0125	0.05	0.1	0.15	0.25
Acc.	27.4	**26.7**	34.1	37.5	38.3	35.1

Table 19: Robust accuracy at different tolerance for reinitialization t during attacking. The evaluated attack is $\text{sPGD}_{\text{proj}}$ generating 5×5 patch ($\epsilon = 1$). The model is sTRADES-l_0 trained on CIFAR-10.

t	3	10	50	100	150
Acc.	36.1	28.7	**26.7**	28.1	33.7

Table 20: Ablation study on multi-ϵ / multi-size strategy during adversarial training against 3×3 patch ($\epsilon = 2$). The model is PRN18 trained by sTRADES with random policy.

Attack	RS	LOAP	$\text{sPGD}_{\text{proj}}$	$\text{sPGD}_{\text{unproj}}$	sAA
Multi-ϵ	**77.5**	**81.4**	**72.8**	**81.7**	**72.3**
Multi-size	75.5	79.2	68.4	79.8	67.9



(a) l_0, $\epsilon = 100$ (b) l_0, $\epsilon = 100$ (c) column, $\epsilon = 1$ (d) column, $\epsilon = 1$

(e) patch, 14×14, $\epsilon = 1$ (f) patch, 14×14, $\epsilon = 1$ (g) patch, 10×10, $\epsilon = 2$ (h) patch, 10×10, $\epsilon = 2$

(i) heart, 20×20, $\epsilon = 1$ (j) heart, 20×20, $\epsilon = 1$ (k) circle, 60×60, $\epsilon_\infty = 64/255$ (l) circle, 60×60, $\epsilon_\infty = 64/255$

Figure 6: Visualization of different sparse adversarial examples in ImageNet-100. The model is vanilla ResNet-34. The predictions before (left) and after (right) attack are (a) loggerhead→stingray, (b) frilled lizard→hornbill, (c) American alligator→African crocodile, (d) toucan→hornbill, (e) red-breasted merganser→drake, (f)peacock→cock, (g) kite→vulture, (h) stingray→electric ray, (i) centipede→stingray, (j) tree frog→tailed frog, (k) chickadee→junco, and (l) ostrich→peacock. **Note that the red squares in (k) and (l) are just for highlighting the perturbation position and not part of perturbations.**

(a) l_0, $\epsilon = 200$ (b) l_0, $\epsilon = 200$ (c) l_0, $\epsilon = 200$ (d) l_0, $\epsilon = 200$

Figure 7: Visualization of successful sparse adversarial examples in ImageNet-100. The model is ResNet-34 trained with sAT-l_0. The predictions before (left) and after (right) attack are (a) alligator lizard→water snake, (b) mud turtle→box turtle, (c) quail→partridge, and (d) black and gold garden spider→garden spider.