ISSN: 2664-9640

Generating Unidirectional Fault Data with Swarm Intelligence-Optimized Generative Adversarial Networks

Zeqing Xiao*

School of Mathematical Sciences, Changsha Normal University, Changsha 410100, Hunan, China

Email: zxiao@csnu.edu.cn

Abstract

The scale of data acquisition is limited by the calculation ability of software (such as Power System Analysis Synthesis Program) or hardware for simulating voltage fault data. By combining generative adversarial networks (GAN) with swarm intelligent algorithm, that is, particle swarm optimization (PSO) algorithm and genetic algorithm (GA), the bus voltage data of unidirectional fault is generated. In the paper, the data are divided into two parts. The first part is the normal data before the fault, and the second part is the data after the fault. Results of PSO optimization, GA optimization, and random number (RN) of GAN parameter values are compared and analyzed. The average relative error is used as the performance evaluation index of the data generated by GAN. Results show that PSO is better than GA in time consumption and generating error. Although the time consumption of RN method is less than that of PSO and GA, it is too random and blind. The swarm intelligent algorithm avoids using artificial experience or repeating trials in selecting parameters. The analysis of normal data before fault shows that compared with GA and RN, the performance of PSO is improved by 85.6% and 94.9%, respectively. In addition, PSO optimization method takes 37.7% less time than GA optimization. The analysis of data after fault shows that compared with GA and RN, the performance of PSO is improved by 79.9% and 91.9%, respectively. In addition, PSO optimization method takes 7.5% less time than GA optimization. The convergence curves of the generator (G) and discriminator (D) of GAN optimized by PSO show that good convergence is achieved when the optimization parameters are taken.

Keywords

Generative adversarial networks, Particle swarm optimization, Genetic algorithm, Unidirectional Fault.

1. Introduction

Goodfellow [1] developed generative adversarial networks (GAN) in 2014 as a novel semi-supervised and unsupervised learning technology. It is characterized by training a pair of competing networks to master the distribution of the overall real data. GAN has attracted many researchers since it was proposed, and they have shown the superiority of GAN in solving data generation problems.

In GAN, the selection of operating parameters has serious impact on the performance of the algorithm. For different optimization problems, the selection of parameters varies. Even if the same kind of optimization problem is concerned, the selection of parameters varies according to the scale of the problem. Given that GAN is relatively new, it has not yet built a complete theoretical system. In addition, the algorithm parameter selection has no theoretical foundation. At present, the GAN algorithm parameters of a problem are usually specified by a mass of experience [2-7]. In this paper, we demonstrate the good performance of GAN algorithms when

ISSN: 2664-9640

parameters are selected by using a systematic procedure. Specifically, we use particle swarm optimization (PSO) and genetic algorithm (GA) to optimize GAN generating bus voltage data of unidirectional fault. In the proposed algorithm, PSO and GA are used to optimize the parameters in the adversarial system to improve the performance of the GAN, which prevents the parameter selection of the GAN from artificial experience or repeating trials seriously, and instead relies on self-adaptive search. In addition, the GAN parameter values are randomly selected to generate sample data. The results of GAN generation by these three methods are compared and analyzed.

2. Gan, Pso, and Ga

2.1. GAN

In this section, we briefly describe GAN algorithm introduced by Goodfellow. Figure 1 shows the architecture of GAN. G is called the generator whose purpose is to make realistic samples. The input of G is random noise vector, which is usually subject to uniform distribution or normal distribution. D is the discriminator, which receives forged samples from the generator and real samples, and aims to distinguish these two types of samples. The output of D is the probability that the samples are real rather than false. G and D are trained at the same time and compete with each other. When D cannot determine whether the data are from the real data set or from G, the Nash equilibrium is achieved. At this time, the obtained G has learned the distribution of the real data, so as to master the distribution of the overall real data. GAN no longer needs massive data with category label like the traditional supervised deep learning, and can be trained without any data annotation, that is, deep learning without supervision. G and D play a minimax game with the following value function $V(G, D)$ [1],

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

where x represents real data; z represents random noise; $D(x)$ represents the probability that D judges the input data as real data; $G(z)$ represents the data generated after z is accepted by G; $p_{data}(x)$ represents the distribution of real data; and $p_z(z)$ represents the distribution of generated data.

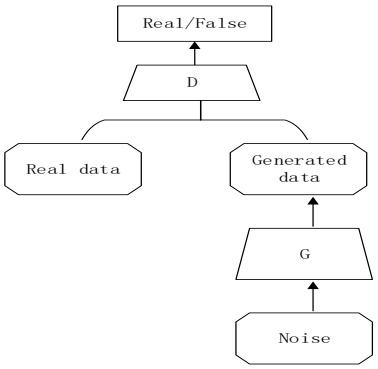

Fig. 1. GAN structure

When G and D are multi-layer perceptrons, applying the adversary modeling framework is easy. To learn the distribution p_g of G on data x, a priori is defined on the input noise variable $p_z(z)$. Then the mapping of data space is expressed as $G(z; \theta_g)$, where G is a differentiable function represented by the multi-layer perceptron of parameter θ_g. The second multi-layer perceptron $D(x; \theta_d)$ that outputs a single scalar is defined. D is trained, so that it can assign the correct label to the training examples and samples from G to the maximum extent. Meanwhile, G is trained to minimize

ISSN: 2664-9640

$\log(1 - \mathrm{D}(\mathrm{G}(\mathrm{z})))$.

The learning rates of G and D (r_G, r_D) are two important parameters in the training process of GAN. Even a small change in learning rate will lead to a fundamental change in the training process of GAN. Improper selection of learning rate of G and D will lead to mode collapse [3]. In the previous studies on GAN, learning rate was determined according to experience or repeated attempts [2,3,6,7]. In addition, batch size (B), another important parameter of GAN, will affect the convergence rate of the network in the GAN, and selecting an inappropriate batch size will even lead to the phenomenon of non-convergence of the network. The determination of B is not guided by systematic theory, but also by experience or repeated attempts, similar to the determination of learning rate.

2.2. PSO and GA

PSO algorithm is a population stochastic heuristic optimization algorithm based on social psychology metaphor [8,9,10]. PSO algorithm has the advantages of simplicity, easy implementation, no gradient information, few parameters, fast search speed, and high efficiency. It has achieved good results in continuous optimization and discrete optimization problems, and its natural real number coding characteristics are more suitable for solving real number optimization problems. In PSO algorithm, the population and individual are called swarm and particle, respectively. Each particle represents a solution to the problem in D dimension space [10]. The state of a particle is characterized by its position and velocity. When iterating k times, the position and velocity of particle i can be represented by D dimension vector $X_i^k = \{x_{i1}^k,\ x_{i2}^k, \cdots, x_{iD}^k\}$ and $V_i^k = \{v_{i1}^k,\ v_{i2}^k, \cdots, v_{iD}^k\}$, respectively. The fitness of each particle can be evaluated by using the objective function of the optimization problem. In each iteration, each particle accelerates to the previous best position and the global best position [11]. The best previous position of the i-th particle until the k-th iteration is expressed as $Pbest_i^k = \{p_{i1}^k,\ p_{i2}^k, \cdots, p_{iD}^k\}$. The best position in the whole group is expressed as $Gbest = \{g_1^k,\ g_2^k, \cdots, g_D^k\}$. To find the best solution, each particle updates its speed and position according to the following two formulas, respectively.

$$v_{id}^k = \omega \cdot v_{id}^{k-1} + c_1 \cdot r_1 \cdot \left(p_{id}^{k-1} - x_{id}^{k-1}\right) + c_2 \cdot r_2 \cdot (g_d^{k-1} - x_{id}^{k-1})$$
$$x_{id}^k = x_{id}^{k-1} + v_{id}^k$$
$$i = 1, 2, \cdots, n; \qquad d = 1, 2, \cdots, D \tag{2}$$

where learning factor c_1 and c_2 are cognitive and social scaling parameters, respectively, which are used to control the maximum step size; r_1 and r_2 are random numbers uniformly distributed in the interval (0, 1). ω is the inertia weight factor, which controls the influence of the particle's previous velocity on the current velocity [10]. Generally, ω decreases linearly with each iteration from ω_{max} to ω_{min}, which can be described as follows:

$$\omega = \omega_{max} - t \cdot (\omega_{max} - \omega_{min})/T_{max} \tag{3}$$

where t and T_{max} represent the current iteration time and the maximum number of iterations, respectively.

GA is a set of optimal process search algorithms based on the mechanism of natural selection and natural genetics [12]. The main characteristics of GA algorithm are the search method among groups and the exchange of individual information in groups. GA is very suitable for solving nonlinear problems that are difficult to be solved by traditional search methods. Figure 2 [13] shows the flowchart of GA.

ISSN: 2664-9640

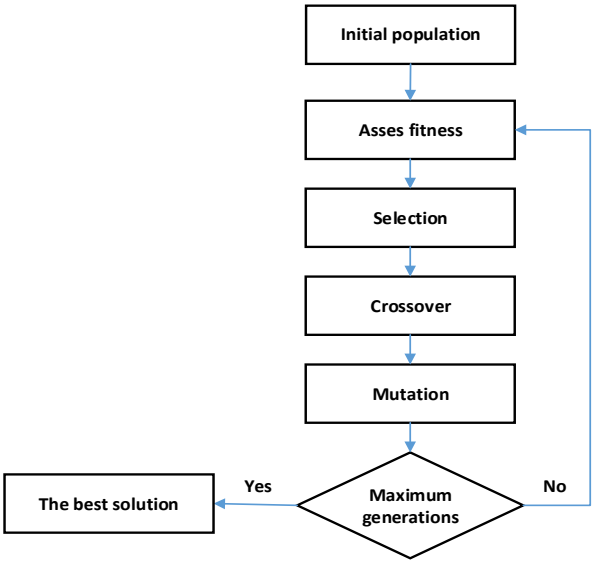

Fig. 2 GA algorithm

3. Algorithm Simulation

The experimental environment of this paper is Intel (R) Xeon (R) Silver 4114 CPU @ 2.20GHz, 20 GB memory, 0.8 GB active memory. The whole model is implemented in the Pytorch deep learning framework.

Figure 3 shows the 36-node single line diagram. A single-phase short-circuit ground fault is set at 50% between bus20 and bus16 lines, which is simulated by Power System Analysis Synthesis Program (PSASP). Figure 4 shows the voltage curve of branch _AC30, which is the object of this paper. Given that short-circuit ground fault occurred at one second indicating sudden change in voltage, this paper will adopt the method of segmented generation, that is, to generate the normal data before the fault (Figure 5) and the data after the fault (Figure 6).

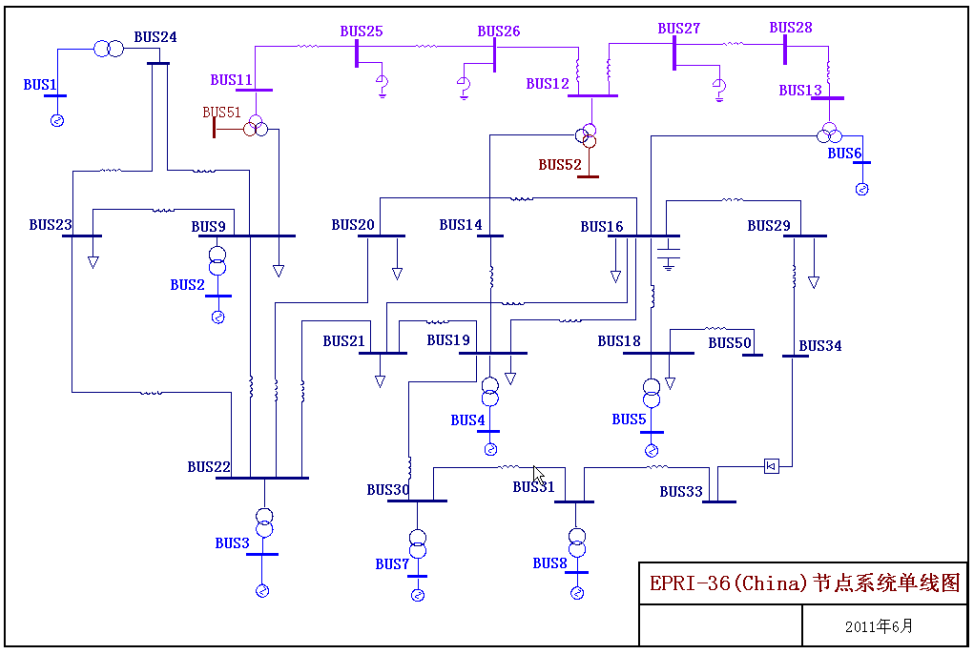

Fig. 3 36-node Single Line Diagram

ISSN: 2664-9640

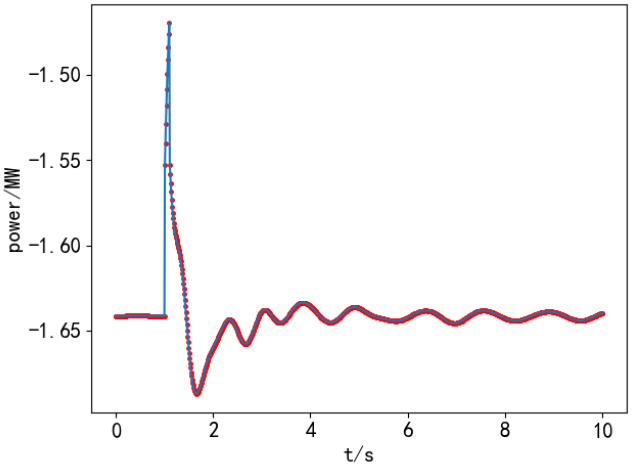

Fig. 4 Voltage Curve of Branch _AC30

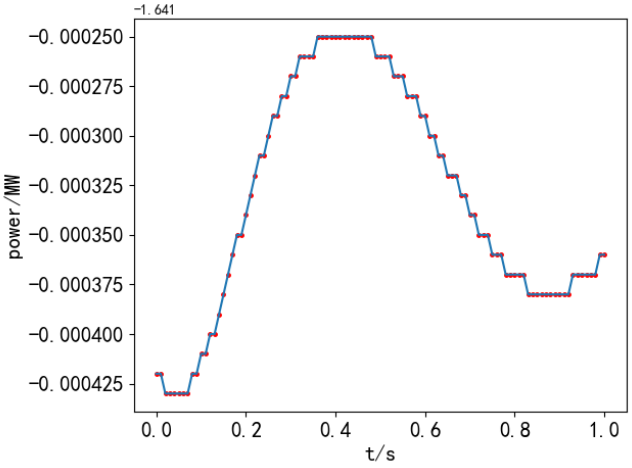

Fig. 5 Normal Data

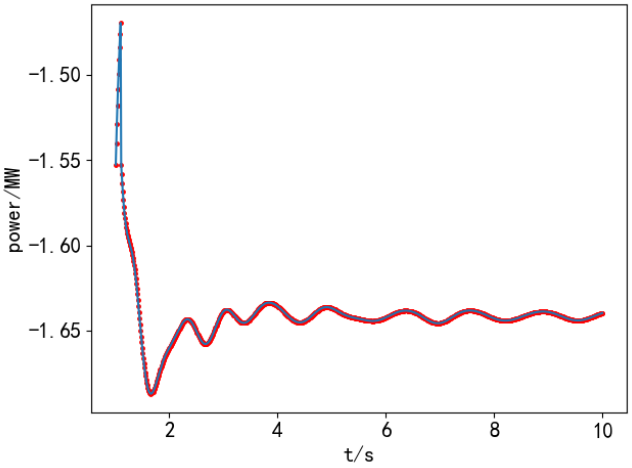

Fig. 6 Data after Fault

In the process of parameter optimization simulation in this paper, parameters r_G, r_D, and B of GAN are objects of optimization. The number of iterations of GAN is 10,000. In the experiment, Adam is used as the optimizer, which can update the weight of neural network iteratively on the basis of training data and minimizes the loss of G and D. Average relative error is adopted as objective function, and the optimal parameter combination is obtained by minimizing the objective function.

ISSN: 2664-9640

The parameters of the PSO algorithm are set as follows. Learning factors c_1 and c_2 decrease and increase linearly with the number of iterations. $c_{min} = 1.4$; $c_{max} = 2.5$; $\omega_{min} = 0.1$; $\omega_{max} = 0.9$; particleNum=100; T_{max} is the max iteration, take 100; t is the current iteration.

The parameters of the GA algorithm are set as follows. $n = 100$; $p\_c = 0.8$; $p\_e = 0.5$; $p\_m = 0.1$; $Num\_iter = 100$.

Improved GAN generating data are obtained by using the optimized parameters. To carry out comparative analysis, PSO and GA are compared for parameter optimization. In addition to the above two optimization methods, GAN parameter values are randomly selected to generate data.

3.1. Generating the normal data

The total number of normal data is 100, 80% of which are randomly selected as training set and the rest as test set. Table I shows the allowed range of parameters.

TABLE I Range of Gan Parameters

Parameters	r_G	r_D	B
Allowed range	[0.00001,0.01]	[0.000001, 0.01]	[64, 100]

Table II shows the results. In the data scenario studied, PSO optimization is the best method for the error of data generation, followed by GA optimization method, and RN method. In terms of time consumed, the running time of RN method is less than that of PSO and GA method, and the time consumed by PSO method is less than that of GA method.

TABLE II Optimization Results

	PSO	GA	RN
r_G	0.000197	0.000253	0.0003345
r_D	0.000025	0.000032	0.0000526
B	75	90	82
ERR (%)	1.38	9.56	26.88
$TIME$ (s)	113928.9	182783.5	27895.5

Figure 7 and Figure 8 present the results optimized by using PSO and GA algorithm, respectively. Figure 9 shows the generated result obtained by randomly taking the parameter values of the GAN. The red line represents the real data, and the green line represents the data generated by G. Figure 7 and Figure 8 show that the generator has learned the approximate distribution of the real samples after 5,000 iterations. However, the generated data deviated greatly from the real samples. After 7,500 repetitions of training, the generated samples are basically the same as the real samples. The results of PSO optimization method are obviously better than those of GA method and RN method. Figure 10, Figure 11, and Figure 12 illustrate the graphs of the convergence and error of PSO-optimized GAN-generated data with the number of iterations, which indicates that convergence works well.

ISSN: 2664-9640

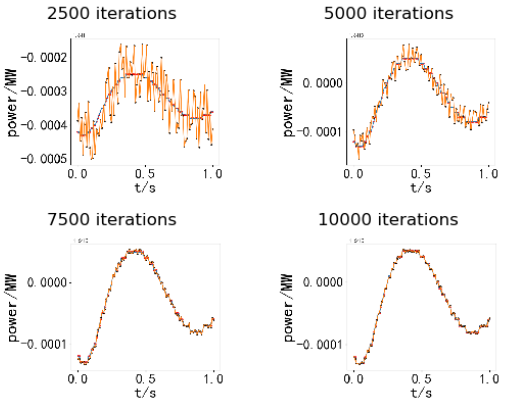

Fig. 7 PSO-GAN

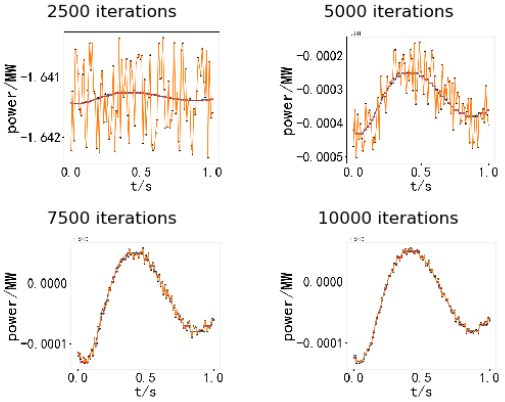

Fig. 8 GA-GAN

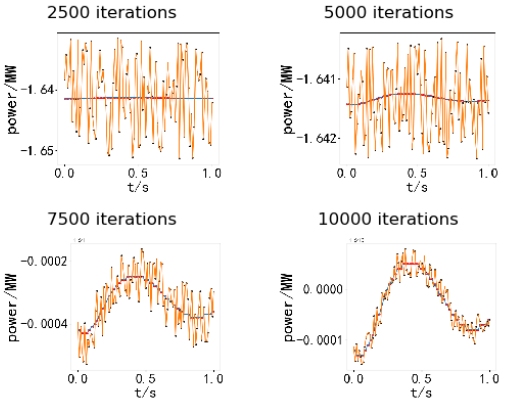

Fig. 9 RN-GAN

ISSN: 2664-9640

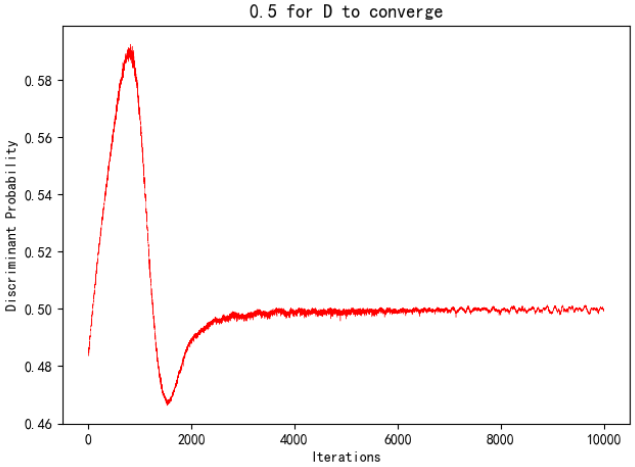

Fig.10 Discriminant Probability curve

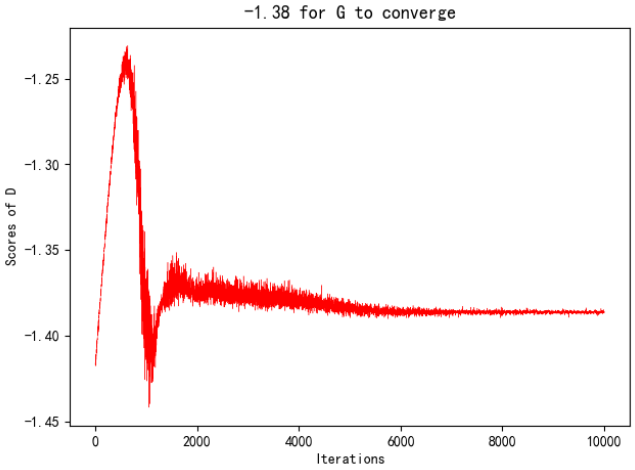

Fig.11 Scores of D

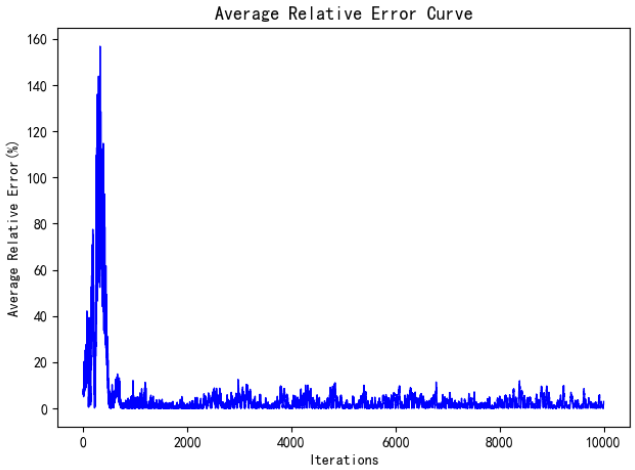

Fig.12 Average Relative Error Curve

3.2. Generating post-fault data

The total number of normal data is 900, 80% of which are randomly selected as training set and the rest as test set. Table III shows the allowed range of parameters.

ISSN: 2664-9640

TABLE III Range of Gan Parameters

Parameters	r_G	r_D	B
Allowed range	[0.00001, 0.01]	[0.000001, 0.01]	[450, 900]

Table IV shows the results. In the data scenario studied, PSO optimization method is the best for the error of data generation, followed by GA optimization method and RN method. In terms of time consumed, the running time of RN method is far less than that of PSO and GA method, and the time consumed by PSO method is less than that of GA method.

TABLE IV Optimization Results

	PSO	GA	RN
r_G	0.000238	0.000261	0.0000968
r_D	0.0000243	0.0000286	0.0000976
B	824	887	684
ERR (%)	2.49	12.38	30.65
$TIME$ (s)	526842.6	569376.5	62928.7

Figure 13 and Figure 14 show the results optimized by using PSO and GA algorithm, respectively. Figure 15 presents the generated result obtained by randomly taking the parameter values of the GAN. The red line represents the real data, and the green line represents the data generated by G. Figure 13 and Figure 14 show that the generator has learned the approximate distribution of the real samples after 5,000 iterations. However, the generated data deviated greatly from the real samples. After 7,500 repetitions of training, the generated samples are basically the same as the real samples. The results of PSO optimization method are obviously better than those of GA method and RN method. Figure 16, Figure 17, and Figure 18 illustrate the graphs of the convergence and error of PSO-optimized GAN-generated data with the number of iterations, which indicted that convergence works well.

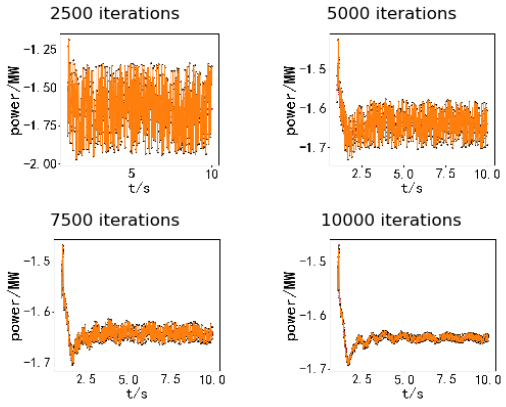

Fig. 13 PSO-GAN

ISSN: 2664-9640

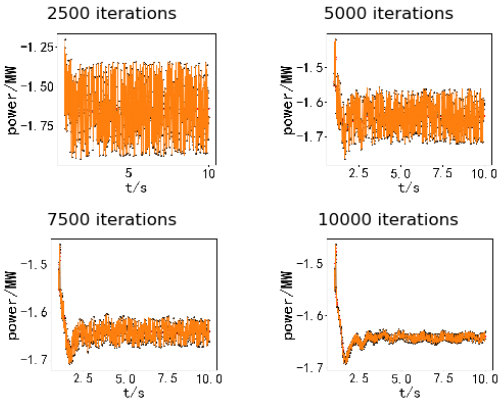

Fig. 14 GA-GAN

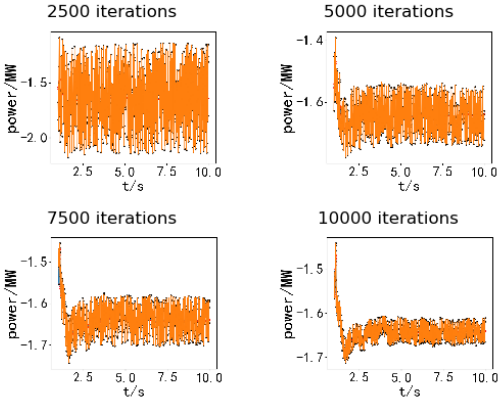

Fig. 15 RN-GAN

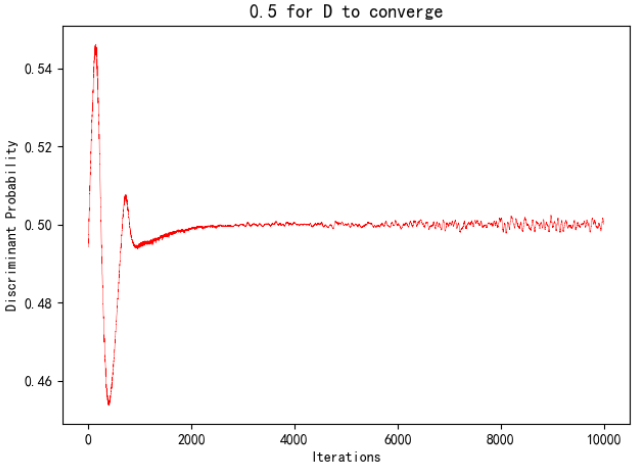

Fig.16 Discriminant Probability curve

ISSN: 2664-9640

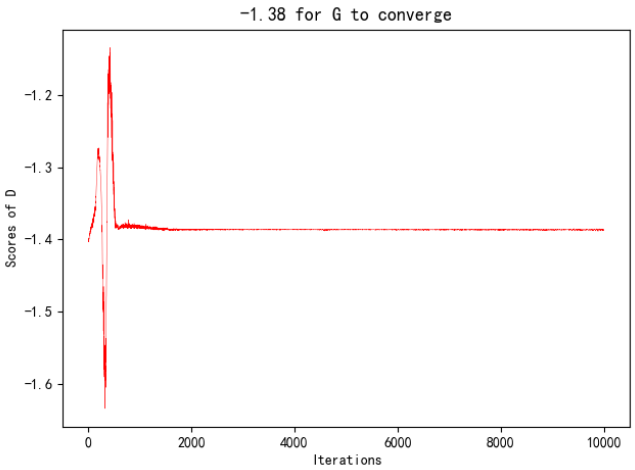

Fig.17 Scores of D

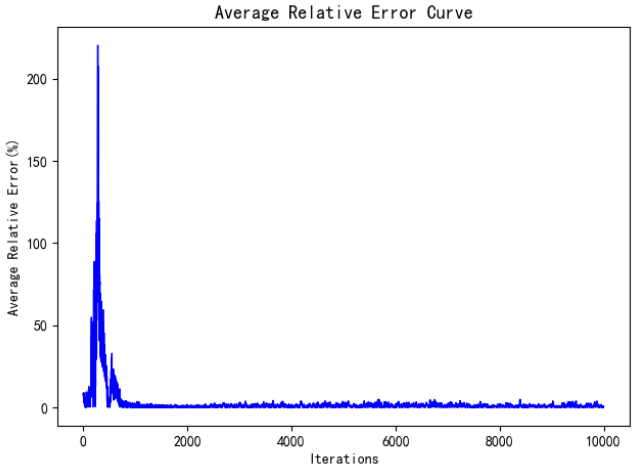

Fig.18 Average Relative Error Curve

4. Conclusion

In this paper, PSO and GA optimization methods are used to optimize the GAN parameters (r_G, r_D, and B) to generate bus voltage data of unidirectional fault. In addition, the method of randomly taking the GAN parameter values is used to generate these data. The results suggest that PSO optimization parameters can improve the performance of GAN algorithms significantly not only in time consumption, but also in generating data accuracy compared with GA. Owing to the blindness of the randomized parameter method, controlling the accuracy of the data generated by GAN is difficult, and the generation error is often large. Therefore, GA is not recommended. The convergence curve of GAN also shows that the PSO optimization method has good convergence.

Acknowledgements

This work is supported by Scientific Research Fund of Hunan Provincial Education Department (Grant No. 20A037).

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

ISSN: 2664-9640

[2] Unterthiner T , Nessler B , Seward C , et al. Coulomb GANs: Provably Optimal Nash Equilibria via Potential Fields[J]. 2017.
[3] Heusel M , Ramsauer H , Unterthiner T , et al. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium[J]. 2017.
[4] Berthelot D , Schumm T , Metz L . BEGAN: Boundary Equilibrium Generative Adversarial Networks[J]. 2017.
[5] J. Luo. Learning inverse mapping by autoencoder based generative adversarial nets. arXiv preprint arXiv:1703.10094, 2017.
[6] Locatello F , Vincent D , Tolstikhin I , et al. Competitive Training of Mixtures of Independent Deep Generative Models[J]. 2018.
[7] Song Y , Ma C , Wu X , et al. VITAL: Visual Tracking via Adversarial Learning[J]. 2018.
[8] Ajdani M , Ghaffary H . Introduced a new method for enhancement of intrusion detection with random forest and PSO algorithm[J]. Security and Privacy, 2021(1).
[9] Dereli S , Kker R . Strengthening the PSO algorithm with a new technique inspired by the golf game and solving the complex engineering problem[J]. Complex & Intelligent Systems, 2021.
[10] Venkatesh A , Pradeepa H , Chidanandappa R , et al. Brushless Motor Performance Optimization by Eagle Strategy with Firefly and PSO[J]. 2021.
[11] Su S , Zhai Z , Wang C , et al. Improved Fractional-Order PSO for PID Tuning[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2021.
[12] Portaluri G. Data Center Resource Allocation: a Genetic Algorithm Approach.[J]. 2017.
[13] Liu L , Moayedi H , Rashid A S A , et al. Optimizing an ANN model with genetic algorithm (GA) predicting load-settlement behaviours of eco-friendly raft-pile foundation (ERP) system[J]. Engineering With Computers, 2019.