A Black-Box Attack on Neural Networks Based on Swarm Evolutionary Algorithm

Xiaolei Liu[1](✉), Teng Hu[1,2], Kangyi Ding[2], Yang Bai[2,3], Weina Niu[2], and Jiazhong Lu[4]

[1] Institute of Computer Application, China Academy of Engineering Physics, Mianyang, China
luxaole@gmail.com, mailhuteng@gmail.com

[2] University of Electronic Science and Technology of China, Chengdu, China
kangyiding@gmail.com, alicepub@163.com, niuweina1@126.com

[3] China Electronic Technology Cyber Security Co., Ltd., Chengdu, China

[4] Chengdu University of Information Technology, Chengdu, China
ljz@cuit.edu.cn

Abstract. Neural networks play an increasingly important role in the field of machine learning and are included in many applications in society. Unfortunately, neural networks suffer from adversarial examples generated to attack them. However, most of the generation approaches either assume that the attacker has full knowledge of the neural network model or are limited by the type of attacked model. In this paper, we propose a new approach that generates a black-box attack to neural networks based on the swarm evolutionary algorithm. Benefiting from the improvements in the technology and theoretical characteristics of evolutionary algorithms, our approach has the advantages of effectiveness, black-box attack, generality, and randomness. Our experimental results show that both the MNIST images and the CIFAR-10 images can be perturbed to successful generate a black-box attack with 100% probability on average. In addition, the proposed attack, which is successful on distilled neural networks with almost 100% probability, is resistant to defensive distillation. The experimental results also indicate that the robustness of the artificial intelligence algorithm is related to the complexity of the model and the data set. In addition, we find that the adversarial examples to some extent reproduce the characteristics of the sample data learned by the neural network model.

Keywords: Adversarial examples · Neural networks · Deep learning · Swarm evolutionary algorithm

1 Introduction

In recent years, neural network models have been widely applied in various fields, especially in the field of image recognition, such as image classification [11,31]

X. Liu—This research was supported by Director of Computer Application Research Institute Foundation (SJ2020A08, SJ2019A05).

© Springer Nature Switzerland AG 2020
J. K. Liu and H. Cui (Eds.): ACISP 2020, LNCS 12248, pp. 268–284, 2020.
https://doi.org/10.1007/978-3-030-55304-3_14

and face recognition [4]. However, users of such model are more concerned about the performance of the model and largely ignore the vulnerability and robustness of the model. In fact, most existing models are easily misled by adversarial examples deliberately designed by attackers and enable the attackers to achieve the purpose of bypassing the detection [20,30]. For example, in an image classification system, by adding the disturbance information to the original image, attackers can achieve the goal of changing image classification results with high probability [21]. The generated adversarial examples can even be classified with an arbitrary label according to the purpose of an attacker, making this type of attack a tremendous threat to the image classification system [6]. More seriously, printing the generated images of adversarial examples and then photographing them with a camera, the captured images are still misclassified, confirming the presence of adversarial examples in the real world [15]. These vulnerability problems make people raise the question on whether neural networks can be applied to security-critical areas.

Several papers have studied related security issues [16,17,19]. Unfortunately, in most previous generation approaches of adversarial examples, when ϵ is fixed, the similarity of the sample is fixed: in the algorithm's calculation, it won' change dynamically. This may cause the image to be disturbed so much that it can be visually distinguishable [22]. Moreover, the existing approaches mainly use gradient information to transform the original samples into the required adversarial examples. If the parameters of the model are unknown, the attackers cannot generate effective adversarial examples [7,12]. Others also proposed some black-box attack approaches [24,25]. However, Papernot [25] takes the transferability assumption. If transferability of the model to be attacked is reduced, the effectiveness of the attack will be reduced. LSA [24] cannot simply modify the required distance metrics, such as L_0, L_2, L_{max}. In most cases, it is only guaranteed that the disturbance is successful at Lmax, but not guaranteed that the disturbance can be kept minimum under other distance functions.

In this paper, we propose a new approach that generates a *black-box attack* to deep neural networks. Our approach is named BANA, denoting A (B)lack-box (A)ttack on (N)eural Networks Based on Swarm Evolutionary (A)lgorithm. Compared with the previous approaches [2,8,26,30], our approach has the following main advantages:

Effectiveness. The adversarial examples generated by our approach can misclassify the neural networks with 100% probability both on non-targeted attacks and targeted attacks. The L_2 distance between adversarial examples and original images is less than 10 on average, indicating that images can be disturbed with so small changes that are not to be undetectable. If we continue to increase the number of iterations of our proposed algorithm, we expect to achieve even better results.

Black-Box Attack. adversarial examples can be generated without the knowledge of the internal parameters of the target network, such as gradients and structures. Existing attacks such as Carlini and Wagner's attacks [2] usually require such information.

Generality. Our proposed attack is a general attack to neural networks. For the attack, we can generate effective adversarial examples of DNNs, CNNs, etc. We have even tested our proposed attack in a wider range of machine learning algorithms and it still misleads the model with 100% probability.

Randomness. Benefiting from the characteristics of evolutionary algorithms, the adversarial examples generated each time are different for the same input image, so they are able to resist defensive mechanisms such as defensive distillation.

In particular, our proposed attack is based on the swarm evolutionary algorithm [1]. The swarm evolutionary algorithm is a population-based optimization algorithm for solving complex multi-modal optimization problems. It can transform the optimization problems into the individual fitness function and has a mechanism to gradually improve individual fitness. Evolutionary algorithms do not require the use of gradient information for optimization and do not require that the objective function be differentiable or deterministic. Different from another approach also based on an evolutionary algorithm [29], our approach focuses on the optimization of results rather than the number of disturbed pixels. Therefore, we have completely different optimization function and iterative processes from the one pixel attack. Without knowing the parameters of the model, our proposed approach uses the original sample as the input to apply to generate an adversarial example of the specific label. The used information is only the probability of the various labels produced by the model.

Our attack also addresses technical challenges when applying the swarm evolutionary algorithm to generate the adversarial examples. The improvements made in our approach include the optimization of calculation results and convergence speed (see more details in Sect. 3).

The rest of the paper is organized as follows. Section 2 introduces the related work of adversarial examples. Section 3 presents SEAA (Swarm Evolutionary Algorithm For Black-box Attacks to Deep Neural Networks. Section 4 presents and discusses our experimental results. Section 5 concludes.

2 Related Work

The adversarial examples of deep neural networks have drawn the attention of many researchers in recent years. [30] used a constrained L-BFGS algorithm to generate adversarial examples. L-BFGS requires that the gradient of the model can be solved, limiting the diversity of the model and the objective function, and making this approach computationally expensive to generate adversarial examples. [8] proposed the fast gradient sign method (FGSM). However, this approach is designed without considering the similarity of the adversarial examples: the similarity of the generated adversarial samples may be low. The consequence is that the generated adversarial samples may be detected by defensive approaches or directly visually distinguished. An adversarial example attack named the Jacobian-based Saliency Map Attack (JSMA) was proposed by [26]. JSMA also requires the gradient of the model to be solved, and the approach is limited to the L_0 distance, and cannot be generated using other distance algorithms [2]. These approaches all assume that the attackers have full access to the parameters of the model. [23] proposed a non-targeted attack approach named Deepfool. This approach assumes that the neural network is linear and makes a contribution to the generation of adversarial examples, while actually neural networks may be not linear. Besides, this approach also does not apply to non-neural network model. Some previous research focused on generating adversarial examples to the malware detection models [3,18,32]. These adversarial examples also successfully disrupted the model's discriminant results, showing that the common models of machine learning are vulnerable to attacks.

Some recent research aimed to defend against the attack of adversarial examples and proposed approaches such as defensive distillatione [5,10,27]. However, experiment results show that these approaches do not perform well in particular situations due to not being able to defend against adversarial examples of high quality [9].

3 Methodology

3.1 Problem Description

The generation of adversarial examples can be considered as a constrained optimization problem. We use L_p distance (which is L_p norm) to describe the similarity between the original images and the adversarial images. Let f be the m-class classifier that receives n-dimensional inputs and gives m-dimensional outputs. Different from L-BFGS [30], FGS [8], JSMA [26], Deepfool [23] and Carlini and Wagner's attack [2], our approach is a black-box attack without using the gradient information. This optimization problem is formalized as follows:

$$F = D(x, x') + M \times loss(x') \tag{1}$$

where for a non-targeted attack (whose purpose is to mislead the classifier to classify the adversarial examples as any of the error categories), $loss(x')$ is defined as

$$loss(x') = max([f(x')]_r - max([f(x')]_{i \neq r}) \ , 0) \tag{2}$$

and for a targeted attack (whose purpose is to mislead the classifier to classify the adversarial examples as a specified category), $loss(x')$ is defined as

$$loss(x') = max(max([f(x')]_{i \neq t}) - [f(x')]_t \ , 0) \tag{3}$$

and $x = (x_1, ..., x_n)$ is the original image, $x' = (x'_1, ..., x'_n)$ is the adversarial example to be produced and $D(x, x')$ is the L_p distance. M is a positive number much larger than $D(x, x')$, r is the real label, and t is the target label. The output of $[f(x)]_r$ is the probability that the sample x is recognized as the label r and the output of $[f(x)]_{i \neq r}$ is the probability set that the sample x is separately recognized as other labels. Since $loss(x') \geq 0$, we discuss the case of $loss(x') > 0$ and $loss(x') = 0$ for the targeted attacks, respectively. The non-targeted attacks are the same.

(1) When $loss(x') > 0$, the $[f(x')]_t$ is not the maximum in Eq. 3, indicating that the adversarial example x' is not classified as the targeted label at this time. Since M is much larger than $D(x, x')$, the objective function in Eq. 1 is approximately equal to the latter half. In this case, it is equivalent to optimizing x' to minimize Q, i.e., increasing the probability that the classifier identifies the sample x' as being a class t.

$$minimize\ loss(x') \tag{4}$$

(2) When $loss(x) = 0$, the adversarial example has been classified as the target label at this time. In this case, it is equivalent to optimizing x' to minimize the value of $D(x, x')$, i.e., to improve the similarity between the adversarial example and the original sample as much as possible.

$$minimize\ D(x, x') \tag{5}$$

Through the preceding objective function, the population is actually divided into two sections, as shown in Fig. 1. The whole optimization process can be divided into three steps.

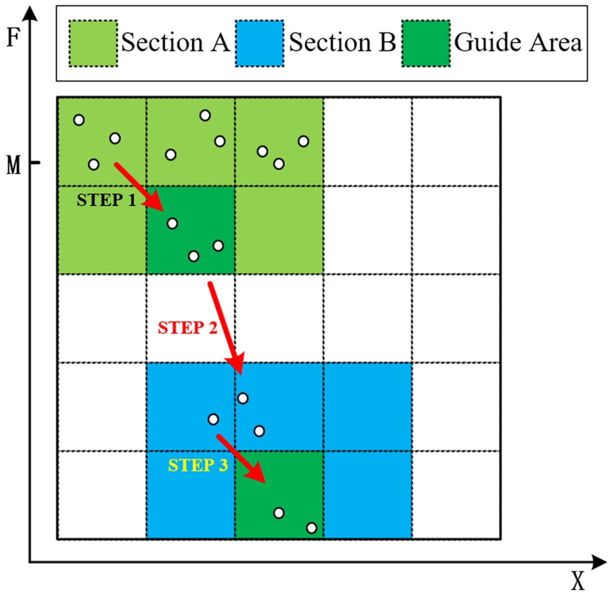


Fig. 1. Individuals distribution diagram

Step 1. At this time the adversarial example cannot successfully mislead the classifier. Individuals at the top of section A gradually approach the bottom through crossover and mutation operators.

Step 2. The individuals move from Section A to Section B, indicating that $loss(x') = 0$, i.e., the adversarial examples generated at this time can successfully mislead the classifier.

Step 3. Individuals at the top of Section B gradually approach the bottom, indicating the improvement of the similarity between the adversarial image and the original image.

Eventually, the bottom individual of Section B becomes the optimal individual in the population, and the information that it carries is the adversarial example being sought out.

3.2 Our BANA Approach

As the generation of adversarial examples has been considered as an optimization problem formalized as Eq. 1, we solve this optimization problem by the swarm evolution algorithm. In this algorithm, fitness value is the result of F, population is a collection of x' and many individuals make up the population. By constantly simulating the process of biological evolution, the adaptive individuals which have small fitness value in the population are selected to form the subpopulation, and then the subpopulation is repeated for similar evolutionary processes until the optimal solution to the problem is found or the algorithm reaches the maximum number of iterations. After the iterations, the optimal individual obtained is the adversarial example x'. As a widely applied swarm evolutionary algorithm, such genetic algorithm is flexible in coding, solving fitness, selection, crossover, and mutation. Therefore, in the algorithm design and simulation experiments, we use the following improved genetic algorithm as an example to demonstrate the effectiveness of our BANA approach. The advantages of this approach are not limited to the genetic algorithm. We leave as the future work the investigation of the effects of different types of swarm evolutionary algorithms on our approach.

Algorithm Workflow. The whole algorithm workflow is shown in Fig. 2. Classifiers can be logistic regression, deep neural networks, and other classification models. We do not need to know the model parameters and just set the input and output interfaces. Each individual is transformed into an adversarial example and then sent to the classifier to get the classification result. After that, the individual fitness value is obtained through solving the objective function. The individuals in the population are optimized by the genetic algorithm to solve the feasible solution of the objective function (i.e., the adversarial example of the image).

The workflow of our BANA approach is as follows:

Step 1. Population Initialization. One gene corresponds to one pixel, and for the grayscale images of (28, 28), there are a total of 784 genes, and there are $32 \times 32 \times 3 = 3072$ genes for the color image of (32, 32).

Step 2. Calculate the Fitness Value. Calculate the value of the fitness function according to the approach described in Sect. 3.1 and take this value as the fitness

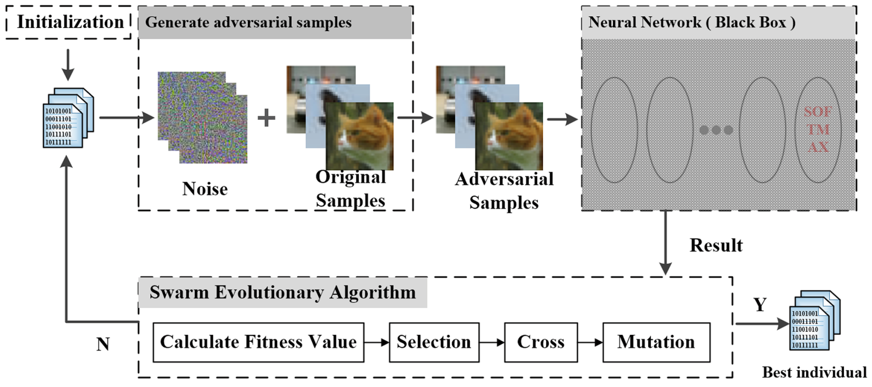


Fig. 2. Algorithm workflow of our BANA approach

of the individual. Since this problem is a minimization problem, the smaller the value, the better the individual's fitness. After that, the best individual with the minimum fitness value in the current population is saved as the optimal solution.

Step 3. Select Operation. According to the fitness of individuals in the population, through the tournament algorithm, individuals with higher fitness are selected from the current population.

Step 4. Cross Operation. Common crossover operators include single-point crossover, multi-point crossover, and uniform crossover. Our algorithm uses uniform crossover. That is, for two random individuals, each gene crosses each independently according to the probability p. Due to the large number of genes that each individual carries, uniform crossover allows for a greater probability of generating new combinations of genes and is expected to combine more beneficial genes to improve the searching ability of genetic algorithms.

Step 5. Mutation Operation. In order to speed up the search ability of genetic algorithms, combining with the characteristics of the problem to be solved, the operator adopts a self-defined Gaussian mutation algorithm. In the process of mutation, Gaussian noise $gauss(m, s)$ is randomly added to the individual (shown in Eq. 6 below), where m is the mean of Gaussian noise and s is the standard deviation of Gaussian noise:

$$x_{mutation} = x_{origin} \pm gauss(m, s) \tag{6}$$

The reason for adopting this mutation operation is that the resulting adversarial example inevitably has a high degree of similarity with the input sample, and a feasible solution to the problem to be solved must also be in the vicinity. This technique can effectively reduce the number of iterations required to solve the problem.

Step 6. Terminate the Judgment. The algorithm terminates if the exit condition is satisfied, and otherwise returns to Step 2.

Improvements. There are two major technical improvements made in our approach.

Improvements of Result. In order to improve the optimization effect of BANA, we adopt a new initialization technique. Considering the problem to be solved requires the highest possible degree of similarity, this technique does not use random numbers while using the numerical values related to the original pixel values. Let x be the original image, and x' be the initialized adversarial image. Then $x' = x + \epsilon$, where ϵ is a very small value.

Improvements of Speed. In order to speed up the convergence of BANA, on one hand, we constrain the variation step of each iteration in the mutation stage. On the other hand, we try to keep the point that has the pixel value of 0, because it is more likely that such a point is at the background of the picture. These improvements help the algorithm converge faster to the optimal solution.

4 Experiments

The datasets used in this paper include MNIST [14], CIFAR-10 [13] and ImageNet [28]. 80% of the data are used as a training set and the remaining 20% as a test set. In order to assess the effectiveness of the adversarial examples, we attack a number of different classifier models. The used classifiers include logistic regression (LR), fully connected deep neural network (DNN), and convolutional neural network (CNN). We evaluate our BANA approach by generating adversarial examples from the MNIST and CIFAR10 test sets.

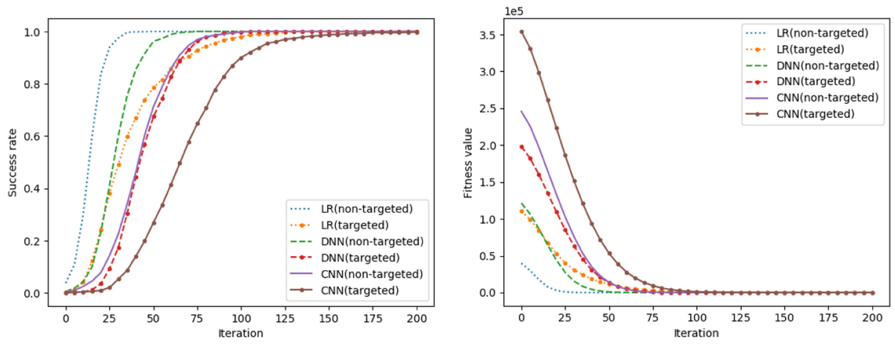
The parameters used by BANA are shown in Table 1. The experimental results show that different parameters affect the convergence rate of BANA. However, with the increase of the iterations, the results would eventually be close. The parameters listed in Table 1 are our empirical values.

Table 1. The parameters of BANA

Database	MNIST	CIFAR-10	ImageNet
Population	100	200	300
Genes number	$28 \times 28 \times 1 = 784$	$32 \times 32 \times 3 = 3072$	$200 \times 200 \times 3 = 120000$
Cross probability	0.5	0.5	0.5
Mutation probability	0/05	0/05	0/05
Iterations	200	200	100
Gaussian mean	0	0	0
Gaussian variance	30	20	40

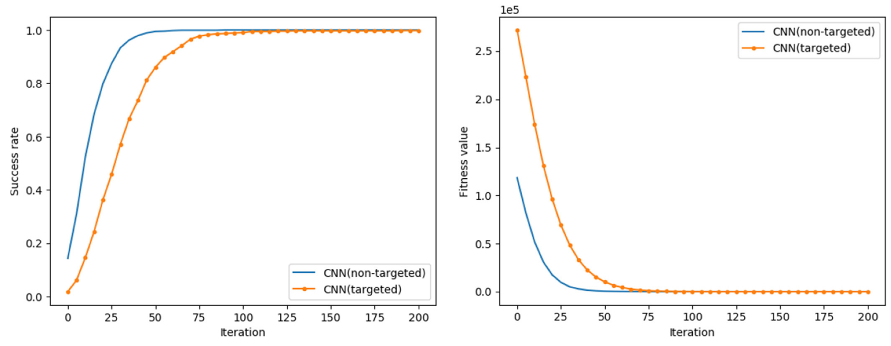
4.1 Adversarial Example Generation on MNIST

In the first experiment, the used dataset is MNIST. The used classification models are LR, DNN, and CNN. We train each of these models separately and then



(a) The trend of success rate. (b) The trend of best fitness value.

Fig. 3. The results of targeted attacks and non-targeted attacks for each undistilled model on MNIST.



(a) The trend of success rate. (b) The trend of best fitness value.

Fig. 4. The results of targeted attacks and non-targeted attacks for each undistilled model on CIFAR-10.

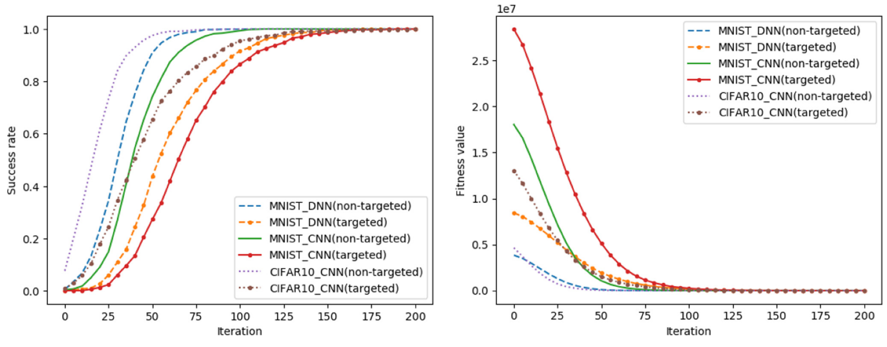
test the accuracy of each model on the test set. Logistic Regression (LR), DNN, and CNN achieve the accuracy of 92.46%, 98.49%, and 99.40% respectively. In the generation of adversarial examples, we set the number of iterations of the genetic algorithm is 200, and the sample with the smallest objective function value generated in each iteration is selected as the optimal sample. For a targeted attack, we select first 100 samples initially correctly classified from the test set to attack. Each of the samples generates adversarial examples from 9 different target labels, resulting in 100 * 9 = 900 corresponding target adversarial examples. For non-targeted attacks, we select the first 900 samples initially correctly classified from the test set to attack. Each sample generates a corresponding adversarial example, resulting in 900 non-target adversarial examples.

The results are shown in Table 2 and Fig. 3. For each model, our attacks find adversarial examples with less than 10 in the L_2 distance, and succeed with

Table 2. Comparison of our attacks with previous work for a number of MNIST models.

MNIST		Models							
		LR		DNN		CNN		CNN**	
		UD(Undistilled)	D(Distilled*)	UD	D*	UD	D*	UD	D*
Non-targeted attack	Mean	0.82	–	2.48	2.91	3.90	3.99	1.76	2.20
	SD	0.62	–	1.67	2.34	2.46	2.70	–	–
	Prob	100%	–	100%	99.89%	100%	100%	100%	100%
Targeted attack	Mean	3.65	–	5.04	7.93	7.60	8.33	–	–
	SD	3.80	–	2.88	5.95	4.12	5.34	–	–
	Prob	100%	–	100%	99.78%	100%	100%	–	–

(* The details of distilled model are shown in Sect. 4.3. There is no distilled LR model.
*** This model is attacked by the approach proposed by Carlini and Wagner [2].)



(a) The trend of success rate. (b) The trend of best fitness value.

Fig. 5. The results of targeted attacks and non-targeted attacks for each distilled model on MNIST and CIFAR-10. Compare to Fig. 3 and Fig. 4 for undistilled models.

100% probability. Compared with the results generated by Carlini and Wagner's attack [2], our perturbations are slightly larger than their results. However, both of our attacks succeed with 100% probability and our BANA is a black-box attack. Besides, there is no visual difference between the adversarial examples. Figure 3(a) and Fig. 3(b) show that as the model becomes more complex, the number of iterations required to produce an effective adversarial example increases. The distribution of the 900 best fitness values after 200 iterations is shown in Fig. 6(a). The figure indicates that the more complex the model, the larger the mean and standard deviation. The reason is that simple classification models do not have good decision boundaries. For the same classification model, non-targeted attacks require fewer iterations than targeted attacks, resulting in about 2× lower distortion and stability. Such result indicates that for the attacker the targeted adversarial example is generated at a higher cost. However, with the increasing of iterations, all the best fitness values tend to be 0. The difficulty caused by the targeted attack can be overcome by increasing the number of iterations. Overall, BANA is able to generate effective adversarial examples for LR, DNN, and CNN on MNIST.

By comparing the trend of success rate and best fitness values for targeted attack and non-targeted attack, respectively, it can be seen that the robustness of the classification model against adversarial examples is related to the complexity of the model, and the more complex the model, the better the robustness of the corresponding classification model.

4.2 Adversarial Example Generation on CIFAR-10 and ImageNet

In the second experiment, the used dataset is CIFAR-10. Our purpose is to find whether BANA is able to generate effective adversarial examples on CIFAR-10. Considering the conclusion in Sect. 4.1, we choose CNN as the classification model to be attacked. Our CNN achieves an accuracy of 77.82% on CIFAR-10. After generating the adversarial examples with BANA, we get the results shown in Fig. 3. Our attacks find adversarial examples with less than 2 in the L_2 distance and succeed with 100% probability. We can find the same conclusion as Sect. 4.1 from Fig. 4 and Table 3.

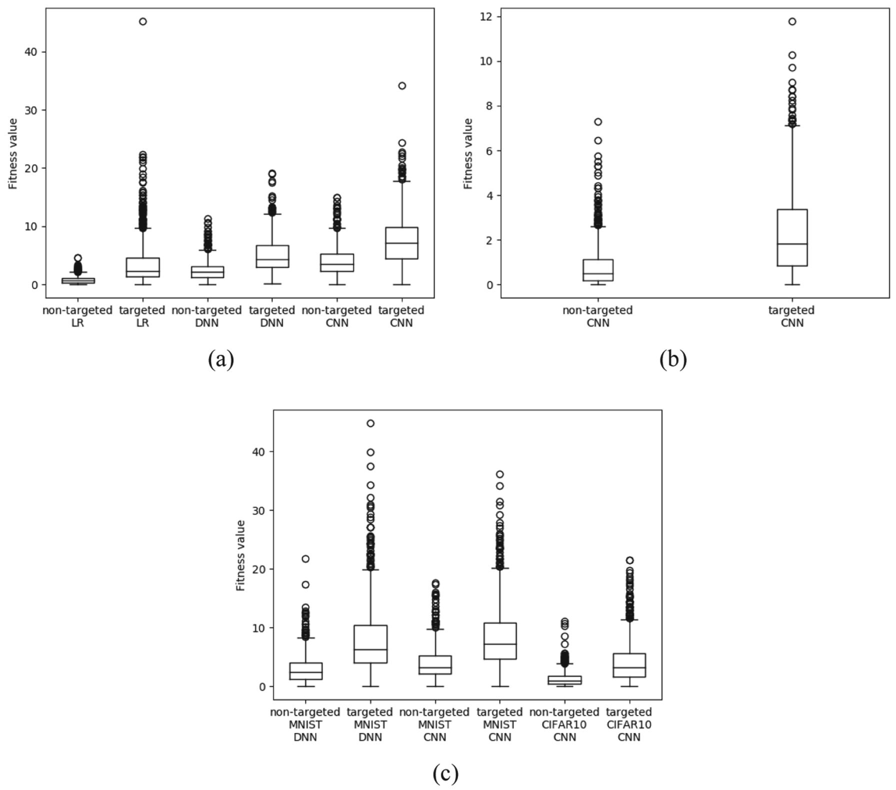


Fig. 6. The distribution of best fitness in Fig. 3, Fig. 4 and Fig. 5.

Figure 7 shows a case study of our BANA on ImageNet. As shown in Fig. 7, there is no visual difference between the original images and the perturbed images. Figure 7 shows that our attack is able to generate adversarial examples with small visually invisible perturbations even on complex datasat.

More importantly, by comparing the experimental results for CNN on MNIST and CIFAR, it can be seen that the average best fitness value and the standard deviation on CIFAR are smaller than them on MNIST, indicating that the adversarial examples generated on CIFAR dataset are more likely to be misleading and more similar to the original data. We find that the robustness of the classification model against adversarial examples is not only related to the complexity of the model but also to the trained data set; however, not the more complex the data set, the better the robustness of the generated classification model.

4.3 Defensive Distillation

We train the distilled DNN and CNN, using $softmax$ at temperature $T = 10$. The experimental results are shown in Tables 2 and 3. The observation is that the average fitness value and standard deviation of undistilled models are smaller than those of distilled model both on targeted attacks and non-targeted attacks. However, the attack success rate of the adversarial examples produced by BABA on the distilled model is still 100% or close to 100%. Our attack is able to break defensive distillation. The reason may be related to the randomness of the swarm evolutionary algorithm. Even with the same model and data, BANA produces a different adversarial example each time, making it effective against defensive distillation.

Table 3. Comparison of our attacks with previous work for a number of CIFAR models.

Models		CIFAR-10					
		Non-targeted attack			Targeted attack		
		Mean	SD	Prob	Mean	SD	Prob
CNN	Undistilled model	0.82	0.93	100%	2.33	1.89	100%
	Distilled model*	1.26	1.29	100%	4.18	3.57	99.89%
CNN**	Undistilled model	0.33	–	100%	–	–	–
	Distilled model*	0.60	–	100%	–	–	–

(* The details of distilled model are shown in Sect. 4.3.
*** This model is attacked by the approach proposed by Carlini and Wagner [2].)

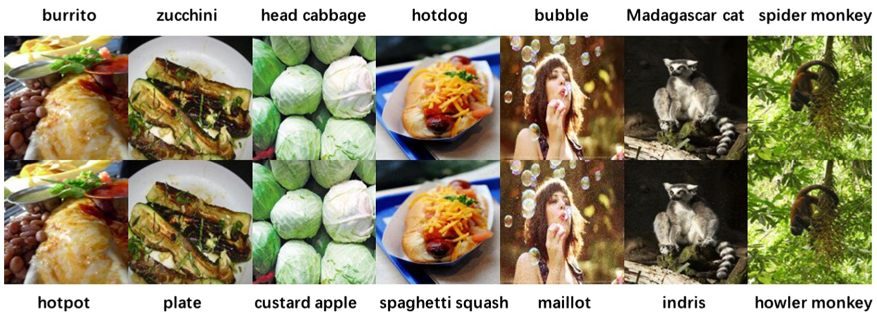


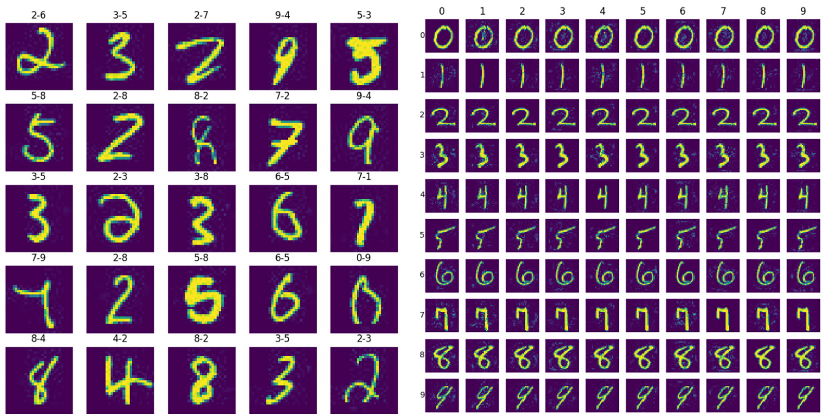
Fig. 7. A case study of our BANA on ImageNet. The top row shows the original images and the bottom row shows the perturbed images attacked by our approach.

5 Conclusions

In this paper, we have presented a new approach that generates a black-box attack to neural networks based on the swarm evolutionary algorithm. Our experimental results show that our approach generates high-quality adversarial examples for LR, DNN, and CNN, and our approach is resistant to defensive distillation. Finally, our results indicate that the robustness of the artificial intelligence algorithm is related to the complexity of the model and the complexity of the data set. Our future work includes designing an effective defense approach against our proposed attack.

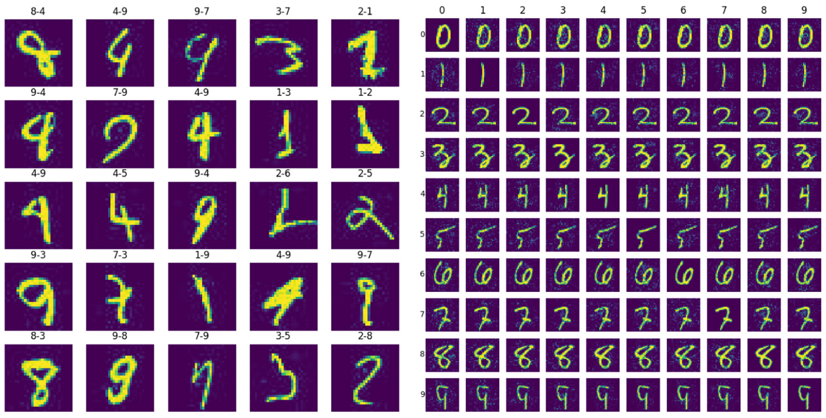
Appendix

Here we give some case studies of our experiment results to let you have a more intuitive visual experience, which are shown from Fig. 8, 9, 10, and 11. For reproducibility, all of these adversarial examples are generated by our method with the parameters shown in Table 1.



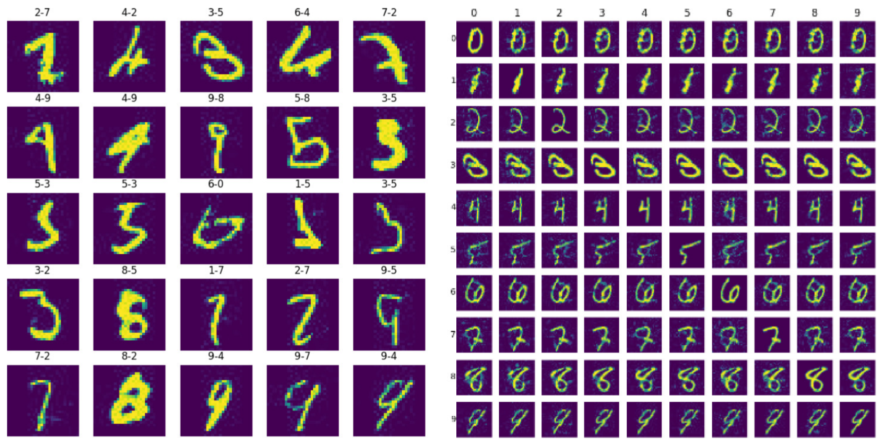
(a) non-targeted (b) targeted

Fig. 8. A case study of targeted attacks and non-targeted attacks for LR model on MNIST.



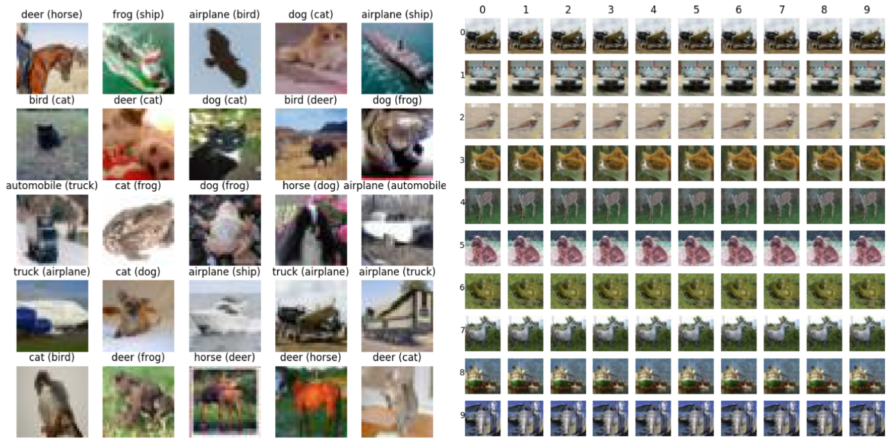
(a) non-targeted (b) targeted

Fig. 9. A case study of targeted attacks and non-targeted attacks for DNN model on MNIST.



(a) non-targeted (b) targeted

Fig. 10. A case study of targeted attacks and non-targeted attacks for CNN model on MNIST.



(a) non-targeted (b) targeted

Fig. 11. A case study of targeted attacks and non-targeted attacks for CNN model on CIFAR-10.

References

1. Bansal, J.C., Singh, P.K., Pal, N.R. (eds.): Evolutionary and Swarm Intelligence Algorithms. SCI, vol. 779. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91341-4
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57 (2017)

3. Demontis, A., et al.: Yes, machine learning can be more secure! a case study on android malware detection. IEEE Trans. Dependable Secure Comput. **16**, 711–724 (2017)
4. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4690–4699 (2019)
5. Dong, Y., Pang, T., Su, H., Zhu, J.: Evading defenses to transferable adversarial examples by translation-invariant attacks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4312–4321 (2019)
6. Eykholt, K., et al.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1625–1634 (2018)
7. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014). arXiv preprint arXiv:1412.6572
9. He, W., Wei, J., Chen, X., Carlini, N., Song, D.: Adversarial example defense: ensembles of weak defenses are not strong. In: 11th USENIX Workshop on Offensive Technologies WOOT 2017 (2017)
10. Hendrycks, D., Gimpel, K.: Early methods for detecting adversarial images. arXiv preprint arXiv:1608.00530 (2016)
11. Hossain, M.Z., Sohel, F., Shiratuddin, M.F., Laga, H.: A comprehensive survey of deep learning for image captioning. ACM Comput. Surv. (CSUR) **51**(6), 1–36 (2019)
12. Hu, W., Tan, Y.: Generating adversarial malware examples for black-box attacks based on GAN. arXiv preprint arXiv:1702.05983 (2017)
13. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report (2009)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
15. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., Jana, S.: Certified robustness to adversarial examples with differential privacy. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 656–672. IEEE (2019)
16. Li, T., Ruan, D., Geert, W., Song, J., Xu, Y.: A rough sets based characteristic relation approach for dynamic attribute generalization in data mining. Knowl. Based Syst. **20**(5), 485–494 (2007)
17. Liu, W., Luo, Z., Li, S.: Improving deep ensemble vehicle classification by using selected adversarial samples. Knowl. Based Syst. **160**, 167–175 (2018)
18. Liu, X., Du, X., Zhang, X., Zhu, Q., Wang, H., Guizani, M.: Adversarial samples on android malware detection systems for IoT systems. Sensors **19**(4), 974 (2019)
19. Liu, X., et al.: TLTD: a testing framework for learning-based IoT traffic detection systems. Sensors **18**(8), 2630 (2018)
20. Liu, X., Zhang, X., Wan, K., Zhu, Q., Ding, Y.: Towards weighted-sampling audio adversarial example attack. arXiv preprint arXiv:1901.10300 (2019)
21. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. arXiv preprint (2017)
22. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P., Soatto, S.: Analysis of universal adversarial perturbations. arXiv preprint arXiv:1705.09554 (2017)
23. 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, pp. 2574–2582 (2016)

24. Narodytska, N., Kasiviswanathan, S.P.: Simple black-box adversarial attacks on deep neural networks. In: CVPR Workshops, pp. 1310–1318 (2017)
25. Papernot, N., et al.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519 (2017)
26. Papernot, N., et al.: The limitations of deep learning in adversarial settings. In: IEEE European Symposium on Security and Privacy (EuroS&P) 2016, pp. 372–387 (2016)
27. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 582–597 (2016)
28. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
29. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. **23**(5), 828–841 (2019)
30. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
31. de Vos, B.D., et al.: A deep learning framework for unsupervised affine and deformable image registration. Med. Image Anal. **52**, 128–143 (2019)
32. Yang, W., Kong, D., Xie, T., Gunter, C.A.: Malware detection in adversarial settings: exploiting feature evolutions and confusions in android apps. In: Proceedings of the 33rd Annual Computer Security Applications Conference, pp. 288–302 (2017)