applied sciences

MDPI

Article

You Only Attack Once: Single-Step DeepFool Algorithm

Jun Li *,†, Yanwei Xu †, Yaocun Hu, Yongyong Ma and Xin Yin

School of Management Science and Information Engineering, Jilin University of Finance and Economics, Changchun 130117, China; xuyanwei@s.jlufe.edu.cn (Y.X.); huyaocun@s.jlufe.edu.cn (Y.H.); mayongyong@s.jlufe.edu.cn (Y.M.); yinxin@s.jlufe.edu.cn (X.Y.)

* Correspondence: lijun@jlufe.edu.cn

† These authors contributed equally to this work.

Abstract: Adversarial attacks expose the latent vulnerabilities within artificial intelligence systems, necessitating a reassessment and enhancement of model robustness to ensure the reliability and security of deep learning models against malicious attacks. We propose a fast method designed to efficiently find sample points close to the decision boundary. By computing the gradient information of each class in the input samples and comparing these gradient differences with the true class, we can identify the target class most sensitive to the decision boundary, thus generating adversarial examples. This technique is referred to as the "You Only Attack Once" (YOAO) algorithm. Compared to the DeepFool algorithm, this method requires only a single iteration to achieve effective attack results. The experimental results demonstrate that the proposed algorithm outperforms the original approach in various scenarios, especially in resource-constrained environments. Under a single iteration, it achieves a 70.6% higher success rate of the attacks compared to the DeepFool algorithm. Our proposed method shows promise for widespread application in both offensive and defensive strategies for diverse deep learning models. We investigated the relationship between classifier accuracy and adversarial attack success rate, comparing the algorithm with others. Our experiments validated that the proposed algorithm exhibits higher attack success rates and efficiency. Furthermore, we performed data visualization on the ImageNet dataset, demonstrating that the proposed algorithm focuses more on attacking important features. Finally, we discussed the existing issues with the algorithm and outlined future research directions. Our code will be made public upon acceptance of the paper.

Keywords: adversarial examples; adversarial attacks; deep learning; computer vision

check for updates

Academic Editor: Andrea Prati

Received: 11 December 2024
Revised: 24 December 2024
Accepted: 27 December 2024
Published: 31 December 2024

Citation: Li, J.; Xu, Y.; Hu, Y.; Ma, Y.; Yin, X. You Only Attack Once: Single-Step DeepFool Algorithm. *Appl. Sci.* **2025**, *15*, 302. https://doi.org/10.3390/app15010302

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

With the rapid advancement of computer technology, deep learning has found extensive applications in various domains, particularly excelling in tasks like image classification. However, their high accuracy and robustness can be compromised by adversaries who deliberately manipulate the input data to deceive the model. Such attacks could have severe implications in real-world applications, such as in autonomous driving systems [1], where adversarial attacks could mislead the vehicle's AI perception system, causing it to overlook obstacles or misinterpret road signs, leading to serious consequences. Alternatively, they can deceive security systems employing deep neural networks (DNNs) for face recognition [2]. Attackers may carry out such actions with malicious intent to cause harm or to exploit the artificial intelligence system for personal gain. Adversarial attacks are not only present in the field of computer vision but also exist in other areas, such as quantum cryptography attacks [3,4]. Understanding the vulnerabilities of DNNs to

adversarial attacks and developing effective defense mechanisms has become a crucial research area in the field of artificial intelligence (AI) and computer security. Adversarial samples are constructed by introducing minor perturbations to the original data, which trick the classifier into making incorrect classifications. By investigating the generation algorithms of adversarial examples, research can advance defenses against such examples, consequently bolstering the security of deep neural network models.

Adversarial sample generation algorithms play a crucial role in understanding the limitations of neural networks and enhancing network security. A growing number of startups are involved in a diverse range of computer vision tasks. DeepFool is an algorithm proposed by Moosavi-Dezfooli et al. [5] that iteratively determines the minimum perturbation needed to move a given input image into a misclassified region of the feature space. Although the DeepFool generation process typically requires multiple iterations to achieve high attack success rates, conducting numerous iterations to generate adversarial samples substantially escalates computational complexity. This necessitates greater computational resources, particularly in the present era characterized by rapid growth in big data and large-scale models. For startups, this presents a significant challenge in improving model security. Therefore, a pressing need exists for a rapid and efficient method to seek adversarial perturbations, to bolster adversarial training techniques, fortify model security, and investigate model robustness against adversarial perturbations. Within the domain of adversarial attacks, methods for generating adversarial samples through single-step iteration remain relatively scarce. In this study, we modified the iteration direction of the DeepFool algorithm. By leveraging gradient differences to identify the direction of samples closest to the decision boundary and searching for adversarial samples along this direction, we can accelerate the generation speed of adversarial samples.

While the DeepFool algorithm is effective in identifying minimal perturbations for misclassifying data, a more expedited methodology is imperative. Our main contributions are as follows:

- As shown in Figure 1, We enhanced the DeepFool algorithm by computing the gradient information of each class in input samples and comparing these gradient differences with the true class that generates adversarial samples in a single iteration, achieving a significant 68% increase in the success rate of adversarial sample generation compared to the original algorithm.
- We demonstrated through experiments that the algorithm has a simpler time complexity than the DeepFool algorithm. By conducting comparative experiments with other related research, we showed that the YOAO algorithm exhibits superior attack efficiency and is more effective at misleading various deep neural networks into misclassification.

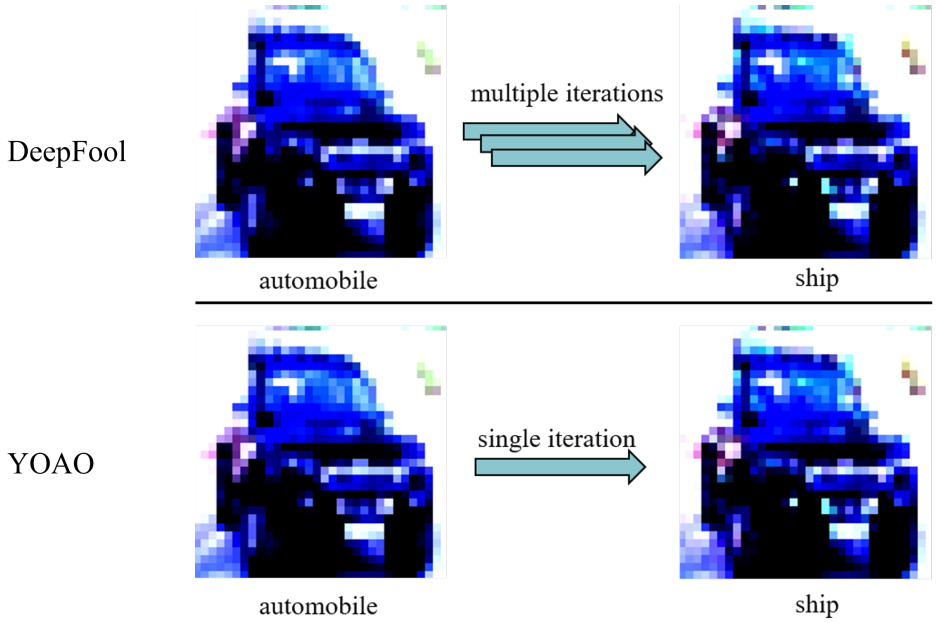


Figure 1. Disparity between the DeepFool algorithm and the algorithm introduced in this study. Initially, a sample is classified as belonging to the "automobile" class by the classifier. Following the creation of adversarial samples using both the DeepFool and YOAO algorithms, the resulting adversarial sample is misclassified as belonging to the "ship" class by the classifier. The upper subfigure illustrates the necessity for multiple iterations by the DeepFool algorithm to progressively approach the decision boundary and generate adversarial samples. In contrast, the lower subfigure presents the YOAO algorithm introduced in this study, which requires only a single iteration to effectively generate an adversarial sample. Clearly, the YOAO algorithm notably enhances computational efficiency and practical applicability.

2. Related Works

2.1. Adversarial Attacks

Adversarial attacks can be categorized into distinct types based on various criteria. Segregated by the understanding of the model's internal architecture, they are delineated as white-box attacks and black-box attacks [6]. A white-box attack constitutes an adversarial attack algorithm that requires understanding the model's internal structure and gradients to generate adversarial samples. Conversely, a black-box attack operates without needing knowledge of the internal structure and gradient information of the targeted model. Further classification of adversarial attacks is based on whether the attack algorithm can misclassify the model into a specific class, leading to the distinction between targeted attacks and non-targeted attacks. Targeted attacks aim to misclassify input data into a specific target class by following certain constraints. In contrast, non-targeted attacks focus solely on potentially misclassifying input data without aiming for a specific target class.

The defense of adversarial samples has driven the evolution of adversarial sample attacks. Likewise, ensuring the robustness and security of deep learning models requires a more powerful adversarial attack algorithm. A significant breakthrough has been achieved in uncovering vulnerabilities in multiple state-of-the-art deep neural networks (DNNs), signifying notable advancements in the domain of adversarial machine learning. Despite being trained meticulously, DNNs exhibit distinct blind spots and are vulnerable to adversarial attacks. Adversaries exploit these weaknesses to generate adversarial examples, capable of misleading machine learning models through slight alterations to the input data distribution. This intrinsic feature of DNNs is utilized in various adversarial attack algorithms to produce adversarial examples.

The Fast Gradient Sign Method (FGSM), proposed by Goodfellow et al. [7], is based on the concept of gradients and can rapidly and effectively generate adversarial examples. However, this method requires the manual selection of a perturbation coefficient, leading to the addition of relatively large perturbations. The Iterative Fast Gradient Sign Method (I-FGSM), proposed by Kurakin et al. [8], enhances the FGSM by iteratively applying perturbations, typically resulting in smaller perturbations and stronger white-box attack capabilities compared to the FGSM. However, this approach demands more computational resources and time to achieve the iterations. The Trans-IFFT-FGSM (Transformer Inverse Finite Fourier Transform Fast Gradient Sign Method), presented in the work by Naseem [9], incorporates multiple modules and retains input noise information, enhancing the attack success rate (ASR) with smaller perturbations, but at the cost of increased algorithm complexity and computational expense.

DeepFool, proposed by Moosavi-Dezfooli et al. [5], aims to introduce the minimal perturbation necessary to generate adversarial examples. As a white-box untargeted attack, the fundamental concept of DeepFool involves perturbing the input image in a manner that guides it across the separating boundary. While this algorithm can generate adversarial examples with minimal perturbations, its high computational cost arises from the need for multiple iterations. Building upon the concept of DeepFool, Moosavi-Dezfooli et al. [10] proposed a universal perturbation method that calculates the sum of vectors representing the distances from multiple original inputs to the decision boundary to generate a universal perturbation. This algorithm successfully perturbs a majority of images in the dataset, demonstrating a certain level of generalization. By iteratively finding the minimal perturbation and summing the found minimal perturbations, the final universal perturbation is obtained. To address the shortcomings of the DeepFool algorithm, Gajjar et al. [11] proposed a targeted version called Targeted DeepFool, which can misdirect misclassification towards a specific target class, realizing a targeted attack approach for DeepFool. Labib et al. [12] introduced an enhanced version of Targeted DeepFool, which increases the ability to misclassify specific classes and introduces a minimum confidence score requirement hyperparameter to enhance flexibility, albeit at the cost of increased algorithm complexity. In the latest research on adversarial sample transferability, Ling et al. [13] proposed a simple channel switching method called CS-MI-FGSM, which obtains variants of the input image and mixes them during momentum updating to improve the transferability of adversarial samples.

In our study comparing the strengths and limitations of existing algorithms, we propose the YOAO (You Only Attack Once) algorithm, which not only rapidly generates adversarial examples but also ensures their potent attack capability. This algorithm is a gradient-based, fast DeepFool method that calculates the maximum gradient disparity across different classes to more swiftly identify the minimal perturbation. By generating adversarial samples in a single-step iteration, it addresses the slow attack speed issue of the original DeepFool algorithm, significantly enhancing the algorithm's attack efficiency.

2.2. Adversarial Defense

Adversarial sample defense refers to the measures taken to protect deep neural networks (DNNs) from adversarial attacks, which are maliciously crafted input samples aimed at misleading the neural network's output [14]. The significance of adversarial sample defense lies in ensuring that DNNs can maintain robustness and reliability when faced with adversarial attacks, thereby ensuring their correct operation and resilience against malicious manipulations in practical applications [15]. By implementing effective adversarial sample defense mechanisms, the security of deep learning systems can be enhanced.

Adversarial training is the most direct and commonly used defense technique to enhance the adversarial robustness of models. The fundamental principle involves augmenting training data by introducing additional adversarial samples. Initially proposed by Szegedy et al. [16], this concept aimed at training neural networks with robustness for classification tasks. Adversarial training aids in reducing the loss curvature in both input space and model parameter space, thereby narrowing the adversarial robustness gap between train and test data. These findings have spurred the development of a range of regularization techniques that enhance adversarial defense capabilities by explicitly addressing the curvature of the loss function or other geometric features [15]. Many regularization techniques can reduce computational complexity without increasing the training set, such as local linear regularization (LLR) [17] and input gradient regularization (IGR) [18].

In addition to adversarial training, there exist alternative approaches to improve a model's adversarial robustness, such as topology-aligned adversarial training (TAAT) [19], feature-decoupled networks (FDNet), and attention-guided reconstruction loss (AIAF-Defense) [20], which aim to smooth out the loss landscape in the model parameter space. Other defense strategies include feature denoising [21], domain adaptation [22], and ensemble defenses [15].

3. Method

3.1. Introduction to the Primitive Method

3.1.1. Mathematical Representation of Adversarial Examples

The goal is to carefully craft extremely small perturbations to be applied to the original samples, thereby creating adversarial examples. The characteristic of these perturbations is their tiny magnitude, such that they are nearly imperceptible to human vision, yet for machine learning models, they lead to inconsistent classification results compared to the original class. The task of generating adversarial examples is as follows:

$$f(x) \neq f(x+r) \text{ subject to } r \leq \epsilon \tag{1}$$

where $f(x)$ represents the class predicted by the classifier for the original image sample, $f(x+r)$ represents the class predicted by the classifier for the adversarial sample with added perturbation, and ϵ represents a predefined very small parameter.

3.1.2. The Relationship Between Adversarial Examples and Decision Boundaries

In the training process of neural networks, original low-dimensional data such as hand-written digits or images from natural scenes undergo multi-layered, non-linear transformations. During this process, the data gradually map from a low-dimensional space to a high-dimensional feature space, forming feature maps. These feature maps not only contain basic information from the original images, such as edges, textures, and shapes, but also integrate complex patterns learned by the neural network during the training process and the inherent relationships between classes, such as abstract concepts and different ways of combining features. In the high-dimensional feature space, the neural network gradually learns one or multiple decision boundaries by optimizing the loss function. As shown in Figure 2, these decision boundaries partition the feature space into different regions, with each region corresponding to a class. The shapes and positions of these boundaries depend on the distribution of the training data, the architecture of the network, and the choice of optimization algorithm. By continuously adjusting network parameters, the neural network can find the optimal decision boundaries, thus achieving accurate differentiation between different classes.

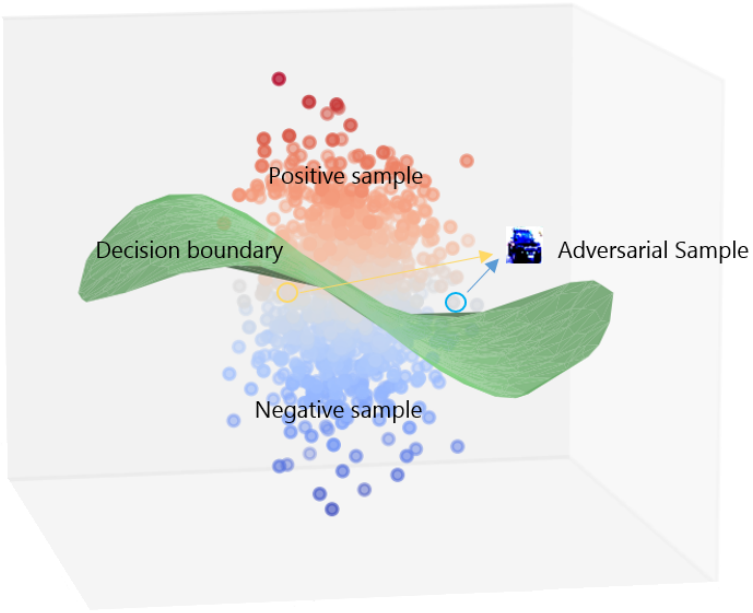


Figure 2. Data space distribution. This figure illustrates the existence of adversarial examples. The red data points represent positive samples, the blue data points represent negative samples, and the green curve represents the decision boundary of the classifier. In the data distribution, the samples above the decision boundary are classified as positive, while those below are classified as negative. Upon examining the figure, we can observe the presence of some negative samples near the decision boundary, indicating samples where data points originally classified as negative are misclassified as positive by the classifier.

3.1.3. Introduction to the DeepFool Algorithm

Moosavi-Dezfooli et al. [5] proposed a sophisticated adversarial attack technique designed to introduce imperceptible small perturbations to benign inputs. Based on the concept of hyperplane-based classification, it iteratively linearizes the classifier to generate minimal perturbations sufficient to alter the classification label. Moosavi-Dezfooli et al. systematically investigate the minimal perturbations required to traverse the decision boundary of the target classifier. For binary classification problems, it identifies the nearest decision boundary and computes the minimal perturbation from the current point to that boundary to determine the adversarial sample. In the case of multiclass classification problems, the issue is decomposed into multiple binary classification problems, with the minimum perturbation calculated for each class individually.

In order to extend the applicability of this approach to non-linear classifiers, DeepFool employs an iterative strategy that approximates the boundary of the initial correct class label associated with the point of interest. This iterative process draws inspiration from Newton's iterative algorithm, enhancing the algorithm's capacity to navigate the complex decision boundaries inherent in non-linear classification scenarios.

3.2. Algorithm Improvement

Search for the Sample Point Closest to the Decision Boundary

The core idea of the DeepFool algorithm is to find the adversarial examples that are closest to the classification boundary. In this work, the proposed algorithmic improvements to the DeepFool algorithm enhance its efficiency in finding adversarial examples and constructing effective universal perturbations.

We proposed improvements to the iterative process for finding the sample point closest to the decision boundary. In each iteration step, we calculate the gradient information of the input sample with respect to each class, quantifying the sensitivity of each class to small

changes in the input sample. We then compare the gradient differences between different classes to identify the largest one. These large gradient differences typically indicate that the decision boundary for these classes is more sensitive to the input sample. Therefore, searching along the direction of the class with the largest gradient difference can efficiently find the sample point that has a significant impact on the classification decision.

Figure 3 illustrates the algorithmic process, wherein we take a three-class classification task as an example. We calculate the gradient differences between other classes and the original sample class, quantifying the sensitivity of each class to small changes in the input sample. Specifically, we determine the distances of the sample point x_0 from the class regions where sample points x_1 and x_2 are located. This allows us to identify the point x_1 that has the greatest discrepancy from the current image class. Subsequently, by computing in the direction from the original sample point x_0 towards x_1, we can rapidly and effectively generate adversarial samples.

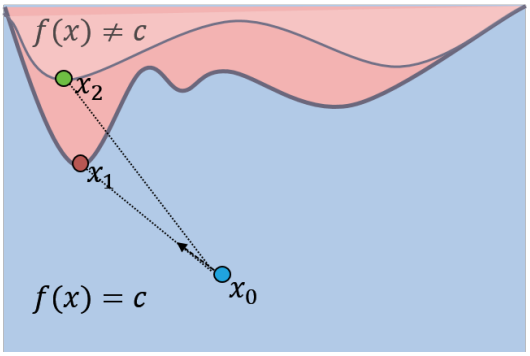


Figure 3. In this process, x_0 denotes the original sample point, with the blue region representing the original class and the pink region indicating other classes. Sample poinSts x_1 and x_2 are situated in regions corresponding to classes other than the original sample class, and they represent samples from distinct classes. By identifying the decision boundary closest to the sample point as the direction for the proposed algorithmic iteration, we aim to expedite the generation of adversarial samples.

We improved the process of finding adversarial samples that are closest to the classification boundary in the DeepFool algorithm. In this work, we utilized gradient information by computing the gradients of the input sample with respect to each class, thereby quantifying the sensitivity of each class to small perturbations in the input sample. This information enables the identification of the class most sensitive to variations in the input samples. By comparing the gradient differences among different classes, we can more effectively search for and locate sample points that significantly impact the classification decision, thereby finding points closest to the decision boundary. Having identified sample points closest to the decision boundary, we employ a minimal number of iterations (even potentially just one iteration) to generate adversarial samples, a method referred to as "You Only Attack Once" (YOAO). The aim of this approach is to enhance the efficiency of finding effective adversarial samples while reducing the number of iterations required.

The gradient difference between classes can be expressed as follows:

$$\Delta_{c,\hat{c}}(x) = \nabla f_c(x) - \nabla f_{\hat{c}}(x) \tag{2}$$

where x represents the original image, $\hat{c}$ represents the current class, c represents the original class, and $\nabla f_{\hat{c}}(x)$ represents the gradient information of input x with respect to the current class $\hat{c}$.

The perturbation direction is computed using the following formula:

$$sign(\Delta_{c,\hat{c}}(x)) = \frac{\Delta_{c,\hat{c}}(x)}{\| \Delta_{c,\hat{c}}(x) \|} \tag{3}$$

where $\Delta_{c,\hat{c}}(x)$ represents the difference in the gradient between the current class $\hat{c}$ and the original class c for input x, and $\| \Delta_{c,\hat{c}}(x) \|$ denotes its norm.

The formula for determining the direction of perturbation in the adversarial optimization process is as follows:

$$r_i = r_{i\text{-}1} \times sign(\Delta_{c,\hat{c}}(x)) \tag{4}$$

In Equation (4), the variable r_i signifies the perturbation at the the i-th iteration, while $r_{i\text{-}1}$ represents the perturbation at the $(i-1)$-th iteration.

We can restate the optimization problem as follows:

$$\begin{aligned} &\text{minimize} \quad \|r\|_2 \\ &\text{subject to} \quad f(x+r) \neq f(x) \\ &\qquad \|r\|_2 \leq \epsilon \\ &\qquad r = -\alpha \cdot \text{sign}(\Delta_{c,\hat{c}}(x)) \end{aligned} \tag{5}$$

where $\|r\|_2$ represents the Euclidean norm (L2 norm) of the perturbation r, which indicates the magnitude of the perturbation. $f(x+r) \neq f(x)$ is a constraint that ensures the perturbed sample has a different classification result from the original sample. $\|r\|_2 \leq \epsilon$ is another constraint that ensures the magnitude of the perturbation r is within a pre-defined threshold ϵ, which defines the "smallness" of the perturbation. In Equation (2), $\Delta_{c,\hat{c}}(x)$ represents the difference between the gradients of the original class c and the current class $\hat{c}$. α is a scalar that controls the step size or magnitude of the perturbation. In Equation (3), $\text{sign}(\Delta_{c,\hat{c}}(x))$ ensures that the direction of the perturbation is aligned with the gradient difference direction.

As shown in Algorithms 1 and 2, by introducing a single-step iterative generation scheme, we successfully eliminated the cumbersome loop structure in the algorithm. Specifically, we streamlined the loop condition in line 5 and the loop in line 6 of the original DeepFool algorithm, no longer tediously calculating gradient information for the top n non-target classes with the highest probabilities. Instead, we directly compute gradients for the class that exhibits the maximum gradient difference from the original class, thereby significantly reducing the time complexity. This step is crucial as it provides us with the most sensitive class information regarding the input sample. Subsequently, we utilize the gradient information w and confidence information f of this class to perform iterative calculations. This process not only enhances the efficiency of the algorithm but also ensures that we can accurately pinpoint the sample points that have the most significant impact on classification decisions.

This enhancement has endowed the proposed algorithm with the ability to rapidly and efficiently obtain adversarial samples, even when limiting the maximum number of iterations to 1.

Algorithm 1 DeepFool algorithm determining the magnitude and direction of perturbations by calculating the minimum distance from the input sample to the nearest decision boundary

procedure DEEPFOOL(x, f)
  Input: Image x, classifier f.
  Output: Perturbation $\hat{r}$.
  Initialize: $x_0 \leftarrow x, i \leftarrow 0$.
  while $\hat{k}(x_i) = \hat{k}(x_0)$ **do**
    for $k \neq \hat{k}(x_0)$ **do**
      $w'_k \leftarrow \nabla f_k(x_i) - \nabla f_{\hat{k}(x_0)}(x_i)$
      $f'_k \leftarrow f_k(x_i) - f_{\hat{k}(x_0)}(x_i)$
    end for
    $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(x_0)} \frac{|f'_k|}{\|w'_k\|_2}$
    $r_i \leftarrow \frac{|f'_{\hat{l}}|}{\|w'_{\hat{l}}\|_2^2} w'_{\hat{l}}$
    $x_{i+1} \leftarrow x_i + r_i$
    $i \leftarrow i + 1$
  end while
  return $\hat{r} = \sum_i r_i$
end procedure

Algorithm 2 YOAO algorithm, based on DeepFool algorithm, optimizes the direction of searching for perturbations through gradient difference information

procedure YOAO(x, f)
  Input: Image x, classifier f.
  Output: Perturbation $\hat{r}$.
  Initialize: $x_0 \leftarrow x, i \leftarrow 0$.
  for $k \neq \hat{k}(x_0)$ **do**
    $\hat{k} \leftarrow \arg\max_{\hat{k}(x_i) \neq \hat{k}(x_0)} \left\| \nabla f_{\hat{k}(x_i)}(x_i) - \nabla f_{\hat{k}(x_0)}(x_i) \right\|$
    $w'_k \leftarrow \nabla f_{\hat{k}}(x_i) - \nabla f_{\hat{k}(x_0)}(x_i)$
    $f'_k \leftarrow f_{\hat{k}}(x_i) - f_{\hat{k}(x_0)}(x_i)$
    $\hat{l} \leftarrow \frac{|f'_k|}{\|w'_k\|_2}$
    $r_i \leftarrow \frac{|f'_{\hat{l}}|}{\|w'_{\hat{l}}\|_2^2} w'_{\hat{l}}$
    $x_{i+1} \leftarrow x_i + r_i$
    $i \leftarrow i + 1$
  end for
  return $\hat{r} = \sum_i r_i$
end procedure

4. Experiments

4.1. Datasets

In the context of classification tasks, we employ the validation images from the ILSVRC2012 [23], CIFAR-10 [24], and MNIST [25] datasets for our experiments. As target classifiers, we utilize pretrained ResNet18, VGG16 models.

ILSVRC2012 [23]: The dataset comprises 50,000 images spanning a thousand distinct classes. Widely recognized as a benchmark dataset in the field of computer vision, it has been instrumental in driving the progress of deep learning models. With its extensive size and diversity, the dataset serves as a comprehensive depiction of real-world visual data.

CIFAR-10 [24]: The CIFAR-10 dataset comprises 60,000 color images, each with a resolution of 32 × 32 pixels distributed across 10 distinct classes. Specifically, the dataset is

partitioned into 50,000 images designated for training purposes and 10,000 images reserved for testing.

MNIST [25]: The MNIST dataset is a well-known dataset in the field of machine learning and computer vision. It consists of a large collection of handwritten digits (0 to 9) that are commonly used for training and testing machine learning models, particularly for image classification tasks. Each image in the MNIST dataset is a grayscale image with a resolution of 28 × 28 pixels.

4.2. Experimental Settings

We conduct experiments on the experimental settings to evaluate the classifiers against this attack. The experimental configurations includes an Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz processor, an NVIDIA Tesla V100 GPU, and 16 GB of RAM. PyTorch 2.0.1 and Torchmetrics 0.15.2 libraries are installed on these experimental settings systems, with Python version 3.6 being utilized.

4.3. Evaluation Metrics

The success rate of adversarial attacks:

$$\rho(f) = \frac{n}{N} \tag{6}$$

where ρ represents the success rate of adversarial attacks on the classifier f, and N represents all the image samples used for testing, and n represents the number of samples that, generated by the adversarial attack algorithm, successfully induce misclassification by the classifier.

Perturbation: The magnitude of perturbations added to an image, known as "perturbation", quantifies the level of changes required to deceive a classifier. We calculate the difference between the perturbed image and the original image, and then take the L2 norm to quantify the amount of change in the image.

Steps: The number of iterations needed to perturb an image is recorded. This metric indicates the average number of iterations required to achieve a successful misclassification.

Time: This metric is used to denote the computational time needed to execute an attack against a single image.

4.4. Experimental Results

4.4.1. Experimental Comparison with Classical Algorithms

As shown in Table 1, we conducted a comparative analysis of various adversarial attack algorithms to assess their efficacy in classification tasks. The algorithms under consideration include PGD (Projected Gradient Descent) [26], FGSM (Fast Gradient Sign Method) [7], BIM (Basic Iterative Method) [27], FFGSM (Fast Fourier Gradient Sign Method) [28], APGD (Adaptive Projected Gradient Descent) [29], SparseFool [30], OnePixel [31], Pixel [32], PIFGSMPP (Patch-based Iterative Fast Gradient Sign Method with Patch Prior) [33], and DeepFool [5]. These algorithms encompass a spectrum of adversarial attack strategies, ranging from gradient-based methods to iterative approaches, as well as techniques leveraging image patches and sparsity. The selection of these algorithms is justified by their distinct attack mechanisms and their significant impact within the academic community, allowing us to challenge and evaluate the robustness of classification models from multiple perspectives. Through these comparative experiments, we aim to gain a comprehensive understanding of the effectiveness of different attack strategies and to provide valuable insights for enhancing model robustness.

Table 1. Comparative analysis of adversarial attack algorithms on classifiers.

Algorithms	Datasets	Success Rate	L2 Norm	Steps	Time
PGD [26]	CIFAR-10	72.5%	41.25	10	0.005 s
FGSM [7]	CIFAR-10	63.5%	41.26	1	0.003 s
BIM [27]	CIFAR-10	55.2%	17.49	10	0.004 s
FFGSM [28]	CIFAR-10	72.1%	41.25	1	0.003 s
APGD [29]	CIFAR-10	58.19%	10.42	10	0.004 s
Sparsefool [30]	CIFAR-10	91.0%	54.18	10	1.962 s
OnePixel [31]	CIFAR-10	47.9%	2.58	10	0.076 s
Pixel [32]	CIFAR-10	52.9%	12.66	10	0.129 s
PIFGSMPP [33]	CIFAR-10	82.3%	0.05	10	0.003 s
DeepFool [5]	CIFAR-10	23.15%	0.05	1	0.056 s
YOAO (Ours)	CIFAR-10	93.81%	3.83	1	0.062 s
PGD	MNIST	3.5%	0.61	10	0.003 s
FGSM	MNIST	3.5%	0.65	1	0.003 s
BIM	MNIST	4.1%	0.63	10	0.003 s
FFGSM	MNIST	3.4%	0.64	1	0.003 s
APGD	MNIST	4.1%	0.02	10	0.003 s
OnePixel	MNIST	1.7%	0.16	10	0.616 s
Pixel	MNIST	89.9%	6.04	10	0.723 s
DeepFool	MNIST	30.2%	1.53	1	0.040 s
YOAO (Ours)	MNIST	66.5%	9.01	1	0.091 s

We trained a classifier with an accuracy rate of 60% on the CIFAR dataset. It can be observed that the performance of various adversarial attack algorithms is high. Our YOAO algorithm ranked second, surpassing the DeepFool algorithm, and was only next to the PIFGSMPP algorithm. On the other hand, we trained a classifier on the MNIST dataset to achieve an accuracy rate of 99%. Interestingly, when the classifier accuracy is high, the success rate of adversarial attack algorithms significantly decreases. However, both the proposed algorithm and the Pixel algorithm maintained high attack success rates on high-accuracy classifiers. Compared to the Pixel algorithm, our method generates each adversarial sample image in just 0.091 s, making it 7.9 times faster. This demonstrates significant progress in both efficiency and attack performance of the proposed algorithm.

4.4.2. Performance Comparison Between the YOAO (You Only Attack Once) Algorithm and the DeepFool Algorithm

As shown in Figure 4, based on the data presented in Figure 4a,b, it is observed that the proposed algorithm (YOAO) exhibits an increase in the time required to generate a single adversarial sample as the maximum number of iterations increases, relative to the original DeepFool algorithm. Further observation of Figure 4c,d reveals that the algorithm maintains a high attack success rate across different maximum iteration settings. Particularly noteworthy is that, in the case of a single iteration, the algorithm achieves a higher attack success rate compared to the original algorithm, which is one of the reasons why we named the proposed algorithm "You Only Attack Once (YOAO)".

These data are sourced from the CIFAR-10 dataset, from which 500 images were selected for experimentation, leading to the aforementioned conclusions. These results demonstrate that the YOAO algorithm has more efficient and effective characteristics than the traditional DeepFool algorithm when generating adversarial examples. By dynamically selecting the gradient update direction for each input data and reducing the search space, The YOAO algorithm can quickly find perturbations that cause misclassification to the target class.

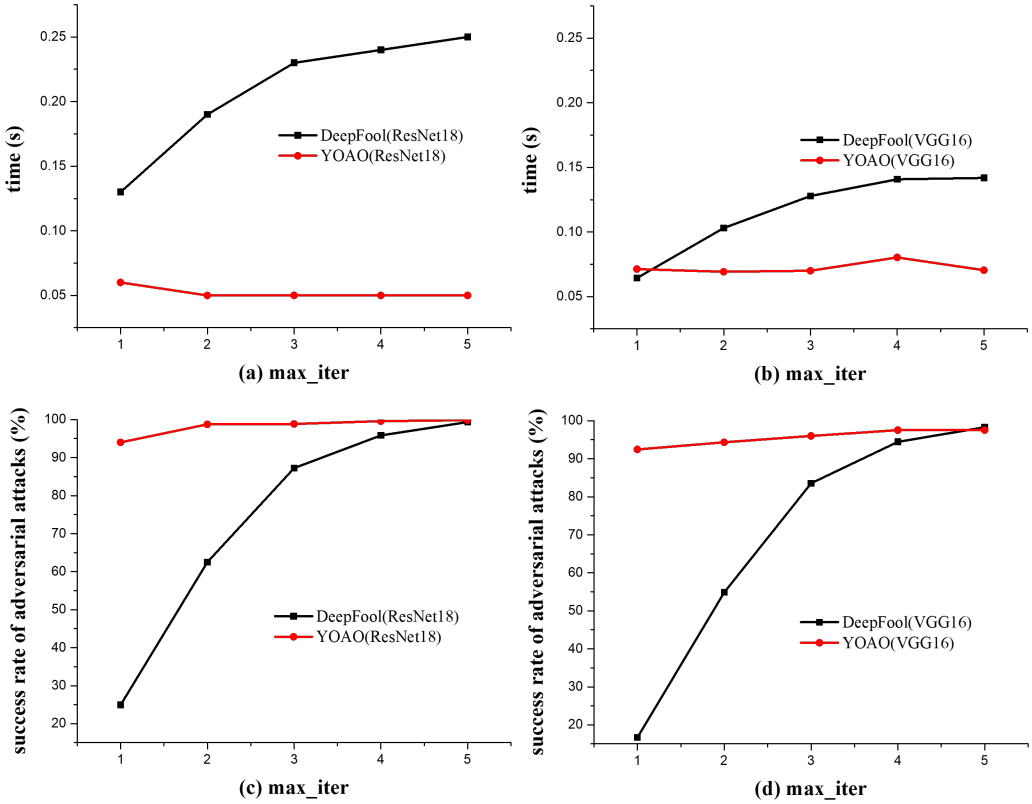


Figure 4. The impact of the maximum iteration parameter (max-iter) on the original DeepFool algorithm and the YOAO algorithm is illustrated in the four subfigures. (**a**,**b**) demonstrate the speed of generating a single adversarial sample at different maximum iteration values, while (**c**,**d**) show the success rate of the algorithm's attacks under varying maximum iteration settings. In our experiments, we employed two classifiers, ResNet18 and VGG16.

4.4.3. Performance Variation in YOAO and DeepFool Algorithms Under Different Parameters

As shown in Figure 5, to investigate the impact of classifier accuracy on adversarial attack algorithms, experiments were conducted using ResNet and VGG networks with an accuracy of 60% as classifiers, comparing the performance of the DeepFool algorithm and the YOAO algorithm on the CIFAR-10 dataset. In this section, we employed the ResNet classifier with an accuracy of 95.45% to evaluate the attack success rates and efficiency of both the DeepFool and YOAO algorithms on the CIFAR-10 dataset. It was observed that, at a maximum iteration count (max-iter) of 1, the attack success rate of the YOAO algorithm dropped to 38%. This indicates that, on well-trained classifiers, the attack success rate of the YOAO algorithm may decrease. However, as the max-iter parameter is increased, the attack success rate for the YOAO algorithm also improves. Notably, under the same parameter settings, the attack success rate of the YOAO algorithm consistently surpasses that of the DeepFool algorithm. Additionally, the time required to generate a single adversarial sample was calculated for both algorithms, revealing that, even with high-accuracy classifiers, the YOAO algorithm requires significantly less time to produce a single adversarial sample compared to the DeepFool algorithm. This underscores the superior performance of the proposed YOAO algorithm.

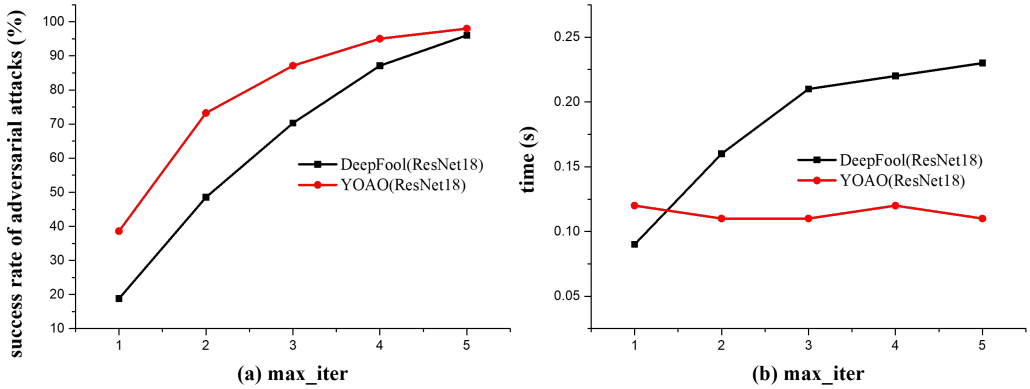


Figure 5. The two subfigures explain the influence of the maximum iteration parameter (max_iter) on both the original DeepFool algorithm and the YOAO algorithm within a classifier with an accuracy of 95.45%. (**a**) delineates the success rate of generating adversarial samples across various max-iter values, while (**b**) portrays the velocity of producing individual adversarial samples at different max-iter settings.

4.4.4. Visualization of Adversarial Samples Generated by YOAO and DeepFool Algorithm

As shown in Figure 6, in the two sets of comparison images provided, each set consists of three small subfigures: the original image, the adversarial example, and the pixel-wise difference map between them, and the top labels clearly indicate the classification results by their corresponding classifiers. It is evident that the classification result of the adversarial example significantly differs from that of the original image, indicating the efficacy of the algorithm used to generate the adversarial sample.

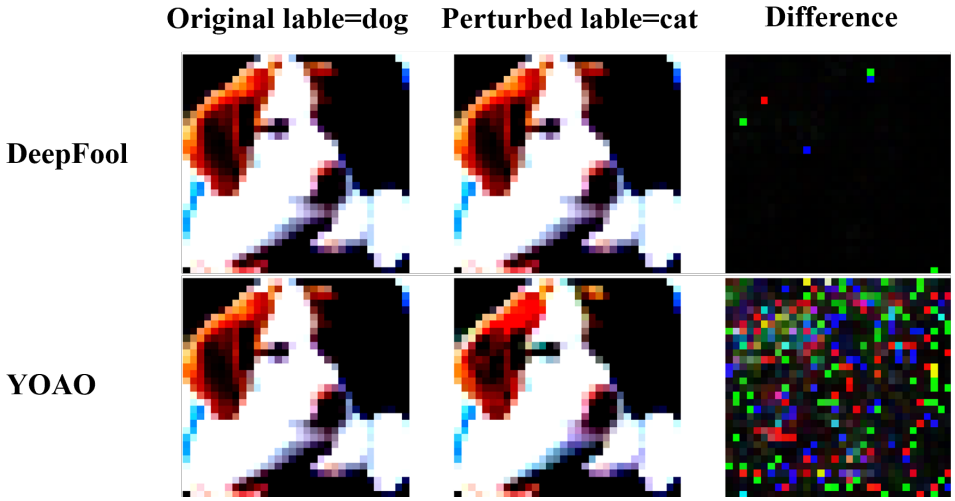


Figure 6. There are two groups of images in the figure: the top three sets are relaxed by the DeepFool algorithm, while the bottom three sets are relaxed by the YOAO algorithm, demonstrating its perturbation effects on the CIFAR-10 dataset.

These results were obtained with the max-iter parameter set to 1, signifying that the algorithm underwent a single-step iteration. This suggests that both the YOAO algorithm and the DeepFool algorithm can find adversarial samples within a single iteration, with the algorithm demonstrating higher computational efficiency under this single-step iteration scenario. Despite a slight increase in the magnitude of perturbation, it remains effectively constrained within a small range, making it imperceptible to human observers. This breakthrough not only demonstrates the efficiency of the algorithm, but also underscores its potential in the field of adversarial example generation through optimized algorithmic

design. We have achieved results comparable to or even surpassing traditional methods in just a single iteration. Such efficiency improvements not only save computational resources, but also accelerate the application and research progress of adversarial example generation.

In order to extend the applicability of the algorithm to broader domains, we have chosen to employ the ImageNet dataset, which closely mirrors real-world scenarios. As shown in Figure 7, it is evident that the adversarial samples we generate bear a striking resemblance to the original images, rendering them nearly imperceptible to the naked eye and possessing a high degree of obfuscation. Furthermore, we conducted a comparative analysis of the differences between the original images and the adversarial samples, visualizing these differences. Through visualization, we are able to discern that the colored regions represent the added perturbation pixels, which closely align with the original image and are predominantly concentrated on the image's class-specific features. The algorithm perturbs crucial feature components, resulting in misclassification of the images. It is precisely due to the interference with vital features by the algorithm that it exhibits robust adversarial capabilities, even on classifiers boasting an accuracy as high as 99%, while significantly reducing the time required for generating adversarial samples.

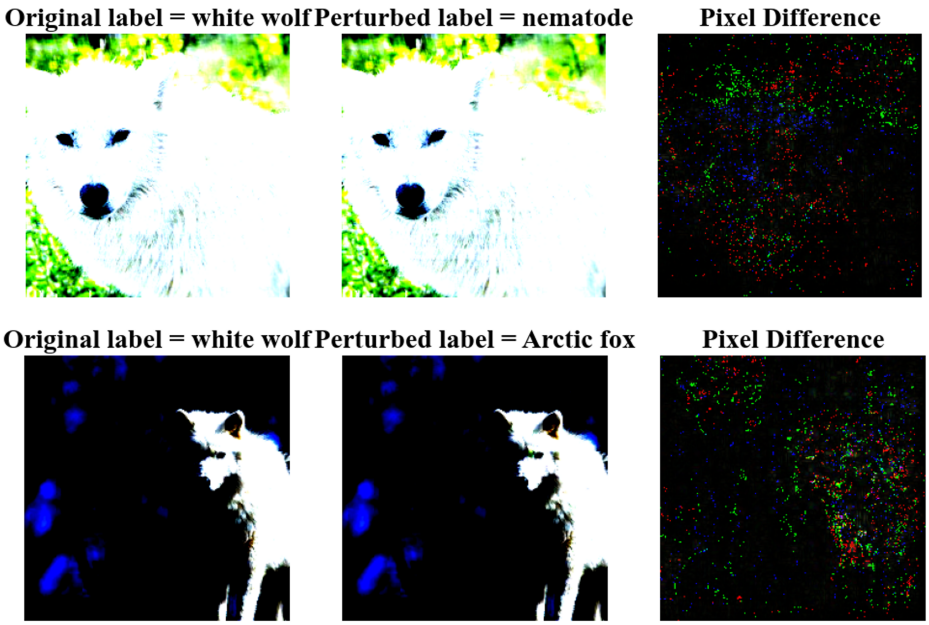


Figure 7. This figure illustrates the performance of the proposed algorithm on the ImageNet dataset. The three columns present distinct information: the left column displays the original images, the middle column showcases the adversarial samples generated by the YOAO algorithm, and the right column illustrates the differences between the adversarial samples and the original images.

By delving into the mechanisms of adversarial sample generation algorithms, we delved into how classifiers are perturbed, a pursuit that holds significant implications for elucidating the operational principles of models and enhancing their security. A thorough exploration of adversarial sample generation algorithms not only aids in comprehending the vulnerabilities of the model, but also offers crucial insights for bolstering model robustness. We are not merely attacking models, we are continuously deepening our understanding of the intrinsic mechanisms of machine learning models, thereby laying a foundation for constructing AI systems that are more secure and reliable.

5. Discussion and Future Work

In this section, we extensively discussed our contributions and highlighted directions for future research. Additionally, we briefly outlined the limitations and fundamental assumptions of our model.

5.1. Discussion on Experiments

We carefully designed a series of experiments to evaluate the effectiveness of the YOAO (You Only Attack Once) algorithm. Initially, we conducted a thorough comparison between YOAO and other advanced algorithms, assessing their performance in terms of attack success rates, attack times, and L2 norm perturbation sizes. The experiments confirmed that the YOAO algorithm performs better than other algorithms, achieving not only a higher success rate in attacks, but also quicker attack times.

Subsequently, we conducted a detailed comparison between the YOAO algorithm and DeepFool algorithms across different datasets and classifiers, validating the reasons behind YOAO's superior attack efficiency. This further emphasizes the importance of optimizing the search for the maximum gradient differences, while also confirming that classes with the greatest gradient differences from the original sample are more likely to impact the classifier's decisions.

Lastly, we visualized and compared the adversarial samples created by YOAO algorithms and DeepFool algorithms. While YOAO algorithms introduced more changes to the pixels in the generated adversarial samples, we noticed that the perturbation size remained within a small range, making it undetectable to the human eye. This observation underscores the strong attack capabilities of YOAO algorithms, as the perturbations, although slightly higher than those of DeepFool algorithms, are still subtle enough to avoid visual detection.

5.2. Discussion on the Model

In algorithm design, the FGSM (Fast Gradient Sign Method) algorithm computes the sign of the gradient, seeking the direction in each input dimension that maximally increases the loss function, and adds perturbations along that direction to generate adversarial samples. We observe that FGSM algorithm leverages the model's sensitivity to input variations, where directions with the largest gradients are more sensitive to inputs and thus more likely to produce adversarial samples. Inspired by this observation, we make a similar assumption that, in the DeepFool algorithm, directions with the largest gradient differences are more sensitive to inputs. Therefore, we enhanced the DeepFool algorithm by searching for the class that maximizes the gradient difference from the original sample's class, enabling faster generation of adversarial samples. Our enhancement leverages the maximization of gradient differences to expedite the generation of adversarial samples. This method exploits the sensitivity differences of the model across different classes, enhancing the targeted and effective generation of adversarial samples.

5.3. Limitations and Future Works

The proposed algorithm requires computing the class with the maximum gradient difference from the original sample class in each run to expedite the generation of adversarial samples. In datasets with fewer classes, we observed faster attack speeds and higher attack success rates. However, when dealing with datasets with a large number of classes, such as image datasets, the process of computing class gradient differences consumes a significant amount of time, resulting in slower attack speeds. We propose the introduction of precomputed gradients or gradient caching mechanisms. Before conducting adversarial attacks, we can perform a single forward and backward pass on samples in the dataset,

calculating and storing the gradient of each sample with respect to its correct class. During actual adversarial attack execution, we can directly utilize these precomputed gradients, eliminating the need for repeated forward and backward passes. In terms of adversarial sample detection, the algorithm generates more targeted and effective adversarial samples. Adversarial training, as a crucial defense mechanism, effectively enhances a model's resilience against adversarial attacks. In the future, we aim to further explore the field of adversarial training, integrating the proposed YOAO algorithm with existing techniques to strengthen the robustness and security of classifier models. Additionally, we plan to expand our work focus to the robustness and security of object detection models, which is equally critical. By applying our adversarial training approach to object detection models, we hope to enhance their performance when facing adversarial attacks and provide more reliable protection for practical application scenarios.

6. Conclusions

Adversarial attacks expose the latent vulnerabilities within artificial intelligence systems, compelling a reevaluation and enhancement of model robustness to ensure the reliability and security of deep learning models against malicious attacks. The "You Only Attack Once" (YOAO) algorithm, which is central to our research, exploits gradient differences to accurately pinpoint the most susceptible class and effectively craft adversarial examples using a minimal iteration approach. The algorithm plays a role in augmenting the accuracy and efficiency of adversarial attack strategies, as empirical data showcase its superior performance compared to the DeepFool algorithm across a variety of scenarios, particularly in resource-constrained environments. To comprehensively investigate the impact of the YOAO algorithm on classifiers under various conditions, we initially employed an underfitted ResNet model with an accuracy of 60% as the classifier in Figure 4, comparing the performance of the YOAO algorithm with that of the DeepFool algorithm. Under the condition of a single iteration, the attack success rate of the YOAO algorithm increased significantly from 24.93% to 94.0% compared to the DeepFool algorithm, achieving an increase of 69.07%. Furthermore, by adjusting the number of iterations to ensure an attack success rate of 99%, the YOAO algorithm was able to generate an adversarial sample in only 0.05 s, while the DeepFool algorithm required 0.25 s, indicating that the YOAO algorithm is 0.2 s faster than the DeepFool algorithm in generating an adversarial sample. Additionally, in Figure 5, it can be seen that it was tested using a well-trained ResNet classifier with an accuracy of 95.45%.Under the condition of a single iteration, the attack success rate of the YOAO algorithm increased from 16.81% to 38.61% compared to the DeepFool algorithm, with an increase of 21.8%. When the attack success rates of both algorithms approached 98%, the YOAO algorithm only needed 0.11 s to generate a adversarial sample, compared to the 0.23 s required by the DeepFool algorithm, making the YOAO algorithm 0.12 s faster in generating an adversarial sample. These enhancements are crucial for both defensive and attack strategies in a myriad of machine learning models.

This work further explores the relationship between classifier accuracy and the effectiveness of attacks, comparing the proposed algorithm with existing methods. The experimental results confirm the higher success rate and speed of the proposed approach. Additionally, visualizing data on the ImageNet dataset demonstrates the ability of the YOAO algorithm to target important features. Although YOAO can significantly accelerate the generation of adversarial samples, when generating corresponding adversarial samples for batch data, there is redundant computation of gradient differences, which can slow down the attack speed. In the future, we will explore the introduction of precomputed gradients or gradient caching mechanisms to address this issue. By directly utilizing these precomputed gradients, redundant calculations can be eliminated. In summary, this study

not only addresses the current challenges of the algorithm but also lays the foundation for further research, which is of great significance for enhancing the security of deep learning models and artificial intelligence.

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

Funding: This work is supported by the Jilin Province Budgetary Capital Construction Fund Project (No. 2024C008-4).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article, and further inquiries can be directed to the corresponding author.

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

References

1. Jumaatuden, D.M.H. ; Ali, M.A.M.; Isa, H.N. Balancing Efficiency and Effectiveness: Adversarial Example Generation in Pneumonia Detection. In Proceedings of the 2024 IEEE Symposium on Wireless Technology & Applications (ISWTA), Kuala Lumpur, Malaysia, 20–21 July 2024; pp. 206–210. [CrossRef]
2. Vyas, D.; Kapadia, V.V. Designing defensive techniques to handle adversarial attack on deep learning based model. *PeerJ Comput. Sci.* **2024**, *10*, e1868. [CrossRef]
3. Wang, C.; Rahman, A. Quantum-Enabled 6G Wireless Networks: Opportunities and Challenges. *IEEE Wirel. Commun.* **2022**, *29*, 58–69. [CrossRef]
4. Rozenman, G.G.; Kundu, N.K.; Liu, R.; Zhang, L.; Maslennikov, A.; Reches, Y.; Youm, H.Y. The quantum internet: A synergy of quantum information technologies and 6G networks. *IET Quantum Commun.* **2023**, *4*, 147–166. [CrossRef]
5. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. Deepfool: A simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2574–2582. Available online: https://openaccess.thecvf.com/content_cvpr_2016/html/Moosavi-Dezfooli_DeepFool_A_Simple_CVPR_2016_paper.html (accessed on 28 December 2024).
6. Chakraborty, A.; Alam, M.; Dey, V.; Chattopadhyay, A.; Mukhopadhyay, D. A survey on adversarial attacks and defences. *CAAI Trans. Intell. Technol.* **2021**, *6*, 25–45. [CrossRef]
7. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2015**, arXiv:1412.6572.
8. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial machine learning at scale. *arXiv* **2017**, arXiv:1611.01236.
9. Naseem, M.L. Trans-IFFT-FGSM: A novel fast gradient sign method for adversarial attacks. *Multimed. Tools Appl.* **2024**, *83*, 72279–72299. . [CrossRef]
10. Moosavi-Dezfooli, S.M.; Fawzi, A.; Fawzi, O.; Frossard, P. Universal adversarial perturbations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1765–1773. Available online: https://openaccess.thecvf.com/content_cvpr_2017/html/Moosavi-Dezfooli_Universal_Adversarial_Perturbations_CVPR_2017_paper.html (accessed on 28 December 2024).
11. Gajjar, S.; Hati, A.; Bhilare, S.; Mandal, S. Generating Targeted Adversarial Attacks and Assessing their Effectiveness in Fooling Deep Neural Networks. In Proceedings of the 2022 IEEE International Conference on Signal Processing and Communications (SPCOM), Bangalore, India, 11–15 July 2022; pp. 1–5. [CrossRef]
12. Labib, S.; Mondal, J.J.; Manab, M.A. Tailoring Adversarial Attacks on Deep Neural Networks for Targeted Class Manipulation Using DeepFool Algorithm. *arXiv* **2024**, arXiv: 2310.13019. http://arxiv.org/abs/2310.13019.
13. Ling, J.; Chen, X.; Luo, Y. Improving the transferability of adversarial samples with channel switching. *Appl. Intell.* **2023**, *53*, 30580–30592. [CrossRef]
14. Costa, J.C.; Roxo, T.; Proença, H.; Inácio, P.R. How deep learning sees the world: A survey on adversarial attacks & defenses. *IEEE Access* **2024**, *12*, 61113–61136. [CrossRef]

15. Deng, Y.; Mu, T. Understanding and improving ensemble adversarial defense. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 58075–58087. Available online: https://proceedings.neurips.cc/paper_files/paper/2023/file/b589d92785e39486e978fa273d0dc343-Paper-Conference.pdf (accessed on 28 December 2024).
16. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2014**, arXiv:1312.6199.
17. Qin, C.; Martens, J.; Gowal, S.; Krishnan, D.; Dvijotham, K.; Fawzi, A.; De, S.; Stanforth, R.; Kohli, P. Adversarial robustness through local linearization. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 13824–13833. Available online: https://proceedings.neurips.cc/paper_files/paper/2019/file/0defd533d51ed0a10c5c9dbf93ee78a5-Paper.pdf (accessed on 28 December 2024).
18. Najafi, A.; Maeda, S.I.; Koyama, M.; Miyato, T. Robustness to adversarial perturbations in learning from incomplete data. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 5541–5551.
Available online: https://proceedings.neurips.cc/paper_files/paper/2019/file/60ad83801910ec976590f69f638e0d6d-Paper.pdf (accessed on 28 December 2024).
19. Kuang, H.; Liu, H.; Lin, X.; Ji, R. Defense against adversarial attacks using topology aligning adversarial training. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 3659–3673. . [CrossRef]
20. Shi, C.; Liu, Y.; Zhao, M.; Pun, C.M.; Miao, Q. Attack-invariant attention feature for adversarial defense in hyperspectral image classification. *Pattern Recognit.* **2024**, *145*, 109955. [CrossRef]
21. Jung, S.; Chung, M.; Shin, Y.G. Adversarial example denoising and detection based on the consistency between Fourier-transformed layers. *Neurocomputing* **2024**, *606*, 128351. [CrossRef]
22. Shi, L.; Liu, W. Adversarial self-training improves robustness and generalization for gradual domain adaptation. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 37321–37333. Available online: https://proceedings.neurips.cc/paper_files/paper/2023/file/75b0edb869e2cd509d64d0e8ff446bc1-Paper-Conference.pdf (accessed on 28 December 2024).
23. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
24. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; University of Toronto: Toronto, ON, Canada, 2009; pp. 1–58 .
25. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
26. Mądry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. *Stat* **2017**, *1050*, 1–27.
27. Kurakin, A.; Goodfellow, I.J.; Bengio, S. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2018; pp. 99–112. Available online: https://www.taylorfrancis.com/chapters/edit/10.1201/9781351251389-8/adversarial-examples-physical-world-alexey-kurakin-ian-goodfellow-samy-bengio (accessed on 28 December 2024).
28. Wong, E.; Rice, L.; Kolter, J.Z. Fast is better than free: Revisiting adversarial training. *arXiv* **2020**, arXiv:2001.03994.
29. Croce, F.; Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 2206–2216. Available online: https://proceedings.mlr.press/v119/croce20b.html (accessed on 28 December 2024).
30. Modas, A.; Moosavi-Dezfooli, S.M.; Frossard, P. Sparsefool: A few pixels make a big difference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9087–9096. Available online: https://openaccess.thecvf.com/content_CVPR_2019/html/Modas_SparseFool_A_Few_Pixels_Make_a_Big_Difference_CVPR_2019_paper.html (accessed on 28 December 2024).
31. Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841. [CrossRef]
32. Pomponi, J.; Scardapane, S.; Uncini, A. Pixle: A fast and effective black-box attack based on rearranging pixels. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–7. [CrossRef]
33. Gao, L.; Zhang, Q.; Song, J.; Shen, H.T. Patch-wise++ perturbation for adversarial targeted attacks. *arXiv* **2020**, arXiv:2012.15503.

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.