

Article

RISOPA: Rapid Imperceptible Strong One-Pixel Attacks in Deep Neural Networks

Wonhong Nam ¹, Kunha Kim ¹, Hyunwoo Moon ¹, Hyeongmin Noh ¹, Jiyeon Park ¹ and Hyunyoung Kil ^{2,*}¹ Department of Computer Science and Engineering, Konkuk University, Seoul 05029, Republic of Korea; wnam@konkuk.ac.kr (W.N.)² Department of Software, Korea Aerospace University, Goyang 10540, Republic of Korea

* Correspondence: hykil@kau.ac.kr

Abstract: Recent research has revealed that subtle imperceptible perturbations can deceive well-trained neural network models, leading to inaccurate outcomes. These instances, known as adversarial examples, pose significant threats to the secure application of machine learning techniques in safety-critical systems. In this paper, we delve into the study of one-pixel attacks in deep neural networks, recently reported as a kind of adversarial examples. To identify such one-pixel attacks, most existing methodologies rely on the differential evolution method, which utilizes random selection from the current population to escape local optima. However, the differential evolution technique might waste search time and overlook good solutions if the number of iterations is insufficient. Hence, in this paper, we propose a gradient ascent with momentum approach to efficiently discover good solutions for the one-pixel attack problem. As our method takes a more direct route to the goal compared to existing methods relying on blind random walks, it can effectively identify one-pixel attacks. Our experiments conducted on popular CNNs demonstrate that, in comparison with existing methodologies, our technique can detect one-pixel attacks significantly faster.

Keywords: one-pixel attack; adversarial example; neural network; image recognition; CIFAR-10

MSC: 68T07



Citation: Nam, W.; Kim, K.; Moon, H.; Noh, H.; Park, J.; Kil, H. RISOPA: Rapid Imperceptible Strong One-Pixel Attacks in Deep Neural Networks. *Mathematics* **2024**, *12*, 1083. <https://doi.org/10.3390/math12071083>

Academic Editor: Marjan Mernik

Received: 27 February 2024

Revised: 23 March 2024

Accepted: 24 March 2024

Published: 3 April 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the past decade, machine learning techniques, including deep neural networks, have achieved remarkable progress [1,2]. The accuracy of artificial intelligence, especially in image classification, recommendation systems, and natural language processing, matches or even surpasses human cognitive abilities. Despite these accomplishments, recent research has unveiled that imperceptible alterations to input images can render well-trained networks unstable. In other words, *adversarial perturbations* lead them to misclassify input images [3,4]. Consequently, *adversarial examples* containing imperceptible perturbations are viewed as significant barriers to the integration of neural networks into safety-critical systems, such as autonomous vehicles and air traffic collision avoidance systems.

In this study, we delve into *one-pixel attacks* [5] within the domain of adversarial examples. These are input images that vary from the original input image by *just one pixel*, yet cause a given neural network to lead to misclassify them. Most existing methods [5,6] for identifying one-pixel attacks utilize the differential evolution (DE) algorithm [7], which is a metaheuristic approach for solving global optimization problems. This algorithm iteratively refines candidate solutions through an evolutionary process. DE endeavors to discover an optimal solution by initializing a population of candidate solutions, generating new candidates through a combination of randomly chosen current solutions, followed by a crossover operation, and then selecting the most suitable solution based on a specified condition. However, the way the DE algorithm performs to identify the optimal solution, which does not utilize the gradient of an objective function but rather chooses the current

solutions randomly and combines them, might waste search time. Additionally, it might occasionally overlook good/optimal solutions if the number of iterations is not sufficient. Therefore, first, we thoroughly observe the change in confidence scores of neural network models by altering the attack pixels and their pixel values. Our observation shows that the distribution of confidence scores has only a few local optima. Based on this observation, our algorithm adopts a gradient ascent with momentum method [1] to find one-pixel attacks faster than the existing works [5,6]. In addition, while the previous work OPA2D [6] uses the sensitivity of each pixel in a given input image to select candidate pixels, we propose a candidate pixel map that represents how much each pixel in the input image increases the confidence score for the target class when its color is maximally altered. Another direction of this study is that we consider three distinct objectives, as in the one-pixel attack problem, users (such as attackers) may desire different one-pixel attack strategies depending on their situations. Our rapid attack (RA) version terminates our investigation and returns the current result as soon as the attack succeeds. On the other hand, the imperceptible attack (IA) aims to identify the one-pixel attack that is closest to a given input. Lastly, the strong attack (SA) continues searching to find the optimal confidence score among attacks. We present our experimental results on popular CNNs using the CIFAR-10 dataset [8] to demonstrate that our technique outperforms the existing methods. Specifically, compared with existing methods, our SA discovers similar quality attacks 4.8 times faster and IA identifies them 2.6 times faster on average, respectively. We remark that while most works [9–11] for adversarial attacks studied a white-box attack, which uses the internal information of the neural network model, the one-pixel attack we study in this paper is a black-box attack, wherein attacker algorithms can utilize *only* the output of the neural network model.

The contributions of this paper are as follows:

- This study conducts a thorough experiment with popular CNNs for one-pixel attacks using the CIFAR-10 dataset. The experimental results provide deep insights into one-pixel attacks for digital images, which can enhance the robustness of neural networks.
- Based on our observations, we propose a novel rapid imperceptible strong one-pixel attack algorithm for three distinct objectives.
- We develop an efficient tool to identify one-pixel attacks using the proposed algorithm. Compared to existing methods, it can identify one-pixel attacks on average 4.8 times faster for SA and 2.6 times faster for IA, respectively.

The remainder of this paper is organized as follows: Section 2 presents the related research. In Section 3, we formalize the one-pixel attack problem studied in this paper. In Section 4, we present our observations regarding confidence score by changing pixel values. Section 5 introduces our rapid imperceptible strong one-pixel attack algorithm, which is based on a candidate pixel map and a gradient ascent with momentum method. In Section 6, we present the experimental results to validate the proposed technique. Finally, the conclusions are presented in Section 7.

2. Related Work

Recently, Szegedy et al. [3] highlighted the existence of adversarial examples, which are input images with imperceptible perturbations that lead neural network models to misclassify them. These adversarial examples were generated to be visually similar to the original input yet caused misclassification across a number of well-trained networks. Subsequent research [11–16] has aimed to identify, detect, and mitigate these adversarial examples. Zhang et al. [14] proposed to treat the DNN logits as a vector for feature representation, and derived a method using random source images as a proxy dataset to generate targeted UAPs (Universal Adversarial Perturbations) without the original training data. Ho et al. [13] addressed the selection of challenging negative pairs, by introducing a new family of adversarial examples for contrastive learning and using these examples to define a new adversarial training algorithm for self-supervised learning. Hendrycks et al. [15] introduced datasets collected with a simple adversarial filtration technique to create datasets

with limited spurious cues. Xie et al. [12] proposed an adversarial training method that treats adversarial examples as additional examples to prevent overfitting. Their method employs a separate auxiliary batch norm for adversarial examples since they have different underlying distributions to normal examples. Furthermore, other studies [11,16] proposed efficient techniques to enhance neural network robustness by defending against subtle adversarial attacks.

Su et al. [5] first introduced the concept of a one-pixel attack problem and utilized a differential evolution approach to perturb input images. Nguyen-Son et al. [6] proposed the OPA2D framework, a comprehensive solution that generates, detects, and defends one-pixel attacks. Their method identifies vulnerable pixels by analyzing discrepancies in confidence scores. Similarly, Korpihalkola et al. [17] employed the differential evolution technique for digital pathology images, building upon the foundation laid by [5]. While previous endeavors [5,6,17] proposed heuristic methodologies utilizing differential evolution to identify one-pixel attacks, that of Nam et al. [18] was based on rigorous experimentation on the MNIST dataset and proposed an adjustable exhaustive search method, leveraging parallelism in conjunction with the inherent properties of one-pixel attacks.

The formal verification of neural networks [19–24] is another valuable research direction for enhancing neural network robustness. Pulina and Tacchella [19] introduced a pivotal study on formal verification for neural networks. Their work [19,20] proposed a method for verifying local and global invariants in multi-layer perceptions. Gehr [21] introduced AI², the first sound and scalable analyzer for deep neural networks. AI² is capable of automatically proving safety properties for realistic CNNs through the abstraction of input points into zonotopes. For fine-grained abstraction, Muller et al. [22] employed polyhedra with intervals, while Tran et al. [23] utilized ImageStar, a representation for a set of input images. Pham et al. [24] introduced an approach to verify whether a given neural network is free from a backdoor with a certain level of success rate. Their approach integrates statistical sampling and abstract interpretation.

3. One-Pixel Attacks

An *adversarial example* [3] is an input to a neural network that includes a slight perturbation from the original input, yet confounds the neural network by causing it to classify the input into an incorrect class. As an extremely limited scenario of adversarial examples, Su et al. [5] first proposed *one-pixel attacks* in the image classification field.

Now, we formalize the concept of one-pixel attacks that we study in this paper.

Definition 1 (Neural network as an image classifier). *Given an input image $\mathbf{x} = (x_1, \dots, x_n)$ which is a vector wherein each scalar element x_i represents a pixel, a neural network f classifies $\mathbf{x}$ into a class t that corresponds to the index of the node in the output layer with the largest value, i.e., $t = \arg\max y_i$, where each y_i is the output value of f and is referred to as the confidence score.*

Definition 2 (Adversarial perturbation). *An additive adversarial perturbation for the input $\mathbf{x}$ is a vector $\mathbf{e} = (e_1, \dots, e_n)$ with the same size as $\mathbf{x}$.*

Definition 3 (Adversarial example). *Given a neural network f and its input $\mathbf{x}$ which is classified as t_x , an adversarial example with regard to f and $\mathbf{x}$ is an input $\mathbf{x}' = \mathbf{x} + \mathbf{e}$ such that f classifies $\mathbf{x}'$ into the class $t_{x'}$, $t_x \neq t_{x'}$, and $\|\mathbf{e}\| \leq \delta$ for some distance threshold δ .*

Definition 4 (One-pixel attack). *As a special case of adversarial examples, for a given neural network f and its input $\mathbf{x}$, a one-pixel attack is an input $\mathbf{x}' = \mathbf{x} + \mathbf{e}$ such that f classifies $\mathbf{x}'$ into the class $t_{x'}$, where $t_x \neq t_{x'}$, and $\|\mathbf{e}\|_0 \leq 1$.*

Intuitively, a one-pixel attack is an input image that differs from the original input by only one pixel.

Definition 5 ((General) One-pixel attack problem). *Given a neural network f and its input $\mathbf{x}$, the one-pixel attack problem studied in this paper aims to identify a one-pixel attack $\mathbf{x}'$ with regard to f and $\mathbf{x}$. In this problem, the solution one-pixel attack $\mathbf{x}'$ can be represented by the corresponding perturbation $\mathbf{e}$ such that $\mathbf{x}' = \mathbf{x} + \mathbf{e}$, and $\mathbf{e}$ can be described simply by the index and value of the nonzero element in $\mathbf{e}$.*

Definition 6 (Targeted/Untargeted one-pixel attack problem). *Given a neural network f , its input $\mathbf{x}$, and a target class t , the targeted one-pixel attack problem aims to identify a one-pixel attack $\mathbf{x}'$ such that f classifies $\mathbf{x}'$ into the given class t . The cases where there is no specific target class are referred to as the untargeted one-pixel attack problem.*

Since the one-pixel attack problem generally has a set of solutions, we might desire several objectives for these solutions. As a result, we define more specific one-pixel attack problems.

Definition 7 (Confidence score-based one-pixel attack problem). *Given a neural network f , its input $\mathbf{x}$ and a target class t , the confidence score-based one-pixel attack problem is to find a one-pixel attack $\mathbf{x}'$ that maximizes the confidence score for the target class t among one-pixel attacks succeeded.*

Definition 8 (Distance-based one-pixel attack problem). *Given a neural network f and its input $\mathbf{x}$, the distance-based one-pixel attack problem is to identify a one-pixel attack $\mathbf{x}'$ that minimizes the distance $\|\mathbf{x} - \mathbf{x}'\|$ among one-pixel attacks succeeded.*

Figure 1 illustrates an example of a one-pixel attack on a well-trained network using the CIFAR dataset: (a) depicts the original input classified as a *horse* with a confidence score of 0.69, (b) presents a one-pixel attack classified incorrectly as an *airplane* with a confidence score of 0.81 and a distance of 109, (c) displays a confidence score-based one-pixel attack with a confidence score of 0.90 and a distance of 217, and (d) demonstrates a distance-based one-pixel attack with a confidence score of 0.51 and a distance of 4.

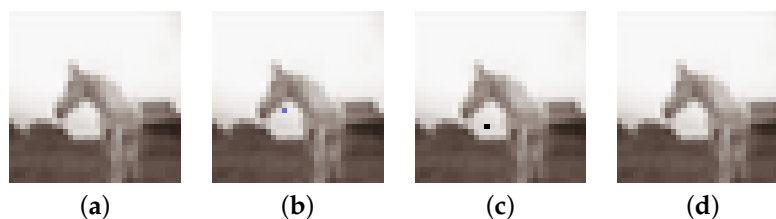


Figure 1. Examples of one-pixel attacks. (a) Original input; (b) one-pixel attack; (c) confidence score-based one-pixel attack; (d) distance-based one-pixel attack.

4. Candidate Pixel Map and Confidence Score

Given a network f and its input $\mathbf{x}$, the solution to the one-pixel attack problem is a one-pixel different image $\mathbf{x}'$ such that f classifies $\mathbf{x}'$ into a different class from the class of $\mathbf{x}$. The solution $\mathbf{x}' = \mathbf{x} + \mathbf{e}$ can be represented by the corresponding perturbation $\mathbf{e}$, and $\mathbf{e}$ can be simply described by the index and value of the nonzero element in $\mathbf{e}$; finally, the solution can be represented as (v, h, r, g, b) , where v and h are the vertical and horizontal coordinate values of the perturbed pixel, and r, g, b are its RGB color values, respectively. To identify such a solution, we propose two methods, namely, a candidate pixel map for determining v and h , and a gradient ascent with momentum method [1] for identifying r, g, b , respectively.

In this section, we present a preliminary experiment that serves as the foundation for the technique we propose, and explain our observations based on the experimental results.

4.1. Candidate Pixel Map

As the first work on the one-pixel attack problem, Su et al. [5] utilized the *differential evolution algorithm* to identify each element of (v, h, r, g, b) for the solution of the problem. However, OPA2D [6], which uses the *sensitivity* of each pixel in the input image to identify v and h , can identify higher-quality solutions than [5], and do so even faster. In this section, we propose a candidate pixel map that is more direct to the aim of the one-pixel attack.

Given an input image, some pixels in the image affect the class decision of the network for the given input more than the other pixels do. Such a sensitive pixel [6] has a property; that is, compared to a less sensitive pixel, when the RGB color value of the sensitive pixel change, the confidence score of the neural network also fluctuates significantly. Since a sensitive pixel is more vulnerable to the attack, an attack algorithm can find a one-pixel attack solution earlier by inspecting sensitive pixels first. The sensitivity of a certain pixel is defined by the change in the network's confidence score when the RGB values of that pixel are maximally altered. Intuitively, the sensitivity map can approximately represent the degree to which each pixel affects the class decision.

Given a network f and an input image $\mathbf{x}$ (with m rows, n columns and its class $t_{\mathbf{x}}$), the sensitivity map $M \in \mathcal{R}^{m \times n}$ with regard to f and $\mathbf{x}$ is computed as follows. For each pixel $x_i \in \mathbf{x}$ ($\forall i \in \{1, \dots, m \times n\}$), let the RGB color of x_i be $(r_{x_i}, g_{x_i}, b_{x_i})$, and let $\mathbf{x}'_i$ denote the one-pixel different image of $\mathbf{x}$, which has the new RGB color $(r'_{x_i}, g'_{x_i}, b'_{x_i})$ at the pixel x_i but has the same RGB color value at the remaining pixels as the original input $\mathbf{x}$. For each pixel $x_i \in \mathbf{x}$ ($\forall i \in \{1, \dots, m \times n\}$), the new RGB color $(r'_{x_i}, g'_{x_i}, b'_{x_i})$ at the pixel x_i is transformed from the RGB color $(r_{x_i}, g_{x_i}, b_{x_i})$ such that we assign 0 to $r'_{x_i}, g'_{x_i}, b'_{x_i}$ if $r_{x_i}, g_{x_i}, b_{x_i}$ are greater than or equal to 128, respectively, and we assign 255 to them if $r_{x_i}, g_{x_i}, b_{x_i}$ are less than 128, respectively. Now, we have $m \times n$ one-pixel different images $\{\mathbf{x}'_1, \dots, \mathbf{x}'_{m \times n}\}$. For each $i \in \{1, \dots, m \times n\}$, we feed the image $\mathbf{x}'_i$ to the given network f and obtain the difference of the confidence score $d_{x_i} = ||f_{t_{\mathbf{x}}}(\mathbf{x}) - f_{t_{\mathbf{x}}}(\mathbf{x}'_i)||$ where $f_{t_{\mathbf{x}}}(\cdot)$ is the confidence score for the original predicted class $t_{\mathbf{x}}$. Finally, each pixel in a sensitivity map corresponds to the difference d_{x_i} for the confidence score, where i is the index for the corresponding vertical and horizontal coordinates.

The basic idea of the sensitivity map is to select pixels that decrease the confidence score for the original class $t_{\mathbf{x}}$ more. However, the precise aim of the one-pixel attack problem is to identify pixels that increase the confidence score of a given target class rather than ones that decrease the confidence score of the original class, even though they are similar in many cases. Therefore, we introduce a *candidate pixel map* which represents how much each pixel can increase the confidence score when it is maximally altered. Formally, given a network f , an input image $\mathbf{x}$ and a target class t , the candidate pixel map $C \in \mathcal{R}^{m \times n}$ is computed as follows. We first compute $m \times n$ one-pixel different images $\{\mathbf{x}'_1, \dots, \mathbf{x}'_{m \times n}\}$ like the sensitivity map. Then, we feed each image $\mathbf{x}'_i$ to the given network f and obtain the confidence score for the target class $f_t(\mathbf{x}'_i)$ rather than the difference of the confidence score $||f_{t_{\mathbf{x}}}(\mathbf{x}) - f_{t_{\mathbf{x}}}(\mathbf{x}'_i)||$. After all, if the value of each pixel in the candidate pixel map is large, the pixel is likely to increase the confidence score for the target class, so it is better for the one-pixel attack algorithm to inspect these pixels first.

Figure 2 presents candidate pixel maps; (a) depicts the original input classified as a horse, a deer, and a cat, respectively, and (b) presents their corresponding candidate pixel maps where the target classes are a cat, a bird, and a dog, respectively. The pixels in the dark area in Figure 2b are more likely to cause the neural network to mistake the image for the target class when the pixel's color changes since they contribute more to increasing the score of the target class compared to other pixels.

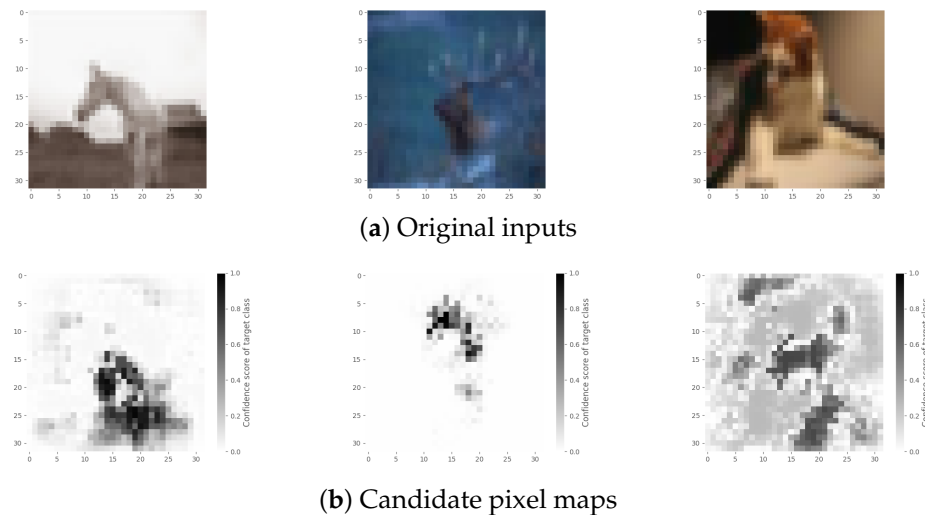


Figure 2. Candidate pixel maps.

4.2. Confidence Score

To identify the optimal solution to the problem, OPA2D [6] first computes the sensitivity of each pixel, and then it applies the differential evolution (DE) algorithm in order of the most sensitive pixels to find the RGB color value as the solution. At each generation g , the DE algorithm produces the population for the next generation using the following formula:

$$x_i(g+1) = x_{r_1}(g) + F \cdot (x_{r_2}(g) - x_{r_3}(g)) \quad (1)$$

where x_i denotes the i -th candidate solution in the population, r_1, r_2, r_3 are distinct random numbers, F is the differential weight parameter, and g is the current generation's index. To evolve the next candidates, the DE algorithm chooses random candidates from the current generation and improves them by mutation, crossover and selection. Since DE does not utilize gradients, it can sometimes easily deviate from local optima. However, due to such randomness, it may take a long time to find good solutions. In general, the DE algorithm may efficiently find the optimal solution in the domains where there are a number of local optima, such as Ackley functions [25] (Figure 3 presents a 2D Ackley function). Otherwise, the DE may waste time in identifying the optimal solution.

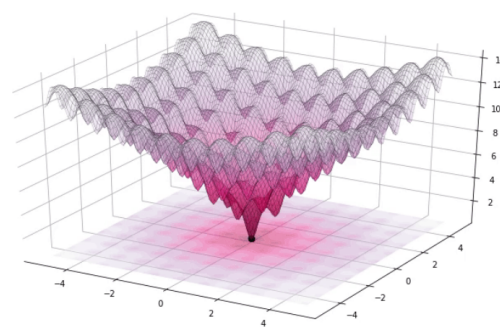


Figure 3. A 2D Ackley function.

Hence, for this question, we investigate the confidence scores of well-trained neural networks (i.e., ResNet [26], LeNet [27], PureCNN [28], NiN [29], DenseNet [30], and WideResNet [31]) using the CIFAR-10 dataset. For a pre-selected target class for which a given network can be attacked, we first compute the candidate pixel map of a randomly selected input image. Then, for top ranked pixels, we measure the confidence scores for the target class by changing the RGB value of the pixel. The results of this experiment should be presented as a single 4D plot, but for the sake of visibility, 27 3D plots are presented with a particular color value fixed. As a representative instance among a number of experimental

results, Figure 4 shows the confidence score of ResNet for input 384 (*bird*) of CIFAR-10. Figure 4a presents how the confidence score for the target class *dog* changes when for the pixel (17, 14) of the input 384, the blue value is fixed at 0 and the red and green values are changed. Additionally, Figure 4b–aa present the confidence scores by fixing the blue values at 10 to 255, respectively. Figure 4ab depicts the original input image. These plots demonstrate that, unlike the Ackley function, there are only a few shallow local optima for the confidence scores. That is, when the blue value is 160 (see Figure 4q), the number of local optima is less than 10, which is the case with the most local optima among them. Even in this case, a simple optimization method such as the gradient ascent with momentum can more efficiently identify optimal solutions than DE.



Figure 4. Change in confidence score of target class.

5. Rapid Imperceptible Strong One-Pixel Attack Algorithm

Even if the domain of each color value is limited to integer values $\{0, \dots, 255\}$ rather than real values, the search space of the one-pixel attack problem is $32 \times 32 \times 256 \times 256 \times 256 = 2^{34}$ in the case of CIFAR-10 data (the image size of 32×32). To handle the large search space, our method tackles the one-pixel attack problem by a two-phase algorithm; that is, in the first phase, we compute a candidate pixel map for a given input image, and then in the second phase, we identify an RGB color value for the top ranked pixels in the map by using a gradient ascent with momentum method [1].

In the first phase, by computing the candidate pixel map, we calculate approximately a possibility of the one-pixel attack for each pixel x_i in the input image $\mathbf{x}$, which is the confidence score $f_t(\mathbf{x}'_i)$ for the target class t , where $\mathbf{x}'_i$ is the one-pixel different image constructed by maximally altering the RGB color value of x_i . OPA2D [6] selects the top 30 sensitive pixels and tries them in a sensitivity order to identify their RGB color value by using the differential evolution method. On the other hand, our algorithm picks the top 30 pixels in the candidate pixel map that is more precise for the goal of the one-pixel attacks since the top ranked pixels in our candidate pixel map increase the confidence score for the target class.

Once in the first phase, we compute a candidate pixel map and decide the set of top ranked pixels; we check whether we can succeed the one-pixel attack with each pixel in the set by using a gradient ascent with momentum method. Let us assume that the selected pixel has originally an RGB value $\mathbf{p}_0 = (r_0, g_0, b_0)$. Now, our algorithm needs to change the RGB value to succeed the one-pixel attack by increasing the confidence score of a given model with regard to a given target class. For an untargeted attack, we try the targeted attack for each class iteratively. From now on, we only describe targeted attacks. Remark that in the confidence score-based one-pixel attack problem, our goal is to identify an RGB value to maximize the confidence score for the target class, but in the distance-based one-pixel attack problem, our goal is to find the RGB value that is closest to the original RGB value among the RGB values that succeed in the attacks. Our algorithm tries to identify such an optimal solution by a gradient ascent with momentum method, which improves the current solution by the following equations:

$$\mathcal{G}_t = \gamma \mathcal{G}_{t-1} + \eta \nabla f_t(\mathbf{x}'_{t-1}) \quad (2)$$

$$\mathbf{p}_t = \mathbf{p}_{t-1} + \mathcal{G}_t \quad (3)$$

where $\mathcal{G}_t$ is a momentum-added gradient, γ is a momentum decay factor, η is a learning rate, and $\mathbf{x}'_{t-1}$ is the one-pixel different image at the previous iteration. That is, in each iteration, the current RGB value $\mathbf{p}_t$ of the corresponding pixel is obtained by adding the previous pixel value $\mathbf{p}_{t-1}$ to the momentum-added gradient $\mathcal{G}_t$. The momentum-added gradient $\mathcal{G}_t$ is computed by adding the previous momentum-added gradient $\gamma \mathcal{G}_{t-1}$ to the gradient of the model's confidence score $\eta \nabla f_{t_x}(\mathbf{x}'_{t-1})$.

Algorithm 1 presents our RISOPA (rapid imperceptible strong one-pixel attack) algorithm for CIFAR-10 data, which is for the targeted attack. Given a neural network f , an input image $\mathbf{x}$, and a target class t , and parameters n_c , n_i , γ , and η , it identifies a perturbation $\mathbf{e} = (v, h, r, g, b)$ as the solution for the one-pixel attack problem, where the parameter n_c is the number of candidate pixels we are considering, n_i is the number of iterations for the gradient ascent method, γ is a momentum decay factor, and η is a learning rate. In the first phase (Lines 1–10), it generates the candidate pixel map and selects the top n_c pixels according to the confidence score for the given target class t when the RGB value of each pixel is maximally changed. That is, for each pixel $x_i \in \mathbf{x}$, the corresponding one-pixel different image $\mathbf{x}'_i$ is generated by copying from the original input image $\mathbf{x}$ in Line 2 and then modifying the pixel color value $\mathbf{x}'_i[i]$ to the new values (r', g', b') in Lines 4–7. After that, for the corresponding one-pixel different image $\mathbf{x}'_i$, we compute the confidence score for the target class t in Line 8. Once we compute the confidence score for each one-pixel

different image, we select the top n_c pixels according to their score, and store their indices to C in Line 10.

Algorithm 1: RISOPA (rapid imperceptible strong one-pixel attack) algorithm for CIFAR-10 data

Input : a neural network f , an input image $\mathbf{x}$, a target class t and parameters n_c , n_i , γ , and η .

Output: a perturbation $\mathbf{e} = (v, h, r, g, b)$.

```

1 for  $i = 1$  to  $32 \times 32$  do
2    $\mathbf{x}'_i = \mathbf{x}$ ;
3   // Let  $(r_i, g_i, b_i)$  be the RGB color value of each pixel  $x_i$  in the original input  $\mathbf{x}$ .
4   if  $(r_i \geq 128)$  then  $r'_i = 0$ ; else  $r'_i = 255$ ;
5   if  $(g_i \geq 128)$  then  $g'_i = 0$ ; else  $g'_i = 255$ ;
6   if  $(b_i \geq 128)$  then  $b'_i = 0$ ; else  $b'_i = 255$ ;
7    $\mathbf{x}'_i[i] = (r'_i, g'_i, b'_i)$ ;
8    $s[i] = f_t(\mathbf{x}'_i)$ ;
9 end for
10  $C = \text{pickTopRanks}(s, n_c)$ ;
11 for  $i = 1$  to  $n_c$  do
12    $\text{index} = C[i]$ ;
13   // The initial pixel value  $(\mathbf{p}_0^r, \mathbf{p}_0^g, \mathbf{p}_0^b)$  is given according to the attack objective.
14    $\mathbf{p}_0 = (\mathbf{p}_0^r, \mathbf{p}_0^g, \mathbf{p}_0^b)$ ;
15    $\mathcal{G}_0 = (0, 0, 0)$ ;
16    $\text{opt} = 0$ ;
17   for  $j = 1$  to  $n_i$  do
18      $\mathbf{x}'_j = \mathbf{x}$ ;
19      $\mathbf{x}'_j[\text{index}] = \mathbf{p}_{j-1}$ ;
20      $\text{score} = f(\mathbf{x}'_j)$ ;
21     if  $(\text{update}(\text{opt}, \text{score}, \mathbf{x}'_j))$  then
22        $\text{opt} = \text{score}[t]$ ;
23        $\mathbf{e} = ((\text{index}/32) + 1, \text{index} \bmod 32, \mathbf{p}_{j-1}^r, \mathbf{p}_{j-1}^g, \mathbf{p}_{j-1}^b)$ ;
24     end if
25     if  $((\text{result} = \text{terminate}()) == \text{"terminate"})$  then return  $\mathbf{e}$ ;
26     else if  $(\text{result} == \text{"break"})$  then break;
27      $\mathcal{G}_j = \gamma \mathcal{G}_{j-1} + \eta \nabla f_t(\mathbf{x}'_j)$ ;
28      $\mathbf{p}_j = \mathbf{p}_{j-1} + \mathcal{G}_j$ ;
29   end for
30 end for
31 return  $\mathbf{e}$ ;
```

Once the candidate pixels are determined in the first phase, our algorithm tries to attack each pixel in the candidate set C from the top ranked pixel to n_c -th pixel in Lines 11–30. In Line 14, it sets the initial RGB color value $\mathbf{p}_0$ for the pixel being attacked according to the attack objective. Remark that our one-pixel attack method supports three distinct objectives; namely, the rapid attack (RA) aims to identify a successful one-pixel attack as soon as possible, the imperceptible attack (IA) finds the one-pixel attack that is closest to a given input image, and the goal of the strong attack (SA) is to identify the highest confidence score among successful attacks. While for RA and SA, we initialize $\mathbf{p}_0$ to the RGB value maximally changed from the original RGB color value, we initialize $\mathbf{p}_0$ to the original RGB color value for IA. Additionally, the initial momentum-added gradient $\mathcal{G}_0$ is set to $(0, 0, 0)$, and the current optimal confidence score opt is set to 0 in Lines 15–16. Then, by the gradient ascent with momentum method, it repeats to improve the confidence score

for the target class until it reaches the given number of iterations n_i . In Lines 18–20, our algorithm generates the current one-pixel different image $\mathbf{x}'_j$ by updating the pixel color at *index* and computes the confidence score for the target class of $\mathbf{x}'_j$. Then, in Line 21, the algorithm determines whether to store the current perturbation as the best value observed so far by using $\text{update}(\text{opt}, \text{score}, \mathbf{x}'_j)$. The function $\text{update}()$ returns *true* according to the attack objective as follows. For RA, if the current confidence score *score* corresponds to an attack success (i.e., the score of the target class is greater than other classes), it returns *true*. For SA, the function returns *true* if the attack succeeds and $\text{score}[t]$ is greater than the current best score *opt*. For IA, the function $\text{update}()$ returns *true* if the attack succeeds and $\mathbf{x}'_j$ is closer to the original image than the current best. In Lines 22–23 (when the function $\text{update}()$ in Line 21 returns *true*), the algorithm stores the current score to *opt*, and the corresponding index and pixel RGB values to *e*, respectively. In Line 25, it checks a termination/break condition according to the kind of the current attack, and if the corresponding condition holds, the algorithm terminates with the current perturbation *e* or breaks the current loop. The termination/break condition according to the current attack objective we use in this algorithm is as follows:

- RA: According to Definition 5 (the general one-pixel attack), as long as a given neural network classifies the one-pixel different image $\mathbf{x}'_j$ into a different class from the original input *x*, the current attack succeeds and the function $\text{terminate}()$ returns “*terminate*”.
- SA: In the strong attack that corresponds to Definition 7 (the confidence score-based one-pixel attack), our goal is to identify the successful one-pixel attack $\mathbf{x}'_j$ that improves the confidence score for the target class the most. Since our algorithm searches the pixel color value in the direction to increase the confidence score for the target class by the gradient ascent method, if we stop when a change in the confidence score is close to 0, the solution is the best one we can find. Therefore, the function $\text{terminate}()$ returns “*break*” if the current change in the confidence score is less than a given threshold ϵ because we may find a stronger attack in other candidate pixels.
- IA: In the imperceptible attack which corresponds to Definition 8 (the distance-based one-pixel attack), our goal is to identify the successful one-pixel attack $\mathbf{x}'_j$ that is the closest to the original input image *x*. Since in this attack we start the pixel value for $\mathbf{p}_0$ from the original pixel color, we stop searching as soon as we identify a one-pixel attack. However, because we may find a more imperceptible attack in other candidate pixels, the function $\text{terminate}()$ returns “*break*” rather than “*terminate*”, and the algorithm continues searching a better solution from the next candidate.
- In all attack objectives (i.e., RA, IA, and SA), the function $\text{terminate}()$ returns “*break*” if the current change in the confidence score is less than a given threshold ϵ since in this case, the gradient ascent method cannot improve the solution anymore with the current candidate pixel.

If the termination/break condition does not hold, the algorithm updates the current momentum-added gradient $\mathcal{G}_j$ and the current RGB color $\mathbf{p}_j$ by Equations (2) and (3), respectively (Lines 27–28). That is, the new $\mathcal{G}_j$ is obtained by adding the previous momentum-added gradient $\mathcal{G}_{j-1}$ multiplied by the momentum decay factor γ and the gradient $\nabla f_t(\mathbf{x}'_{t-1})$ of the confidence score multiplied by the learning rate η . The pixel RGB value $\mathbf{p}_j$ is updated by adding the previous pixel RGB value $\mathbf{p}_{j-1}$ and the momentum-based gradient $\mathcal{G}_j$ computed above. Finally, our algorithm terminates with the current optimal perturbation *e* after all the candidates are investigated, unless any inspection for the termination condition holds.

6. Experiment

In this section, we demonstrate the effectiveness of our method to identify one-pixel attacks for well-trained popular neural networks with CIFAR-10 data. We implemented a tool for our RISOPA (rapid imperceptible strong one-pixel attack) algorithm described in Section 5, and experimented with it on the following networks: ResNet [26], LeNet [27],

PureCNN [28], NiN [29], DenseNet [30], and WideResNet [31]. All the experiments were performed on a PC using a 3.10 GHz 16 core Xeon processor, 256 GB memory, and a 4090 GPU. To analyze the performance of our RISOPA algorithm, we pre-executed each neural network with the 10,000 CIFAR-10 testing data, and used randomly selected 100 images that each target neural network correctly classified.

Table 1 shows the neural network architectures used in our experiments. The architectures considered in our experiments contain up to 11.3M parameters and 354 layers. Their accuracies for CIFAR-10 data are 74.88% to 95.34%.

Table 1. Neural network architectures.

Model	Num. of Parameters	Num. of Layers	Accuracy (CIFAR-10)
ResNet [26]	470,218	113	92.31%
LeNet [27]	62,006	8	74.88%
PureCNN [28]	1,369,738	16	88.77%
NiN [29]	972,658	33	90.74%
DenseNet [30]	850,606	354	94.67%
WideResNet [31]	11,318,026	54	95.34%

6.1. The Size of Candidate Pixel Set

Since our algorithm selects a candidate pixel set and then tries to attack each pixel in the set, the size of the candidate pixel set is an important parameter for the performance of our algorithm. For the same reason, the OPA2D method picks the top 30 pixels according to their sensitivities. To decide the size of the candidate pixel set, we run our tool for the untargeted attack with 100 randomly selected input images. In this preliminary experiment, we measure how much the average success rate increases as the size of the candidate pixel set changes.

Figure 5 shows the results of this experiment; Figure 5a–f present the change of the average success rate for ResNet, LeNet, PureCNN, NiN, DenseNet, and WideResNet, respectively. As the size of the candidate pixel set increases, the average success rate also increases in the first few top pixels ($n_c \leq 30$) and becomes saturated to the maximum value after that point. In the case of LeNet, the average success rate increases for the last time when the set size is around 80, but the increment is relatively small. Consequently, we conclude that it is a reasonable decision to try to identify a solution until $n_c = 30$.

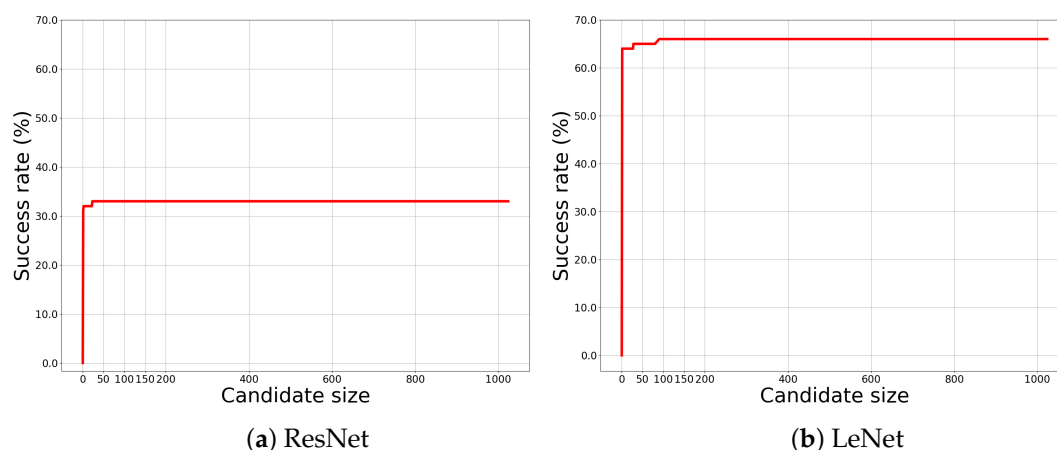


Figure 5. Cont.

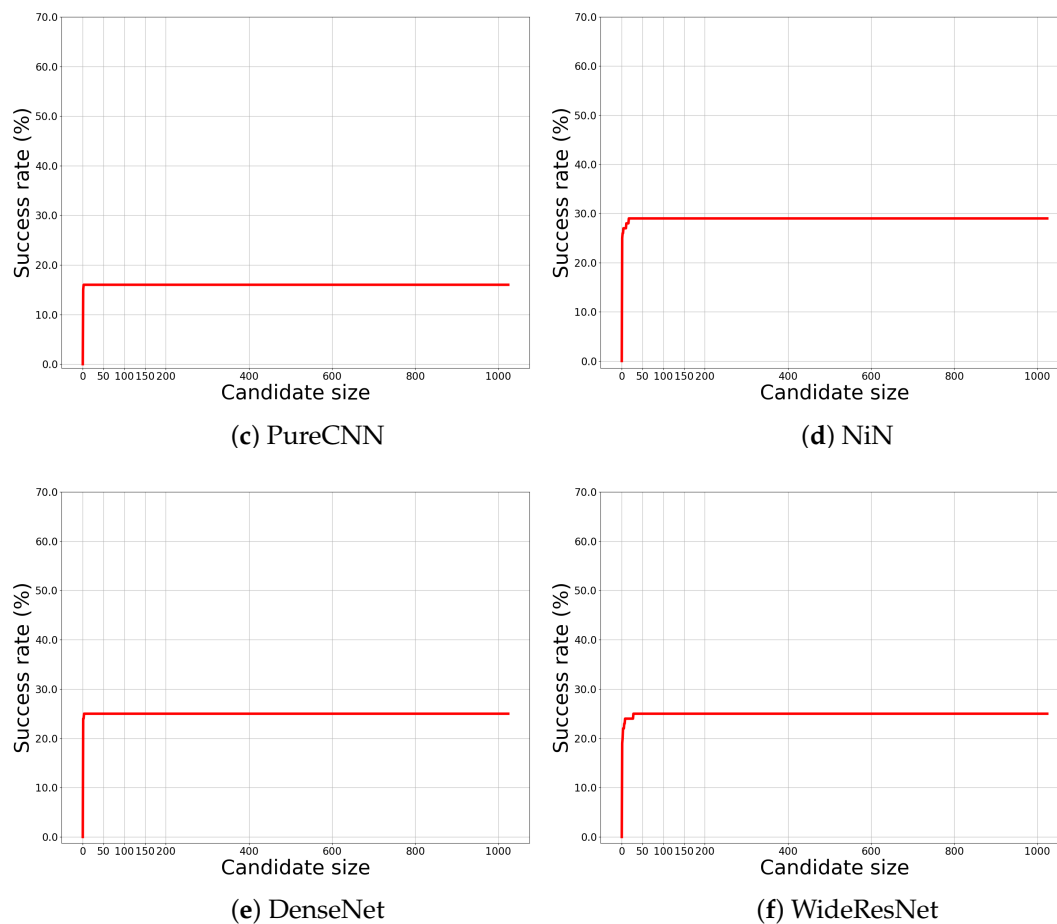


Figure 5. Experiment result for the size of candidate pixel set.

6.2. Performance Comparison

To demonstrate the efficiency of our method, we compare our tool with two existing methods for the one-pixel attack problem—e.g., Su et al.'s work [5] and OPA2D [6]. Note that the authors in [5] did not explicitly name their technique, so in this paper we refer to it as DEOPA, which means the differential evolution-based one-pixel attack method. Remark that while DEOPA aims to find the one-pixel attack with the highest confidence score for the target class, OPA2D tries to identify the one-pixel attack that is closest to the original input image. Therefore, for the imperceptible attack (IA) we compare our tool with OPA2D, and we compare ours with DEOPA for the strong attack (SA). Since both DEOPA and OPA2D do not support the rapid attack (RA), we do not directly compare them with our tool for RA.

Table 2 presents a performance comparison for the untargeted attack. For the parameters of DEOPA, we set the population size and the number of iterations as 400 and 100 as their paper, respectively. For OPA2D, we set the size of the candidate set, the population size, and the number of iterations as 30, 400 and 100 as their paper, respectively. Also, for our RISOPA, we set the size of the candidate set and the number of iteration as 30 and 50, respectively. For the sake of fairness, we only compare them with inputs for which all three methods successfully execute the corresponding attack. The table shows the running time and the quality of attacks for each attack type, and the success rate. For RA, because both DEOPA and OPA2D do not support this type of attack, we present only our running time. Since our rapid attack terminates once we discover any one-pixel attack, the running time is on average 14.3 times faster and 12.1 times faster than our IA and SA, respectively. Furthermore, it is on average 36.0 times more rapid than OPA2D and 57.4 times faster than DEOPA. For IA, since DEOPA does not support the imperceptible attack, we only compare

with OPA2D. Our IA identifies similar quality attacks on average 2.5 times faster than OPA2D. The difference in attack quality is under 3%. Next, because OPA2D identifies only distance-based attacks, we compare with DEOPA for SA. Our SA discovers very similar quality attacks 4.8 times faster than DEOPA, and the difference of attack quality is under 1%. All three methods achieve almost the same success rate.

Table 2. Untargeted attack performance.

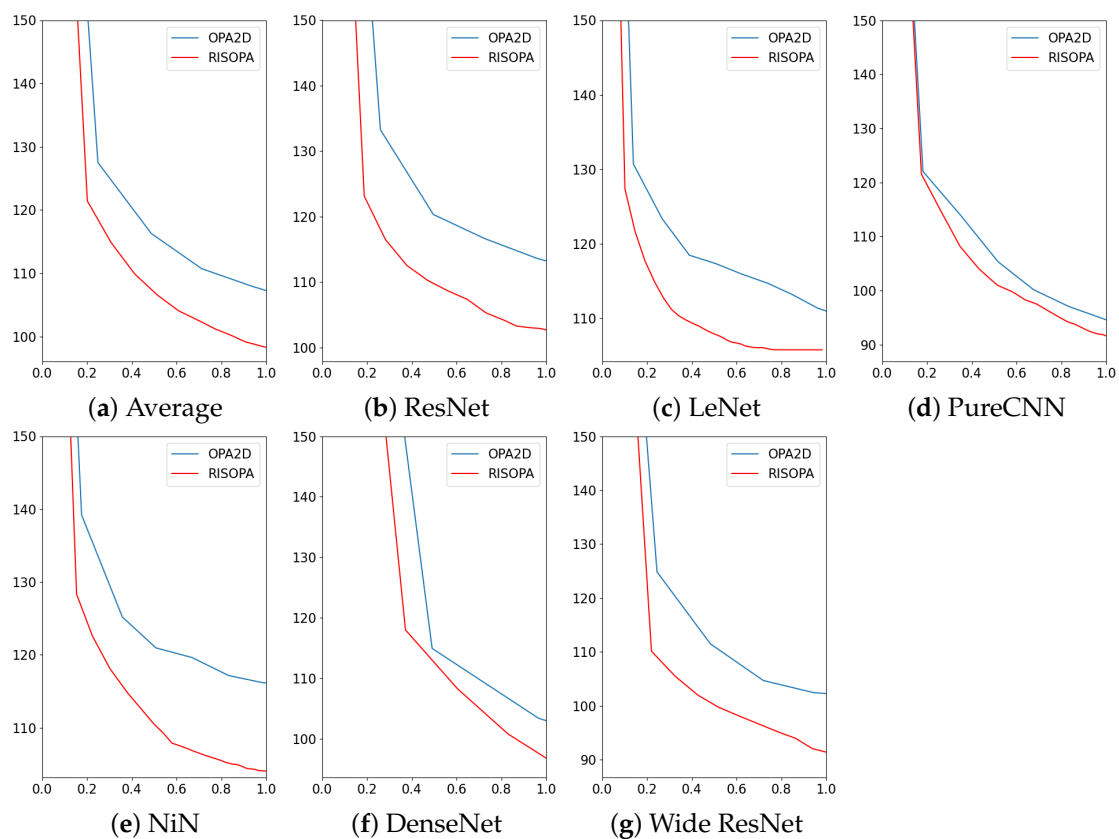
Model	Method	RA Time	IA Distance	Time	SA Score	SA Time	Success Rate (%)
ResNet	DEOPA	–	–	–	0.93	62.40	32.0
	OPA2D	–	98.0	42.47	–	–	32.0
	RISOPA	1.26	102.8	16.14	0.92	10.21	32.0
LeNet	DEOPA	–	–	–	0.79	37.11	64.0
	OPA2D	–	99.9	21.77	–	–	65.0
	RISOPA	0.53	104.3	8.36	0.79	6.13	65.0
PureCNN	DEOPA	–	–	–	0.84	49.62	20.0
	OPA2D	–	83.6	28.20	–	–	19.0
	RISOPA	0.63	83.9	11.20	0.84	8.94	19.0
NiN	DEOPA	–	–	–	0.86	53.35	30.0
	OPA2D	–	112.6	29.40	–	–	29.0
	RISOPA	0.74	113.2	11.31	0.84	9.29	29.0
DenseNet	DEOPA	–	–	–	0.92	112.75	25.0
	OPA2D	–	87.2	76.83	–	–	25.0
	RISOPA	2.25	90.4	31.02	0.92	27.94	25.0
Wide ResNet	DEOPA	–	–	–	0.89	63.93	25.0
	OPA2D	–	73.8	38.70	–	–	25.0
	RISOPA	1.28	75.6	17.43	0.87	17.14	26.0

Table 3 explains the performance comparison for the untargeted attack, where we select the same parameters with the untargeted attack experiment. Our rapid attack terminates on average 7.5 times faster and 4.7 times faster than our IA and SA, respectively. Moreover, it is on average 20.8 times faster than OPA2D and 22.6 times more rapid than DEOPA. Our IA method discovers similar quality attacks on average 2.8 times faster than OPA2D, and our SA discovers them 4.8 times faster than DEOPA, respectively. The difference in attack quality is similar with that of the untargeted attack. Finally, the three methods accomplish almost the same success rate.

Although existing methods have found solutions of similar quality to that of our tool, such quality can typically be achieved only with very generous time limits. For the imperceptible attack, Figure 6 presents an environment with strict deadlines, for example, in which one wants to identify a one-pixel attack within 1 s. Figure 6a illustrates the average distance of attacks found within a 1 s deadline by OPA2D and our tool. Our IA method identifies better attacks earlier than OPA2D; more specifically, our IA discovers attacks of quality that are 9.12% closer on average at the end of the deadline. Figure 6b–g show the result for each model, where the IA tool finds attacks of better quality than OPA2D.

Table 3. Targeted attack performance.

Model	Method	RA	IA		SA		Success Rate (%)
		Time	Distance	Time	Score	Time	
ResNet	DEOPA	–	–	–	0.88	6.92	6.2
	OPA2D	–	93.0	7.00	–	–	5.6
	RISOPA	0.24	97.5	2.37	0.87	1.11	6.1
LeNet	DEOPA	–	–	–	0.64	4.12	15.9
	OPA2D	–	102.2	3.3	–	–	14.4
	RISOPA	0.16	104.5	1.11	0.65	0.68	16.7
PureCNN	DEOPA	–	–	–	0.72	5.50	3.6
	OPA2D	–	84.2	4.73	–	–	3.0
	RISOPA	0.22	82.9	1.79	0.72	1.00	3.3
NiN	DEOPA	–	–	–	0.84	5.92	5.0
	OPA2D	–	102.7	4.50	–	–	4.6
	RISOPA	0.22	103.6	1.58	0.81	1.03	4.9
DenseNet	DEOPA	–	–	–	0.89	12.53	4.4
	OPA2D	–	78.2	12.60	–	–	4.0
	RISOPA	0.62	81.9	4.49	0.88	3.07	4.1
Wide ResNet	DEOPA	–	–	–	0.84	7.1	4.6
	OPA2D	–	74.5	6.53	–	–	4.4
	RISOPA	0.40	77.2	2.58	0.81	1.87	4.7

**Figure 6.** Experiment result for a strict deadline (1 s).

6.3. Discussion

In this subsection, we present a discussion comparing the results of our RISOPA with the existing methods (DEOPA [5] and OPA2D [6]) as a summary. Because our RISOPA

supports three kinds of attacks (i.e., the rapid attack (RA), the imperceptible attack (IA), and the strong attack (SA)), we explain the performance difference from the perspective of these three attacks. First, for RA, since both of DEOPA and OPA2D do not support RA, we cannot directly compare our RISOPA with them for RA. However, to understand its advantages, we can simply compare its running time with our IA and SA, DEOPA, and OPA2D, even though they keep searching until identifying an optimal solution. Since RA terminates its search as soon as it discovers any successful one-pixel attack, the running time is 10.9 times faster than our IA and 8.4 times faster than our SA. Moreover, our RA terminates 28.4 times faster than OPA2D and 40.0 times more rapidly than DEOPA. Hence, if one prioritizes finding answers quickly over the quality of the answers, using this method would enable us to find solutions much faster than both methods DEOPA and OPA2D.

For IA, we compare our IA method with OPA2D. Although they both employ similar map approaches to identify vertical and horizontal coordinate values of the perturbed pixels, our candidate pixel map is more direct to the aim of one-pixel attacks. Moreover, to discover RGB values, while OPA2D exploits the differential evolution approach (DE), our IA uses the gradient ascent with momentum method. If sufficient iterations are allowed, the DE method will converge to an optimal solution, but the execution time will be significantly prolonged. Our experiment shows that our IA identifies similar quality attacks 2.7 times faster than OPA2D.

For SA, we compare our SA method with DEOPA. While OPA2D employs the sensitivity map and the DE approach, DEOPA utilizes the DE approach to find vertical and horizontal coordinate values as well as their RGB values. Similar to OPA2D, DEOPA will also converge to an optimal solution if sufficient iterations are allowed. However, its execution time is significantly slower than our SA approach. Our experiment presents that our SA identifies similar quality attacks 4.8 times faster than DEOPA.

7. Conclusions

In this study, we have proposed a novel technique *RISOPA* for one-pixel attacks in deep neural networks. Our algorithm selects a set of promising pixels by constructing a candidate pixel map, and then identifies the RGB color value for the pixels by a gradient ascent with momentum method. Moreover, we have implemented an efficient tool for the algorithm, and the experiment has yielded promising results that our technique outperforms the existing methods with regard to three objectives—i.e., rapid attacks, imperceptible attacks and stronger attacks.

There are several noteworthy issues for future study. First, although our technique has been applied to the CIFAR-10 dataset, various datasets need to be investigated for one-pixel attacks; i.e., high-resolution images such as ImageNet [32], many-class datasets like CIFAR-100 [8], and medical data like the tumor dataset [33]. In addition, to explore our algorithm's scalability and its effectiveness, we plan to apply our methodology to a broader range of neural network architectures. Second, we wish to study an efficient method to adapt our technique to more state-of-the-art models like transformer models and contrastive learning. Third, we plan to conduct ample experiment to improve the accuracy and gain of the proposed technique. Lastly, conducting experiments with recent adversarial defense mechanisms would be valuable research to gain a clearer understanding of the performance of our algorithm.

Author Contributions: Conceptualization, W.N. and H.K.; methodology, W.N. and H.K.; software, K.K., H.M., H.N. and J.P.; validation, K.K., H.M., H.N. and J.P.; formal analysis, W.N. and H.K.; investigation, W.N. and H.K.; resources, W.N. and H.K.; data curation, K.K., H.M., H.N. and J.P.; writing—original draft preparation, W.N. and H.K.; writing—review and editing, W.N. and H.K.; visualization, W.N.; supervision, W.N. and H.K.; project administration, W.N. and H.K.; funding acquisition, W.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI ²	Abstract Interpretation for Artificial Intelligence
CIFAR-10	Canadian Institute for Advanced Research, 10 classes
CNN	Convolution Neural Networks
DE	Differential Evolution
DEOPA	Differential Evolution-based One-Pixel Attack
DNN	Deep Neural Network
IA	Imperceptible Attack
MNIST	Modified National Institute of Standards and Technology
OPA2D	One-Pixel Attack, Detection, and Defense
RA	Rapid Attack
RISOPA	Rapid Imperceptible Strong One-Pixel Attack
SA	Strong Attack
UAP	Universal Adversarial Perturbations

References

- Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
- Miikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzyan, A.; Duffy, N.; et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*; Elsevier: Amsterdam, The Netherlands, 2024; pp. 269–287.
- Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the 2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
- Huang, X.; Kroening, D.; Ruan, W.; Sharp, J.; Sun, Y.; Thamo, E.; Wu, M.; Yi, X. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Comput. Sci. Rev.* **2020**, *37*, 100270. [\[CrossRef\]](#)
- Su, J.; Vargas, D.V.; Sakurai, K. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841. [\[CrossRef\]](#)
- Nguyen-Son, H.; Thao, T.P.; Hidano, S.; Bracamonte, V.; Kiyomoto, S.; Yamaguchi, R.S. OPA2D: One-Pixel Attack, Detection, and Defense in Deep Neural Networks. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), Virtual, 18–22 July 2021; pp. 1–10.
- Storn, R.; Price, K.V. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
- Krizhevsky, A. CIFAR Data Set. 2009. Available online: <https://www.cs.toronto.edu/~kriz/cifar.html> (accessed on 1 May 2023).
- Mingxing, D.; Li, K.; Xie, L.; Tian, Q.; Xiao, B. Towards multiple black-boxes attack via adversarial example generation network. In Proceedings of the 29th ACM International Conference on Multimedia, Virtual, 20–24 October 2021; pp. 264–272.
- Suya, F.; Chi, J.; Evans, D.; Tian, Y. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Berkeley, CA, USA, 12–14 August 2020; pp. 1327–1344.
- Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [\[CrossRef\]](#) [\[PubMed\]](#)
- Xie, C.; Tan, M.; Gong, B.; Wang, J.; Yuille, A.L.; Le, Q.V. Adversarial Examples Improve Image Recognition. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 816–825.
- Ho, C.; Vasconcelos, N. Contrastive Learning with Adversarial Examples. In Proceedings of the Annual Conference on Neural Information Processing Systems 2020, (NeurIPS), Online, 6–12 December 2020; pp. 17081–17093.
- Zhang, C.; Benz, P.; Imtiaz, T.; Kweon, I.S. Understanding Adversarial Examples From the Mutual Influence of Images and Perturbations. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 14509–14518.
- Hendrycks, D.; Zhao, K.; Basart, S.; Steinhardt, J.; Song, D. Natural Adversarial Examples. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 15262–15271.
- Luo, W.; Zhang, H.; Kong, L.; Chen, Z.; Tang, K. Defending Adversarial Examples by Negative Correlation Ensemble. In Proceedings of the International Conference on Data Mining and Big Data, Beijing, China, 21–24 November 2022; pp. 424–438.

17. Korpikalkola, J.; Sipola, T.; Kokkonen, T. Color-Optimized One-Pixel Attack Against Digital Pathology Images. In Proceedings of the 29th Conference of Open Innovations Association, (FRUCT), Tampere, Finland, 12–14 May 2021; pp. 206–213.
18. Nam, W.; Kil, H. AESOP: Adjustable Exhaustive Search for One-Pixel Attacks in Deep Neural Networks. *Appl. Sci.* **2023**, *13*, 5092. [CrossRef]
19. Pulina, L.; Tacchella, A. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In Proceedings of the 22nd International Conference of Computer Aided Verification (CAV), Edinburgh, UK, 15–19 July 2010; pp. 243–257.
20. Guidotti, D.; Pulina, L.; Tacchella, A. pyNeVer: A Framework for Learning and Verification of Neural Networks. In Proceedings of the 19th International Symposium on Automated Technology for Verification and Analysis (ATVA), Gold Coast, Australia, 18–22 October 2021; pp. 357–363.
21. Gehr, T.; Mirman, M.; Drachler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; Vechev, M.T. AI²: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–23 May 2018; pp. 3–18.
22. Müller, M.N.; Makarchuk, G.; Singh, G.; Püschel, M.; Vechev, M.T. PRIMA: general and precise neural network certification via scalable convex hull approximations. *Proc. ACM Program. Lang.* **2022**, *6*, 1–33. [CrossRef]
23. Tran, H.; Bak, S.; Xiang, W.; Johnson, T.T. Verification of Deep Convolutional Neural Networks Using ImageStars. In Proceedings of the 32nd International Conference of Computer Aided Verification (CAV), Los Angeles, CA, USA, 21–24 July 2020; pp. 18–42.
24. Pham, L.H.; Sun, J. Verifying Neural Networks Against Backdoor Attacks. In Proceedings of the 34th International Conference on Computer Aided Verification, (CAV), Haifa, Israel, 7–10 August 2022; pp. 171–192.
25. Ackley, D.H. *A Connectionist Machine for Genetic Hillclimbing*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1987.
26. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
27. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
28. Kausar, A.; Sharif, M.; Park, J.; Shin, D.R. Pure-CNN: A Framework for Fruit Images Classification. In Proceedings of the 2018 International Conference on Computational Science and Computational Intelligence, Las Vegas, NV, USA, 13–15 December 2018; pp. 404–408.
29. Lin, M.; Chen, Q.; Yan, S. Network In Network. In Proceedings of the 2nd International Conference on Learning Representations, (ICLR), Banff, AB, Canada, 14–16 April 2014.
30. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 (IEEE) Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
31. Zagoruyko, S.; Komodakis, N. Wide Residual Networks. In Proceedings of the British Machine Vision Conference, York, UK, 19–22 September 2016.
32. Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255.
33. Kaggle. Brain Tumor Data Set. 2020. Available online: <https://www.kaggle.com/datasets/jakeshbohaju/brain-tumor> (accessed on 1 September 2023).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.