# An Improved Genetic Algorithm and Its Application in Neural Network Adversarial Attack

**Dingming Yang** 202071544@yangtzeu.edu.cn
School of Computer Science, Yangtze University, Jingzhou, 434023, China
**Zeyu Yu** yuzeyu_jz@163.com
School of Electronic & Information, Yangtze University, Jingzhou, 434023, China
**Hongqiang Yuan** 429809060@qq.com
School of Urban Construction, Yangtze University, Jingzhou, 434000, China **Yanrong Cui** cyanr@yangtzeu.edu.cn
School of Computer Science, Yangtze University, Jingzhou, 434023, China

**Abstract**
The choice of crossover and mutation strategies plays a crucial role in the search ability, convergence efficiency and precision of genetic algorithms. In this paper, a new improved genetic algorithm is proposed by improving the crossover and mutation operation of the simple genetic algorithm, and it is verified by four test functions. Simulation results show that, comparing with three other mainstream swarm intelligence optimization algorithms, the algorithm can not only improve the global search ability, convergence efficiency and precision, but also increase the success rate of convergence to the optimal value under the same experimental conditions. Finally, the algorithm is applied to neural networks adversarial attacks. The applied results show that the method does not need the structure and parameter information inside the neural network model, and it can obtain the adversarial samples with high confidence in a brief time just by the classification and confidence information output from the neural network.

**Keywords**
Genetic algorithms, swarm intelligence optimization algorithms, algorithm improvement, neural network adversarial attack.

## 1 Introduction

In real life, optimization problems such as shortest path, path planning, task scheduling, parameter tuning, etc. are becoming more and more complex and have complex features such as nonlinear, multi-constrained, high-dimensional, and discontinuous (Deng et al., 2021). Although a series of artificial intelligence algorithms represented by deep learning can solve some optimization problems, they lack mathematical interpretability due to the existence of a large number of nonlinear functions and parameters inside their models, so they are difficult to be widely used in the field of information security. Traditional optimization algorithms and artificial intelligence algorithms can hardly solve complex optimization problems with high dimensionality and nonlinearity in the field of information security.

Therefore, it is necessary to find an effective optimization algorithm to solve such problems. In this background, various swarm intelligence optimization algo-

rithms have been proposed one after another, such as Particle Swarm Optimization(PSO)(Kennedy and Eberhart, 1995; Eberhart and Kennedy, 1995),Grey Wolf Optimizer(GWO)(Mirjalili et al., 2014), etc. Subsequently, a variety of improved optimization algorithms also have been proposed one after another. For example, the improved genetic algorithm for cloud environment task scheduling(Zhou et al., 2020), the improved genetic algorithm for flexible job shop scheduling(Zhang et al., 2020), the improved genetic algorithm for green fresh food logistics(Li et al., 2020), etc.

However, these improved optimization algorithms are improved for domain-specific optimization problems, and there is no improvement on the precision, convergence efficiency and generalization of the algorithms themselves. In this paper, the crossover operator and mutation operator of the genetic algorithm are improved to improve the convergence efficiency and precision of the algorithm without affecting the effectiveness of the improved genetic algorithm on most of optimization problems. The effectiveness of the improved genetic algorithm is also verified through many comparison experiments and applications in the field of neural network adversarial attacks. The source code of this paper has been released on Github(Yang, 2021).

## 2 Related Works

### 2.1 Genetic Algorithm

Genetic Algorithm is a series of simulation evolutionary algorithms proposed by Holland et al. (1975), and later summarized by DeJong, Goldberg and others. The general flowchart of Genetic Algorithm is shown in Figure 1. The Genetic Algorithm first encodes the problem, then calculates the fitness, then selects the parent and the mother by roulette, and finally generates the children with high fitness by crossover and mutation, and finally generates the individuals with high fitness after many iterations, which is the satisfied solution or optimal solution of the problem. Simple Genetic Algorithm (SGA) uses single-point crossover and simple mutation to embody information exchange between individuals and local search, and does not rely on gradient information, so SGA can find the global optimal solution.

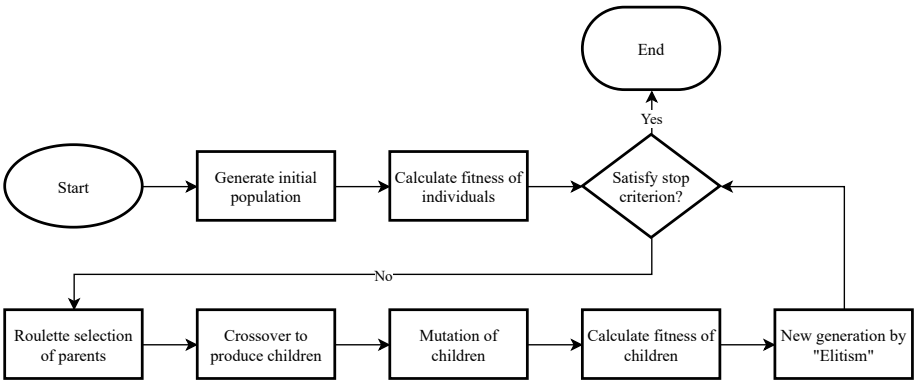



Figure 1: Genetic algorithm flowchart

### 2.2 Neural Network Adversarial Attack

Szegedy et al. (2013) first demonstrated that a highly accurate deep neural network can be mislead to make a misclassification by adding a slight perturbation to an image

that is imperceptible to the human eye, and also found that the robustness of deep neural networks can be improved by adversarial training. Such phenomena are far-reaching and have attracted many researchers in the area of adversarial attacks and deep learning security. Akhtar and Mian (2018) surveyed 12 attack methods and 15 defense methods for neural networks adversarial attacks. The main attack methods are finding the minimum loss function additive term (Szegedy et al., 2013), increasing the loss function of the classifier (Kurakin et al., 2016), the method of limiting the l_0 norm (Papernot et al., 2016), changing only one pixel value (Su et al., 2019), etc.

Nguyen et al. (2015) continued to explore the question of "what differences remain between computer and human vision" based on Szegedy et al. (2013). They used the Evolutionary Algorithm to generate high-confidence adversarial images by iterating over direct-encoded images and CPPN (Compositional Pattern-Producing Network) encoded images, respectively. They obtained high-confidence adversarial samples (fooling images) using the Evolutionary Algorithm on a LeNet model pre-trained on the MNIST dataset (LeCun, 1998) and on an AlexNet model pre-trained on the ILSVRC 2012 ImageNet dataset (Deng et al., 2009; Russakovsky et al., 2015), respectively.

Neural network adversarial attacks are divided into black-box attacks and white-box attacks. Black-box attacks do not require the internal structure and parameters of the neural network, and the adversarial samples can be generated with optimization algorithms as long as the output classification and confidence information are known. The study of neural network adversarial attacks not only helps to understand the working principle of neural networks, but also increases the robustness of neural networks by training with adversarial samples.

## 3 Approaches

This section improves the single-point crossover and simple mutation of SGA. The fitness function is used as the evaluation index of the crossover link, and the crossover points of the whole chromosome are traversed to improve the efficiency of the search for the best. Selective mutation is performed for each gene of the children's chromosome, and the mutation rate of the latter half of the chromosome is set to twice that of the first half to improve the global search under the stable situation of local optimum.

### 3.1 Improved Crossover Operation

As shown in algorithm 1 is the Python pseudocode for the improved crossover algorithm. The single-point crossover of SGA is to generate random number within the parental chromosome length range, and then intercept the first half of the father's chromosome and the second half of the mother's chromosome to cross-breed the children according to the generated random number. In this paper, the algorithm is improved by trying to cross genes within the parental chromosome length range one by one, calculating the fitness, and picking out the highest fitness children individuals. Experimental data show that such an improvement can reduce the number of iteration and speed up the convergence of fitness.

**Algorithm 1** Crossover with fitness as evaluation.

**Input:** Father's gene, mother's gene, fitness function;
**Output:** Child's gene;

**function** CROSSOVER($father, mother, fitness$)
    $best\_fitness = float.MIN\_VALUE$;
    $best\_child = np.zeros(father.size)$;
    **for** $i = 0 \rightarrow father.size$ **do**
        $current\_child = np.zeros(father.size)$;
        $current\_child = np.append(father[0:i], mother[i:])$;
        $current\_fitness = fitness(current\_child)$;
        **if** $current\_fitness > best_fitness$ **then**
            $best\_fitness = current\_fitness$;
            $best\_child = current\_child.copy()$;
        **end if**
    **end for**
    **return** $best_child$
**end function**

### 3.2 Improved Mutation Operation

As shown in algorithm 2 is the pseudocode of the improved mutation algorithm. The simple mutation of SGA sets a relatively large mutation rate, and mutates any one gene of the incoming children's chromosome when the generated random number is smaller than the mutation rate. In this paper, we improve the algorithm by setting a small mutation rate and then selectively mutating each gene of the incoming children's chromosome. That is, when the generated random number is smaller than the mutation rate, the gene is mutated, and when the traversed gene position is larger than half of the chromosome length, the mutation rate is set to twice the original one (the second half of the gene has relatively less influence on the result). This ensures that the first half of the gene and the second half of the gene have equal chance of mutation respectively, and can mutate at the same time. When the gene length is $784$, the mutation rate of the whole chromosome is $1-(1-0.025)^{392}(1-0.05)^{392}$, which greatly improves the species diversity and at the same time ensures the stability of the species (in the stable situation of the local optimum improves the global search ability), and experimental data show that it can improve the search ability.

**Algorithm 2** Mutate child with alter each gene if rand number less than mutate rate.

**Input:** Child's gene;
**Output:** Mutated child's gene;

```
function MUTATE(child)
    mutate_rate = 0.025;
    for i = 0 → child.size do
        if i > child.size//2 then
            mutate_rate = 0.05;
        end if
        if random.random() < mutate_rate then
            child[i] =!child[i];//child[i] equals 0 or 1
        end if
    end for
    return child
end function
```

## 4 Numerical Experiments and Analysis

### 4.1 Test Functions

In order to evaluate the optimization performance of the proposed improved genetic algorithm, four representative test functions from Wikipedia (2021) are selected in this paper. Since the proposed improved genetic algorithm is mainly used for the neural network adversarial attack problem, and the neural network has multi-dimensional parameters, the low-dimensional test functions are not selected. The expressions of the four test functions are shown in the formula (1)(2)(3)(4), the name, global minimum and search domain of the test functions are shown in Table 1, and the images of the corresponding test functions are shown in Figure 2.

$$f(x,y) = x^2 + y^2 \tag{1}$$

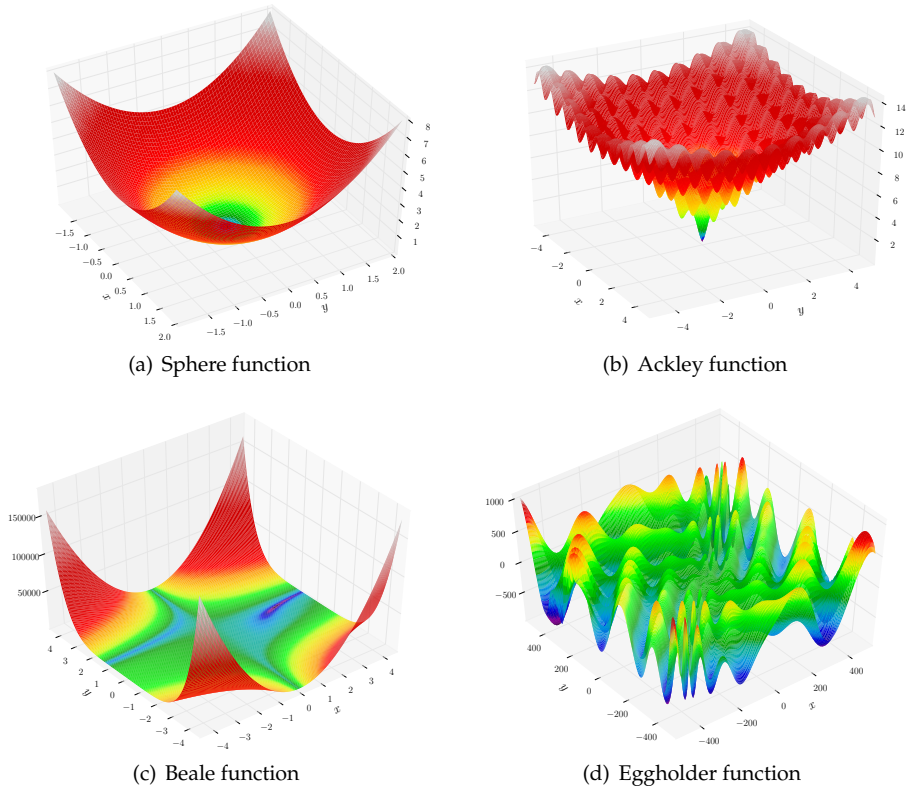
$$f(x,y) = -20\exp\left[-0.2\sqrt{0.5\left(x^2+y^2\right)}\right] - \exp[0.5(\cos 2\pi x + \cos 2\pi y)] + e + 20 \tag{2}$$

$$f(x,y) = (1.5 - x + xy)^2 + \left(2.25 - x + xy^2\right)^2 + \left(2.625 - x + xy^3\right)^2 \tag{3}$$

$$f(x,y) = -(y+47)\sin\sqrt{\left|\frac{x}{2} + (y+47)\right|} - x\sin\sqrt{|x-(y+47)|} \tag{4}$$

Table 1: Test functions

| Name | Global minimum | Search domain |
|---|---|---|
| Sphere function | $f(0,0) = 0$ | $-5 \le x, y \le 5$ |
| Ackley function | $f(0,0) = 0$ | $-5 \le x, y \le 5$ |
| Beale function | $f(3, 0.5) = 0$ | $-4.5 \le x, y \le 4.5$ |
| Eggholder function | $f(512, 404.2319) = -959.6407$ | $-512 \le x, y \le 512$ |



(a) Sphere function (b) Ackley function

(c) Beale function (d) Eggholder function

Figure 2: Schematic diagrams of test function

### 4.2 Experimental Environment

The hardware environment of the experiment includes 8G of RAM, i7-4700MQ CPU; the software environment includes Windows 10 system, and the version of Python is 3.8.8. In order to compare the optimization performance of IGA, SGA (Simple Genetic Algorithm), PSO (Particle Swarm Optimization) and GWO (Grey Wolf Optimizer) are selected as the experimental objects for comparison experiments in this paper. The parameters of the 4 optimization algorithms are shown in Table 2, and the population size and the number of iterations are kept the same for the convenience of comparison. The other parameters in PSO are as follows: $w = 1$, $c_1 = c_2 = 1.49445$. The other parameters of GWO are set to typical values: $\vec{C} = \mathrm{Rand}(0, 2)$, $\vec{a} = \mathrm{Rand}(-a, a)$, $a = 2 \to 0$.

Table 2: The parameter settings

| Algorithm | Iteration | Population size | Gene length | Mutation rate |
|---|---|---|---|---|
| IGA | 101 | 50 | 30 | 0.05 |
| SGA | 101 | 50 | 30 | 0.2 |
| PSO | 101 | 50 | - | - |
| GWO | 101 | 50 | - | - |

### 4.3 Experimental Results and Analysis

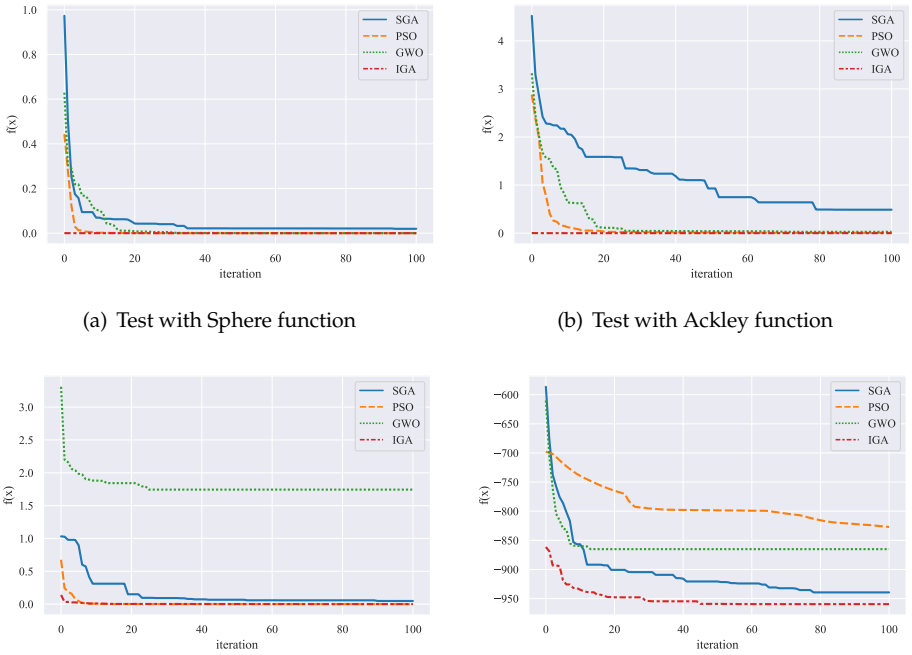
The average convergence curves of each optimization algorithm tested 10 times with the four test functions under the conditions of the same experimental environment are shown in Figure 3. From the figure, it can be seen that: among the four tested functions, IGA is converged before the other three optimization algorithms, and the precision after convergence is better. As shown in Table 3, the comparison results of the four optimization algorithms after 101 iterations are shown. From the table, we can see that the convergence success rate of IGA in the condition with precision of 0.15 among the four test functions is $100\%$. Two independent sample t-test were done using the formula (5), where $S_1^2$ and $S_2^2$ are the two samples' variance; $n_1$ and $n_2$ are the two samples' volume. The P value of the t-test in the table show that IGA performs as well as GWO in $f_1$ and $f_2$, and far better than the other three optimization algorithms in $f_4$. It is noteworthy that IGA has a very significant performance advantage in $f_4$; its performance in $f_3$, where the gradient is less pronounced, is not as well as PSO, but is also much better than SGA and GWO.

As shown in Figure 4, the population distributions of the four optimization algorithms at the last iteration in the four test functions are shown. Among them, Figure 4(a)4(b)4(c)4(d) is the scatter plot of the distribution of all individuals for each optimization algorithm in 10 experiments, and the formula for the density is shown in (6), $population\_size = 50$. Figure 4(e)4(f)4(g)4(h) shows the scatter plot of the distribution of the optimal individuals for each experiment, and the formula for calculating the intensity is shown in (7), $test\_n = 10$. From the figure, we can see that the density of optimal individuals for each round of experimental IGA is better than the other three optimization algorithms, and also retains a strong global search capability in the last iteration. As shown in Figure 4(d)4(h), SGA, PSO and GWO fall into local optimum several times, among them, PSO has the population distribution near the local optimum in the last iteration, and the global search ability is weak because it does not have the function of adaptive variation.

In general, IGA has better iteration efficiency, global search capability, and convergence success rate than the other three optimization algorithms.

Table 3: The comparison optimization results on $f_1$–$f_4$ with $iteration = 101$

| Fun | Alg | Prob(.15) | Min | Max | Mean | Median | Std | T-test | |
|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | SGA | **100** | 2.54E-04 | 6.31E-02 | 1.95E-02 | 6.19E-03 | 2.41E-02 | 3.08E-02 | - |
| | PSO | **100** | 1.89E-07 | 3.39E-05 | 1.26E-05 | 6.56E-06 | 1.33E-05 | 1.58E-02 | - |
| | GWO | **100** | 6.36E-18 | 1.97E-03 | 2.35E-04 | **4.81E-11** | 6.21E-04 | **2.62E-01** | = |
| | IGA | **100** | **0.00E+00** | **9.31E-08** | **6.52E-08** | 9.31E-08 | **4.50E-08** | **1.00E+00** | = |
| $f_2$ | SGA | 10 | 4.34E-02 | 7.16E-01 | 4.88E-01 | 5.31E-01 | 2.04E-01 | 3.53E-05 | - |
| | PSO | **100** | 1.51E-03 | 1.37E-02 | 6.55E-03 | 6.44E-03 | 3.92E-03 | 7.75E-04 | - |
| | GWO | 90 | 1.95E-07 | 2.17E-01 | 2.86E-02 | **2.16E-04** | 6.81E-02 | **2.23E-01** | = |
| | IGA | **100** | **4.44E-16** | **8.66E-04** | **4.33E-04** | 4.33E-04 | **4.56E-04** | **1.00E+00** | = |
| $f_3$ | SGA | 90 | 8.29E-04 | 1.54E-01 | 4.80E-02 | 2.40E-02 | 5.81E-02 | 2.94E-02 | - |
| | PSO | **100** | **2.14E-06** | **1.29E-04** | **4.82E-05** | **3.86E-05** | **3.95E-05** | 2.06E-02 | + |
| | GWO | 0 | 2.32E-01 | 5.95E+00 | 1.74E+00 | 1.13E+00 | 1.74E+00 | 1.14E-02 | - |
| | IGA | **100** | 2.50E-05 | 1.44E-03 | 4.90E-04 | 3.25E-04 | 4.98E-04 | **1.00E+00** | = |
| $f_4$ | SGA | 0 | -9.54E+02 | -9.02E+02 | -9.39E+02 | -9.45E+02 | 1.64E+01 | 3.67E-03 | - |
| | PSO | 10 | **-9.60E+02** | -5.72E+02 | -8.27E+02 | -8.81E+02 | 1.43E+02 | 1.66E-02 | - |
| | GWO | 10 | **-9.60E+02** | -7.48E+02 | -8.65E+02 | -8.73E+02 | 8.05E+01 | 4.87E-03 | - |
| | IGA | **100** | **-9.60E+02** | **-9.60E+02** | **-9.60E+02** | **-9.60E+02** | **7.05E-03** | **1.00E+00** | = |




(a) Test with Sphere function

(b) Test with Ackley function

(c) Test with Beale function

(d) Test with Eggholder function

Figure 3: The average convergence curves of 4 different optimization algorithms

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{(n_1-1)S_1^2+(n_2-1)S_2^2}{n_1+n_2-2}\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}} \tag{5}$$

$$\text{density} = \frac{1}{population\_size}\sum_{i=1}^{population\_size} \text{dist}\left(a_i, o\right) \tag{6}$$

$$\text{density} = \frac{1}{test\_n}\sum_{i=1}^{test\_n} \text{dist}\left(a_{best}, o\right) \tag{7}$$

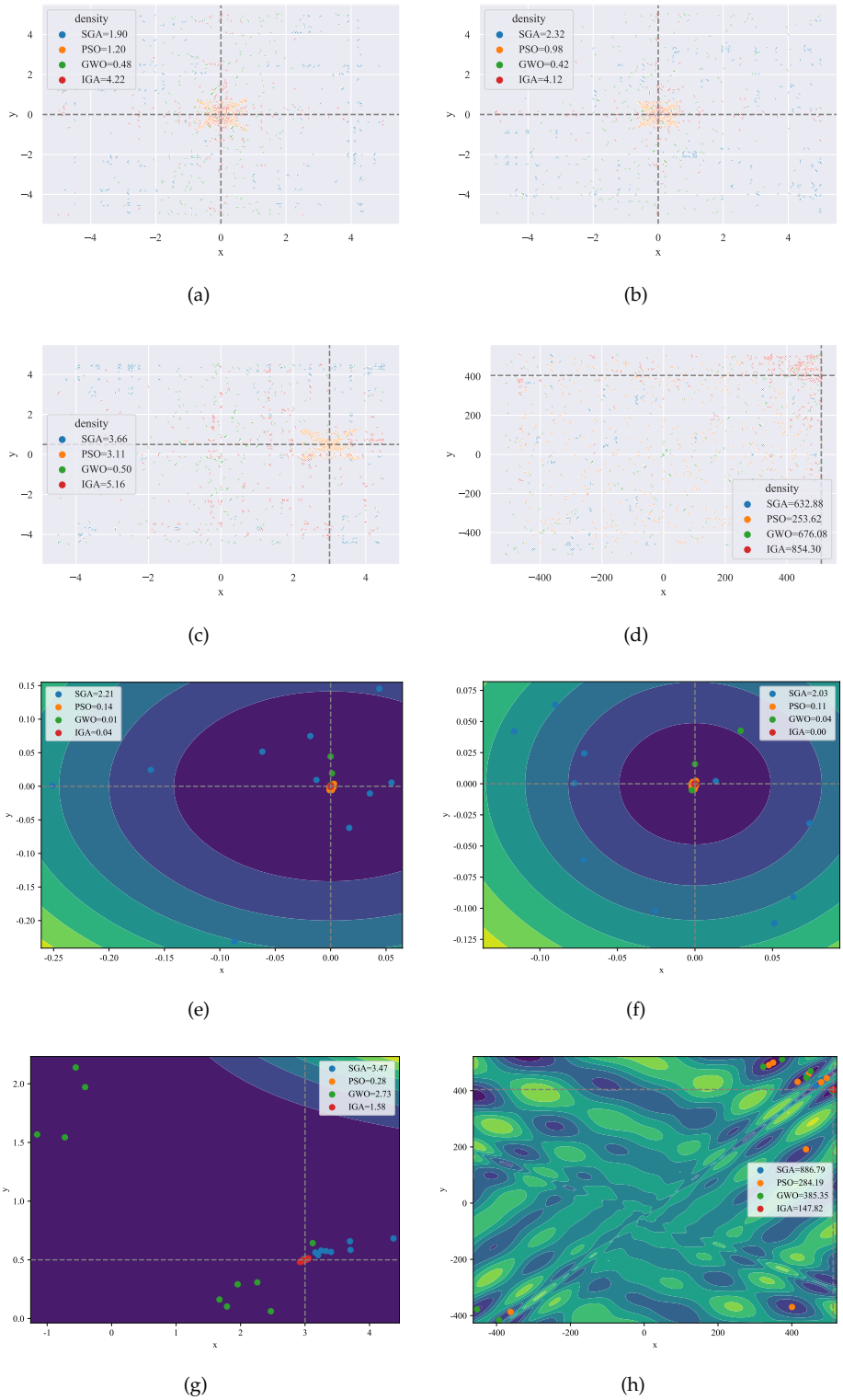



Figure 4: Final population in last iteration

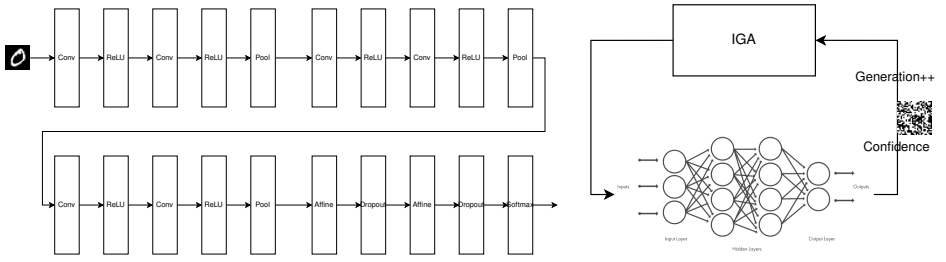
## 5 Application in Neural Network Adversarial Attack

### 5.1 MNST Dataset

The MNST dataset (Mixed National Institute of Standards and Technology database)(LeCun, 1998) is one of the most well-known datasets in the field of machine learning and is used in applications from simple experiments to published paper research. It consists of handwritten digital images from 0-9. The MNIST image data is a single-channel grayscale map of $28 \times 28$ pixels, with each pixel taking values between 0 and 255, with 60,000 samples in the training set and 10,000 samples in the test set. The general usage of the MNIST dataset is to learn with the training set first and then use the learned model to measure how well the test set can be correctly classified (Yasue, 2018).

### 5.2 Implementation

As shown in Figure 5(a), the Deep Convolutional Neural Network (DCNN) pre-trained on the MNST dataset (LeCun, 1998) is used as the experimental object in this paper, and the accuracy of the model is $99.35\%$ with a Loss value of 0.9632. As shown in Figure 5(b), the model of network adversarial attack is shown. The number of populations of a specific size (set to 100 in this paper) is first generated and then input to the neural network to obtain the confidence of the specified labels. To reduce the computational expense, the input is reduced to a binary image of $28 \times 28$ and the randomly generated binary image is iterated using the IGA proposed in this paper. Among the 100 individuals, the fathers and mothers with relatively high confidence are selected by roulette selection, and then the children are generated by using the improved crossover link in this paper, and the children form new population by improving the mutation link until the specified number of iterations. Finally, the individual with the highest confidence is picked from the 100 individuals, which is the binary image with the highest confidence after passing through the neural network.




(a) The structure of DCNN for experiment

(b) The model of network adversarial attack

Figure 5: The model of network adversarial attack

### 5.3 Result

As shown in Figure 6, the confidence after 99 iterations of DCNN is $99.98\%$ for sample "2". Sample "6" and sample "4" have the slowest convergence speed, and the confidence of sample "6" is $78.84\%$ after 99 iterations, and the confidence of sample "4" is $78.84\%$ after 99 iterations.

The statistics of the experimental results are shown in Table 4. The binary image

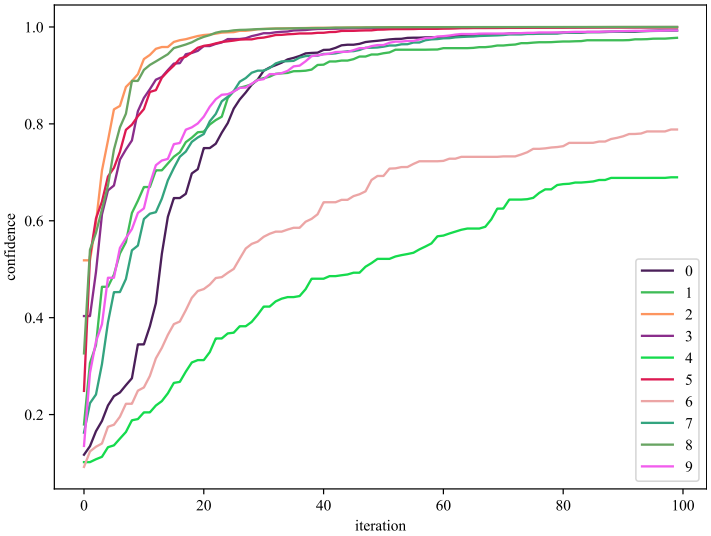



Figure 6: The confidence change of the binary image after iteration

of sample "1" generated after 999 iterations has a confidence of $99.94\%$ after passing DCNN, which is much higher than the confidence of sample "1" in the MNIST test set in the DCNN control group. In the statistics of the results after initializing the population with the MNIST test set, because the overall confidence of the population initialized with the test set is higher, the increase in confidence during iteration is smaller. The confidence of the sample selected from the MNIST test set is $99.56\%$, and after 10 iterations the confidence of the sample is $99.80\%$, and the number "1" becomes vertical; after 89 iterations the confidence is $99.98\%$, and the number "1 " has a tendency to "decompose" gradually.

Table 4: Statistical table of experimental results

| Figure | Label | Initialize | Iteration | Confidence |
|---|---|---|---|---|
|  | 0 | random | - | 96.71% |
|  | 0 | random | 204 | 99.61% |
|  | 1 | random | - | 99.44% |
|  | 1 | random | 999 | 99.94% |
|  | 1 | test dataset | - | 99.56% |
|  | 1 | test dataset | 10 | 99.80% |
|  | 1 | test dataset | 89 | 99.98% |

As shown in Figure 7, the reason for this situation is probably that the confidence as a function of the image input is a multi-peak function, and the interval in which the

test set images are distributed is not the highest peak of the confidence function. This causes the initial population of the test set to ”stray” from some pixels in the images generated by the IGA.

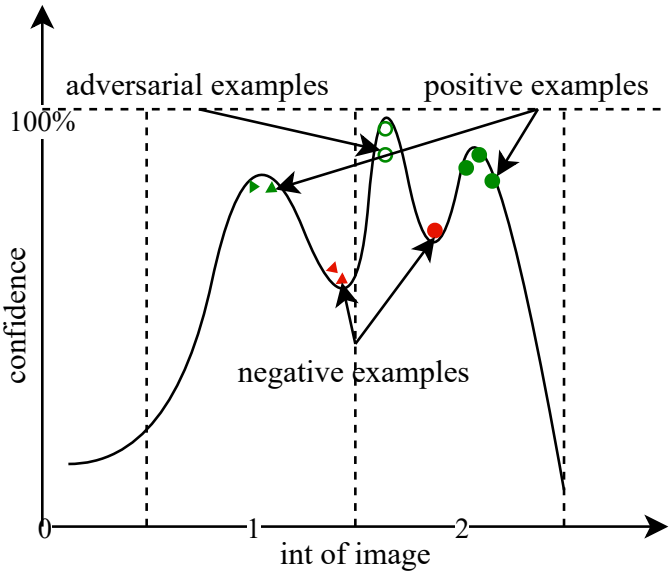



Figure 7: The confidence curve of a binary image

## 6 Conclusion

The comparison and simulation experiments show that the improved method proposed in this paper is effective and greatly improves the convergence efficiency, global search ability and the convergence success rate. Applying IGA to the field of neural network adversarial attacks can also quickly obtain adversarial samples with high confidence, which is meaningful for the improvement of the robustness and security of neural network models.

With the widely application of artificial intelligence and deep learning in the field of computer vision, face recognition has outstanding performance in access control systems and payment systems, which require fast response to the input face image, but this has instead become a drawback to be hacked. For face recognition systems without in vivo detection, using the method in this paper only requires output labels and confidence information can obtain high confidence images quickly. In summary, neural networks have many pitfalls due to their uninterpretability and still need to be considered carefully for using in important areas.

## References


Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Deng, W., Shang, S., Cai, X., Zhao, H., Song, Y., and Xu, J. (2021). An improved differential evolution algorithm and its application in optimization problem. *Soft Computing*, 25(7):5277–5298.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee.

Holland, J. H. et al. (1975). Adaptation in natural and artificial systems.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

Kurakin, A., Goodfellow, I., Bengio, S., et al. (2016). Adversarial examples in the physical world.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

Li, D., Cao, Q., Zuo, M., and Xu, F. (2020). Optimization of green fresh food logistics with heterogeneous fleet vehicle route problem by improved genetic algorithm. *Sustainability*, 12(5):1946.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69:46–61.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Wikipedia (2021). Test functions for optimization. Website. `https://en.wikipedia.org/wiki/Test_functions_for_optimization`.

Yang, D. (2021). Neural network adversarial attack method based on improved genetic algorithm. Website. `https://github.com/huangyebiaoke/adversarial-attack-method-based-on-IGA`.

Yasue, S. (2018). *Deep Learning from Scratch*. "Beijing: Posts and Telecom Press".

Zhang, G., Hu, Y., Sun, J., and Zhang, W. (2020). An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. *Swarm and Evolutionary Computation*, 54:100664.

Zhou, Z., Li, F., Zhu, H., Xie, H., Abawajy, J. H., and Chowdhury, M. U. (2020). An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Computing and Applications*, 32(6):1531–1541.