

## Article

# ABCAttack: A Gradient-Free Optimization Black-Box Attack for Fooling Deep Image Classifiers

Han Cao <sup>1</sup>, Chengxiang Si <sup>2</sup>, Qindong Sun <sup>1,3,\*</sup>, Yanxiao Liu <sup>1</sup>, Shancang Li <sup>4</sup> and Prosanta Gope <sup>5</sup>

- <sup>1</sup> Key Laboratory of Network Computing and Security, Xi'an University of Technology, Xi'an 710048, China; caohan@stu.xaut.edu.cn (H.C.); liuyanxiao@xaut.edu.cn (Y.L.)
- <sup>2</sup> National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC), Beijing 100029, China; sichengxiang@cert.org.cn
- <sup>3</sup> School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China
- <sup>4</sup> The Department of Computer Science and Creative Technology, University of the West of England, Bristol BS16 1QY, UK; Shancang.Li@ieee.org
- <sup>5</sup> Department of Computer Science, University of Sheffield, Sheffield S10 2TN, UK; Prosanta.nitdgp@gmail.com
- \* Correspondence: qdongsun@xjtu.edu.cn

**Abstract:** The vulnerability of deep neural network (DNN)-based systems makes them susceptible to adversarial perturbation and may cause classification task failure. In this work, we propose an adversarial attack model using the Artificial Bee Colony (ABC) algorithm to generate adversarial samples without the need for a further gradient evaluation and training of the substitute model, which can further improve the chance of task failure caused by adversarial perturbation. In untargeted attacks, the proposed method obtained 100%, 98.6%, and 90.00% success rates on the MNIST, CIFAR-10 and ImageNet datasets, respectively. The experimental results show that the proposed ABCAttack can not only obtain a high attack success rate with fewer queries in the black-box setting, but also break some existing defenses to a large extent, and is not limited by model structure or size, which provides further research directions for deep learning evasion attacks and defenses.

**Keywords:** deep neural networks; adversarial examples; image classification; information security; black-box attack



**Citation:** Cao, H.; Si, C.; Sun, Q.; Liu, Y.; Li, S.; Gope, P. ABCAttack: A Gradient-Free Optimization Black-Box Attack for Fooling Deep Image Classifiers. *Entropy* **2022**, *24*, 412. <https://doi.org/10.3390/e24030412>

Academic Editor: Gholamreza Anbarjafari

Received: 16 February 2022

Accepted: 12 March 2022

Published: 15 March 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 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 recent years, deep learning has developed rapidly and successfully applied to a number of complex problems such as image classification [1] and traffic classification [2]. However many studies on adversarial examples [3,4] have proclaimed the vulnerability of DNN models, which brings great security risks to the deployment of DNN models in practical applications. Even in a DNN-based system with communication security defenses [5], attackers can break it by adding adversarial perturbations to the transmitted data in advance [6,7]. For example, when a DNN model is used for traffic sign recognition in an automatic driving system, attackers can maliciously modify the captured road sign image, causing the model to give false inference, which leads to the decision system making wrong judgments. It greatly increases the probability of safety accidents. Therefore, the security problems caused by adversarial samples in machine learning system cannot be ignored.

The adversarial sample refers to a synthetic sample formed by adding subtle perturbation to the original sample, and the visual difference between the adversarial sample and the original sample is imperceptible to humans. The concept of adversarial samples was first proposed by Szegedy et al. [4]. Adversarial samples have a significant research space in the field of computer vision and images [8]. The methods of generating adversarial samples can be divided into white-box and black-box. The white-box attacks require the attacker to have full knowledge of the target model, such as parameters, structure, types,

objective functions, etc. The clean sample can be modified by calculating the true gradient of the victim model to generate the adversarial image. The black-box attacks mean that the attacker can only access the model input and feedback. In the real world, the assumption of white-box attacks is difficult to satisfy. Therefore, researchers have explored black-box attacks.

At present, the existing black-box attacks are to break the target model with adversarial samples carefully crafted on the substitute model [9], gradient evaluation [10] or swarm intelligence. However, generating adversarial examples on the substitute model to attack the target model actually builds a remote hosting system, but requires access to the target model to gather a dataset to train the substitute model. These attacks add extra overhead for gathering data and training a substitute model. After that, the adversarial samples generated by the white-box attacks on the substitute model are used to attack the victim model which is based on the transferability of adversarial examples. However, the success rate is relatively low, especially in targeted attacks, and evaluating the gradient using finite difference requires a higher number of queries and computational cost. Attacks adopting warm intelligence can perform in different DNN models because these do not rely on model structure and other detailed information, and the crafted adversarial samples are random, which can break some of the existing state-of-the-art defense strategies to a certain extent [11,12].

Motivated by the above, we propose a black-box adversarial attack using Artificial Bee Colony (ABC), called ABCAttack. It formalizes the adversarial sample generation problem into an optimization problem, and uses the ABC algorithm to solve it. This method needs no gradient information in the process of generating the adversarial samples, so it is gradient-free. The performance of the algorithm is evaluated on MNIST, CIFAR-10 and ImageNet datasets. The results show that ABCAttack can obtain a better attack success rate with fewer queries. In addition, we also found that classes with similar appearance characteristics are more likely to be crafted to each other. Due to its gradient-free nature, ABCAttack is robust to some defenses. We performed untargeted attacks on models which have different structures and sizes, and it can still achieve a high success rate with fewer queries. This is because the process of generating adversarial samples does not rely on gradient calculation or gradient evaluation. Therefore, it has a strong applicability. The main contributions are summarized as:

1. Aiming at reducing the reliance of white-box attacks on the attacker's knowledge and overcoming the shortcomings of black-box attacks that evaluate gradients and train a substitute model to generate adversarial samples, a new black-box adversarial sample generation method, ABCAttack, is proposed.
2. ABCAttack transforms adversarial sample generation into an optimization problem by using an improved search strategy to continuously iterate to obtain the adversarial image that is adversarial samples if successfully attacking the targeted model. Adversarial samples can be generated only by accessing the input and softmax probability of the model, without other detailed information, so gradient evaluation and training a substitute model are avoided, which can effectively improve the generation efficiency of adversarial samples.
3. We evaluated the proposed ABCAttack using existing datasets: MNIST, CIFAR-10 and ImageNet. The results demonstrate the effectiveness and efficiency under the black-box setting, that is, in both targeted and untargeted attacks, ABCAttack uses fewer queries and lower time consumption to generate adversarial samples, which destroys the trained DNN models and greatly reduces its credibility.
4. We further highlight the effectiveness of ABCAttack in defenses, namely defence-Gan, Stochastic Activation Pruning (SAP), Local Intrinsic Dimensionality (LID), non-differentiable input transformations and others. Although these defenses claim to be robust, our attack can still greatly reduce the accuracy of defense models. ABCAttack is gradient-free, so it has wide applicability, which means that as long as we obtain

the input and feedback, we can successfully craft adversarial examples without caring about the specific details of the model.

In Section 2, we describe the work related to adversarial samples, and Section 3 introduces the standard ABC algorithm. Section 4 formalizes the description of adversarial samples and discusses the specific details of our proposed algorithm. Section 5 presents the experimental results and uses multiple indicators to evaluate the performance of the algorithm. Section 6 discusses the effect of population size and the maximum amount of change allowed by adversarial perturbation on attack performance. Finally, we summarize the work of this paper and look forward to the future work in Section 7.

## 2. Related Work

The emergence of adversarial samples provides an alarm for the security of ML systems and promotes research on both attacks and defenses. The difference between white-box and black-box attacks is that in a white-box setting, the attacker has all the knowledge of the target model.

Szegedy et al. [4] proposed L-BFGS, which uses box-constrained optimization to generate adversarial examples, and pointed out that adversarial examples are transferable, that is, in two models that perform the same task, the adversarial samples that affect one model usually affect the other model, even if the two models have different architectures or are trained on non-duplicate training sets. Goodfellow et al. [13] described the rationale of adversarial samples, explaining that the high-dimensional linearity of deep neural networks is the reason for the existence of adversarial samples, and proposed the Fast Gradient Symbol Algorithm (FGSM). Kurakin et al. [7] proposed I-FGSM on the basis of FGSM, and used a cell phone camera to take an image of a printed adversarial sample as the input of the image classification neural network. As a result, most of the captured images were still misclassified, which confirms the existence of adversarial samples in the real world. Nguyen et al. [14] used evolutionary algorithms or gradient ascent algorithms on a successfully trained CNN model to find special images that humans cannot recognize but the DNN classifies with high confidence. Moosavi-Dezfooli et al. [15] proposed an untargeted attack, DeepFool, which can calculate a smaller perturbation than FGSM. JSMA [16], as a typical  $L_0$ -norm white-box attack, introduced Saliency Map [17], which seeks to minimize the number of pixels modified. C&W [18] is generally regarded as one of the most powerful white-box attacks, supporting  $L_0$ -norm,  $L_2$ -norm and  $L_\infty$ -norm attacks, which break the effectiveness of defensive distillation [19] for most attacks.

In the black-box setting, the attacker can only access the input and feedback. In view of the transfer nature of adversarial samples, Papernot et al. trained a local model to substitute for the target DNN, using synthetic inputs generated by an attacker and labeled by the target DNN [9]. Then, they introduced reservoir sampling to greatly improve the accuracy and reduce the computational cost of the substitute learning technique [20]. Chen et al. [10] only accessed the model input and feedback information to evaluate the gradient using the finite difference, but a lot of queries are required. Bhagoji et al. [21] also used the finite difference to evaluate the gradient for black-box attacks, and Principal Component Analysis (PCA) and random grouping were introduced to reduce queries. Narodytska et al. [22] randomly selected a single pixel or a small number of pixels to add perturbation, and then used the idea of greedy local search to craft a small group of pixels to improve the attack efficiency. Brendel et al. [23] proposed a decision-based black-box attack, starting with a large adversarial perturbation, and then sought to reduce the perturbation while maintaining the adversarial nature. Ilyas et al. [24] defined three realistic threat models: the query-limited setting, the partial information setting and the label-only setting, and developed new attacks under these restrictive threat models, which demonstrates that ML systems are still vulnerable to attacks even under limited queries and information.

In addition, Su et al. [25,26] merely modified a single pixel of the original image using differential evolution to attack. GenAttack [11], AdversarialPSO [27,28] used genetic and particle swarm optimization algorithms, respectively, to make adversarial samples. Both

attacks greatly reduce the number of queries. In addition, Liu et al. [12] put forward a method of crafting adversarial samples based on the population evolutionary algorithm, which is resistant to defense distillation. It is found that the adversarial samples reproduce the features of the data learned by the DNN model to a certain extent.

Recently, Moosavi-Dezfooli et al. [29] and Mopuri et al. [30] proved the existence of universal adversarial perturbations, which are both network-agnostic and image-agnostic. With the studies on adversarial samples, defense algorithms have also emerged, which hope to improve the robustness of the model adversarial samples [31–34].

### 3. ABC

Artificial bee colony is a meta-heuristic swarm intelligence optimization algorithm proposed by Professor *Karaboga* in 2005 [35]. It is inspired by the principles of a bee colony's foraging behavior through information sharing and communication. A bee colony collects nectar efficiently and collaboratively. In the process, Employed Bees (EBs) collect the nectar and return to the hive and share information with onlooker bees through dance. Onlooker Bees (OBs) choose the nectar according to the observed dance. Under certain conditions, the employed bee is transformed into a Scout Bee (SB), which randomly searches for new nectar near the hive. This enables the entire bee colony to complete the work of collecting honey in a collaborative manner efficiently. Inspired by this, the minimum search model for a bee colony to realize swarm intelligence includes four elements: food source, employed bees, onlooker bees and scout bees and, among them, OBs and SBs are called unemployed bees. Each food source is a candidate solution to the optimization problem being solved. If the number of food sources is  $SN$ , the numbers of EBs and OBs are both  $SN/2$ . Under special circumstances, the individuals in the colony will transform into SBs to search for a new food source. Suppose the dimension of the problem to be solved is  $d$ , and the position of the food source  $i$  in the  $t$  iteration is  $X_i^t = [x_{i1}^t, x_{i2}^t, \dots, x_{id}^t]$ . The main steps of the ABC algorithm are as follows:

**Step 1: Initialization.** The ABC algorithm uses Equation (1) to initialize  $EN$  food sources, and calculate the fitness for every food source.

$$x_{ij} = x_{ij}^{\min} + \text{rand}(0, 1) \times (x_{ij}^{\max} - x_{ij}^{\min}) \quad (1)$$

where  $x_{ij}$  is the component of food source  $i$  in the  $j$ -th dimension,  $i \in \{0, 1, 2, \dots, EN - 1\}$ ,  $j \in \{0, 1, 2, \dots, d - 1\}$ .  $x_{ij}^{\min}$  and  $x_{ij}^{\max}$  are lower limit and upper limit of the  $j$ -th component, respectively.  $\text{rand}(0, 1)$  represents a floating-point random number obtained randomly in the range of  $[0, 1]$ .

**Step 2: Sending EBs.** Each EB exploits a new food source according to Equation (2),

$$x_{ij}^{t+1} = x_{ij}^t + \varphi_{ij} (x_{ij}^t - x_{kj}^t) \quad (2)$$

where  $i, k \in \{0, 1, 2, \dots, EN - 1\}$ ,  $k \neq i$ ,  $\varphi_{ij}$  is a floating-point random number in the range  $[-1, 1]$ . After that, the fitness value is calculated for the new food source, and each  $x_i^{t+1}$  is compared with the corresponding  $x_i^t$  to keep the better one. It is judged whether to update the feasible solution.

**Step 3: Sending OBs.** EBs share the information with OBs, and the OBs selectively mine new food sources according to probability.

$$\text{fitness}_i^{t+1} = \begin{cases} \frac{1}{1+g_i^{t+1}} & g_i^{t+1} \geq 0 \\ 1 + |g_i^{t+1}| & g_i^{t+1} < 0 \end{cases} \quad (3)$$

$$P_i^{t+1} = \frac{\text{fitness}_i^{t+1}}{\sum_{i=0}^{EN-1} \text{fitness}_i^{t+1}} \quad (4)$$

In Equation (4),  $fitness_i^{t+1}$  is the fitness value of the food source in the generation  $t + 1$ ,  $P_i^{t+1}$  is probability of the  $i$ -th solution and  $g_i$  is an objective function. The food sources to be exploited are selected according to  $P_i$ , the search formula is the same as step 2. After that, the remaining procedures are the same as sending EBs.

**Step 4: Sending SBs.** Assuming the quality of the food source has not improved after reaching the limit evaluation, the corresponding EB becomes a SB, the current food source is abandoned and a new food source is generated according to Equation (1).

The ABC algorithm is to continuously mine and explore the food sources by bees with three different responsibilities, iteratively executing steps 2 to 4 to obtain the final solution.

## 4. Methodology

### 4.1. Adversarial Sample

The adversarial sample can be formally described as follows:

$$x_{adv} = x + \delta \quad (5)$$

where  $x$  is the original image, and it can be classified as true label  $y$  by  $f$ .  $\delta$  is the perturbation,  $x_{adv}$  is a carefully crafted adversarial sample.

The attacks can be divided into targeted and untargeted attacks according to whether the attacker specifies the model's prediction label for the adversarial sample. Given the classifier  $f(x)$  and the original sample  $x$ , the goal of the untargeted attack is for  $x_{adv}$  to be classified by  $f(x)$  into any other label except  $y$ , that is,  $f(x_{adv}) \neq f(x) = y$ . The targeted attack is to specify the prediction label  $y^*$  of the  $x_{adv}$  before the algorithm runs, i.e.,  $f(x_{adv}) = y^*$ . In contrast, targeted attacks are more difficult.  $x_{adv}$  is an adversarial sample made by the attacker on the model  $f_1(x)$ . Typically, it still has an attack effect for another model  $f_2(x)$  that performs the same task as  $f_1(x)$ , which is called the transferability of adversarial samples.

On the basis of a successfully attack, in order to ensure the information invariance of the whole vision, that is, a human can correctly classify the adversarial samples, the attack algorithms will limit the perturbations added. Generally, the following formula is used to limit the perturbation:

$$L_p = \|x - x_{adv}\|_p \quad (6)$$

1. When  $p = 0$ , it is the  $L_0$ -norm of the matrix, indicating the number of non-zero elements in a vector. This constraint limits the dimension of the modified vector, but there is no limit on the modification range of each dimension when the adversarial sample is crafted.
2. When  $p = 2$ , it is the  $L_2$ -norm of the matrix, representing the Euclidean distance between the original image and the adversarial sample. This restriction allows the algorithm to modify all dimensions, but limits the magnitude.
3. When  $p = \infty$ , it is the  $L_\infty$ -norm of the matrix, which represents the maximum amount of modification between the original sample and adversarial sample, that is, the maximum absolute value of the added perturbation, which can modify all dimensions.

### 4.2. Problem Description

The essence of adversarial example generation is to find the adjacent images that match the target within the clean image neighborhood. In this study, ABC is used to continuously search for better candidate solutions in the search space by minimizing the fitness function until an adversarial sample is found or termination condition is reached.

We consider the generation of adversarial examples in a black-box setting. The attacker does not have any knowledge of the model and can only access input and softmax output. Assuming that the target classification model  $f : \mathbb{R}^d \rightarrow [0, 1]^k$ , where  $d$  is the input dimension and  $k$  is the output dimension and is also the number of classes, and  $f$  returns the probability of each label. Assuming that the original image  $\mathbf{x} = \{x_1, x_2, \dots, x_d\}$  can be

correctly classified as true label  $y$  by the model  $f$ . The purpose of the attack is to find the perturbed vector  $\mathbf{x}_{adv}$ , and the prediction label of the  $\mathbf{x}_{adv}$  is not  $y$  for untargeted attacks, or equal to the pre-defined target label  $y^*$  for targeted attacks. In addition, the attacker should constrain the  $L_p$  distance between clean the adversarial samples as much as possible to ensure that the visual difference between them is as imperceptible as possible. In this paper, we limit the  $L_\infty$  distance.

For untargeted attacks:

$$\arg \max_{c \in \{1, \dots, k\}} f(\mathbf{x}_{adv})_c \neq y \text{ such that } \|\mathbf{x} - \mathbf{x}_{adv}\|_\infty \leq \delta \tag{7}$$

For targeted attacks, set  $y^*$  as the target label:

$$\arg \max_{c \in \{1, \dots, k\}} f(\mathbf{x}_{adv})_c = y^* \text{ such that } \|\mathbf{x} - \mathbf{x}_{adv}\|_\infty \leq \delta \tag{8}$$

### 4.3. ABCAttack Algorithm

ABC is used in many fields to solve optimization problems. This paper introduces ABCAttack that relies on ABC to solve the optimization problems of Equations (7) and (8) to craft adversarial images. The process of ABC to solve the problem is mainly initializing, exploiting EBs, calculating selection probability and exploiting OBs and SBs to iterate continuously to find the adversarial image. Assuming that  $\mathbf{x}$  and  $\mathbf{x}_{adv}$  are the clean image and the corresponding adversarial image, respectively,  $\mathbf{x}$  is correctly classified as the real label  $y$ , and the target label pre-defined by the attacker is  $y^*$ . As shown in Figure 1, ABCAttack takes the original image as the input. According to the input domain of the target model, the clean image is first mapped to the input space of the model. It is assumed that the input space range of the model is  $[-0.5, 0.5]$ , therefore, we use  $\mathbf{x}' = (\mathbf{x}/255.0 - 0.5)$  to complete the normalization, and add random disturbance to  $\mathbf{x}'$  to initialize the food source. After that, we iteratively optimize to obtain the adversarial sample. After the attack is successful, we then map it to the RGB space to observe the visual difference from the clean image. The algorithm will terminate the loop in advance when the adversarial sample is crafted successfully. The adversarial sample generation will fail if the maximum for queries is reached.

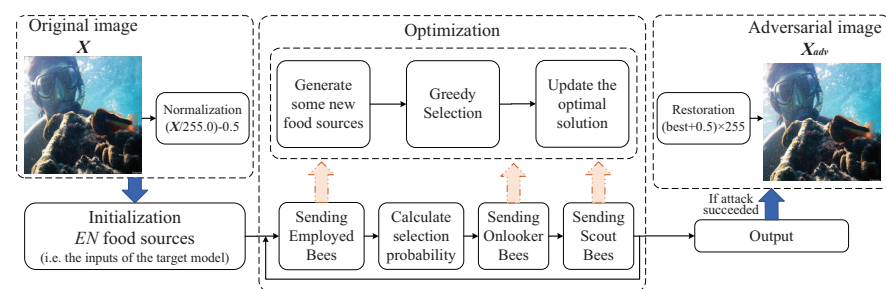


Figure 1. The process of adversarial attacks using ABCAttack.

In order to further control the change range of the original image during the algorithm iteration process, we define the maximum change  $\delta_{max}$ . ABCAttack crafts adversarial samples under the  $L_\infty$ -norm. The  $L_\infty$ -norm is defined as the maximum change in  $\mathbf{x}_{adv}$  relative to  $\mathbf{x}$ , expressed as  $\delta_{max} = \max(|x_1 - x_{adv}^1|, |x_2 - x_{adv}^2|, \dots, |x_d - x_{adv}^d|)$ .

We use the clip operation to limit changes in each candidate solution. If we set  $\delta_{max} = B$ , the upper and lower limits of the original image are  $UP$  and  $LP$ , respectively, and the next candidate solution is  $\mathbf{x}^{t+1} = clipByTensor(\mathbf{x}^{t+1}, clip(\mathbf{x} - B, LP, UP), clip(\mathbf{x} + B, LP, UP))$ , where  $clip$  is the truncation function  $clip(vector, min, max)$ , and its purpose is to assign  $min$  to elements smaller than  $min$  in the vector, and assign  $max$  to elements larger than  $max$ .

For each image, we use Algorithm 1 to generate its adversarial sample. In ABCAttack, we respectively define fitness functions for targeted and untargeted attacks. In untargeted

attacks, the object function  $g(\cdot)$  is the probability  $p_{cand}^y$  that the original image is predicted as true label  $y$ . In targeted attacks, the object function  $g(\cdot)$  is  $1 - p_{cand}^{y^*}$ , where  $p_{cand}^{y^*}$  is the probability of the pre-defined target label  $y^*$  of the current candidate solution. In ABCAttack, the ranges of  $p_{cand}^y$  and  $1 - p_{cand}^{y^*}$  are both  $(0, 1)$ , and there is no case where  $g(\cdot)$  is less than 0, so we choose the fitness function as:

$$\text{calFitness}(\mathbf{x}_{cand}) = g(\mathbf{x}_{cand}) = \begin{cases} p_{cand}^y, & \text{untargeted} \\ 1 - p_{cand}^{y^*}, & \text{targeted} \end{cases} \quad (9)$$

---

**Algorithm 1** Adversarial attack-based ABCAttack.

---

**Input:** original image  $\mathbf{x}$ , the true label  $y$  and its probability  $prob_{ori}$ , target label  $y^*$ , number of food source  $SN$ , Limit number  $l$ , max change  $\delta_{max}$ , the initialization amplitude limit factor  $\alpha$ , upper bounds  $UP$  and lower bounds  $LP$ .

**Output:** *BestSolution*

```

1: Poputions  $\leftarrow$  []
2: BestSolution, bestFit  $\leftarrow$   $\mathbf{x}$ , 100,000
3: aMin, aMax  $\leftarrow$  clip( $\mathbf{x} - \delta_{max}$ , LP, UP), clip( $\mathbf{x} + \delta_{max}$ , LP, UP)
4: EN  $\leftarrow$   $SN/2$ 
   // Initializing
5: for  $i = 0$  to  $EN - 1$  do
6:   Poputions[ $i$ ] =  $\mathbf{x} + \mathcal{U}(-\alpha \times \delta_{max}, \alpha \times \delta_{max})$ 
7:   Poputions[ $i$ ] = clipByTensor(Poputions[ $i$ ], aMin, aMax)
8:   Poputions[ $i$ ].fitness = calFitness(Poputions[ $i$ ])
9:   updateBestSolution(Poputions[ $i$ ], BestSolution)
   // Optimization
10: while stopOption() do
11:   for  $i = 0$  to  $EN - 1$  do
12:     next = clipByTensor(searchStrategy( $i$ , 1), aMin, aMax)
13:     fitness = calFitness(next)
14:     greedySelection()
15:     updateBestSolution(next, BestSolution)
     Poputions.p = selectProbabilities()
16:   for  $i = 0$  to  $EN - 1$  do
17:     if Poputions[ $i$ ].p > rand(0, 1) then
18:       next = clipByTensor(searchStrategy( $i$ , 2), aMin, aMax)
19:       fitness = calFitness(next)
20:       greedySelection()
21:       updateBestSolution(next, BestSolution)
22:   sendScoutBees()
23:   updateBestSolution(Poputions, BestSolution)
return BestSolution

```

---

Therefore, the fitness value for targeted and untargeted attacks decreases as the cycle goes on, and the smaller the fitness, the better the current solution.

It can be seen from Algorithm 1 that ABCAttack consists of two main parts: initialization and optimization. The *searchStrategy()* is the implementation of search strategy. The *calFitness()* function calculates the fitness of the candidate solution by Equation (9). The *updateBestSolution()* updates the current optimal solution, and *stopOption()* is a function that judges whether the current cycle is stopped according to the termination conditions. The termination conditions include the successful generation of adversarial samples and queries reaching the maximum. The specific details of the algorithm are as follows.

**(1) Initialization**

In this paper, the initialization of the standard ABC algorithm (Equation (1)) is not used, but the candidate solutions are initialized by uniform distribution, as shown in Equation (10):

$$\mathbf{x}_i^0 = \mathbf{x} + \mathcal{U}(-a \times \delta_{\max}, a \times \delta_{\max})(i = 1, 2, \dots, EN) \tag{10}$$

Define the number of EBs and OBs  $EN$ , the maximum modification  $\delta_{\max}$ , that is,  $L_{\infty} = \delta_{\max}$ , and the initialization amplitude limiting factor  $a$ .

**(2) Optimization**

Iteratively optimizing solving adversarial examples using ABCAttack can be divided into the following stages: sending EBs, probability calculation, sending OBs and sending SBs. In the basic ABC algorithm, the search strategies of the EBs and the SBs are the same, both of which are Equation (2). In this paper, we improve the search strategy of the EBs and OBs to enhance the optimization ability and convergence speed.

Sending EBs stage: According to the ratio of the fitness of the current optimal solution to probability of the original image classified into the true label, the candidate solutions are divided into two parts,  $S_1$  and  $S_2$ , and different search strategies are used for generating new candidate solutions. The specific division method is as follows:

$$N = \max\left((EN - 1) \times \frac{bestFit}{prob_{ori}} - 1, 1\right) \tag{11}$$

$$S_1, S_2 = S[: N], S[N :] \tag{12}$$

The search method of the candidate solution subset  $S_1$  is the same as Equation (2), and the candidate solution subset  $S_2$  is applied to Equation (13) to generate next generation candidate solutions, which use the current optimal solution to guide optimization:

$$\mathbf{x}_i^{next} = S^{best} + \varphi_i(\mathbf{x}_i^t - \mathbf{x}_k^t), (i \in N, \dots, EN, k \neq i) \tag{13}$$

It can be seen that in the initial iteration of the algorithm, assuming that there is an untargeted attack, and  $bestFit = prob_{ori}$ , the number of EBs that use the original search strategy in Equation (2) to exploit is  $EN - 2$ , and the number of EBs that use the current optimal solution to guide the exploitation is 2. As the algorithm runs,  $bestFit$  gradually decreases with the increase in iterations, the number of EBs using the original search strategy in Equation (2) to exploit is gradually reduced and using optimal guided searches will increase.

After generating the next generation candidate solution, the fitness is calculated and the final next generation candidate solution based on greedy selection in Equation (14) is determined. If the candidate solution is replaced, its *trial* is set to 1, otherwise it is trial plus 1. This is the *greedySelection()* in Algorithm 1. Equation (15) is applied to update the current optimal solution if the  $fit_i^{t+1}$  is less than  $bestFit$ .

$$\mathbf{x}_i^{t+1}, trial_i^{t+1} = \begin{cases} \mathbf{x}_i^t, trial_i^t + 1, & \text{if } fit_i^{next} > fit_i^t \\ \mathbf{x}_i^{next}, 1, & \text{if } fit_i^{next} \leq fit_i^t \end{cases} \tag{14}$$

$$bestFit = \begin{cases} bestFit, & \text{if } fit_i^{t+1} > bestFit \\ fit_i^{t+1}, & \text{if } fit_i^{t+1} \leq bestFit \end{cases} \tag{15}$$

Probability calculation and sending OBs stage: Calculate the selection probability for each candidate solution according to Equation (4), that is, the *selectProbabilities()* function in Algorithm 1. Select the candidate solution that satisfies the  $rand(0, 1) < Popution[i].p$  condition for further exploitation. First, randomly select  $(rated)/2 + 1$  elements with the probability rate for adaptive optimal guided local optimization. The *rate* can be obtained from Equation (16):

$$rate = \max\left(\frac{MaxEvaluation - evalCount}{MaxEvaluation}, 0\right) \tag{16}$$



where  $MaxEvaluation$  is the maximum queries,  $evalCount$  represents queries currently used. The  $rate$  shows a decreasing trend as the algorithm continues to iterate. Therefore, when the current optimal solution is closer to the attack target, the number of modified pixels should be reduced as much as possible to enhance the ability of local optimization. Let  $Points$  be the selected elements indexes set, and use the following equation to search new solutions for the sending OBs stage:

$$\mathbf{x}_i^{next}[Points] = S^{best}[Points] + \varphi_i \times (\mathbf{x}_i^t[Points] - \mathbf{x}_k^t[Points]), i \neq k. \quad (17)$$

Then, calculating the corresponding fitness, carry out greedy selection according to Equation (14), and update the current optimal solution as seen in Equation (15).

**Sending SBs stage:** The numbers of evaluations of all current candidate solutions are checked. If the  $trial$  corresponding to a certain candidate solution is greater than the  $limitTrial$ , the current candidate solution is discarded, and a new candidate solution is generated by Equation (10).

The above steps are carried out iteratively until the adversarial sample is crafted or the termination condition of the algorithm is reached. Equation (2) generates a new food source near the randomly selected individuals, which has strong global search ability and can maintain the diversity of the population, but the convergence speed is slow. In the optimal solution guided search strategy, new food sources are generated near the optimal individual, which has strong local search ability, but it easily falls into local optimization. In the EB search stage, this paper adopts the combination of the original ABC search strategy (Equation (2)) and the search strategy guided by the current optimal solution (Equation (13)), which can ensure adjusting the population diversity and search ability according to the current optimal solution. We also compare the hybrid search strategy in this paper with the search strategy using only Equation (2) or Equation (13) on MNIST and CIFAR-10 datasets. With the same parameters, whether an untargeted or targeted attack, we prove that the algorithm can effectively improve the convergence speed, and the maximum reduction in average queries is more than 300, although this is limited. However, for lack of space, we will not show the specific experimental results in the experimental part.

## 5. Experimental Verification and Result Analysis

### 5.1. Experimental Dataset and Environment

We use MNIST [36], CIFAR-10 [37] and ImageNet [38] datasets to do comparative experiments to verify the performance of ABCAttack in ZOO [10], C&W [18], AdversitiaoPSO [27], GenAttack [11] and other methods. The models used by MNIST and CIFAR-10 are the same as those used in ZOO, C&W, AdversitiaoPSO and GenAttack. The specific details of the models can be found in [18], and their accuracies are 99.51% and 78.91% on the test datasets for MNIST and CIFAR-10, respectively. ImageNet uses the InceptionV3 model as the target attack model.

The experimental data used in this paper are the same as AdversitiaoPSO for MNIST and CIFAR-10. Under untargeted attacks, the first 1000 correctly predicted test samples in MNIST and CIFAR-10 are used to generate adversarial samples, while under targeted attacks, the first 111 test samples with correct prediction are used to attack. For one image, the pre-specified label of the targeted attack is set to all labels except the real label, and nine targeted attacks are performed. Therefore, the total number of targeted attacks is  $111 \times 9 = 999$ . The ImageNet dataset has 1000 classes. In this paper, 500 random samples with correct classifications are selected for attack. The following experiments on the MNIST and CIFAR-10 datasets were completed on a desktop computer with Intel Core i5-8400 @2.80 GHz CPU and 16 GB RAM, and for the experiments on the ImageNet, we completed the experiments on a machine with Intel Core i7-7700 @3.60 GHz  $\times$  8 CPU and GeForce RTX 1080Ti GPU.

In order to evaluate the performance of the algorithm, this paper introduces several metrics to measure the effectiveness of ABCAttack:

Attack success rate: *ASR* refers to the proportion of successful adversarial samples in all samples.

Average queries: We record an image individually input into the model as a query. The query count statistics are calculated only for successful attacks. The *AvgQueries* refers to the sum of the queries required for each successful adversarial example divided by the number of successful adversarial examples.

At the same time, we also calculated the  $L_2$ -norm of the adversarial perturbation, and also aimed to calculate the *AvgL<sub>2</sub>* on the successful sample. However, we pay more attention to the *AvgQueries* and the *ASR*.

### 5.2. Parameter Analysis

This attack first determines the number of food sources *SN*. We set the number of food sources to start from 10, with a step size of 2 to 30, perform untargeted attacks on the MNIST, CIFAR-10 and ImageNet datasets and observe the *AvgQueries*. We first describe the attack success rate, that when  $SN = 10$  and  $SN = 12$ , respectively, the attack success rate is small, 92.9% and 99.5% on MNIST ( $\alpha = 1.0$ ), 94.7% and 98.5% on CIFAR-10 ( $\alpha = 1.0$ ), 76.67% and 84.67% on ImageNet. When  $SN \geq 14$ , the attack success rate of the algorithm tends to be stable. After that, the accuracy will fluctuate slightly as a result of some random factors in the algorithm. This is a normal phenomenon, because the algorithm contains randomness, but the change in accuracy is very slight. For example, the success rate on MNIST is between 99.9% and 100%. Figure 2 shows the relationship between *SN* and *AvgQueries* on MNIST, CIFAR-10 and ImageNet datasets.

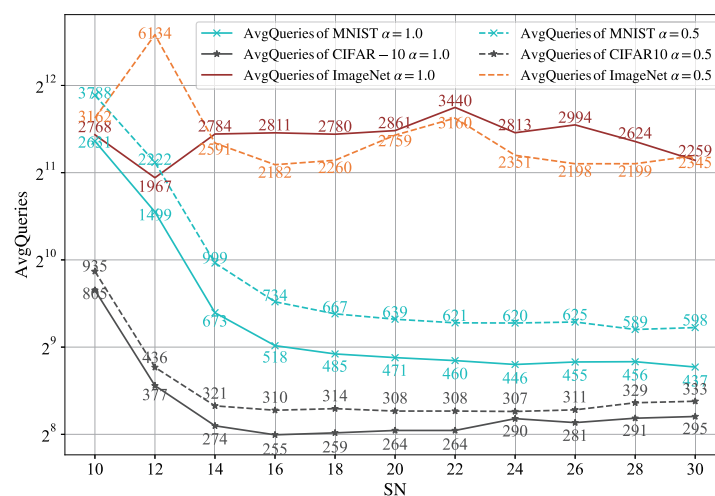


Figure 2. The relationship between the number of food sources and average queries.

It can be seen from the Figure 2 that, at the beginning, the *AvgQueries* decrease with the increase in *SN*. However, when it increases to a certain value, the *AvgQueries* increase slowly. When  $\alpha = 1.0$ , the optimal number of food sources is 30, 16 and 12 and the *ASR* is 99.9% 98.5% and 84.67% and, when  $\alpha = 0.5$ , they are 28, 24 and 16 and the *ASR* is 100%, 98.4% and 86.67%, respectively on MNIST, CIFAR-10 and ImageNet. However, with the increase in *SN*, the average  $L_2$ -norm also increases, even if it does not increase significantly ( $\alpha = 1.0$ , the average  $L_2$ -norms of MNIST and CIFAR-10 increase by 0.2526 and 0.0545. For  $\alpha = 0.5$ , the difference is 0.3605 and 0.1547, respectively). The trend of *AvgQueries* on ImageNet is not obvious, because when  $SN = 22$ , the *AvgQueries* have a significant increase, and the other points are relatively stable after removing this point. Although the *AvgQueries* at this point have increased, the *ASR* has not changed much. Considering the *ASR*, *AvgQueries* and computing load, we finally choose *SN* as 20.

Equation (10) is used to initialize the candidate solution, where  $\alpha$  is closely related to the  $L_2$ -norm of the initial candidate solution. According to the above experimental results of  $SN$ , we fixed  $SN$  at 20, and increased  $\alpha$  from 0.1 to 1.0 in steps of 0.1. Figure 3 shows the results of untargeted attacks on MNIST, CIFAR-10 and ImageNet datasets, including the  $AvgQueries$  and  $AvgL_2$  under different  $\alpha$ . In MNIST and CIFAR-10 dataset experiments, we first show that while  $\alpha$  changes, the  $ASR$  does not fluctuate much. When  $\alpha = 0.8$  and  $\alpha = 0.9$ , the  $ASR$  of MNIST is 99.9%, and in other cases it is 100%. The  $ASR$  of CIFAR-10 is maintained at 98.3–98.5%. It can be seen from Figure 3 that the  $AvgQueries$  decrease monotonically with the increase in  $\alpha$ , while the average  $L_2$ -norm increases monotonously. The following conclusions can be drawn: the  $L_2$ -norm of the initial candidate solution is directly proportional to the final adversarial perturbation  $L_2$ -norm, and is inversely proportional to  $AvgQueries$  used for successful attacks. The smaller the  $L_2$ -norm of the initial candidate solution is, the smaller the  $L_2$ -norm of the final adversarial perturbation that will be crafted, but the  $AvgQueries$  used for successful attack will also increase. On the contrary, the larger the  $L_2$ -norm of the initial candidate solution is, the  $AvgQueries$  used for successful attack are correspondingly lower.

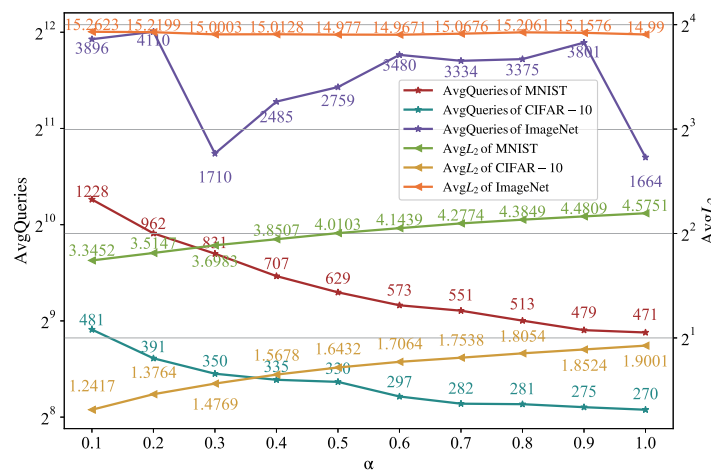


Figure 3. The relationship between  $\alpha$  and average queries and  $L_2$ -norm of adversarial perturbation.

However, the experimental results for the ImageNet dataset are not obvious. When  $\alpha = 0.3$  and  $\alpha = 1.0$ , the  $AvgQueries$  decrease because the  $ASR$  of these two data points decreases, which is 83.33% and 84.67%, respectively. Compared with other points, the  $ASR$  of 89.33–92.00% is lower, resulting in low  $AvgQueries$ . However, unlike the MNIST and CIFAR-10, the  $AvgQueries$  and  $AvgL_2$  have no particularly obvious trend on ImageNet. The  $AvgL_2$  shows a downward trend from 0.1–0.5. However, the  $AvgL_2$  shows a downward trend from 0.6–0.9, and the downward trend is not continuous. We think this is partly due to the increase in  $AvgQueries$ , because, in general, the more iterations, the more the  $L_2$ -norm will also increase. Considering  $ASR$ ,  $AvgQueries$  and  $AvgL_2$ , we can see that  $\alpha = 0.5$  is a better choice in the ImageNet dataset.

Parameter setting: In black-box attacks, the  $L_2$ -norm of perturbation cannot be balanced with the query cost. We can only try to trade off between them. Therefore, according to the above experimental results, we set  $SN = 20$ ,  $\alpha = 0.5$ , so the number of EBs and OBs is 10, and  $limitTrial = 10$ . In MNIST and CIFAR-10, the  $MaxEvaluation$  is set to 10,000, and for ImageNet, the  $MaxEvaluation$  is set to 30,000. The image size of the MNIST is  $28 \times 28$ , the maximum change  $\delta_{max} = 0.3$ , the image size is  $32 \times 32 \times 3$  in CIFAR-10 and  $299 \times 299 \times 3$  in ImageNet and the maximum change in these is  $\delta_{max} = 0.05$ . When the model input range is  $[0, 1]$  or  $[-0.5, 0.5]$ , and if the model input range is  $[0, 255]$ , the  $\delta_{max} = 0.05 \times 255 = 12$ .

### 5.3. Comparison and Analysis of Experimental Results

#### 5.3.1. Attack on MNIST, CIFAR-10 Classification Models

As can be seen in Table 1, ABCAttack uses 629 and 1695 average queries in both untargeted and targeted attacks on the MNIST dataset, achieving 100% and 94.89% attack success rates, uses 330 and 851 average queries on the CIFAR-10 dataset, achieving 98.6% and 82.3% attack success rates, respectively. Compared with ZOO that uses 384,000 average queries, the average queries of untargeted attacks are reduced by about 609 times, and the average queries of targeted attacks are reduced by about 255 times in ABCAttack on the MNIST dataset. Compared with C&W that uses 4650 average queries, the average queries of ABCAttack are reduced by 6.3 times in untargeted attacks, and the average queries are reduced by about 1.74 times in targeted attacks. It is obvious from the table that this phenomenon is the same in the CIFAR-10 dataset. The average queries used by ZOO and C&W are several times that of ABCAttack, whether it is an untargeted or targeted attack.

**Table 1.** Result comparison in MNIST and CIFAR-10. The experimental results of ZOO attack are black-box (ZOO-ADAM), and in the experimental results of C&W, the average time is calculated as the total time for training the surrogate model and generating adversarial samples. The bolded item indicates the best result.

MNIST								
Untargeted					Targeted			
Attack	ASR	AvgL <sub>2</sub>	AvgTime	AvgQueries	ASR	AvgL <sub>2</sub>	AvgTime	AvgQueries
ZOO	<b>100%</b>	<b>1.4955</b>	1.38 min	384,000	<b>98.90%</b>	<b>1.987068</b>	1.62 min	384,000
Black-box (Substitute Model + FGSM)	40.6%	–	0.002 s (+6.16 min)	–	7.48%	–	0.002 s (+6.16 min)	–
Black-box (Substitute Model + C&W)	33.3%	3.6111	0.76 min (+6.16 min)	4650	26.74%	5.272	0.80 min (+6.16 min)	4650
GenAttack	–	–	–	–	94.45%	5.1911	–	1801
AdversarialPSO	96.30%	4.1431	0.068 min	<b>593</b>	72.57%	4.778	0.238 min	1882
ABCAttack ( $\alpha = 0.5$ )	<b>100%</b>	4.01033	<b>0.018 min</b>	629	94.89%	4.7682	<b>0.048 min</b>	<b>1695</b>
SWISS	<b>100%</b>	3.4298	0.087 min	3043	19.41%	3.5916	0.345 min	20,026
ABCAttack ( $\alpha = 0.1$ )	<b>100%</b>	3.34524	0.045 min	1228	94.99%	4.5617	0.051 min	2066
CIFAR-10								
Untargeted					Targeted			
Attack	ASR	AvgL <sub>2</sub>	AvgTime	AvgQueries	ASR	AvgL <sub>2</sub>	AvgTime	AvgQueries
ZOO	<b>100%</b>	<b>0.19973</b>	3.43 min	128,000	<b>96.80%</b>	<b>0.39879</b>	3.95 min	128,000
Black-box (Substitute Model + FGSM)	76.1%	–	0.005 s (+7.81 min)	–	11.48%	–	0.005 s (+7.81 min)	–
Black-box (Substitute Model + C&W)	25.3%	2.9708	0.47 min (+7.81 min)	4650	5.3%	5.7439	0.49 min (+7.81 min)	4650
GenAttack	–	–	–	–	98.09%	1.3651	–	1360
AdversarialPSO	99.60%	1.414	0.139 min	1224	71.97%	2.925	0.6816 min	6512
ABCAttack ( $\alpha = 0.5$ )	98.60%	1.64319	<b>0.0233 min</b>	<b>330</b>	82.3%	1.910103	<b>0.0615 min</b>	<b>851</b>
SWISS	99.80%	2.3248	0.1264 min	2457	31.93%	2.9972	1.623 min	45,308
ABCAttack ( $\alpha = 0.1$ )	98.40%	1.24167	0.031 min	481	80.88%	1.644	0.0654 min	1102

In the experiment with MNIST, although the *AvgQueries* of ABCAttack are slightly lower than those of AdversarialPSO (ABCAttack: 629, AdversarialPSO: 593), the *ASR* of ABCAttack is better than that of AdversarialPSO. In targeted attacks, ABCAttack achieves a 94.89% attack success rate, and the average queries used are lower. In the experiment with CIFAR-10, only the *ASR* and *AvgL<sub>2</sub>* of untargeted attacks are slightly lower than the *AvgL<sub>2</sub>* (ABCAttack: 98.6%, 1.64319, AdversarialPSO: 99.6%, 1.4140). When  $\alpha = 0.1$ , in the MNIST experiment, the success rates of both SWISS and ABCAttack are 100%, and the *AvgL<sub>2</sub>* is comparable, but the average queries used by SWISS are about 2.47 times those of ABCAttack. However, in the targeted attack, the *AvgL<sub>2</sub>* of the adversarial samples generated by ABCAttack is 1 larger than that of SWISS. ABCAttack has more advantages in attack success rate and average queries. When  $\alpha = 0.1$ , in the CIFAR-10 dataset, only ABCAttack's *ASR* (98.4%) is lower than that of SWISS (99.6%), and the rest are the best.

Secondly, in MNIST and CIFAR-10 experiments, Substitute+FGSM and Substitute+C&W both use white-box attacks to generate adversarial samples on the trained substitute models to attack the target model. It can be seen from Table 1 that the *ASR* needs to be improved. In the MNIST dataset, transferability reaches 84.24% for the first substitute DNN and 78.72% for the second, with input variations of  $\delta_{max} = 0.3$  [9]. The *ASR* of transfer-based attacks depends on similarity between the model generating adversarial samples and the target model. With the same model structure, the more possibilities there are for successful transfer. However, the *ASR* with different structures is lower, especially in targeted attacks, because the decision boundaries of different models may be quite different, which cannot guarantee the success of target attacks. How to improve the transferability of adversarial samples is also a hot issue in current research.

Table 1 shows that only the *AvgL<sub>2</sub>* between the adversarial samples generated by ZOO and the original samples is the smallest, but the *AvgQueries* used are the largest. ABCAttack has more advantages in running time and queries.

### 5.3.2. Attacks on Large-Scale Image Dataset ImageNet

The attack results of the large-scale ImageNet dataset are shown in Table 2. The target models of ZOO, AdversarialPSO and SWISS are all InceptionV3. Our attack algorithm also takes VGG-16 and VGG-19 as the target model to facilitate comparison with the UAP algorithm. The input of the InceptionV3 model is  $299 \times 299 \times 3$  or  $224 \times 224 \times 3$ , and the input of VGG-16, VGG-19 and MobileNet is  $224 \times 224 \times 3$ . In this experiment, we do not use dimension reduction or hierarchical attack to improve the attack efficiency. Therefore, the perturbation dimension is the same size as the image size. Parameters *SN* and  $\alpha$  are still set to 20 and 0.5, and the maximum for queries is limited to 30,000. The first 500 randomly selected images predicted correctly by InceptionV3 in the ImageNet dataset are used as the original images to be input into the ABCAttack algorithm. The *ASR* and *AvgQueries* are 90.00% and 2759, respectively. In contrast, ZOO used 1,280,000 to produce adversarial samples with *AvgL<sub>2</sub>* of 1.19916 on the ImageNet dataset, and the *ASR* was as high as 88.9%, while the AdversarialPSO only used 2833 average queries, but the *ASR* was 82.00%. The *L<sub>2</sub>*-norm in ABCAttack is much larger than that of ZOO and AdversarialPSO. However, the difference between the adversarial samples generated by this attack algorithm and the original samples cannot be perceived by human eyes, as shown in Figure 4. The main reason for the larger *L<sub>2</sub>*-norm of the adversarial samples generated by ABCAttack is the larger perturbation dimension. ABCAttack used 2759 *AvgQueries* to achieve 90.00% *ASR*, while ZOO used 1,280,000 average queries, which is an order of magnitude difference. Compared with AdversarialPSO, our algorithm is about 70 fewer queries than that of AdversarialPSO, but has higher success rate, and the difference in *AvgQueries* is not significant, which is also related to the experimental samples.

Since different models have inconsistent processing schemes for input images, we set  $\delta_{max}$  according to the model input. When the model input is between  $[0, 1]$ , we set  $\delta_{max} = 0.05$ , and when the model input is in the  $[-1, 1]$  range, we set  $\delta_{max}$  to 0.1, but when the model input is  $[0, 255]$ , we set it to 12, but this is basically consistent for the maximum change in the image in RGB space.

The UAP algorithm generates a universal perturbation without generating for each image. The accuracies of UAP [29] on VGG-16 and VGG-19 models are 78.3% and 77.8%. Some of the training images are used to craft universal perturbations, and [30] shows that universal perturbations can be crafted with non-training set images, which removes the limitation of knowing part of the training set data for universal perturbation generation, but as a cost, *ASR* decreases significantly. Tran-TAP and Tran-UAP [39] adopt transfer-based attacks. It can be seen that the *ASR* from VGG-16 to Inc-V3 is 76.1% and 68.8%. Although UAP is still transferable, its *ASR* is reduced. The *ASR* is only 34.3% when adversarial samples generated by Inc-V3-FGSM transfer to ResNet-V2. The difference in model structure is the bottleneck to improve the *ASR* of transfer-based attacks.

In this study, our ABCAttack ( $a = 0.5$ ) could obtain 99.6%, 99.0% and 100% success rates in VGG-16, VGG-19 and MobileNet, respectively, with average queries of 1839, 1501 and 568 when the maximum query *MaxEvaluation* is set to 30000. The attack success rate is significantly higher than that of the UAP algorithm.

**Table 2.** Result comparison in ImageNet dataset.

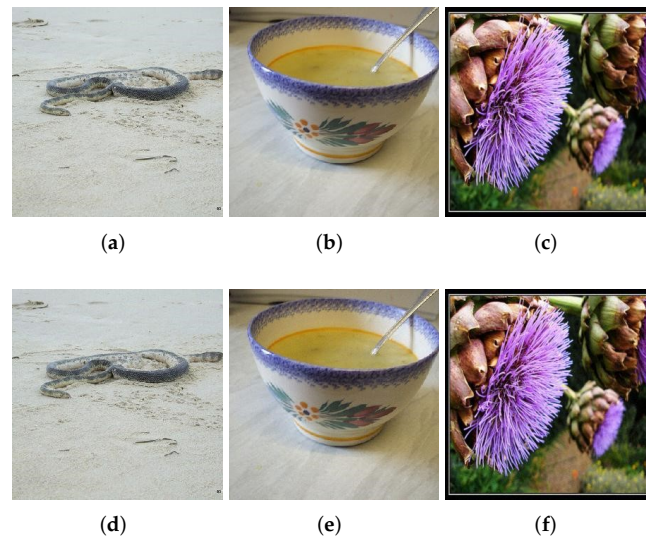
Method	MaxEvaluation = 30,000			MaxEvaluation = 50,000		
	ASR	AvgQueries	AvgTime	ASR	AvgQueries	AvgTime
UAP [29] (VGG-16)	78.3%	-	-	-	-	-
UAP [29] (VGG-19)	77.8%	-	-	-	-	-
UAP-Fast Feature Fool [30] (VGG-16)	47.10%	-	-	-	-	-
UAP-Fast Feature Fool [30] (VGG-19)	43.62%	-	-	-	-	-
Tran-TAP [39] VGG-16 to Inc-v3	76.1%	-	-	-	-	-
Tran-TAP [39] Inc-v3 to VGG-16	70.6%	-	-	-	-	-
Tran-UAP [39] VGG-16-UAP to Inc-v3	68.8%	-	-	-	-	-
Tran-UAP [39] Inc-v3-UAP to VGG-16	60.2%	-	-	-	-	-
ZOO	88.9%	1,280,000	8.031 min	-	-	-
AdversarialPSO	82.00%	2833	3.181 min	-	-	-
SWISS	70.66%	8429	5.014 min	-	-	-
ABCAttack (VGG-16)	99.60%	1839	2.07 min	99.8%	1901	2.1276 min
ABCAttack (VGG-19)	99.00%	1501	1.7298 min	99.6%	1698	1.8832 min
ABCAttack (MobileNet-v3)	100%	568	0.6714 min	100%	568	0.6714 min
ABCAttack (Inc-v3, image size is 299)	90.00%	2759	3.172 min	92.00%	2971	3.238 min
ABCAttack (Inc-v3, image size is 224)	98.4%	899	1.0086 min	98.4%	899	1.0086 min

In terms of attack cost time, ZOO needs to evaluate the gradient. For large-scale datasets, multi-dimensional gradient evaluation is quite time-consuming even with dimensionality reduction, so the attack time cost of the ZOO method is high. Compared with AdversarialPSO on the ImageNet dataset, we have a slight advantage in its average queries and average time spent, but we have an 8% increase in attack success rate, and regardless of whether we use VGG-16, VGG19, MobileNet or InceptionV3, our attack success rate is over 90.00%.

We conducted experiments with a greater query limit *MaxEvaluation* = 50,000. The experimental results are shown in Table 2. From the perspective of attack success rate, even if the maximum query *MaxEvaluation* is increased by 20,000, the increase in attack success rate is only slight. When VGG-16 and VGG-19 are used as target models, when the maximum query increases to 50,000, the attack success rate increases by 0.2% and 0.6%, respectively. When taking InceptionV3 as the target model, the attack success rate did not rise when the image size was 224. Even when the maximum query is increased to 70,000, a few samples still cannot obtain the corresponding adversarial samples. Compared with the maximum limit change  $\delta_{\max}$ , increasing the maximum limit query *MaxEvaluation* will not significantly improve the attack success rate, and will lead to an increase in the average queries. Similarly, we intentionally increased the *MaxEvaluation* when attacking the defense strategy, and the results can be seen in Table 3 and are consistent with the results on ImageNet.

However, we did not do any optimization in the ImageNet dataset, so the quality of generated adversarial images is not high. Improving the quality of adversarial samples,

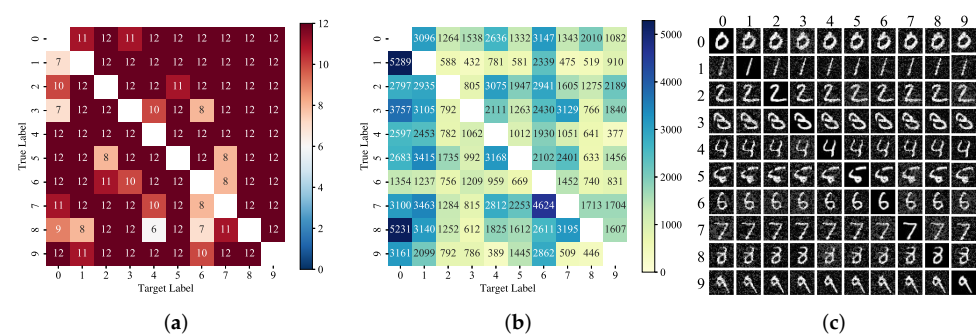
reducing the dimensions of algorithm input and reducing the average queries in a large-scale dataset in a black-box setting that does not rely on gradient calculations will be our future focus.



**Figure 4.** Original samples (top) and their corresponding adversarial samples (bottom) in ImageNet dataset. (a) Predicted label: sea snake. (b) Predicted label: soup bowl. (c) Predicted label: artichoke. (d) Predicted label: Indian cobra. (e) Predicted label: red wine. (f) Predicted label: cartoon.

### 5.3.3. Targeted Attack Analysis

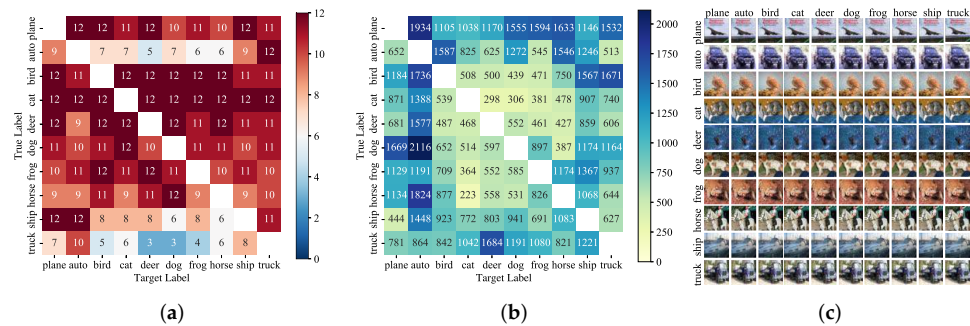
This experiment selects the first 12 images of each class in the MNIST and CIFAR-10 datasets to carry out a targeted attack, and each image uses the remaining class as the pre-set label of the targeted attack. Therefore,  $12 \times 10 \times 9 = 1080$  attacks are executed. The *ASR* is 93.98% and 83.24%, and the corresponding *AvgQueries* and *AvgL<sub>2</sub>* are 1744 and 915, and 4.844 and 1.911, respectively. Figures 5 and 6 show the analysis of the experimental results of a targeted attack on MNIST and CIFAR-10 datasets, respectively, where (a) is the heat map of the number of successful attacks, and the total number of attacks for each square is 12. Panel (b) shows the corresponding average queries. It can be seen from the figure that the original images predicted as auto and truck in the CIFAR-10 dataset are more difficult to be made into adversarial samples, while some samples are easier to be crafted adversarial images to attack models.



**Figure 5.** Analysis of targeted attack results in MNIST dataset. The row label is true label and column label is target label. (a) Heat map of the number of successful attacks in the MNIST. (b) The average queries used by each different original–target class pairs in MNIST. (c) The original images in the MNIST test set and the generated adversarial images by ABCAttack.

In the MNIST dataset, although the *ASR* is higher, when the pre-specified label is 0, the *AvgQueries* used for successful attack is more, which means that, in comparison, the clean

samples of any other class are more difficult to be made into adversarial samples predicted to be 0. Compared with CIFAR-10, the *AvgQueries* used for successful attack in the MNIST dataset are more, and the heap map of both of them has slight symmetry, which is more obvious in CIFAR-10. In (c), the images on the main diagonal are the original images, and the other images are the generated corresponding adversarial images. It can be seen that there is a slight difference between the adversarial images generated by ABCAttack and the original images. However, the model classifies 10 different images into 10 different classes which can be correctly recognized as the same labels by a human. This further proves that DNN models still have many vulnerabilities and defects.



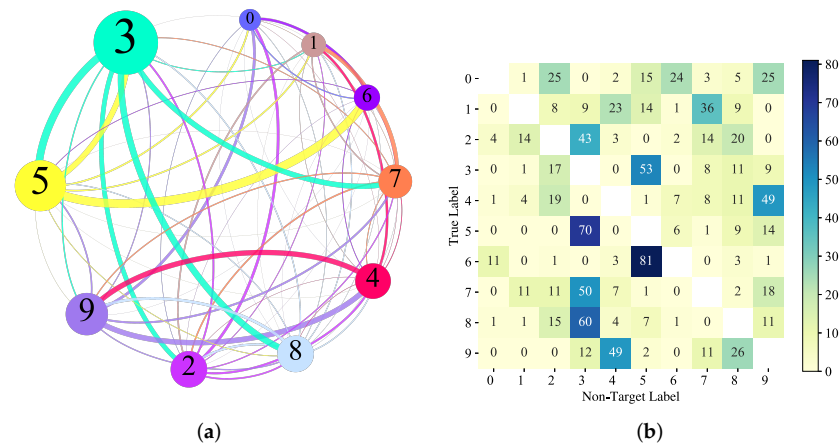
**Figure 6.** Analysis of targeted attack results in CIFAR-10 dataset. The row label is true label and column label is target label. (a) Heat map of the number of successful attacks in the CIFAR-10. (b) The average queries used by each different original–target class pairs in CIFAT-10. (c) The original images in the CIFAR-10 test set and the generated adversarial images by ABCAttack.

### 5.3.4. Untargeted Attack Analysis

We select the first 100 correct prediction samples in each class of MNIST and CIFAR-10, and  $10 \times 100 = 1000$  images are selected to craft untargeted adversarial images. The *AvgQueries* using 664 and 295 can achieve 99.9% and 98.3% ASR. Figures 7 and 8 show the statistics of untargeted attacks on MNIST and CIFAR-10 datasets, respectively. Panel (a) is the relationship between the true labels of original images and predicted labels of corresponding adversarial images in the untargeted attack. Each node in (a) represents a class, and the node size indicates the number of adversarial samples that are predicted to be in the class. The larger the node, the more crafted adversarial images are predicted to be in the class, the color of the edge is consistent with the target node and the thickness of the edge represents the number of original–predicted label of adversarial image pairs. More detailed results can be seen in (b), which shows the relationship between the true class and predicted labels of the adversarial samples.

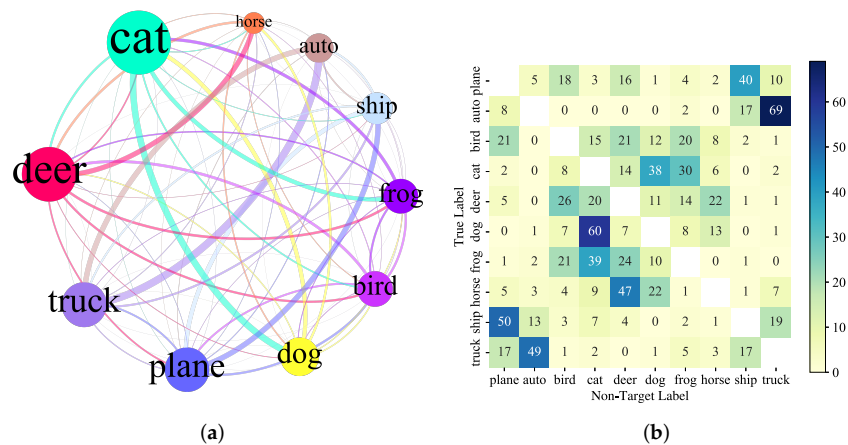
From Figures 7a and 8a, it can be seen that on MNIST and CIFAR-10 datasets, the ABCAttack more easily creates adversarial images that are recognized by the model as the digit 3 and a cat. In the MNIST experiment, original images with class 3, 5, 7 and 8 are easily crafted into adversarial images recognized as 3. In the CIFAR-10 experiment, the original images with class deer, dog and frog are more easily crafted into adversarial images recognized as a cat. From Figures 7b and 8b, it can be seen that, similar to the targeted attack, the heat map of original–predicted classes of adversarial sample pairs has a symmetrical relationship, especially in the CIFAR-10 dataset.





**Figure 7.** Analysis of untargeted attack results in MNIST dataset. (a) The relationship between the label of original images and predicted label of corresponding adversarial images in MNIST dataset. (b) The number of successful crafted adversarial images for each class of MNIST dataset in untargeted attacks.

In MNIST, original images with true label 4 are easier to be crafted into adversarial samples with predicted label 9, and original images with true label 3 are easier to be crafted into adversarial samples with predicted label 5 in untargeted attacks, and vice versa. The reason may be because the digit 4 is similar to the digit 9, and the digit 3 is similar to the digit 5. However, this symmetry is not absolute. Among the 100 original images with class 6, 81 adversarial samples produced in untargeted attacks are predicted to be 5, but only six in the opposite case. Perhaps, in comparison, the handwritten digit 5 and the handwritten digit 3 are more similar than the handwritten digit 6, which to a certain extent shows that the DNN model has indeed learned the characteristics of each class. In the CIFAR-10 dataset, there are cat–dog class pairs that exhibit the same properties, which is consistent with the description in [26].



**Figure 8.** Analysis of untargeted attack results in CIFAR-10 dataset. (a) The relationship between the label of original images and predicted label of corresponding adversarial images in CIFAR-10 dataset. (b) The number of successful crafted adversarial images for each class of CIFAR-10 dataset in untargeted attacks.

In untargeted attacks, even the same image can be crafted into adversarial images that are predicted to be a different class to a certain extent. This is due to the randomness of ABCAttack.

#### 5.4. Attacking Defenses

While multiple defenses already exist, researchers have been trying to study new attacks that render defenses ineffective. Our proposed ABCAttack can also break some existing defenses. The defense methods we use include adversarial training, input transformation and some overall defense. For all the following attacks against CIFAR-10, we have followed the setting  $\epsilon = 8.0$  or  $\epsilon = 0.031$  and  $MaxEvaluation = 10,000$ , unless otherwise stated. The experimental results can be seen in Table 3.

Athalye et al. [40] point out that although a “confusion gradient” can defeat some attacks based on gradient optimization, the defense relying on this “confusion gradient” can still be avoided. A BPDA white-box attack is proposed, which can break some defenses based on a “confusion gradient”. We selected some of the mentioned defenses to evaluate the effectiveness of our method for defense strategies.

We first evaluated our attack on defense-GAN [31] using the MNIST dataset and adding  $L_2$ -criteria to the fitness function, and we achieved a success rate of 62.4% and a BPDA of 45%. Stochastic Activation Pruning (SAP) [32] introduces randomness into neural networks to resist adversarial samples. The application of SAP will slightly reduce the accuracy of cleaning classification, but increase the robustness. We evaluated ABCAttack’s performance against SAP defense on the CIFAR-10 dataset. When setting parameters  $\epsilon = 0.031$ ,  $MaxEvaluation = 20,000$  and  $\epsilon = 0.031$ ,  $MaxEvaluation = 20,000$  and  $\epsilon = 0.05$ ,  $MaxEvaluation = 10,000$ , the success rate is 64.6%, 67.4% and 88.4%, respectively. Input transformations [33] defend against the adversarial samples by converting inputs. We evaluated the effectiveness of this defense on the ImageNet dataset, and our method can still achieve a 78% success rate in this defense (JPEG and bit depth). Local Intrinsic Dimensionality (LID) [41] is a general metric used to measure the distance from input to adjacent input. The results of the attack on the CIFAR-10 dataset show that our ABCAttack can still reduce the accuracy of the model to 0.01% under the LID defense strategy, and ASR reaches 99.9%.

The focus of [42] is to analyze and evaluate some defenses against adversarial samples in detail, and the study points out that defenses that are claimed to be robust to white-box attacks can still be broken. No attack strategy is sufficient to deal with all defenses, that is, to break a defense, its internal mechanism also needs to be carefully analyzed and adjusted. We evaluated two of these defenses. In  $k$ -winners defense takes all in [42], we obtained 70.4% ASR in the adversarial training model, and the accuracy of the model was reduced to 0.16% in the adversarial training model in [42]. Pang et al. [43] proposed to train an ensemble of models with an additional regularization term that encourages diversity in defense adversarial samples. We set the maximum  $L_\infty$  distance  $\delta_{max} = 0.031$  and  $\delta_\infty = 0.05$  in the CIFAR-10 dataset, and obtained ASR of 71.4% and 96.3%, respectively, when limiting the maximum number of queries to 10,000. When  $MaxEvaluation = 20,000$  and  $\delta_\infty = 0.031$ , the ASR is 98.3%.

As can be seen from Table 3, the black-box ABCAttack of this paper can break the above defenses to a great extent, but the attack cost increases. All the above experiments were performed with untargeted attacks. Compared with the attacks of [40,42], the proposed ABCAttack has a lower attack success rate for the defense model, especially the model of adversarial training. In the future, we will further enhance ABCAttack on the defense model and reduce the attack cost.

**Table 3.** Evaluating the effectiveness of ABCAttack on the existing defense strategy.

Defense	Dataset	Parameter Setting	AvgQueries	ASR
defense-GAN [31]	MNIST	$\delta_{max} = 0.031$ $MaxEvaluation = 20,000$	1066	62.40%
SAP [32]	CIFAR-10	$\delta_{max} = 0.031$ $MaxEvaluation = 10,000$	986	64.60%
SAP [32]	CIFAR-10	$\delta_{max} = 0.031$ $MaxEvaluation = 20,000$	1491	67.40%
SAP [32]	CIFAR-10	$\delta_{max} = 0.05$ $MaxEvaluation = 10,000$	701	88.40%
JPEG and bit depth [33]	ImageNet	$\delta_{max} = 0.05$ $MaxEvaluation = 30,000$	4893	78.00%
LID [41]	CIFAR-10	$\delta_{max} = 0.031$ $MaxEvaluation = 10,000$	362	99.90%
$k$ -winners [42] (adversarial training model)	CIFAR-10	$\delta_{max} = 0.031$ $MaxEvaluation = 10,000$	937	70.40%
Ensemble train [43]	CIFAR-10	$\delta_{max} = 0.031$ $MaxEvaluation = 10,000$	893	89.50%
Ensemble train [43]	CIFAR-10	$\delta_{max} = 0.05$ $MaxEvaluation = 10,000$	448	96.30%
Ensemble train [43]	CIFAR-10	$\delta_{max} = 0.05$ $MaxEvaluation = 20,000$	744	98.30%

### 5.5. The Wide Applicability of ABCAttack to Various DNN Models

Following AdversarialPSO, we used HRNN and MLP as target models on the MNIST dataset, which achieved 98.76% and 97.94% accuracy on the test set, respectively, and used CNNCapsule (with data augmentation) and ResNet (ResNet20V1 with data augmentation), which achieved test accuracy of 81.89% and 90.61%, respectively, as target models on the CIFAR-10 dataset to perform untargeted attacks. The networks and training code can be found at: <https://github.com/rstudio/keras/tree/master/vignettes/examples> (accessed on 4 November 2020). The experimental results are shown in Table 4.

In addition to the target model in Table 4, we also implemented the proposed ABCAttack with different models as target models on MNIST and CIFAR-10 datasets. The experimental results are shown in Table 5. On the MNIST dataset, we performed targeted and untargeted attacks with Lenet5, LSTM and CNN models as victim models. We also used ResNet20V1, ResNet32V1, ResNet44V1, ResNet56V1, ResNet110V1, ResNet20V2, ResNet56V2 and ResNet110V2 as target models on the CIFAR-10 dataset. These ResNet models were trained with data augmentation and, respectively, obtained the following accuracy rates on the CIFAR-10 test set: 90.61%, 91.58%, 91.92%, 92.27%, 91.54%, 91.31%, 92.93%, 92.9%.

It can be seen from Table 4 that under the same target model, our method has a greater advantage in the AvgQueries used to successfully attack. On different target models, the attack success rates of the two methods have their own advantages and disadvantages, but the difference is relatively small. From Tables 4 and 5, it can be seen that the proposed ABCAttack can obtain superior results on target models with different structures, different datasets and different model sizes. This shows that it has good applicability.

**Table 4.** Comparison of the results of ABCAttack and AdversarialPSO on different target models, and the bolded item indicates the best result.

MNIST				
Attack	HRNN		MLP	
	ASR	AvgQueries	ASR	AvgQueries
AdversarialPSO	100%	552	94.70%	548
SWISS	100%	3214	100%	1984
ABCAttack ( $\alpha = 0.5$ )	100%	<b>395</b>	99.40%	<b>412</b>
ABCAttack ( $\alpha = 0.1$ )	100%	1083	99.80%	715
CIFAR-10				
Attack	CNNCapsule		ResNet	
	ASR	AvgQueries	ASR	AvgQueries
AdversarialPSO	97.80%	2052	100%	1723
SWISS	98.90%	3725	100%	1792
ABCAttack ( $\alpha = 0.5$ )	100%	<b>164</b>	99.20%	<b>165</b>
ABCAttack ( $\alpha = 0.1$ )	100%	300	99.20%	290

**Table 5.** The wide applicability of ABCAttack on various target models.

Model	Dataset	Untargeted		Targeted	
		ASR	AvgQueries	ASR	AvgQueries
CNN	MNIST	99.50%	381	80.30%	806
LSTM		100%	184	98.50%	920
Lenet5		99.90%	291	93.90%	1648
ResNet20V1	CIFAR-10	99.10%	161	91.60%	798
ResNet32V1		98.80%	140	93.10%	892
ResNet44V1		99.10%	202	91.40%	900
ResNet56V1		99.60%	171	93.20%	617
ResNet110V1		99.50%	180	90.30%	682
ResNet20V2		99.60%	136	95.30%	547
ResNet56V2		99.40%	150	94.30%	546
ResNet110V2		98.90%	141	95.80%	564

## 6. Discussion

In this paper, the ABC algorithm is used to generate adversarial images. As a swarm intelligence algorithm, the performance of ABC is sensitive to the selection of the hyperparameter of population size. In addition, under the constraint of the maximum change in the image, the value of  $\delta_\infty$  can also affect the performance of ABCAttack. Therefore, in this section, we will discuss the impact of population size, maximum  $L_\infty$ -norm perturbation and maximum queries on the *AvgQueries* and *ASR*.

**Population size:** A larger population size will increase the diversity of candidate solutions, thereby enhancing the exploration ability of the algorithm. In this paper, a single candidate solution input to the target model is recorded as a query, so we need to set the population size reasonably to balance the algorithm performance and the average queries. As can be seen from Figure 2, we increase the population size from 10 to 30 in steps of 2, and

observed the change in the average queries, and found that with the continuous growth of the population size, the average queries used for successful attacks no longer decreased, but remained relatively stable, and even increased slightly. Therefore, we conclude that setting the population size to 20 is a more reasonable choice to balance the convergence speed of the algorithm and the average queries.

$\delta_{max}$ :  $\delta_{max}$  is the maximum absolute value in a single dimension limited to the added perturbation on the clean image, and it is obvious that  $\delta_{max}$  directly affects the adversarial image quality, attack success rate and average queries. As the greater the increased perturbation amplitude, the greater the distance between it and the clean image in the pixel domain, the easier it is to cause the model to give wrong prediction results, and the attack algorithm can obtain higher attack success rate with fewer queries. We show in Table 3 the average queries and success rates of the proposed attack under different  $\delta_{max}$  in breaking SAP and ensemble training defense strategies, which prove this. However, correspondingly, the distance between the generated adversarial image and the clean image is larger, and when the value of  $\delta_{max}$  increases to a certain extent, the generated adversarial samples may cause humans to fail to distinguish their original class. Therefore, the added adversarial perturbation needs to be constrained to be less different from the original image.

## 7. Conclusions and Future Work

In this paper, a black-box attack using the ABC algorithm ABCAttack is proposed, and related experiments are carried out on MNIST, CIFAR-10 and ImageNet datasets. The results of comparison with methods such as ZOO, C&W, AdversarialPSO, GenAttack and UAP show that the proposed ABCAttack can efficiently generate adversarial images with high attack success rates. We analyze the relationship between  $SN$ ,  $\alpha$  and the  $AvgL_2$  and  $AvgQueries$  of the generated adversarial images. It is noticed that the smaller the  $L_2$ -norm between the initial food sources and the original images, the smaller the final  $L_2$ -norm of the adversarial perturbations. By comparing the original–predicted class of adversarial sample pairs of untargeted and targeted attacks, it is found that some images are easier to be made into such adversarial images, where the predicted label of the adversarial examples and the corresponding true label show some visual or appearance similarity for both a CNN and human eyes. To some extent, this reflects that the DNN models has learned the local information of the class. The proposed ABCAttack can still break some defenses to a certain extent, even if these defenses claim to be robust to white-box attacks. This puts forward new requirements for future defenses, and should be evaluated in black-box attacks. It seems that there is a long way to go for the defense of adversarial samples. The experiments in this paper further illustrate the importance of population-based gradient-free black-box attacks in adversarial attacks, and provide a future research direction for ML system attacks and defenses.

Although the experimental results show the effectiveness of ABCAttack, there is still space for further improvement. The existing GenAttack, AdversarialPSO and ABCAttack are all black-box attacks, they can achieve high attack success rates with fewer average queries and the  $L_2$ -norm of perturbations is larger than that of white-box attacks and gradient evaluation methods. ABC has many variants and improvements in this algorithm. In the next step, we will pursue the use of fewer queries to produce higher-quality adversarial images with a smaller distance from the original images. Secondly, it takes time and effort to craft adversarial samples with high-dimensional data. How to improve the applicability of the algorithm to high-dimensional data needs further research under the premise of maintaining the attack success rate. Finally, the defense ability of many existing defense measures against population-based gradient-free black-box attacks is greatly reduced, which undoubtedly increases the urgency of designing more effective defense measures against attacks. In addition, no single defense can effectively defend against all attacks, and there will always be powerful attacks to break the defense, which makes us more vigilant about the security issues of DNN models. In the future, we will also focus on the root

causes and interpretability of adversarial samples, and study more robust defense methods to enhance the security of DNN models.

**Author Contributions:** Data curation, Y.L.; Formal analysis, C.S.; Funding acquisition, Q.S.; Investigation, S.L.; Methodology, H.C.; Supervision, P.G.; Validation, H.C.; Visualization, H.C.; Writing—original draft, H.C.; Writing—review & editing, Q.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** The research presented in this paper is supported in part by Shaanxi Province Key Research and Development Program (No.: 2022ZDLSF07-07) and the Youth Innovation Team of Shaanxi Universities (No.: 2019-38).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Li, S.; Song, W.; Fang, L.; Chen, Y.; Ghamisi, P.; Benediktsson, J.A. Deep learning for hyperspectral image classification: An overview. *IEEE Trans. Geosci Remote Sens.* **2019**, *57*, 6690–6709. [[CrossRef](#)]
2. Aceto, G.; Ciunzo, D.; Montieri, A.; Pescapé, A. DISTILLER: Encrypted traffic classification via multimodal multitask deep learning. *J. Netw. Comput. Appl.* **2021**, *183*, 102985. [[CrossRef](#)]
3. Qiu, H.; Dong, T.; Zhang, T.; Lu, J.; Memmi, G.; Qiu, M. Adversarial Attacks Against Network Intrusion Detection in IoT Systems. *IEEE Internet Things J.* **2021**, *8*, 10327–10335. [[CrossRef](#)]
4. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2013**, arXiv:1312.6199.
5. Sun, Q.; Lin, K.; Si, C.; Xu, Y.; Li, S.; Gope, P. A Secure and Anonymous Communicate Scheme over the Internet of Things. *ACM Trans. Sen. Netw.* **2022**. [[CrossRef](#)]
6. Flowers, B.; Buehrer, R.M.; Headley, W.C. Evaluating Adversarial Evasion Attacks in the Context of Wireless Communications. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 1102–1113. [[CrossRef](#)]
7. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial Examples in the Physical World. *arXiv* **2016**, arXiv:1607.02533.
8. Tabacof, P.; Valle, E. Exploring the space of adversarial images. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 426–433.
9. Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z.B.; Swami, A. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Abu Dhabi, United Arab Emirates, 2–6 April 2017; pp. 506–519.
10. Chen, P.Y.; Zhang, H.; Sharma, Y.; Yi, J.; Hsieh, C.J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November 2017; pp. 15–26.
11. Alzantot, M.; Sharma, Y.; Chakraborty, S.; Zhang, H.; Hsieh, C.J.; Srivastava, M.B. Genattack: Practical black-box attacks with gradient-free optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019; pp. 1111–1119.
12. Liu, X.; Hu, T.; Ding, K.; Bai, Y.; Niu, W.; Lu, J. A black-box attack on neural networks based on swarm evolutionary algorithm. In Proceedings of the Australasian Conference on Information Security and Privacy, Perth, WA, Australia, 30 November–2 December 2020; Springer: Amsterdam, The Netherlands, 2020; pp. 268–284.
13. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2014**, arXiv:1412.6572.
14. Nguyen, A.; Yosinski, J.; Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 427–436.
15. 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, 27–30 June 2016; pp. 2574–2582.
16. Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbruecken, Germany, 21–24 March 2016; pp. 372–387.
17. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv* **2013**, arXiv:1312.6034.

18. Carlini, N.; Wagner, D. Towards evaluating the robustness of neural networks. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 39–57.
19. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016, pp. 582–597.
20. Papernot, N.; McDaniel, P.; Goodfellow, I. Transferability in Machine Learning: From Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv* **2016**, arXiv:1605.07277.
21. Biggio, B.; Rieck, K.; Ariu, D.; Wressnegger, C.; Corona, I.; Giacinto, G.; Roli, F. Poisoning behavioral malware clustering. In Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, Scottsdale, AZ, USA, 7 November 2014; pp. 27–36.
22. Narodytska, N.; Prasad Kasiviswanathan, S. Simple Black-Box Adversarial Perturbations for Deep Networks. *arXiv* **2016**, arXiv:1612.06299.
23. Brendel, W.; Rauber, J.; Bethge, M. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *arXiv* **2017**, arXiv:1712.04248.
24. Ilyas, A.; Engstrom, L.; Athalye, A.; Lin, J. Black-box adversarial attacks with limited queries and information. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 2137–2146.
25. Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841. [[CrossRef](#)]
26. Su, J.; Vargas, D.V.; Sakurai, K. Attacking convolutional neural network using differential evolution. *IPSP Trans. Comput. Vis. Appl.* **2019**, *11*, 1–16. [[CrossRef](#)]
27. Mosli, R.; Wright, M.; Yuan, B.; Pan, Y. They Might NOT Be Giants: Crafting Black-Box Adversarial Examples with Fewer Queries Using Particle Swarm Optimization. *arXiv* **2019**, arXiv:1909.07490.
28. Zhang, Q.; Wang, K.; Zhang, W.; Hu, J. Attacking black-box image classifiers with particle swarm optimization. *IEEE Access* **2019**, *7*, 158051–158063. [[CrossRef](#)]
29. 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.
30. Mopuri, K.R.; Garg, U.; Babu, R.V. Fast feature fool: A data independent approach to universal adversarial perturbations. *arXiv* **2017**, arXiv:1707.05572.
31. Samangouei, P.; Kabkab, M.; Chellappa, R. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv* **2018**, arXiv:1805.06605.
32. Dhillon, G.S.; Azizzadenesheli, K.; Lipton, Z.C.; Bernstein, J.; Kossaifi, J.; Khanna, A.; Anandkumar, A. Stochastic activation pruning for robust adversarial defense. *arXiv* **2018**, arXiv:1803.01442.
33. Guo, C.; Rana, M.; Cisse, M.; Van Der Maaten, L. Countering adversarial images using input transformations. *arXiv* **2017**, arXiv:1711.00117.
34. Xiao, C.; Zhong, P.; Zheng, C. Resisting adversarial attacks by k-winners-take-all. *arXiv* **2019**, arXiv:1905.10510.
35. Karaboga, D. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report; Citeseer. 2005. Available online: [https://abc.erciyes.edu.tr/pub/tr06\\_2005.pdf](https://abc.erciyes.edu.tr/pub/tr06_2005.pdf) (accessed on 24 October 2020).
36. LeCun, Y. The MNIST Database of Handwritten Digits. 1998. Available online: <http://yann.lecun.com/exdb/mnist/> (accessed on 4 May 2020).
37. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. 2009. Available online: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.222.9220> (accessed on 11 January 2021).
38. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
39. Zhou, W.; Hou, X.; Chen, Y.; Tang, M.; Huang, X.; Gan, X.; Yang, Y. Transferable adversarial perturbations. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 452–467.
40. Athalye, A.; Carlini, N.; Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 274–283.
41. Ma, X.; Li, B.; Wang, Y.; Erfani, S.M.; Wijewickrema, S.; Schoenebeck, G.; Song, D.; Houle, M.E.; Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv* **2018**, arXiv:1801.02613.
42. Tramer, F.; Carlini, N.; Brendel, W.; Madry, A. On adaptive attacks to adversarial example defenses. *arXiv* **2020**, arXiv:2002.08347.
43. Pang, T.; Xu, K.; Du, C.; Chen, N.; Zhu, J. Improving adversarial robustness via promoting ensemble diversity. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 4970–4979.