Machine Learning (2025) 114:163
https://doi.org/10.1007/s10994-025-06788-z

# Revealing vulnerable regions through diverse adversarial examples

**Yunce Zhao**[1,2] · **Wei Huang**[3] · **Wei Liu**[2] · **Xin Yao**[4]

Received: 2 August 2024 / Revised: 10 January 2025 / Accepted: 23 April 2025 /
Published online: 6 June 2025


**Abstract**
Current explainable AI approaches to Deep Neural Networks (DNNs) primarily aim to understand network behavior by identifying key input features that influence predictions. However, these methods often fail to identify *vulnerable* regions in the input that are sensitive to minor perturbations and pose significant security risks. The vulnerability of DNNs is typically studied through adversarial examples, but traditional norm-based algorithms, lacking spatial constraints, distribute perturbations across the entire image, obscuring these critical areas. To overcome this limitation, we propose the Vulnerable Region Discovery Attack (VrdAttack), an efficient method that leverages Differential Evolution to generate diverse one-pixel perturbations, enabling the discovery of vulnerable regions and uncovering pixel-level vulnerabilities in Deep Neural Networks (DNNs). Our extensive experiments on CIFAR-10 and ImageNet demonstrate that our proposed VrdAttack outperforms existing methods in identifying diverse critical weak points in an input, highlighting model-specific vulnerabilities, and revealing the impact of adversarial training on these vulnerable regions.

**Keywords** Diverse adversarial examples · Deep neural network vulnerability · Adversarial attacks · Explainability · Differential evolution

## 1 Introduction

Deep Neural Networks (DNNs) have driven significant advancements across various complex fields (He et al., 2016; Liu et al., 2021; Pouyanfar et al., 2017). Despite their impressive performance, fully understanding the decision-making process of these networks remains a major challenge. Recent progress in explainable AI (Selvaraju et al., 2017; Yang et al., 2021) has focused on identifying salient features that influence DNN decisions, typically within the main objects of interest. However, these methods often overlook vulnerable regions that are sensitive to small perturbations.

In this work, we explore the vulnerability of different regions in the input image, specifically focusing on the role of background elements in DNN decision-making. While prior

Editor: Lijun Zhang.

Extended author information available on the last page of the article

studies have targeted salient regions to deceive models (Xu et al., 2018), the sensitivity of background regions remains largely unexplored. If background regions are found to be vulnerable, it suggests that the model's decision-making process may be overly reliant on irrelevant details, raising concerns about its reliability. Identifying these vulnerabilities not only reveals the parts of the input most influential in the model's decisions, but also helps improve interpretability by exposing how background elements can disproportionately affect predictions.

To expose vulnerabilities, traditional techniques often employ adversarial examples (*e.g.*, $\ell_0, \ell_2, \ell_\infty$ adversarial examples). As illustrated in Fig. 1, these methods either densely distribute perturbations across images, obscuring specific vulnerable areas, or sparsely disperse them, hence complicating accurate vulnerability analysis (Carlini & Wagner, 2017; Croce et al., 2022; Madry et al., 2018; Moosavi-Dezfooli et al., 2017). Our work addresses these limitations by employing adversarial perturbations determined in a one-pixel manner to identify vulnerable regions, with each identified pixel capable of independently deceiving DNNs when perturbed. On one hand, this one-pixel based approach streamlines the process of attributing DNN errors to specific input points. It enables a precise assessment of vulnerabilities by examining the impact of these minimal perturbations on DNN predictions. On the other hand, the varied locations of these one-pixel perturbations ensure a comprehensive exploration of potential vulnerabilities, enabling a thorough analysis of these sensitive areas.

To further enhance the efficiency of our method and avoid the prohibitive computational costs associated with a brute-force approach that requires enumerating each pixel value, we leverage the powerful Sharing mechanism to propose the Vulnerable Region Discovery Attack (VrdAttack). Our proposed VrdAttack efficiently evolves a set of perturbed one-pixel perturbations towards a specific adversarial goal, simultaneously obtaining diverse solutions. Furthermore, the algorithm's unique sharing mechanism minimizes the chance of generating similar solutions, such as variations of the same pixel, thereby ensuring diverse one-pixel perturbations. The contributions are summarized as follows:

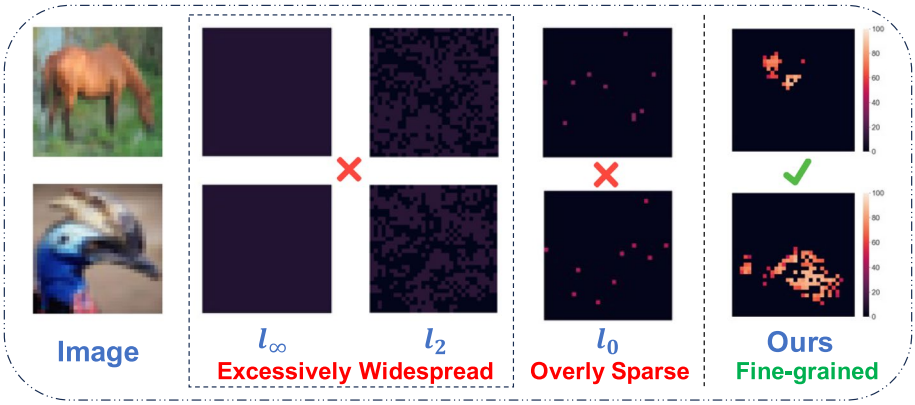


**Fig. 1** Comparing with three different perturbations generated by PGD ($\ell_\infty$) (Madry et al., 2018), CW ($\ell_2$) (Carlini & Wagner, 2017), and RSAttack ($\ell_0$) (Croce et al., 2022) for vulnerable region discovery. Since these methods incorporate different perturbed pixels, we evenly distribute the reduced confidence of the ground truth across perturbed pixels. Notably, as perturbations generated by the PGD attack encompass the entire image, each pixel is thought to contribute just a little. In contrast, our approach reveals detailed vulnerable regions with a bar that indicates the level of vulnerability

- We propose a novel approach for identifying vulnerable regions in DNNs through adversarial perturbations at diverse locations, serving as a complement to current explainability methods.
- We propose the Vulnerable Region Discovery Attack (VrdAttack), which efficiently generates a diverse set of adversarial examples in a single run, providing a detailed pixel-level vulnerability assessment.
- Through extensive experiments on various network architectures and datasets, we demonstrate the effectiveness of our approach. Our findings provide deep insights into the inherent vulnerabilities of DNNs, highlighting critical weak points that could be exploited.

The rest of this paper is organized as follows: Sect. 2 discusses related explainability methods and the vulnerabilities of DNNs. In Sect. 3, we introduce our methodology for identifying vulnerable regions using diverse adversarial examples. Section 4 showcases experimental results that demonstrate the effectiveness of our approach. Finally, Sect. 5 summarizes our contributions and outlines directions for future work.

## 2 Related work

### 2.1 Explainable approach of DNNs

Explainable AI methods are essential for understanding and interpreting the decisions made by DNNs. They identify and visualize key image regions that influence model predictions, enhancing the transparency and trustworthiness of DNNs.

Activation map methods analyze the activations and gradients within the network to identify the image parts that contribute most to the final output. Zhou et al. proposed Class Activation Mapping (CAM) (Zhou et al., 2016), which generates heatmaps by leveraging the global average pooling layer before the final output layer. However, CAM is limited to specific network architectures with a global average pooling layer. Selvaraju et al. proposed Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al., 2017), which computes gradients of the target class score with respect to the feature maps of a convolutional layer. Grad-CAM combines these gradients with the feature maps to produce a coarse localization map of important regions. Srinivas and Fleuret proposed Full-Gradient Representation (Full-Grad) (Srinivas and Fleuret, 2019), which aggregates gradients across all layers and neurons of a network. Full-Grad generates fine-grained saliency maps that highlight both high-level concepts and low-level features contributing to the decision.

Perturbation is another powerful tool for delving deeper into the intricacies of DNNs. It primarily involves removing parts of the input image (i.e., setting the pixel values of the removed parts to fixed values, such as 0) and observing the resulting shifts in predictions. Zeiler et al. proposed the Occlusion method (Zeiler & Fergus, 2014), which uses a gray square mask on parts of the input image to examine variations in model predictions. Ribeiro et al. proposed LIME (Ribeiro et al., 2016), leveraging superpixel occlusion and linear models to accurately emulate the decision boundaries of the original deep model. However, the saliency maps obtained often lack precision due to the coarse-grained nature of superpixels. Considering the morphology of objects, Fong et al. (2019) and Yang et al. (2021) introduced optimization methods to iteratively refine the obtained saliency maps.

Despite their effectiveness in highlighting key features, these methods often overlook the critical aspect of DNN vulnerabilities to adversarial perturbations. Our approach differs by focusing on the specific pixel regions susceptible to adversarial attacks, which optimizes both the positions and values of perturbed pixels to identify and assess pixel-level vulnerabilities in DNNs, thus providing a different perspective of explainability.

### 2.2 Adversarial vulnerabilities

Adversarial attacks, which attempt to fool deep models by exploiting their adversarial vulnerability, can be categorized into white-box and black-box attacks. In a white-box attack, attackers have full access to the DNN's architecture and parameters (Carlini & Wagner, 2017; Madry et al., 2018). Black-box attacks involve limited access to only the DNN's input and output, necessitating numerous queries to the target model (Chen et al., 2017; Guo et al., 2019; Ilyas et al., 2018; Zhang et al., 2024). Recent studies have focused on enhancing the efficiency of generating adversarial examples in black-box attacks (Andriushchenko et al., 2020; Bai et al., 2023; Shi et al., 2022; Sun et al., 2022). While such research excels at finding a single adversarial example to attack the DNN, these methods often distribute perturbations densely across images, obscuring the identification of specific vulnerable regions.

Few-pixel attacks aim to deceive DNNs using minimal perturbations, emphasizing the significant role certain pixels play in misleading these networks (Croce et al., 2022; Rao et al., 2020). Nonetheless, accurately pinpointing how these perturbed pixels induce incorrect predictions remains a significant challenge. Xu et al. (2018) proposed the structured attack, which enhances interpretability by targeting semantically meaningful features. While this work demonstrates the effectiveness of manipulating such features to deceive DNNs, it leaves open the question of whether regions deemed non-important to human observers can still be vulnerable and effective in fooling the network. The One-Pixel Attack (Su et al., 2019) represents the extreme case of few-pixel attacks, where a single pixel is altered to mislead the model. Although this study validates the effectiveness of one-pixel perturbations, it does not explore whether a set of diverse perturbed pixels-i.e., vulnerable regions-could be equally or more effective. To address this gap, we propose generating diverse adversarial perturbations determined in a "one-pixel" manner at varied locations simultaneously, enabling a more comprehensive analysis of vulnerabilities within DNNs.

### 2.3 Adversarial training

Adversarial training, a pioneering technique introduced by Goodfellow et al. (2014), has been a cornerstone in improving the robustness of deep neural networks (DNNs) against adversarial attacks. The core concept involves training the model on adversarial examples generated by perturbing input data to maximize the loss. Madry et al. (2018) expanded on this concept by introducing the Projected Gradient Descent (PGD) adversarial training, considered one of the most effective methods for training robust models. Subsequent research has explored various dimensions of adversarial training. For instance, Xie et al. (2019) introduced feature denoising to improve model robustness. Zhang et al. (2019) proposed TRADES, a theoretical framework balancing model accuracy on clean data with robustness against adversarial examples. Furthermore, Zhou et al. (2023) proposed the LADDER framework, which generates adversarial examples in the latent feature space

guided by the decision boundary, improving the balance between standard accuracy and adversarial robustness. In this work, we also selected two well-known adversarial training algorithms PGD (Madry et al., 2018) and TRADES (Zhang et al., 2019) to explore how the identified vulnerable regions change with different adversarial training algorithms.

## 3 Methodology

In this section, we first define the problem of generating diverse adversarial examples with different perturbed positions for a single image in Sect. 3.1, and then present our method for solving this problem in Sect. 3.2.

### 3.1 Problem definition

Consider a data pair $(x, y)$, where $x \in X \subseteq \mathbb{R}^d$ represents the input (e.g., an image), and $y \in Y \subseteq 1, 2, ..., K$ is the true label corresponding to one of $K$ classes. The classifier $C(x)$ maps the input $x$ to the class label with the highest predicted confidence, i.e., $C(x) = \arg\max_i f(x)_i$, where $f(x)_i$ represents the confidence of the DNN in the i-th class. If the DNN makes a correct prediction, we have $C(x) = y$. The goal of the adversary is to create the adversarial example $x' = x + \delta$ that deceives the classifier into making a false prediction, $C(x') \neq y$. The perturbations $\delta$ are normally bounded by the $\ell_p$ norm to ensure visual invisibility. Consequently, this search problem can be reformulated as a constrained optimization problem (Szegedy et al., 2013). In a non-targeted attack (Carlini & Wagner, 2017; Kurakin et al., 2016), the adversary aims to diminish the model's confidence in the true class, thereby inducing the DNN to predict any arbitrary incorrect class:

$$\arg\min_{\delta} \quad f(x+\delta)_y, \qquad s.t. \quad ||\delta||_p \leq \epsilon_p, \tag{1}$$

where $y$ is the ground truth label and $\epsilon_p$ denotes the maximum allowed perturbation strength. Conversely, in a targeted attack (Carlini & Wagner, 2017; Kurakin et al., 2016), the adversary seeks to deliberately manipulate the DNN's predictions, ensuring the input is misclassified with high confidence into a specific target class predetermined by the attacker, irrespective of its true label:

$$\arg\max_{\delta} \quad f(x+\delta)_t, \qquad s.t. \quad ||\delta||_p \leq \epsilon_p, \tag{2}$$

where $f(x+\delta)_t$ represents the predicted confidence in the targeted false class. In this work, we focus on assessing vulnerabilities by introducing diverse one-pixel adversarial perturbations, which enable the direct attribution of misclassification to specific input locations. Unlike traditional methods that either generate a single adversarial example or distribute perturbations densely across the input, our approach generates perturbations at diverse locations across the input image. Let $\mathbb{A} = \{\delta_1, \delta_2, ..., \delta_n\}$ represent the set of successful one-pixel adversarial perturbations, ensuring that:

$$d(\delta_i, \delta_j) \neq 0, \qquad s.t. \quad \delta_i, \delta_j \in \mathbb{A}. \tag{3}$$

here, $d(\cdot)$ represents the Euclidean distance between two perturbations, ensuring that each perturbation is distinct.

The use of diverse perturbations is crucial for providing a more comprehensive assessment of the model's vulnerabilities. Traditional approaches, such as dense or sparse perturbation methods, often focus on either a small set of perturbations or affect the entire image, which can lead to incomplete vulnerability analysis. By using one-pixel perturbations at diverse locations, we achieve the following benefits:

1. **Comprehensive Coverage of the Input Space** Each perturbation is applied to a different pixel, ensuring that all regions of the image are tested for vulnerabilities. This prevents focusing only on the most obvious regions and helps identify more subtle, yet critical, vulnerabilities in less prominent areas.
2. **Increased Sensitivity in Vulnerability Detection** The diversity of perturbations allows us to uncover vulnerabilities in various parts of the image, including both foreground and background regions. This helps in understanding how different regions of the image contribute to the model's decisions, revealing any over-reliance on irrelevant background details or subtle features.
3. **Improved Interpretability** This diversity in perturbation locations allows us to gain a finer-grained understanding of how the model's predictions are influenced by specific pixels. This makes the model's decision-making process more interpretable.

In summary, the diversity of one-pixel perturbations in our approach ensures a more thorough exploration of the input space, enabling a more complete and detailed vulnerability analysis.

### 3.2 Vulnerable region discovery attack

Generating adversarial examples with diverse locations is a complicated task. It requires not only optimizing the perturbation values but also the positions of the perturbations, which are typically non-differentiable in adversarial settings. An evolutionary algorithm is well-suited for this scenario, as it can provide a set of candidate solutions in the final generation, enabling the acquisition of diverse adversarial examples simultaneously. Additionally, it is a black-box attack that does not rely on internal information about the targeted DNNs, and its search process does not require differentiability. A notation Table can be found in Table 1.

#### 3.2.1 Overview of differential evolution (DE)

Differential evolution (DE) (Storn & Price, 1997) is an evolutionary optimization algorithm that has been successfully applied in various fields. DE maintains a population of candidate solutions, which evolve over time through a cycle of mutation, recombination, and selection. This iterative process gradually guides the population toward the optimal solution.

**Initialization** The DE process begins by initializing a population of $P$ candidate solutions within the search space. This population is represented by the set $\mathbb{S} = \{z_1, z_2, ..., z_P\}$, where each individual $z_i \in \mathbb{R}^D$ is a $D$-dimensional vector. Here, $P$ denotes the population size (i.e., the number of candidate solutions), and $D$ represents the dimensionality of the search space. Each $z_i$ corresponds to a potential solution, such as a perturbation applied to a specific pixel in an image. Once initialized, each candidate solution is evaluated based on the fitness function, which quantifies how well the solution meets the optimization objective. For example, in

**Table 1** Notation table

| Notation | Definition |
|---|---|
| $\mathbb{S}$ | Population of candidate solutions |
| $\mathbb{V}$ | Population of mutant solutions |
| $\mathbb{U}$ | Population of trial solutions |
| $z_i$ | $i$-th solution in the population $\mathbb{S}$ |
| $\boldsymbol{v}_i$ | $i$-th solution in the population $\mathbb{V}$ |
| $\boldsymbol{u}_i$ | $i$-th solution in the population $\mathbb{U}$ |
| $F$ | Raw fitness score |
| $F'$ | Sharing fitness score |
| $d_{i,j}$ | Euclidean distance between two perturbed pixels |
| **Hyperparameters** | **Definition** |
| $N$ | Iterations |
| $P$ | Population Size |
| $W$ | Mutation rate |
| $R$ | Recombination rate |
| $\gamma_s$ | Sharing radius |
| $\alpha$ | Penalty factor |

the context of adversarial attacks, the fitness of each candidate solution is evaluated based on how effectively it causes the DNN to misclassify the image. The fitness score is computed by querying the DNN with the perturbed image and measuring how confident the model is in the true label (or the target label, in the case of targeted attacks). Then, the population of solutions is refined iteratively through the processes of mutation, recombination, and selection.

**Mutation** In the mutation step, we generate a mutant solution for each solution in the population $\mathbb{S}$, resulting in a total of $P$ mutant solutions. For the $i$-th solution $z_i$, the corresponding mutant solution $\boldsymbol{v}_i \in \mathbb{R}^D$ is computed by applying a weighted differential scheme to three randomly selected individuals $z_{r_1}, z_{r_2}, z_{r_3}$ from the population $\mathbb{S}$:

$$\boldsymbol{v}_i = z_{r_1} + W(z_{r_2} - z_{r_3}), \qquad s.t. \quad r_1 \neq r_2 \neq r_3. \tag{4}$$

here, $W$ is the mutation factor, which controls the scaling of the difference between the randomly selected individuals. A larger $W$ results in a larger step size, allowing the mutation to explore more distant regions of the solution space. The indices $r_1$, $r_2$, and $r_3$ are randomly chosen from the population and are distinct from each other. The set of mutant solutions is denoted as $\mathbb{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_P\}$. This mutation step helps explore the search space by generating new solutions based on the difference of two existed solutions.

**Recombination** After mutation, each trial solution $\boldsymbol{u}_i \in \mathbb{R}^D$ is formed by recombining the mutant solution $\boldsymbol{v}_i \in \mathbb{V}$ with the corresponding parent solution $z_i \in \mathbb{S}$. The recombination process follows the equation:

$$\boldsymbol{u}_{i,j} = \begin{cases} \boldsymbol{v}_{i,j}, & c \leq R \text{ or } j = j', \\ z_{i,j}, & \text{otherwise}, \end{cases} \tag{5}$$

where $c$ is a random number within the range [0,1]. The subscript $(i, j)$ denotes the $j$-th variable of the $i$-th individual. $R$ is a hyperparameter known as the crossover rate, indicating the likelihood that a variable in the trial solution originates from the corresponding variable in the mutant solution. $j'$ is an integer randomly sampled from $[1, D]$ and is used to make sure at least one of the variables of the trial solution comes from the mutant solution. The set of trial solutions is denoted as $\mathbb{U} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_P\}$. This step mixes information from both the parent and mutant solutions to create a trial solution. The trial solution $\boldsymbol{u}_i$ is then evaluated based on the fitness function for the assessment of its quality.

**Selection** In the selection process, the fitness of the trial solution $\boldsymbol{u}_i$ is compared with that of the parent solution $\boldsymbol{z}_i$ within the population. If the trial solution exhibits better fitness (i.e., it leads to a more optimal solution), it replaces the parent solution in the population. This mechanism ensures that only solutions that improve the objective are retained, driving the population toward better-performing individuals over time. By favoring solutions with higher fitness, the algorithm encourages the exploration of promising regions in the search space, gradually converging to areas that maximize the objective function. This is done iteratively until a termination condition (such as a maximum number of generations or a desired fitness level) is met.

### 3.2.2 Revisit one-pixel attack (Su et al., 2019)

In the context of the One-Pixel Attack (Su et al., 2019), differential evolution (DE) is employed to generate minimal one-pixel perturbations that are capable of deceiving deep neural networks (DNNs). The primary goal is to identify a perturbation consisting of a single pixel that, when modified, causes the DNN to misclassify the image. Each perturbation is represented by a candidate solution in the population of DE, which is defined as a 5-element tuple $z_i \in \mathbb{R}^5$. This tuple comprises the coordinates and RGB values of the pixel to be perturbed. Specifically, for the $i$-th candidate solution (perturbation), the clean input image is perturbed by replacing the pixel at position $(z_i[1], z_i[2])$ with the RGB values $(z_i[3], z_i[4], z_i[5])$.

The core objective of the attack is to find the most effective one-pixel perturbation that can mislead the DNN. To achieve this, the fitness function is defined differently for non-targeted and targeted attacks. For non-targeted attacks, the fitness function is $F_i = 1 - f(I_i')_y$, where $I_i'$ denotes the image after the perturbation, and $f(I_i')_y$ is the DNN's predicted probability for the true class. For targeted attacks, the fitness function is $F_i = f(I_i')_t$, where $f(I_i')_t$ is the predicted probability for the target class. These fitness definitions guide the evolutionary process, pushing the perturbation towards a solution that either decreases the DNN's confidence in the correct class (for non-targeted) or increases its confidence in the target class (for targeted).

Throughout the evolutionary process, variability is introduced through mutation and recombination, while during the selection phase, solutions that perform poorly in deceiving DNNs are discarded. This iterative process allows the remaining solutions to evolve closer to successfully fooling the DNNs.

**Algorithm 1** Vulnerable region discovery attack

**Input:** Image $I$, deep neural network $f$ for querying, population size $P$, number of generations $N$, mutation rate $W$, crossover rate $R$, sharing radius $\gamma$, hyperparameter $\alpha$

Initialize the candidate solutions for the first generation as $\mathbb{S}_0 = \{\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_P\}$, where each $\boldsymbol{z}_i \in \mathbb{R}^5$.
Evaluate raw fitness $F(\mathbb{S}_0)$ by querying $f$ with Image $I$ and perturbed pixels $\boldsymbol{z} \in \mathbb{S}_0$.
**for** $g = 1, ..., N$ **do** ▷ For each generation
    Execute the mutation process to obtain the set of mutant solutions $\mathbb{V}_g$ using Eq. 4.
    Execute the recombination process to obtain the set of trial solutions $\mathbb{U}_g$ using Eq. 5.
    Evaluate raw fitness $F(\mathbb{U})$ by querying $f$ with Image $I$ and perturbed pixels $\boldsymbol{u} \in \mathbb{U}_g$.
    Merge the trial solutions $\mathbb{U}_g$ and previous generation $\mathbb{S}_{g-1}$ to create an extended set $\mathbb{S}'_{g-1} = \mathbb{S}_{g-1} \cup \mathbb{U}_g$, which contains $2P$ perturbations.
    Compute the Euclidean distance between each two perturbed pixels in $\mathbb{S}'_{g-1}$.
    Calculate the sharing fitness $F'(\mathbb{S}'_{g-1})$ using Eqs. 6 and 7.
    Discard the worse half of the population based on sharing fitness $F'(\mathbb{S}'_{g-1})$ to form the new generation $\mathbb{S}_g$.
    **if** The individual with the highest raw fitness in $\mathbb{S}'_{g-1}$ is removed **then**
        Replace the least fit individual in $\mathbb{S}_g$ with this removed individual.
    **end if**
    Record the raw fitness $F(\mathbb{S}_g)$.
**end for**
Evaluate the final generation $\mathbb{S}_N$ and and select the successful perturbations to form the set $\{\delta_1, ..., \delta_n\}$

**Output:** Return the final set of successful perturbations $\{\delta_1, ..., \delta_n\}$.

### 3.2.3 Vulnerable region discovery attack

However, since the algorithm does not incorporate a penalty for similarity among candidate solutions, it increasingly focuses on exploiting the most promising solutions found early on. This leads to most individuals in the population converging to the same or very similar values. Consequently, the mutation and recombination operators make only minor adjustments to these individuals, refining their values until they all become nearly identical. As a result, the One-Pixel Attack (Su et al., 2019) ultimately converges toward solutions with the highest fitness scores-namely, one-pixel perturbations that cause the most significant degradation to the performance of DNNs. As illustrated in Fig. 2, only a very small number of pixels have been pinpointed as viable for such a One-Pixel Attack (Su et al., 2019), thereby making them unsuitable for uncovering vulnerable regions. To address this issue, we adapt the sharing mechanism to propose **Vulnerable Region Discovery Attack (VrdAttack)** for the diversely located one-pixel perturbations.

To penalties the similarity of the searched candidate solutions, the sharing mechanism is introduced. The sharing mechanism for the evolution algorithm (EA) was originally

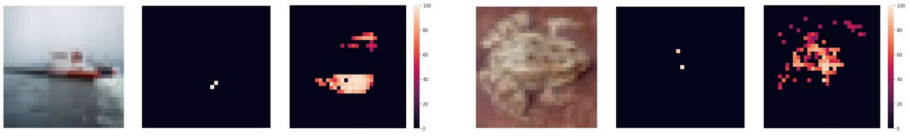

**Fig. 2** Two examples (ship and frog) for both One-Pixel Attack (Su et al., 2019) and our algorithm on $32 \times 32$ images from CIFAR-10. For each example, three columns are shown: (1) the original image, (2) the result of 10 independent One-Pixel Attacks, and (3) the result of a single run of our algorithm. Each point on the heatmap represents a successful one-pixel perturbation, with the brightness indicating the degree of reduction in the DNN's confidence for the ground truth class

proposed in (Goldberg & Richardson, 1987) with the goal of locating diverse solutions simultaneously. The core idea of the fitness sharing is to penalize solutions for occupying the same regions by applying a cost to their fitness scores. In general, fitness sharing serves to reduce the payoff of the densely populated area by dividing the raw fitness of an individual by the approximate number of similar solutions (Darwen & Yao, 1996). In the implement of our Vulnerable Region Discovery Attack, the fitness score of (Su et al., 2019) is replaced by the sharing fitness score to evaluate the solutions. Specifically for the i-th solution, the sharing fitness score is calculated as:

$$F_i' = \frac{F_i}{\sum_{j=1}^{P} sh(d_{i,j})}, \tag{6}$$

where $F_i$ denotes the raw fitness score before sharing, (i.e., the fitness score of (Su et al., 2019)) and $P$ represents the number of candidate solutions in the population. The sharing function, $sh(\cdot)$, is designed to penalize the fitness of similar solutions and is defined as follows:

$$sh(d_{i,j}) = \begin{cases} 1 - (d_{i,j}/\gamma_s)^{\alpha}, & d < \gamma_s, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $d_{i,j}$ denotes the Euclidean distance between the $i$-th candidate solutions and the $j$-th candidate solutions. And $\gamma_s$ is the predefined sharing radius, determining the threshold for how similar solutions must be to be penalized, and $\alpha$ is a positive scalar controlling the shape of the sharing function. When two or more candidate solutions are located within the sharing radius $\gamma_s$, the function $sh(d_{i,j})$, assigns a positive value, effectively reducing their sharing fitness $F_i'$. The closer the solutions are to each other (i.e., the smaller $d_{i,j}$), the greater the penalty imposed. This encourages the algorithm to spread out the population across different vulnerable regions of the image space.

In addition to the fitness function, we have modified One-Pixel Attack (Su et al., 2019) in the following way to mitigate the risk of the algorithm converging prematurely to local optimal solutions (Chiou et al., 2004). Instead of replacing the corresponding parents, all the newly generated offspring are added to the population to obtain sharing fitness value $F'$. To stabilize the population size for the next generation, we remove the bottom 50% candidate solutions of the population based on their sharing fitness scores $F'$. And the Elitism ensures the preservation of the best-found solution throughout the optimization process: the best solution with original fitness $F$ is used to replace the worst individual ranked with sharing fitness score in the left population if it gets removed. The searching process is presented in Algorithm 1. We denote

raw fitness as $F(\cdot)$ and sharing fitness as $F'(\cdot)$. And they are specifically applicable to the set elements indicated within the brackets, providing the object value for the solution selection process.

Once the optimization process of our VrdAttack is completed, the final population of candidate solutions is applied individually to the input image to assess their effectiveness. Perturbations that fail to deceive the DNN are discarded, while the remaining successful perturbations, corresponding to the final set of perturbations $\delta_1, ..., \delta_n$, are retained. These perturbations are represented by their pixel positions and corresponding fitness scores, and are used to map the vulnerable regions within the input image. To represent these vulnerable regions, we aggregate the positions of the successfully perturbed pixels and visualize their distribution as a vulnerability heatmap. Each pixel in the heatmap is assigned a value proportional to the reduction in the DNN's confidence for the true label caused by its perturbation. This process highlights not only the specific locations that are most sensitive to perturbations but also the relative impact of each location, providing a detailed and interpretable representation of the model's weaknesses. By leveraging the diversity of the perturbations generated during optimization, our method ensures that the identified vulnerable regions are spatially diverse, encompassing areas that might not be traditionally associated with critical features, such as the background or peripheral regions of the image. This comprehensive analysis allows us to uncover hidden dependencies in the model's decision-making process, challenging conventional notions of feature importance and enhancing the interpretability of the DNN.

Compared to the traditional One-Pixel Attack (Su et al., 2019), our Vulnerable Region Discovery Attack emphasizes maintaining solution diversity through a sharing fitness function and a structured selection process. By penalizing similar solutions and retaining the top-performing and diverse perturbations, our approach enhances the algorithm's ability to explore multiple vulnerable regions effectively, thereby increasing the effectiveness of the diversely located one-pixel perturbations against deep neural networks.

## 4 Experimental studies

The experimental setup is detailed in Sect. 4.1. In Sect. 4.2, we illustrate our discovered vulnerable regions and compare our proposed methods with ten runs of the One-Pixel Attack (Su et al., 2019) to demonstrate the effectiveness of our approach. We also study the specific locations of these vulnerable regions across different DNNs and datasets in Sect. 4.3. In Sect. 4.4, we investigate the effects of adversarial training on the vulnerable regions to study whether adversarial training can eliminate these vulnerabilities. Finally, in Sect. 4.5, we discuss the differences between important areas identified by explainability methods and the vulnerable regions uncovered by our approach, highlighting the significance of our method. The source code is available at https://github.com/YunCe-Code/Vulnerable-Region-Discovery.git.

Throughout the following discussion, we utilize 'vulnerable pixels' to denote the identified pixels, each of which, when perturbed, can successfully deceive DNNs. The vulnerable region is formed by these vulnerable pixels as illustrated in Fig. 3. We refer to images containing vulnerable regions as 'vulnerable images'.

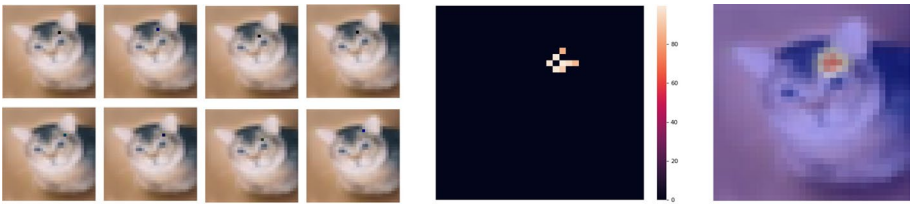

**Fig. 3** Diverse adversarial examples and the corresponding vulnerable region. The set of 8 small images shows the generated adversarial examples. The middle heatmap indicates one-pixel perturbation locations, with colors denoting reduced confidence in the 'cat' label. The rightmost image presents the heatmap overlaid on the image

### 4.1 Experimental setup

**Dataset** We conducted experiments on the CIFAR-10 and ImageNet datasets. The CIFAR-10 dataset comprises 60,000 images, each of $32 \times 32$ dimensions, evenly distributed across 10 distinct classes with 6,000 images per class. This dataset is divided into 50,000 training images and 10,000 testing images. The ImageNet dataset, also known as ILSVRC 2012, contains high-resolution natural images spanning 1,000 classes. These images are resized to dimensions of 224×224 for DNN classification.

**Models** For CIFAR-10, we evaluate our algorithm on VGG16 (86.04% accuracy), ResNet18 (94.03% accuracy), and Network in Network (91.49% accuracy). For ImageNet, we test on AlexNet (56.5% accuracy) and ResNet50 (75.9% accuracy). These models are chosen based on the existing literature Su et al. (2019) to facilitate comparisons. Each model was trained with standard settings and subjected to our VrdAttack method to identify pixel-level vulnerabilities.

**Evaluation Metrics** The experiment is conducted in both targeted and non-targeted scenarios, measuring the ASR (Average Success Rate), APN (Average Pixel Number), ADist (Average Distance)and Cost to evaluate the effectiveness of our approach.

- **ASR (Average Success Rate)** This metric is defined as the percentage of samples that are successfully perturbed to arbitrary false classes in non-targeted attacks and to designed classes in targeted attacks. It is used to measure the proportion of vulnerable images, indicating the overall effectiveness of the adversarial attacks.
- **APN (Average Pixel Number)** This metric represents the average number of different pixels that can be utilized independently for successful attacks, regardless of whether the attacks are targeted or non-targeted. It measures the average size of vulnerable regions for each vulnerable image, providing insight into how widespread the vulnerabilities are within the images.
- **ADist (Average Distance)** This metric calculates the average Euclidean distance between each pair of successful one-pixel perturbations for every successful attack. It reflects the diversity of the acquired solutions.
- **Cost** This metric is defined as the average time (in seconds) required to conduct the attack *per image* in non-targeted scenarios and *per image per designed class* in targeted scenarios. It is used to assess the computational efficiency of our approach.

These metrics provide a comprehensive understanding of its effectiveness, the characteristics of the vulnerable regions, and the computational resources needed. The results presented are the averages from three independent experiments.

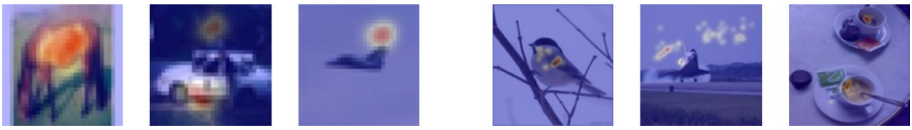

**Fig. 4** Heatmaps of vulnerable regions on CIFAR-10 (left) and ImageNet (right)

**Table 2** Overall results of our VrdAttack on CIFAR-10 dataset

| Non-targeted | ResNet18 | NiN | VGG16 | Targeted | ResNet18 | NiN | VGG16 |
|---|---|---|---|---|---|---|---|
| ASR (%) | $28.4 \pm 0.8$ | $32.6 \pm 0.4$ | $60.2 \pm 1.1$ | ASR (%) | $5.0 \pm 0.3$ | $5.6 \pm 0.2$ | $13.2 \pm 0.6$ |
| APN | $37.3 \pm 2.2$ | $40.8 \pm 1.1$ | $41.7 \pm 2.7$ | APN | $29.4 \pm 0.6$ | $27.9 \pm 0.7$ | $20.6 \pm 0.5$ |
| ADist | $5.81 \pm 0.22$ | $6.53 \pm 0.41$ | $6.37 \pm 0.24$ | ADist | $4.70 \pm 0.14$ | $4.76 \pm 0.31$ | $5.41 \pm 0.17$ |
| Cost (s) | $4.57 \pm 0.01$ | $4.28 \pm 0.03$ | $4.03 \pm 0.03$ | Cost (s) | $4.55 \pm 0.02$ | $4.11 \pm 0.01$ | $4.01 \pm 0.02$ |

Left: non-targeted attacks. Right: targeted attacks

**Implementation details** In our experimental setup, following the One-Pixel Attack (Su et al., 2019), we use a Gaussian distribution represented by $\mathcal{N}(\mu = 128, \delta = 127)$ for initializing the $r$, $g$, $b$ values, whereas the coordinates $x$, $y$ are determined utilizing uniform distributions: $\mathcal{U}(1, 32)$ for CIFAR-10 and $\mathcal{U}(1, 224)$ for ImageNet. The attack budget ($\epsilon_0$) is set to 1, with the perturbations constrained within the valid range of RGB pixel values. For CIFAR-10, we use a population size of 200 individuals and 100 iterations with the Sharing Differential Evolution (DE) procedure. Due to the larger size of ImageNet images (approximately 50 times larger than CIFAR-10), we increase the population size to 800. The mutation rate $W$ and recombination rate $R$ are set to 0.5 and 1.0, respectively. These hyperparameters are based on the recommendations from the One-Pixel Attack (Su et al., 2019). The sharing radius $\gamma_s$ is chosen through grid search to optimize APN number with a radius of 4 pixels for CIFAR-10 and 6 pixels for ImageNet. The parameter $\alpha$, which controls the shape of the sharing function, is set to 1, a commonly used value in the literature (Nguyen et al., 2012; Sareni & Krahenbuhl, 1998). All experiments are conducted using PyTorch on a single Tesla V100 GPU.

## 4.2 Vulnerable regions with diverse adversarial examples

### 4.2.1 Diverse adversarial examples with CIFAR-10

In the non-targeted and targeted attack scenarios, we randomly sample 1000 and 500 correctly classified images from the CIFAR-10 test set, respectively. Specifically, for the targeted attack scenario, we attempt to perturb each of the selected 500 images into each of the other 9 categories. Results are presented in Table 2. For visualization of the vulnerable regions, the heatmap is overlaid on top of the images. The value of the heatmap represents the change in confidence of the class label. We also apply a Gaussian filter to the heatmap for better visualization. Examples of discovered vulnerable regions are shown in Fig. 4. Additional examples can be found in Fig. 14 of Appendix B.

***Superiority in Discovering Diverse Adversarial Examples*** We verify the superiority of our algorithm to compare the state-of-the-art One-Pixel Attack (Su et al., 2019). The

**Table 3** Accumulate results of 10 runs for the One-Pixel Attack (Su et al., 2019)

| Non-targeted | ResNet18 | NiN | VGG16 | Targeted | ResNet18 | NiN | VGG16 |
|---|---|---|---|---|---|---|---|
| ASR (%) | 27.2 ± 1.1 | 32.0 ± 1.4 | 58.4 ± 1.8 | ASR (%) | 5.0 ± 0.4 | 5.7 ± 0.2 | 12.9 ± 0.3 |
| APN | 1.3 ± 0.2 | 1.9 ± 0.3 | 2.7 ± 0.2 | APN | 1.2 ± 0.1 | 1.6 ± 0.1 | 3.2 ± 0.3 |
| ADist | 0.27 ± 0.07 | 0.31 ± 0.02 | 0.37 ± 0.04 | ADist | 0.13 ± 0.02 | 0.21 ± 0.06 | 0.26 ± 0.12 |
| Cost (s) | 42.13 ± 0.03 | 38.71 ± 0.02 | 36.57 ± 0.03 | Cost (s) | 42.08 ± 0.03 | 38.61 ± 0.02 | 36.42 ± 0.03 |

experiment is conducted on CIFAR-10. Our goal is to generate a diverse set of adversarial examples, so we compare our approach with a variation of the One-Pixel Attack (Su et al., 2019). We run the One-Pixel Attack ten times independently and aggregate the results to assess whether multiple runs can identify different vulnerable regions. Results are presented in Table 3. Our algorithm identifies a comparable count of vulnerable images to the multi-run One-Pixel Attack (Su et al., 2019), it discovers a broader variety of vulnerable pixels with fewer computational resources. Notably, the One-Pixel Attack often converges to specific regions on the object across multiple runs. In contrast, our method uncovers a broader set of vulnerable areas. A higher APN indicates that our approach identifies more vulnerable regions, while a higher ADist suggests that it can pinpoint diverse vulnerable regions rather than clustering all vulnerable pixels within a small area, highlighting its effectiveness in discovering vulnerabilities across the entire input.

***Diverse Adversarial Examples in Non-targeted Scenarios*** Among the three DNN types we evaluated, our algorithm consistently demonstrated proficiency in generating diverse one-pixel adversarial examples. Notably, VGG16 emerged as the most vulnerable, with a success rate of 60.2%, almost twice as high as that of the NiN network. This high success rate suggests that a significant portion of data points lies near the decision boundary along a single dimension. ResNet18 performed the best among the three selected DNNs, with the lowest success rate (ASR) in discovering vulnerable regions and the smallest vulnerable region size (APN).

***Diverse Adversarial Examples in Targeted Scenarios*** With the lowest success rate being 5.0%, it's clear that vulnerabilities are class-specific. Particularly, VGG16 exhibits the largest vulnerable regions for non-targeted attacks, averaging 41.7 pixels. However, when considering class-specific vulnerabilities, VGG16's pixel count drops to 20.6, the smallest among the three studied networks. Further analysis reveals that VGG16 has the highest diversity of target classes to which a single sample can be successfully perturbed, averaging 2.0 classes compared to 1.6 for ResNet18 and 1.7 for NiN. This suggests that vulnerabilities are distributed across multiple target classes, and among the three selected DNNs, ResNet18 shows the most class-specific vulnerabilities.

The lowest 5% success rate in targeted scenarios on ResNet18 reflects the challenging nature of the one-pixel perturbation attack, where only a single pixel is allowed to be modified. Although this low ASR might initially appear to be a limitation, it actually highlights the difficulty of finding a minimal perturbation that can deceive the model in a targeted manner. Given the strict one-pixel constraint, achieving any success, even at 5%, indicates that the model has regions of high vulnerability despite its robust architecture. Additionally, this result is consistent with the performance of previous advanced One-Pixel Attack (Su et al., 2019) where success rates tend to be lower in such constrained settings, further validating the robustness of our findings.

**Table 4** Overall results of our VrdAttack and accumulate results of 4 runs for the One-Pixel Attack on ImageNet dataset for non-targeted attacks

| VrdAttack | AlexNet | ResNet50 | One-Pixel | AlexNet | ResNet50 |
|---|---|---|---|---|---|
| ASR (%) | 11.7 ± 0.5 | 10.7 ± 0.3 | ASR (%) | 10.4 ± 0.3 | 10.6 ± 0.4 |
| APN | 28.5 ± 2.3 | 21.6 ± 3.3 | APN | 2.3 ± 0.2 | 2.1 ± 0.1 |
| ADist | 34.23 ± 3.16 | 23.82 ± 2.75 | ADist | 1.13 ± 0.12 | 2.37 ± 0.25 |
| Cost(s) | 48.13 ± 0.04 | 101.36 ± 0.05 | Cost(s) | 181.33 ± 0.05 | 398.72 ± 0.05 |

Moreover, we validated that the ASR can be further boosted by allowing localized patches, where slightly more pixels are perturbed. This increase in perturbation size leads to a higher success rate but also sacrifices granularity since the adversarial example is no longer based on a minimal perturbation but on the cooperative effect of multiple pixels. Appendix A includes the implementation and results of these experiments, demonstrating the trade-off between success rate and the precision level of perturbation in the attack.

#### 4.2.2 Diverse adversarial examples with ImageNet

In the case of non-targeted vulnerable region discovery, we randomly sample 500 correctly classified images. Due to the significant computational load involved in perturbing each correctly classified sample into the remaining 999 classes, we select five parent classes, each including two child classes: Bird (including snowbird, chickadee), Dog (elkhound, malamute), Plane (space shuttle, warplane), Cat (tabby, Persian), and Ship (pirate ship, schooner). Each child class contains 50 images.

***Diverse adversarial examples with non-targeted scenario*** Due to the higher computational cost associated with high-resolution images, we run the One-Pixel Attack four times on ImageNet and accumulate the results for comparison. Despite this, our algorithm remains effective at generating diverse one-pixel adversarial examples for high-resolution images, as demonstrated in Table 4. These results are consistent with our findings on the CIFAR-10 dataset, further showcasing the robustness of our approach. Examples of vulnerable regions for high-resolution images are presented in the right three images of Fig. 4. ResNet50 performs better with lower ASR and APN. However, it's worth noting that the average number of vulnerable pixels accounts for only a small fraction of the total pixels in these images, making the identified vulnerable regions relatively smaller and sparser compared to those on CIFAR-10.

***Diverse adversarial examples with targeted scenario*** While thousands of attacks are applied to each DNN, only four successful attacks occur for AlexNet, and fourteen are observed for ResNet50, resulting in a success rate of under 1%. Notably, all successful targeted attacks happen between pairs of child classes. These results can be attributable to several reasons: (i) DNNs trained on higher-resolution ImageNet images are less vulnerable to one-pixel modifications, as evidenced in the non-targeted attack section. (ii) In our experiments on CIFAR-10, we find that vulnerable regions typically only enable crossing the decision boundaries of a limited number of classes. On average, each sample can be perturbed to 1.7, 1.6, and 2.0 classes for ResNet18, NiN, and VGG16, respectively. For ImageNet models, we calculate an average of 1.04 classes for AlexNet and 1.13 classes for ResNet50 can be perturbed to, based on non-targeted attack results. Considering we select only 10 classes, there's no guarantee that the targeted vulnerable classes are among them.

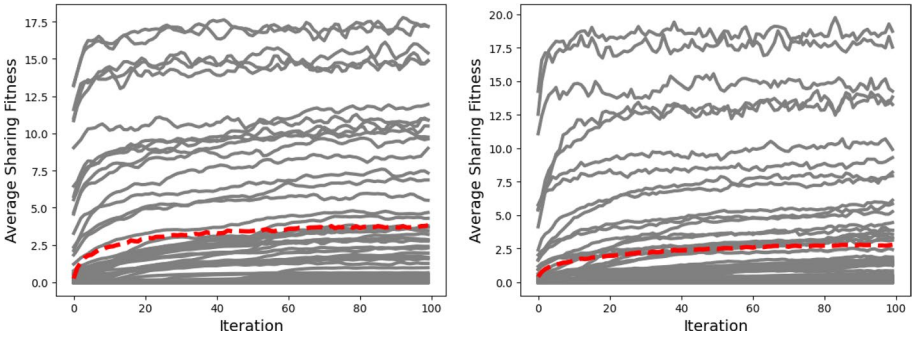


**Fig. 5** Convergence behavior of the algorithm on ResNet-18 (CIFAR-10, left) and ResNet-50 (ImageNet, right). The plots show the average sharing fitness score over iterations. The gray line represents the average fitness on a random subset of test images, while the red dotted line shows the average fitness across all test images (Color figure online)

This phenomenon is not limited to our VrdAttack but is a characteristic of the one-pixel scenario. In fact, even if none of the final acquired perturbations can successfully deceive the DNNs, they are still considered vulnerable, as they can significantly decrease the DNNs' confidence in the true class or increase the DNNs' confidence in the false class. Moreover, effectiveness (ASR) can be improved with more perturbed pixels (e.g., regions with small patches), as validated in Appendix A. However, this comes at the cost of sacrificing granularity, shifting the analysis from the pixel level to the patch level. This can be seen as a trade-off between granularity and effectiveness. In this study, we continue to focus on pixel-level vulnerability analysis and leave comprehensive studies on other levels of vulnerability analysis for future work.

### 4.2.3 Further analysis

***Convergence analysis*** We validate the convergence of our VrdAttack approach on ResNet-18 using CIFAR-10 and ResNet-50 using ImageNet. Since the solution selection is based on its corresponding sharing fitness score, we plot the average sharing fitness score across the population over iterations to validate the convergence behavior. In Fig. 5, the gray line represents the average fitness on a random subset of test images, while the red dotted line shows the average fitness across all test images. As shown in the figure, the average sharing fitness stabilizes before approximately 100 iterations, demonstrating that the algorithm has converged. This suggests that the model has effectively identified vulnerable regions, and additional iterations may yield diminishing improvements in the population quality.

***Selection of sharing radius*** $\gamma_s$ The sharing radius, $\gamma_s$, determines how close two solutions (perturbed pixels) can be before their corresponding fitness scores are penalized. We select the sharing radius based on the ability to identify large vulnerable regions (APN) using different $\gamma_s$ values. This search process is conducted on ResNet-18 with the CIFAR-10 dataset and ResNet-50 with the ImageNet dataset. Due to the differing image resolutions of CIFAR-10 and ImageNet, we set the sharing radius range as follows: for CIFAR-10, $\gamma_s$ ranges from 1 to 8 with a step size of 1, and for ImageNet, $\gamma_s$ ranges from 2 to 16 with a step size of 2. The results presented in Fig. 6 demonstrate that a sharing radius of 4 for CIFAR-10 and 6 for ImageNet allows us to identify the largest vulnerable regions.

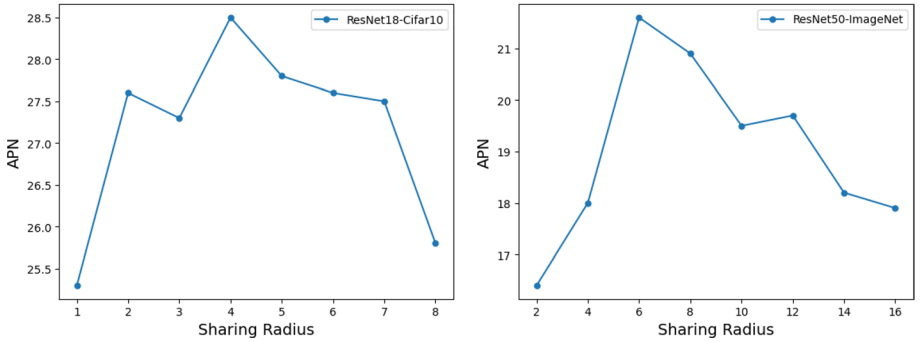


**Fig. 6** Sensitivity of the sharing radius in identifying vulnerable regions

**Table 5** 'Enumerate Pixels' method on CIFAR-10 with non-targeted attack

| Enumerate pixels | ResNet18 | NiN | VGG16 |
|---|---|---|---|
| ASR (%) | $28.5 \pm 0.4$ | $32.1 \pm 0.4$ | $60.0 \pm 0.9$ |
| APN | $23.3 \pm 3.3$ | $31.9 \pm 2.7$ | $30.6 \pm 3.5$ |
| ADist | $4.63 \pm 0.24$ | $5.25 \pm 0.32$ | $5.17 \pm 0.33$ |
| Cost (s) | $8.40 \pm 0.03$ | $8.01 \pm 0.02$ | $7.53 \pm 0.03$ |

***Comparison with Enumerate Pixel Method*** In this section, we compare our proposed VrdAttack method with a baseline approach referred to as the Enumerate Pixels method. In the Enumerate Pixels approach, we iteratively examine each pixel in the input image and optimize the perturbation value for each pixel individually using Differential Evolution (DE). To make this method computationally feasible, we limit the population size to 10, and each pixel perturbation is optimized over 4 iterations. The mutation and recombination rates are set to the same values as those in our VrdAttack method to maintain consistency in the optimization process.

Given an image of size $32 \times 32$ (the CIFAR-10 image resolution), this method results in a query count of 40,960. This is more computationally intensive than our VrdAttack, but still manageable for relatively small images. However, the cost of the Enumerate Pixels approach increases significantly as the image resolution grows. For example, with ImageNet images, which have a resolution of $224 \times 224$, the computation cost increases drastically, leading to a query count that exceeds 2 million. This is more than 25 times the query cost of our VrdAttack method. Due to the excessive computational cost, the Enumerate Pixels method is only validated on CIFAR-10 using non-targeted attacks.

The results for the Enumerate Pixels method are presented in Table 5. Compared to VrdAttack, the Enumerate Pixels method identifies a limited number of vulnerable regions and exhibits less solution diversity. This limitation arises from the relatively low computational budget allocated for perturbation optimization at each pixel, even though the total cost is nearly double that of our approach. In summary, while the Enumerate Pixels method offers a nearly brute-force approach to adversarial attack generation, it becomes computationally prohibitive as image resolution increases. On the other hand, our VrdAttack not only identifies a larger number of vulnerable regions but also does so more efficiently, making it the superior choice for both smaller and larger datasets.

**Fig. 7** Segmentation with grabcut

### 4.3 Vulnerable regions locations

In addition to evaluating the size of vulnerable regions, we are particularly interested in exploring whether these regions might appear in the backgrounds of images. This exploration is driven by our desire to understand the degree to which background elements can influence DNNs' final predictions. If the model is sensitive to background perturbations, it could suggest that it has overfitted to the training data, learning specific background details instead of generalizing to new, unseen backgrounds.

Experiments are conducted with CIFAR-10 for both targeted and non-targeted scenarios. Since CIFAR-10 images are low-resolution and may lack a clear demarcation between foreground and background, we utilize the Grabcut algorithm (Rother et al., 2004) with the human effort to segment the identified objects and background. Examples of this segmentation are provided in Fig. 7.

In the following analysis, we represent the number of vulnerable images with $N$ and the number of vulnerable images with vulnerable regions in the background as $N_{back}$. We denote the total set of vulnerable pixels identified by our VrdAttack as $\mathbb{P}$ and the set of vulnerable pixels in the background as $\mathbb{P}_{back}$. As the vulnerable region is formed by these identified vulnerable pixels, the total vulnerable region size is calculated as $||\mathbb{P}||_0$, denoted as $S$. And the size of vulnerable region in the background is denoted as $S_{back}$. We utilize $\nabla f_i$ to represent the effect caused by perturbing the vulnerable pixel $i$, which indicates the decrease in confidence in the true class for non-targeted attacks and the increase in confidence in the targeted false class for targeted attacks. To evaluate the influence of the location of vulnerable regions, three other metrics are employed:

- **Background Percentage** This metric is defined as $\frac{N_{back}}{N}$. It quantifies the proportion of vulnerable images where vulnerable pixels appear in the background, assessing whether this phenomenon is image-specific.
- **BPixel Percentage** This metric is defined as $\frac{S_{back}}{S}$. It denotes the ratio of vulnerable pixels identified in the background, designed to assess the average size of vulnerable regions in the background of the images.
- **Foreground/Background Effect** These metrics are designed to measure the different effects caused by perturbing vulnerable pixels in various positions. The foreground effect is calculated with $\frac{\sum_{i\in\mathbb{P}, i\notin\mathbb{P}_{back}} \nabla f_i}{S-S_{back}}$, while the background effect is calculated as $\frac{\sum_{i\in\mathbb{P}_{back}} \nabla f_i}{S_{back}}$.

**Table 6** Overall results of vulnerable region location on CIFAR-10 dataset

| Non-targeted | ResNet18 | NiN | VGG16 |
|---|---|---|---|
| Background percentage (%) | 60.6 ± 2.1 | 67.1 ± 1.8 | 66.1 ± 2.4 |
| BPixel percentage (%) | 29.2 ± 3.5 | 39.6 ± 2.3 | 34.7 ± 2.8 |
| Foreground effect (%) | 74.1 ± 2.5 | 64.1 ± 3.0 | 55.2 ± 2.7 |
| Background effect (%) | 58.3 ± 2.2 | 58.1 ± 2.6 | 47.3 ± 3.1 |
| Targeted | ResNet18 | NiN | VGG16 |
| Background percentage (%) | 54.2 ± 1.9 | 60.7 ± 2.0 | 59.1 ± 1.5 |
| BPixel percentage (%) | 30.4 ± 1.2 | 37.5 ± 1.9 | 34.8 ± 1.7 |
| Foreground effect (%) | 73.3 ± 2.1 | 62.6 ± 2.5 | 46.9 ± 3.3 |
| Background effect (%) | 63.2 ± 2.3 | 57.1 ± 2.4 | 41.9 ± 2.8 |

Top: Non-targeted attacks. Bottom: Targeted attacks

#### 4.3.1 Vulnerable regions locations in non-targeted scenario

We re-sample 1000 correctly classified images from the test set (100 images per class) to evaluate all three DNNs. The results, illustrating the impact of perturbation positions, are presented in Table 6.

Among the studied networks, NiN exhibits the highest dependence on vulnerable contextual information. Specifically, 67.1% of the vulnerable images for NiN are identified vulnerable pixels in the background, which is 6.5% higher than that for ResNet18 and 1% higher than that for VGG16. Furthermore, 39.1% of all vulnerable pixels for NiN are detected in the background, surpassing the rates for ResNet18 and VGG16 by 10.4% and 4.9%, respectively. In contrast, ResNet18 performs the best as it shows reduced reliance on vulnerable background features compared to the other models. Even vulnerable pixels in recognized objects (foreground) for ResNet18 result in significantly more misclassifications: 10% more compared to NiN and 18.9% more than VGG16. This observation indicates that although ResNet18 exhibits increased sensitivity to changes in the foreground, it predominantly focuses on object-specific features rather than overly attending to irrelevant background features.

While vulnerable pixels in the foreground do result in a larger drop in the confidence of the ground-truth label, it's noteworthy that perturbations in the image background still lead to a significant average reduction of 54.6% in label confidence. In fact, for over 60% of the images, we identify vulnerable regions in the background.

***Locations with different original classes are quite differently*** The distribution of vulnerable images across different classes is shown in Fig. 8. Certain classes, such as planes, birds, and ships, consistently exhibit a high proportion of vulnerable images with vulnerable pixels located in their background regions. This trend likely stems from the simplistic and recurring background patterns associated with these classes. For instance, images of ships often feature the sea as the background, while plane images frequently include the sky.

To empirically validate the role of these background attributes, we randomly sampled an additional 1000 images (100 per class) from the training set. Among these, we observed that the attribute 'sky' appeared in 81% of sampled plane images, while 62% of sampled ship images included the attribute 'sky,' and 90% included the attribute 'sea.' These findings suggest that the presence of dominant background features in the training data contributes to the model's vulnerability to attacks targeting these regions. These insights

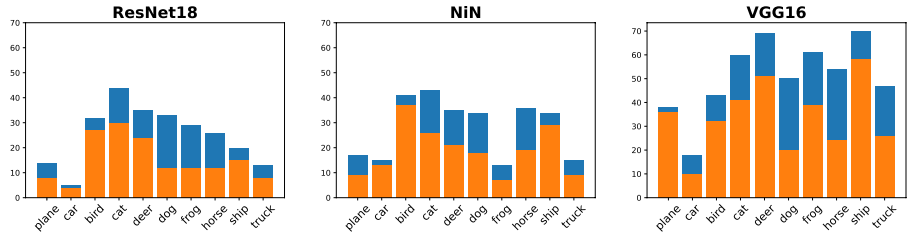


**Fig. 8** Number of successful attacks (vertical axis) for a specific class acting as the original class. Orange bar: Number of vulnerable images where vulnerable regions are found to be located in the image background. Blue bar: Number of vulnerable images where vulnerable regions are found *only* in the foreground

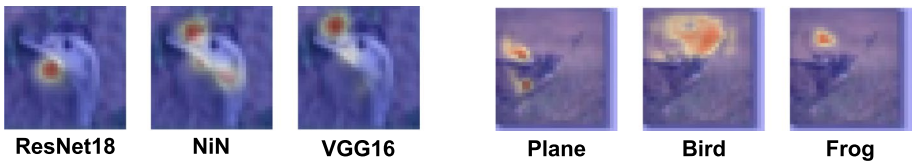


**Fig. 9** Vulnerable regions among different DNNs(left) and targeted classes(right)

**Table 7** Sharing ratio across DNN pairs

| | Res18 &NiN | Res18 &VGG | VGG&NiN | All |
|---|---|---|---|---|
| Image ratio (%) | 14.0 ± 2.1 | 16.5 ± 2.3 | 19.3 ± 1.8 | 9.8 ± 1.5 |
| Pixel ratio (%) | 7.9 ± 1.0 | 6.2 ± 0.7 | 5.4 ± 0.8 | 1.2 ± 0.3 |

emphasize the importance of addressing class-specific vulnerabilities and background-related features in training data. By highlighting how models rely on these contextual patterns for classification, our analysis underscores the need for more diverse and balanced training data. This approach could help ensure models focus on salient object features rather than contextual backgrounds, thereby enhancing both generalization and adversarial robustness.

***If vulnerable regions are shared among different types of DNNs*** We are curious to see if such vulnerable regions are shared across models, given adversarial examples are usually transferred across different models (Inkawhich et al., 2019). The results are presented in Table 7. The 'Image Ratio' represents the percentage of tested images where both DNNs within the DNN pair share at least one vulnerable region. This metric indicates how often both models in a pair are vulnerable in the same image. Meanwhile, 'Pixel Ratio' measures the proportion of overlapping vulnerable regions between the two DNNs within the pair. A higher Pixel Ratio indicates that the same specific regions are vulnerable across both architectures, demonstrating a greater consistency. The VGG & NiN pair has the highest Image Ratio at 19.3%, meaning that in 19.3% of the tested images, both models exhibit vulnerabilities, though the regions may differ. The Res18 & NiN pair shows a lower Image Ratio of 14%, indicating less frequent overlap in the affected images. However, only 9.8% of the images across all pairs show vulnerabilities in both models. For Pixel Ratio, the Res18 &

NiN pair has the highest overlap at 7.9%, suggesting a stronger consistency in the specific pixels targeted by adversarial perturbations. The VGG & NiN pair has a lower Pixel Ratio of 5.4%, showing less overlap in the vulnerable regions. Overall, the Pixel Ratio across all tested DNNs is quite low (1.2%), indicating that while the models share vulnerabilities in the same images, the exact vulnerable regions differ significantly. Heatmaps for a common vulnerable image are provided in the left of Fig. 9. Additional examples can be found in Fig. 16 of Appendix B.

#### 4.3.2 Vulnerable regions locations in the targeted scenario

In the targeted vulnerable region discovery experiment, we re-sample 500 classified images (50 images per class). Table 6 (right) presents the ratio of vulnerable pixels located in the image background and their corresponding effects on different DNNs in the targeted scenario.

***Some vulnerable regions are associated with multiple targeted classes*** We visualize the heatmap of vulnerable regions in the right of Fig. 9, where each pixel's intensity corresponds to an increase in confidence for the target classes. Additional examples are provided in Fig. 15 of Appendix B. With these examples, we discover that there exist common vulnerable regions across different target classes. This indicates that the samples can penetrate the decision boundaries of multiple classes along these specific dimensions. However, a detailed analysis revealed that the percentage of attacked positions, which can lead to perturbations causing more than one target class, is only 3.0%, 1.9%, and 2.9% for ResNet18, NiN, and VGG16, respectively. While vulnerable regions for different target classes are primarily distinct, these findings highlight that certain shared regions exist, representing weaknesses in the model's decision boundaries. These observations underscore the need for further investigation into the nature of such shared vulnerable regions and their role in adversarial susceptibility.

***Locations of vulnerable pixels with true-target Pairs*** The results in Fig. 10 illustrate the average number of vulnerable pixels detected for each ture-target class pair. Beyond findings from the previous work (Su et al., 2019), we have identified additional intriguing properties. For instance, cats (class 3) are more easily perturbed to dogs and vice versa (Su et al., 2019). Interestingly, the perturbations are predominantly located on the cat or dog itself, rather than the background. Additionally, we have observed that images of ships (class 8) are susceptible to being perturbed into the plane class through perturbations in the background. However, the opposite-planes being perturbed to ships-is less common. This difference may be attributed to the similar but not identical background attributes shared by these two classes. Notably, the attribute sky appears more frequently in ship images (60 out of 100), while the presence of the sea is less common in plane images (only 7 out of 100).

### 4.4 Vulnerable regions with adversarial training

Adversarial training is one of the most effective methods for defending against $\ell_\infty$ attacks. In this section, we assess the impact on the vulnerable regions of CIFAR-10, focusing on two leading adversarial training algorithms: PGD (Madry et al., 2018) and TRADES (Zhang et al., 2019), as implemented on ResNet18. The models were trained using SGD with a momentum of 0.9 and a weight decay of $2 \times 10^{-4}$. The initial learning rate was set at 0.1 and was reduced by a factor of 10 at the 75th, 90th, and 100th

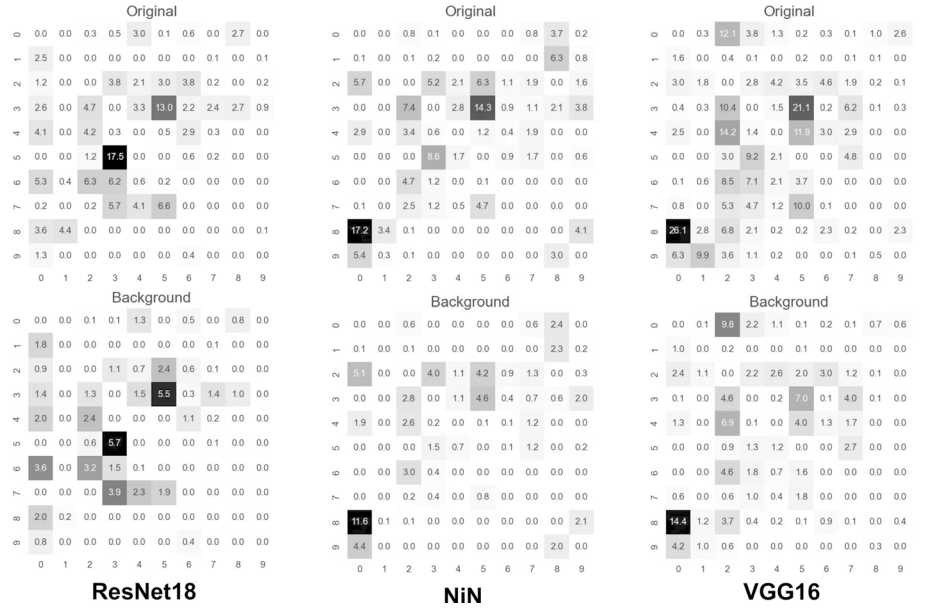


**Fig. 10** The average number of adversarial pixels for the ture-target class pair with standard trained DNNs. Vertical and horizontal indices indicate respectively the original and target classes. Original means the average number of adversarial pixels with the whole image. While 'Original' means the average number of adversarial pixels found on the whole image, 'Background' indicates the average number of adversarial pixels on the background. The classes are identified by numbers from 0 to 9, representing the following classes respectively: plane, car, bird, cat deer, dog, frog, horse, ship, and truck

**Table 8** Overall results of our VrdAttack and the accumulated results from 10 runs of the One-Pixel Attack (Su et al., 2019) with adversarially trained models for non-targeted attacks

| VrdAttack | PGD | TRADES | One-Pixel | PGD | TRADES |
|---|---|---|---|---|---|
| ASR (%) | $14.0 \pm 0.3$ | $9.7 \pm 0.4$ | ASR (%) | $13.9 \pm 0.6$ | $9.2 \pm 0.2$ |
| APN | $29.0 \pm 2.3$ | $17.8 \pm 3.1$ | APN | $1.4 \pm 0.1$ | $1.2 \pm 0.1$ |
| ADist | $5.21 \pm 0.33$ | $6.46 \pm 0.21$ | ADist | $0.37 \pm 0.05$ | $0.52 \pm 0.11$ |

epochs. The training lasted for a total of 120 epochs. The maximum perturbation was set to $\epsilon = 0.031$, and the step size was $\epsilon/4$.

***Adversarial training reduce the vulnerabilities along single pixels*** We apply the non-targeted attack on the same sampled 1000 images for standard trained models. Table 8 presents a comparison with the One-Pixel Attack (Su et al., 2019), revealing consistent results with those observed on standardly trained models. Notably, our VrdAttack identifies, on average, 27.6 more vulnerable pixels than the One-Pixel Attack across 10 runs on PGD-trained models and 16.6 more on TRADES-trained models. Additionally, the ADist metric, which measures the average distance between successful perturbations, highlights our algorithm's ability to identify vulnerable pixels across a broader range of positions. The results of our VrdAttack further reveal a significant reduction in the number of vulnerable images when using adversarially trained models, with decreases of 14.4% for PGD and 18.7% for TRADES. Beyond the reduction in

**Table 9** Location of pixel perturbations with different adversarial trained models

| Non-targeted | PGD | TRADES |
|---|---|---|
| Background percentage (%) | 68.9 ± 2.5 | 65.8 ± 2.3 |
| BPixel percentage (%) | 49.2 ± 3.1 | 53.6 ± 2.8 |
| Foreground effect (%) | 31.4 ± 1.6 | 5.0 ± 0.7 |
| Background effect (%) | 20.9 ± 1.2 | 7.6 ± 1.0 |

vulnerable regions, we also observe that the sensitivity of these regions is substantially weakened. These findings provide additional evidence that $\ell_\infty$ adversarial training effectively smooths the loss landscape not only within the $\ell_\infty$ ball (Madry et al., 2018) but also across individual dimensions of the input images.

***The effect of adversarial training on the location of vulnerable regions*** The severity of vulnerable regions identified by our approach at different positions is presented in Table 9. While the success rate of attacks significantly decreases, vulnerable regions are more likely to appear in background areas compared to standard-trained models. This suggests that adversarial training better smooths the loss landscape along dimensions associated with objects, reducing the model's sensitivity to perturbations in these regions. Notably, the impact on the model's output is largely diminished regardless of the location of the vulnerable pixels. For example, TRADES-trained models exhibit reductions of 69.1% and 50.7% in changes to label confidence.

Interestingly, we observe that vulnerable pixels located in the background exert a slightly larger influence than those in the foreground for TRADES-trained models. This indicates that while TRADES is effective at mitigating the impact of foreground vulnerabilities, it may not fully address biases in background elements. These findings highlight the need for targeted adversarial training strategies focusing on background regions, such as applying larger perturbations to these areas, to further smooth decision boundaries and enhance model robustness. Examples of vulnerable region heatmap for DNNs with different training algorithms are presented in Fig. 12, and more examples can be found in Fig. 17 of Appendix B.

***The effect of adversarial training on vulnerable pairs*** We conduct an experiment to identify targeted vulnerable regions and observe how the average number of vulnerable pixels in various locations changes for different original-target pairs. We reuse the 500 correctly classified images for standard trained and conducted the targeted attack on the adversarial trained models. The experiment results are presented in Fig. 11. From the results, we notice a decrease in the number of diverse vulnerable pixels for most original-target class pairs. TRADES is found to perform better in eliminating the adversarial vulnerable regions. Notably, no vulnerable pixels are found for the original class 'plane' to other target classes. However, there are still some class pairs where the vulnerable regions increase after adversarial training. For the PGD-trained model, the class 'bird', 'cat', and 'deer' (class 2, 3, 4) are more likely to be perturbed than the class 'frog' (class 6). And the class 'bird' (class 2) is more easily to perturbed than the class 'plane' (class 0). This suggests that while adversarial training aims to eliminate a majority of the vulnerabilities, it inadvertently introduces new vulnerabilities between other class pairs.

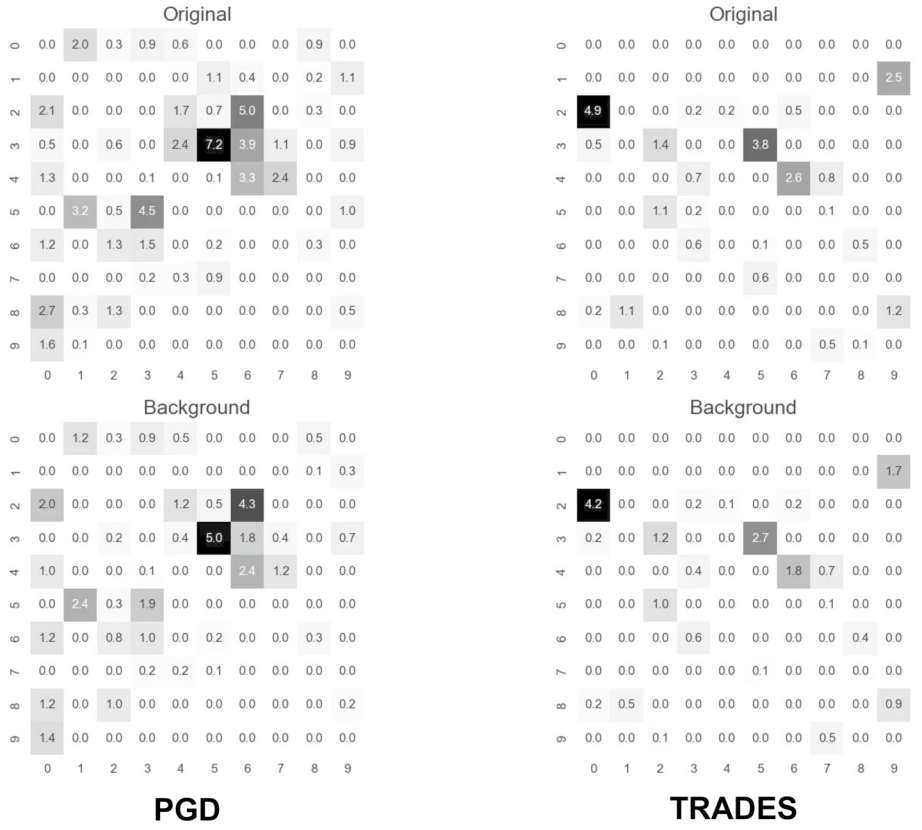


**Fig. 11** The average number of adversarial pixels for the original-target class pair with adversarial trained DNNs


**Fig. 12** Vulnerable regions with differently trained models

### 4.5 Compare with important regions of the explainable approach

Explainable DNNs aim to identify the key features that influence their predictions. In contrast, our research delves deeper by focusing on the detection of vulnerable regions, i.e., a single pixel perturbation can successfully deceive DNNs. Interestingly, these regions may not always be highlighted by standard explainability techniques. As demonstrated in Fig. 13, there is a clear distinction between these vulnerable regions and the regions deemed important by explainable DNNs.

More importantly, the effectiveness of such vulnerable pixels in deceiving DNNs raises critical questions about the assumptions underlying current explainability methods. If these

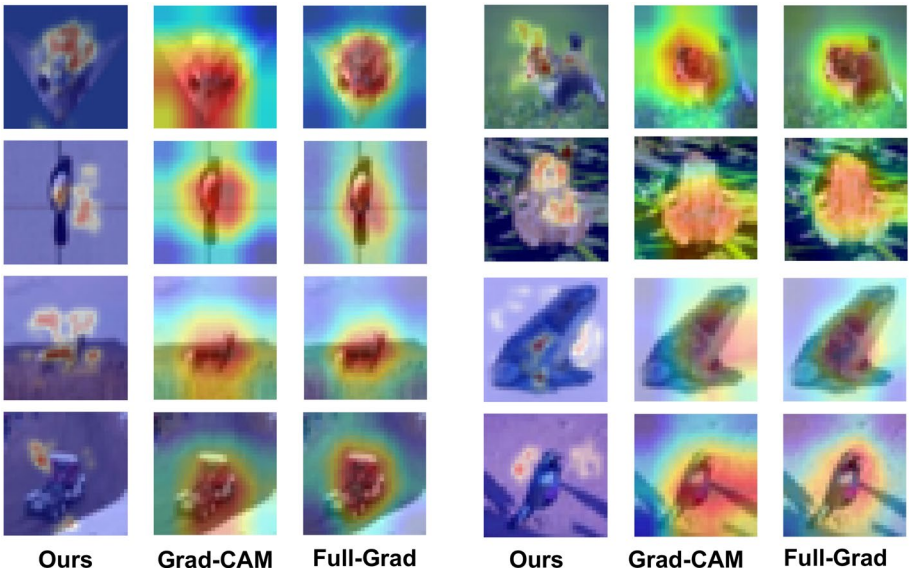


**Fig. 13** Vulnerable regions discovered by our algorithm and Important regions discovered by Grad-CAM (Selvaraju et al., 2017) and Full-Grad methods (Srinivas & Fleuret, 2019)

perturbations occur in regions considered non-essential by explainability standards, it begs the question of whether our understanding of what makes an area of an image significant needs reevaluation. This discrepancy suggests that the robustness and reliability of DNNs could be compromised by overlooking such vulnerabilities. Investigating why DNNs are susceptible to these seemingly trivial manipulations not only challenges the robustness of neural networks but also calls for a deeper integration of security concerns into the field of neural network explainability.

## 5 Conclusion

In this study, we introduced the Vulnerable Region Discovery Attack (VrdAttack), a novel approach designed to efficiently identify vulnerable regions in deep neural networks (DNNs) using diverse adversarial examples. Through extensive experimentation, our results demonstrate that VrdAttack effectively uncovers a significant number of vulnerable regions, both in the primary focus areas of the input and in typically overlooked background regions. This comprehensive identification of vulnerabilities not only improves the robustness of DNNs but also contributes to the interpretability of these models by revealing the specific input areas that disproportionately influence decision-making.

By highlighting the model's sensitivity to areas that may be irrelevant or semantically insignificant, our approach offers valuable insights into the decision process of DNNs, helping to address issues of trust and transparency in AI systems. The discovery of background vulnerabilities, in particular, underscores the need to reconsider traditional interpretability methods, which often focus on salient features, and signals potential areas for improvement in model training and design.

Looking ahead, our work opens several promising directions for future research. These include adapting our methodology to high-resolution images, exploring the impact of different perturbation strategies, and investigating the underlying factors contributing to the vulnerability of specific regions. Ultimately, we aim to refine our approach further, contributing to the development of more robust and interpretable DNNs and advancing the overall security and reliability of AI systems.

## Appendix A

### Additional experiments with diverse adversarial patches

While our proposed approach excels in identifying vulnerable regions susceptible to vulnerable pixels, it does exhibit limitations when applied to higher-resolution images. To further boost the attack success rate of our VrdAttack, we conducted additional experiments by adapting it from pixel-level vulnerability assessment to patch-level vulnerability analysis. In the context of patch attacks, each candidate solution, denoted as $z_i \in \mathbb{S}$, can be modified to a tuple containing the coordinates of the top-left pixel of the patch and the RGB values of different perturbed pixels within the patch. As a result, the tuple comprises $2 + 3 \times n$ elements, where $n$ represents the number of perturbed pixels.

For our CIFAR-10 experiment, we utilized $2 \times 2$ pixel patches with 500 correctly classified images. In the case of ImageNet, where images are approximately 50 times larger than those in CIFAR-10, we employed $4 \times 4$ pixel patches with 100 correctly classified images. The results of these experiments are presented in Tables 10 and 11, where the term 'region size' refers to the average number of distinct perturbed pixels covered by diverse adversarial patches. The improved success rate and expanded region size demonstrate the adaptability of our algorithm to varying requirements. However, this modification shifts our analytical focus from individual pixels to patch-level analysis, striking a balance between achieving a higher success rate and maintaining the granularity of vulnerability assessment. We leave a comprehensive analysis for future work.

**Table 10** Results of attacks with $2 \times 2$ pixel patch with our adapted algorithm on CIFAR-10

| CIFAR-10 | Perturbation Level | Success Rate | Region Size |
|---|---|---|---|
| VGG16 (ours) | 2x2 pixel patch | 83.2% | 171.3 |
| NiN (ours) | 2x2 pixel patch | 57.4% | 122.4 |
| ResNet18 (ours) | 2x2 pixel patch | 51.2% | 105.7 |

Region size refers to the average total number of perturbed pixels covered in each successful attack

**Table 11** Results of attacks with $4 \times 4$ patch with our adapted algorithm on ImageNet

| | Perturbation Level | Success Rate | Region Size |
|---|---|---|---|
| AlexNet (ours) | 4x4 pixel patch | 40.0% | 1106.2 |
| ResNet50 (ours) | 4x4 pixel patch | 28.0% | 1920.5 |

Region size refers to the average total number of perturbed pixels covered in each successful attack

## Appendix B

### Additional examples of vulnerable regions

In this subsection, we provide extended visual illustrations to better highlight the vulnerable regions pinpointed by our algorithm. Figure 14 depicts these areas across varying architectures for both the CIFAR10 and ImageNet datasets. Notably, these regions can be identified on both recognized objects and backgrounds, independent of the image resolution. In Figs. 15 and 16, we exhibit vulnerable regions of the same images across different DNNs and different target classes, respectively,a shared vulnerability pattern. We provide more examples of vulnerable regions for adversarial trained models in Fig. 17, demonstrating that adversarial trained models are more likely to have vulnerable regions in the background.

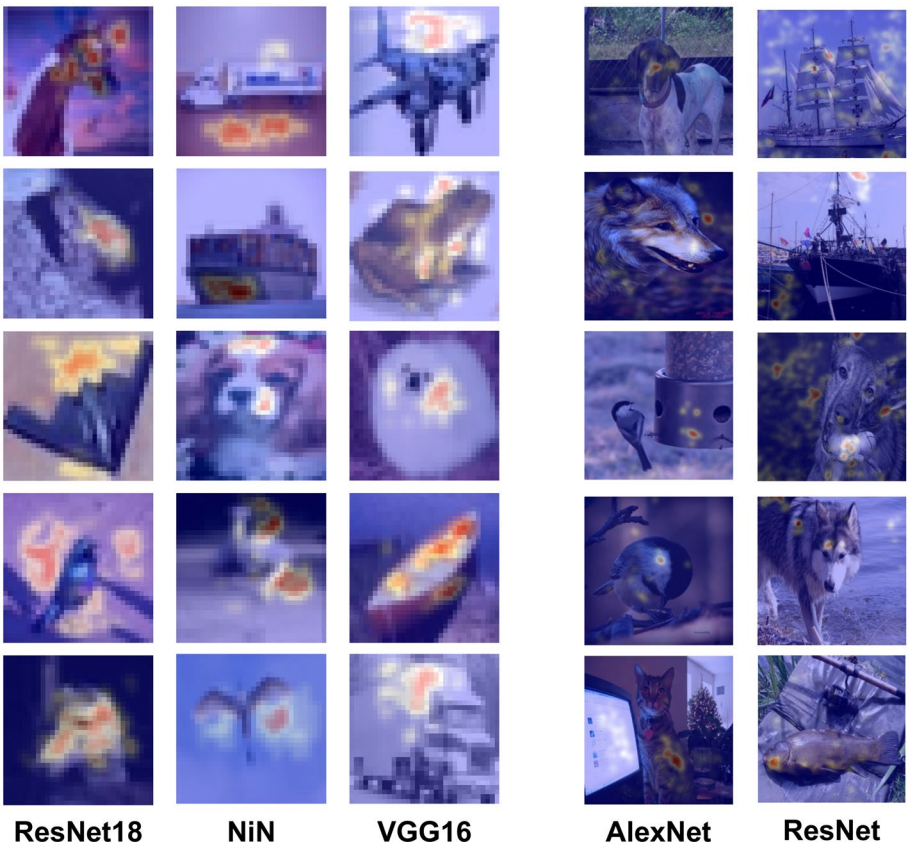


**Fig. 14** Visualization of vulnerable regions with CIFAR10 (left) and ImageNet (right)


**Fig. 15** Visualization of vulnerable regions to different false classes

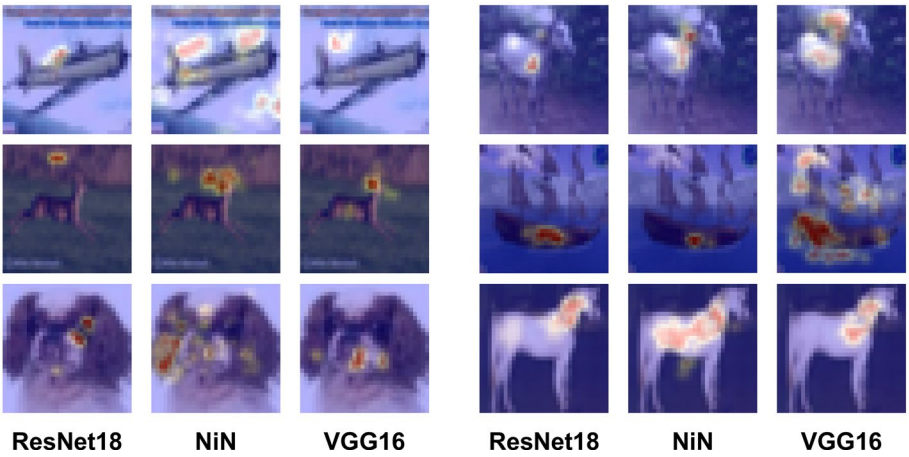


**Fig. 16** Visualization of vulnerable regions for shared vulnerable images by 3 different DNNs

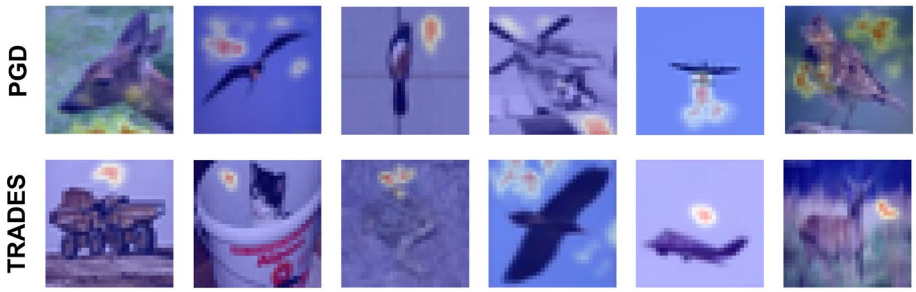


**Fig. 17** Vulnerable regions of adversarial trained ResNet18 Models

**Author contributions** YZ and XY conceived and developed the main ideas for this paper. YZ drafted the initial manuscript and conducted the experiments. XY supervised the project, providing substantial guidance, feedback, and overseeing organization. WH and WL contributed valuable feedback and played key roles in revising the manuscript critically.

**Funding** Open Access Publishing Support Fund provided by Lingnan University. This work was supported by NSFC (Grant No. 62250710682), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386).

**Data availability** The datasets used in this work, CIFAR-10 and ImageNet, are both publicly available. CIFAR-10 dataset can be accessed at https://www.cs.toronto.edu/~kriz/cifar.html and ImageNet dataset at https://www.image-net.org/challenges/LSVRC/index.php.

**Materials availability** Not applicable.

**Code availability** The source code is provided in the associated repository at https://github.com/YunCe-Code/Vulnerable-Region-Discovery.git.

## Declarations

**Conflict of interest** The authors declare no conflict of interest.

**Ethical approval** Not applicable.

**Informed consent** Not applicable.

**Consent for publication** Not applicable.



## References

Andriushchenko, M., Croce, F., Flammarion, N., spsampsps Hein, M. (2020). Square attack: A query-efficient black-box adversarial attack via random search. In: *European conference on computer vision* (pp. 484–501). https://doi.org/10.1007/978-3-030-58592-1_29

Bai, Y., Wang, Y., Zeng, Y., Jiang, Y., & Xia, S. T. (2023). Query efficient black-box adversarial attack on deep neural networks. *Pattern Recognition, 133*, Article 109037. https://doi.org/10.1016/j.patcog.2022.109037

Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In: *2017 IEEE symposium on security and privacy (SP)* (pp. 39–57). https://doi.org/10.1109/SP.2017.49

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., & Hsieh, C.-J. (2017). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: *Proceedings of the 10th ACM workshop on artificial intelligence and security. AISec '17* (pp. 15–26). Association for Computing Machinery. https://doi.org/10.1145/3128572.3140448

Chiou, J.-P., Chang, C.-F., & Su, C.-T. (2004). Ant direction hybrid differential evolution for solving large capacitor placement problems. *IEEE Transactions on Power Systems, 19*(4), 1794–1800. https://doi.org/10.1109/TPWRS.2004.835651

Croce, F., Andriushchenko, M., Singh, N.D., Flammarion, N., & Hein, M. (2022). Sparse-rs: A versatile framework for query-efficient sparse black-box adversarial attacks. In: *Proceedings of the AAAI conference on artificial intelligence* (vol. 36, pp. 6437–6445).

Darwen, P., & Yao, X. (1996). Every niching method has its niche: Fitness sharing and implicit sharing compared. In H.-M. Voigt, W. Ebeling, I. Rechenberg, & H.-P. Schwefel (Eds.), *Parallel problem solving from nature—PPSN IV* (pp. 398–407). Springer. https://doi.org/10.1007/3-540-61723-X_1004

Fong, R., Patrick, M., & Vedaldi, A. (2019). Understanding deep networks via extremal perturbations and smooth masks. In: *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2950–2958).

Goldberg, D.E., & Richardson, J.(1987). Genetic algorithms with sharing for multimodal function optimization. In: *Proceedings of the second international conference on genetic algorithms on genetic algorithms and their application* (pp. 41–49).

Goodfellow, I.J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. Preprint retrieved from http://arxiv.org/abs/1412.6572

Guo, C., Gardner, J., You, Y., Wilson, A.G., & Weinberger, K. (2019). Simple black-box adversarial attacks. In: *International conference on machine learning* (pp. 2484–2493). PMLR.

He, K., Zhang, X., Ren, S., & Sun, J.(2016). Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Ilyas, A., Engstrom, L., Athalye, A., & Lin, J. (2018). Black-box adversarial attacks with limited queries and information. In: *Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research* (vol. 80, pp. 2137–2146).

Inkawhich, N., Wen, W., Li, H.H., & Chen, Y. (2019). Feature space perturbations yield more transferable adversarial examples. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7066–7074).

Kurakin, A., Goodfellow, I., & Bengio, S.(2016). Adversarial examples in the physical world. Preprint retrieved from http://arxiv.org/abs/1607.02533

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B.(2021). Swin transformer: Hierarchical vision transformer using shifted windows. In: *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 10012–10022).

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In: *International conference on learning representations*. https://openreview.net/forum?id=rJzIBfZAb

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1765–1773).

Nguyen, Q. U., Nguyen, X. H., O'Neill, M., & Agapitos, A. (2012). An investigation of fitness sharing with semantic and syntactic distance metrics. In A. Moraglio, S. Silva, K. Krawiec, P. Machado, & C. Cotta (Eds.), *Genetic programming* (pp. 109–120). Springer.

Pouyanfar, S., Chen, S.-C., & Shyu, M.-L.(2017). An efficient deep residual-inception network for multimedia classification. In: *2017 IEEE international conference on multimedia and expo (ICME)* (pp. 373–378). https://doi.org/10.1109/ICME.2017.8019447

Rao, S., Stutz, D., spsampsps Schiele, B. (2020). Adversarial training against location-optimized adversarial patches. In: *European conference on computer vision* (pp. 429–448). Springer. https://doi.org/10.1007/978-3-030-68238-5_32

Ribeiro, M.T., Singh, S., & Guestrin, C. (2016). Why should i trust you? Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144). https://doi.org/10.1145/2939672.2939778

Rother, C., Kolmogorov, V., & Blake, A. (2004). Grabcut interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG), 23*(3), 309–314. https://doi.org/10.1145/1015706.1015720

Sareni, B., & Krahenbuhl, L. (1998). Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation, 2*(3), 97–106. https://doi.org/10.1109/4235.735432

Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: visual explanations from deep networks via gradient-based localization. In: *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Shi, Y., Han, Y., Hu, Q., Yang, Y., & Tian, Q. (2022). Query-efficient black-box adversarial attack with customized iteration and sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(2), 2226–2245. https://doi.org/10.1109/TPAMI.2022.3169802

Srinivas, S., & Fleuret, F.(2019). Full-gradient representation for neural network visualization. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R. (eds.) *Advances in neural information processing systems* (vol. 32). https://proceedings.neurips.cc/paper_files/paper/2019/file/80537a945c7aaa788ccfcdf1b99b5d8f-Paper.pdf

Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359. https://doi.org/10.1023/A:1008202821328

Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation, 23*(5), 828–841. https://doi.org/10.1109/TEVC.2019.2890858

Sun, C., Zhang, Y., Chaoqun, W., Wang, Q., Li, Y., Liu, T., Han, B., & Tian, X. (2022). Towards lightweight black-box attack against deep neural networks. *Advances in Neural Information Processing Systems, 35*, 19319–19331.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R.(2013). Intriguing properties of neural networks. Preprint retrieved from http://arxiv.org/abs/1312.6199

Xie, C., Wu, Y., Maaten, L.v.d., Yuille, A.L., & He, K.(2019). Feature denoising for improving adversarial robustness. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 501–509).

Xu, K., Liu, S., Zhao, P., Chen, P.-Y., Zhang, H., Fan, Q., Erdogmus, D., Wang, Y., & Lin, X .(2018). Structured adversarial attack: Towards general implementation and better interpretability. Preprint retrieved from http://arxiv.org/abs/1808.01664

Yang, Q., Zhu, X., Fwu, J.-K., Ye, Y., You, G., & Zhu, Y. (2021). Mfpp: Morphological fragmental perturbation pyramid for black-box model explanations. In: *2020 25th international conference on pattern recognition (ICPR)* (pp. 1376–1383). IEEE.

Zeiler, M.D., spsampsps Fergus, R. (2014). Visualizing and understanding convolutional networks. In: *European conference on computer vision)* (pp. 818–833). Springer. https://doi.org/10.1007/978-3-319-10590-1_53

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., & Jordan, M.(2019). Theoretically principled trade-off between robustness and accuracy. In: *International conference on machine learning* (pp. 7472–7482). PMLR.

Zhang, S., Gao, H., Shu, C., Cao, X., Zhou, Y., & He, J. (2024). Black-box Bayesian adversarial attack with transferable priors. *Machine Learning, 113*(4), 1511–1528. https://doi.org/10.1007/s10994-022-06251-3

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2921–2929)

Zhou, X., Tsang, I. W., & Yin, J. (2023). Ladder: Latent boundary-guided adversarial training. *Machine Learning, 112*(10), 3851–3879. https://doi.org/10.1007/s10994-022-06203-x



## Authors and Affiliations

**Yunce Zhao[1,2] · Wei Huang[3] · Wei Liu[2] · Xin Yao[4]**

✉ Xin Yao
xinyao@ln.edu.hk

Yunce Zhao
yunce.zhao2023@gmail.com

Wei Huang
wei.huang.vr@riken.jp

Wei Liu
wei.liu@uts.edu.au

[1] Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, Guangdong, China

[2] School of Computer Science, FEIT, University of Technology Sydney, Sydney, NSW 2007, Australia

[3] RIKEN Center for Advanced Intelligence Project (AIP), Tokyo 103-0027, Japan

[4] School of Data Science, Lingnan University, Hong Kong SAR, China