mathematics

MDPI

Article

PEGANs: Phased Evolutionary Generative Adversarial Networks with Self-Attention Module

Yu Xue [1,*], Weinan Tong [1], Ferrante Neri [2] and Yixia Zhang [1]

[1] School of Computer Science, Nanjing University of Information Science & Technology, Nanjing 210044, China
[2] NICE Research Group, Department of Computer Science, University of Surrey, Guildford GU2 7XH, UK
* Correspondence: xueyu@nuist.edu.cn

Abstract: Generative adversarial networks have made remarkable achievements in generative tasks. However, instability and mode collapse are still frequent problems. We improve the framework of evolutionary generative adversarial networks (E-GANs), calling it phased evolutionary generative adversarial networks (PEGANs), and adopt a self-attention module to improve upon the disadvantages of convolutional operations. During the training process, the discriminator will play against multiple generators simultaneously, where each generator adopts a different objective function as a mutation operation. Every time after the specified number of training iterations, the generator individuals will be evaluated and the best performing generator offspring will be retained for the next round of evolution. Based on this, the generator can continuously adjust the training strategy during training, and the self-attention module also enables the model to obtain the modeling ability of long-range dependencies. Experiments on two datasets showed that PEGANs improve the training stability and are competitive in generating high-quality samples.

Keywords: generative adversarial networks; evolutionary computing; deep learning; self-attention mechanism

MSC: 68T07; 68W50

check for updates

Citation: Xue, Y.; Tong, W.; Neri, F.; Zhang, Y. PEGANs: Phased Evolutionary Generative Adversarial Networks with Self-Attention Module. *Mathematics* **2022**, *10*, 2792. https://doi.org/10.3390/math10152792

Academic Editor: Pedro A. Castillo Valdivieso

Received: 2 July 2022
Accepted: 3 August 2022
Published: 5 August 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

Generative adversarial networks (GANs) are generative models originally proposed by Goodfellow et al. [1], which are mainly used to learn the distribution of real data in the real world and learn to generate realistic data. The emergence of GANs has set off a research upsurge in the field of deep learning, which has many successful applications in style transfer [2,3], image generation [4,5], image super-resolution [6,7], etc. GANs is a semi-supervised or unsupervised learning technique. Generally speaking, it consists of two neural networks, a generator and a discriminator, which are used to solve the minimax optimization problem [1]. The generator's task is to learn to generate high-quality samples to deceive the discriminator, and the discriminator's task is to distinguish whether data are from the real samples or a sample generated by the generator.

Ideally, through this adversarial training, the generator will be able to fully understand the distribution of real samples, and then the discriminator will not be able to distinguish between them, which is called reaching a Nash equilibrium [1]. However, in practice, this type of adversarial training system is very fragile, and the phenomenon of mode collapse and gradient disappearance often occurs during the training process [8]. Mode collapse means that the generator tends to generate samples of only a single mode, rather than real samples that tend to be diverse. Gradient vanishing refers to the fact that when the discriminator is stronger than the generator, the discriminator can identify generated samples as fake samples with high probability, so the generator cannot obtain useful gradient information for updating. In order to alleviate these problems, researchers have made many efforts and attempts, including modifying the objective functions [9,10], designing different

architectures [11,12], and using multiple generators [13] or multiple discriminators [14,15] for training. The first method aims to make the generator more stable to reach the equilibrium point by using different objective functions. The second approach is to find a suitable network architecture for the generator. In order to capture different data patterns, the third method trains multiple generators. In the case of using multiple discriminators, the generator can obtain more stable gradient information to stabilize the training process. In addition, another approach is to use regularization techniques such as normalization or gradient penalty on the discriminator [16–19].

However, the above methods are limited by fixed training strategies in the training process, so researchers began to use evolutionary computing techniques to improve the training stability of GANs. Evolutionary computing techniques, which can automatically design hyperparameters and network architectures according to practical problems, have been receiving increasing attention from researchers in deep learning. Wang et al. introduced evolutionary computing technology into GANs [20]. In each iteration, a set of generators were obtained by using different objective functions as mutation operations, and a certain evaluation strategy was used to select the best offspring to enter the next iteration. Through this mutation operation, the adversarial strategy can be dynamically adjusted in the adversarial training, the characteristics of different objective functions can be effectively utilized, and the problems of gradient disappearance and mode collapse can be alleviated to a certain degree. Furthermore, Chen et al. injected the mutation operation into the evolution of the discriminator, and simultaneously evolved the generator and discriminator [21]. However, the traditional evolutionary GANs method adopts the mutation algorithm in each iterative step, and this frequent mutation causes the gradient direction to be unstable and reduces the training efficiency. Therefore, inspired by the evolutionary GANs, this paper proposes the idea of phased evolution and incorporates the self-attention mechanism. In addition, we apply the gradient normalization to further stabilize the training of GANs. The contributions of this paper are as follows.

(1) For the sake of overcoming the problem of a single training strategy and the low efficiency of evolutionary GANs, we propose a more superior training algorithm, namely the phased evolutionary algorithm, which more effectively combines the advantages of gradient descent and evolutionary algorithms, and improves the training efficiency and stability.
(2) The self-attention mechanism and gradient normalization technology are introduced into the improved evolutionary algorithm, which effectively stabilizes the discriminator during training and retains the best offspring through the phased evolution mechanism, and dynamically adjusts the adversarial strategy during training, effectively improving the training stability.
(3) Experiments are conducted on several large datasets, and the results show that our proposed optimization method outperforms existing models of the same architecture. Our code is available at: https://github.com/xueyunuist/PEGANs (accessed on 1 July 2022).

The rest of this paper is organized as follows: Section 2 presents related work of GANs, Section 3 presents our method, and Section 4 experimentally validates the performance of our method and designs comparative experiments. Section 5 summarizes the research and describes some prospects for the future.

2. Related Work

Researchers have performed a great deal of exploration to develop the performance of GANs and try to make the training of GANs more stable. In this section, we present some overview of these explorations from different aspects.

The training strategy of the original GANs was the minimax strategy, but as this strategy has the problems of slow training speed and frequent gradient disappearance, a non-saturation heuristic strategy was proposed in [1], which alleviated the problem of slow convergence to a certain extent. However, the unsaturated heuristic strategy is

prone to gradient instability and reduces the diversity of generated samples, so Mao et al. used least squares as the objective function [9]. The authors theoretically proved that least squares can make the generated samples close to the decision boundary and maintain a certain balance between diversity and quality. However, Arjovsky et al. theoretically analyzed the convergence and training problems of GANs, and proposed to apply the Wasserstein-1 distance as the objective function to measure the two distributions, and fixed the discriminator at the Lipschitz constant K [8,16]. However, because the of way in which it restricts the discriminator is weight clipping, that is, restricting the weight to [−c, c], this can easily lead to an unbalanced weight distribution. Therefore, Gulrajani et al. proposed a gradient penalty term, which calculates its gradient norm as a gradient penalty term by sampling from a certain distribution [17]. Additionally, Miyato et al. adopted the spectral normalization to directly normalize the weights without sampling by dividing the weight matrix of the discriminator by its largest singular value [18], which improves training speed. However, when the neural network becomes deeper, the performance of spectral normalization will be degraded. For this reason, Wu et al. proposed gradient normalization, which constrains the entire model and is non-sampling [19].

Some studies have attempted to introduce multiple generators or multiple discriminators into the framework of GANs. Hoang et al. adopted the strategy of training multiple generators [13]. In addition to distinguishing whether a sample is real or generated, the discriminator also needs to determine which generator generates the generated samples. By maximizing the Jensen–Shannon distance between generators, the mode collapse problem is overcome to a certain extent. Ghosh et al. also used multiple generators for training [22]. The difference is that they extract the feature vectors of the latent space of the generated samples and maximize the distance between the vectors so that different generators capture different data patterns. Nguyen et al. used two discriminators for training, essentially combining KL (Kullback–Leibler) divergence with inverse KL divergence to capture different data patterns more efficiently [15].

Wang et al. designed an evolutionary generative adversarial network framework [20]. By applying different objective functions as mutation strategies, a certain number of generator individuals are generated, and the generators are continuously evolved through fitness evaluation and selection strategies. Chen et al. unified the generator and the discriminator into the evolutionary framework by integrating the evolution of the discriminator into the adversarial framework, and proposed a soft mechanism to assist the evolution of the generator population and the discriminator population [21]. Lin et al. introduced differentiable architecture search into the evolutionary adversarial framework [23].

3. Materials and Methods

3.1. Revisiting GANs

In GANs, taking the noise z that is sampled from a certain random distribution as the input of the generator, and the generator will output the new data $G(z)$. Taking the real data or the generated data x as the input, the discriminator will output a value $D(x)$. In general, the training objective function for GANs is:

$$\min_G \max_D V_{GAN}(D, G) = \mathbb{E}_{x \sim p_{\text{real}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \quad (1)$$

3.2. PE-GANs Framework

Inspired by the success of the evolutionary algorithm and the successful application of different objective functions to improve inherent problems of GANs, we proposed phased evolutionary GANs to further improve it. To further enhance the performance of PEGANs, we used a self-attention mechanism to obtain a large degree of improvement at very little cost. Essentially, to take advantage of different objective functions, our method is to take an evolutionary approach to training the generator and employ different objective functions at different training stages. The framework of PEGANs is shown in Figure 1. Different objective functions are used as different mutation operators. First, a parent

generator G is initialized, and then a certain number of offspring are generated according to different mutation strategies. The number of offspring is consistent with the number of mutation operators. Each of these offspring takes a different variation operator as its objective function. During the process of updating, the discriminator needs to play against multiple generator individuals with different variation operators. After each p steps of adversarial training, the generator descendants are evaluated, that is, the advantages and disadvantages of different variation operators are evaluated, and according to their scores in the current environment, the survival of the fittest strategy is adopted to retain the offspring, and the winning offspring will enter the next round of iterations as the parent. Through this algorithm, the generator can dynamically adjust its training strategy according to the needs of different environments during the training process.

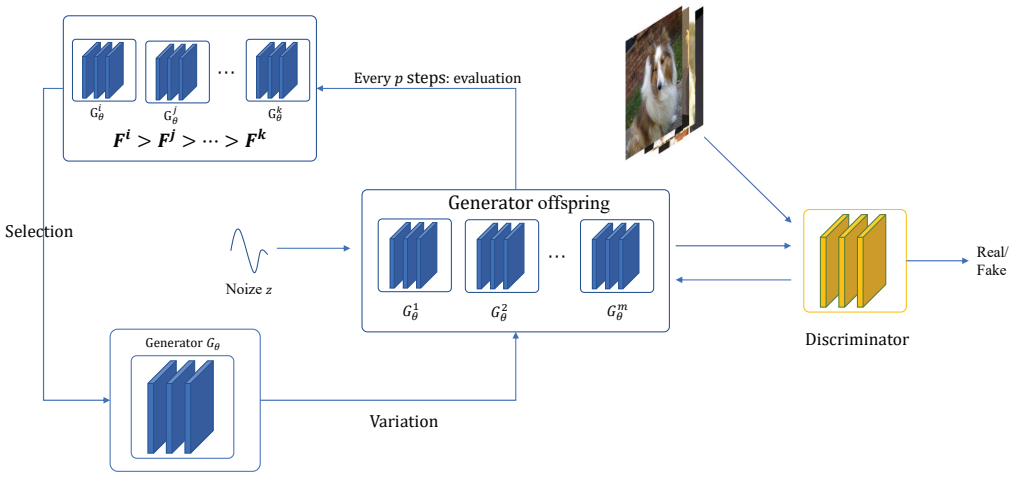

Figure 1. The framework of PEGANs. A generator will produce some offspring that will use different objective functions as update strategies. The discriminator needs to play against several generators at a time. At every certain training interval, the fitness of the generator offspring is evaluated, and the outstanding individuals will be retained and enter the next round of evolution.

3.2.1. Mutations in PEGANs

We used three objective functions that are complementary in performance as variation operators: G-Heuristic mutation, G-Minimax mutation, and G-Wasserstein mutation.

$$\mathcal{M}_G^{\text{Heuristic}} = -\mathbb{E}_{z\sim p_z}[\log D(G(z))] \tag{2}$$

$$\mathcal{M}_G^{\text{Minimax}} = \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))] \tag{3}$$

$$\mathcal{M}_G^{\text{Wasserstein}} = \mathbb{E}_{z\sim p_z}[D(G(z))] \tag{4}$$

When the discriminator has a stronger ability, that is, when it has a higher probability to set the generated data as a fake sample, the objective function of the minimax mutation has a very small gradient at this time. However, in this case a heuristic mutation can provide a large gradient, as illustrated in Figure 2. For the generator to obtain larger gradient information for updating, the generator tends to choose heuristic mutation as its own objective function, but the heuristic mutation may have difficulty to obtain enough gradient information when the generator can fool the discriminator when generating high-quality samples. At this time, the minimax mutation can provide a larger gradient, and the generator is more inclined to choose the minimax mutation. In some cases, a compromise solution is needed. At this time, it can be confirmed from the figure that the Wasserstein mutation can provide more stable gradient information, at this time the generator will choose the Wasserstein mutation. For more details about the objective function and theoretical analysis, please refer to [8].

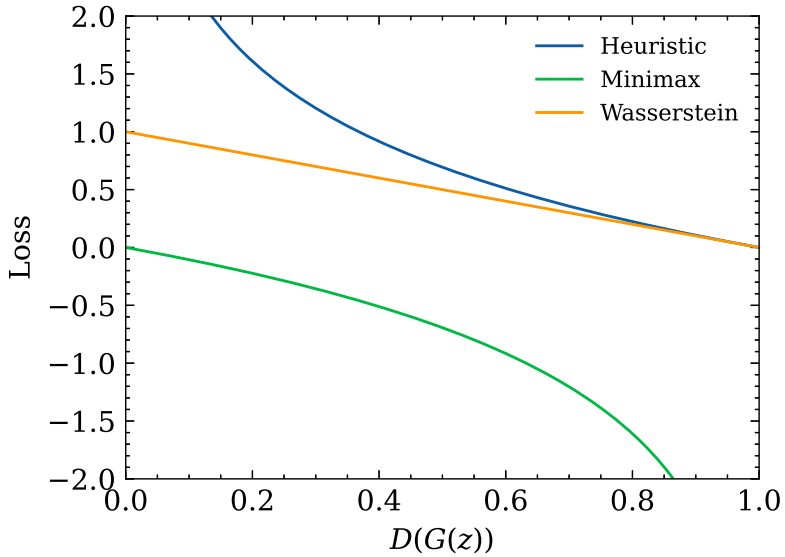

Figure 2. Function graphs for different objective functions.

During each training step, the discriminator D is updated to further distinguish between real data and fake data generated by the generators guided by different objective functions. The objective function of the discriminator is:

$$L_D = \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] - \sum_{i=1}^{3} \mathbb{E}_{z \sim p_z} [\log(1 - D(G_i(z)))] \tag{5}$$

where G_i represents the ith generator offspring. We use the above three mutation operators, so three generator individuals will be created in the training.

3.2.2. Evaluation and Selection

After training p steps of offspring generated by different mutation operators, we evaluate the individual's ability, and the evaluation function is consistent with the generator objective. The fitness evaluation function of the generator individual is:

$$\mathcal{F} = \mathbb{E}_{z \sim p_z} [D(G(z))] \tag{6}$$

In traditional EGANs [20], the fitness evaluation function considers the diversity of samples, that is, they believe that there is a positive correlation between the gradient norm of D and the diversity of generated samples. However, lin et al.proved it is unscientific [24]. For this reason, we abandon the evaluation of diversity and return the problem to the original goal.

3.2.3. Gradient Normalization

Wu et al. originally proposed gradient normalization to limit the gradient of the discriminator within a certain range [19]. By using this method, one can constrain the Lipschitz constant of the model. Compared to other normalization techniques, gradient normalization has no additional hyper-parameter and does not require sampling data from a distribution. Gradient normalization was formulated as:

$$\widehat{D} := \frac{D(x)}{\|\nabla_x D(x)\| + |D(x)|} \tag{7}$$

where $\|\nabla_x D(x)\|$ represents the norm of the gradient of D with respect to the input x. The details of PEGANs' algorithm can be seen in Algorithm 1.

Algorithm 1 PEGANs. Default Values: $\alpha = 0.0002, \beta_1 = 0, \beta_2 = 0.9, M_G = 128, M_D = 64, p = 10, m = 3$

Require: the batch size of generators M_G. the batch size of discriminator M_D. the number of offspring of the generator m. Adam hyper-parameters α, β_1, β_2. the hyper-parameter p of the number of interations between each evaluation. the number of total iterations N. discriminator's parameters ω. generators' parameters $\theta^1, \theta^2, ..., \theta^m$.

$\widehat{D} := D(x)/(\|\nabla_x D(x)\| + |D(x)|)$

for $step = 1$ to N **do**

Sample a batch of $\left\{x^{(i)}\right\}_{i=1}^{M_D} \sim p_{\text{data}}$ (real samples), and $\left\{z^{(i)}\right\}_{i=1}^{M_D} \sim p_z$ (noise samples)

$g_\omega \leftarrow \nabla_\omega \left[\frac{1}{M_D}\sum_{i=1}^{M_D} \log\left(\widehat{D}\left(x^{(i)}\right)\right) + \frac{1}{M_D}\sum_{i=1}^{M_D/m}\sum_{j=1}^{m} \log\left(1-\widehat{D}\left(G^j\left(z^{(i)}\right)\right)\right)\right)$

$\omega \leftarrow \text{Adam}(g_\omega, \omega, \alpha, \beta_1, \beta_2)$

for $j = 1$ to m **do**

$g_{\theta^j} \leftarrow \nabla_{\theta^j} \frac{1}{M_G}\mathcal{M}^j\left(\sum_{i=1}^{M_G} \widehat{D}\left(G^j\left(z^{(i)}\right)\right)\right)$

$\theta^j \leftarrow \text{Adam}\left(g_{\theta^j}, \omega, \alpha, \beta_1, \beta_2\right)$

end for

if $step \,\%\, p == 0$ **then**

for $j = 1$ to m **do**

Sample a batch of $\left\{z^{(i)}\right\}_{i=1}^{M_G} \sim p_z$

$\mathcal{F}^j \leftarrow \frac{1}{M_G}\sum_{i=1}^{M_G} D\left(G^j\left(z^{(i)}\right)\right)$

end for

$\left\{\mathcal{F}^{j_1}, \mathcal{F}^{j_2}, \ldots\right\} \leftarrow \text{sort}\left(\left\{\mathcal{F}^j\right\}\right)$

$\theta^1, \theta^2, \ldots, \theta^m \leftarrow \theta_{best}$

end if

end for

3.2.4. Self-Attention Module

The convolution operation has strong local information processing ability, but it needs to rely on deep convolutions or large convolution kernels when processing global information. This inevitably leads to low computational efficiency. Therefore, it is difficult for GANs that rely solely on convolution operations to deal with the relationship between details and the whole. Therefore, we adopt the non-local modules with self-attention mechanism [25,26] to introduce into our PEGANs framework, enabling the model to gain the ability to process global information more efficiently. As shown in Figure 3, the input $\boldsymbol{x}$ for the module is the feature map (i.e., $\boldsymbol{x} \in \mathbb{R}^{C\times H\times W}$).

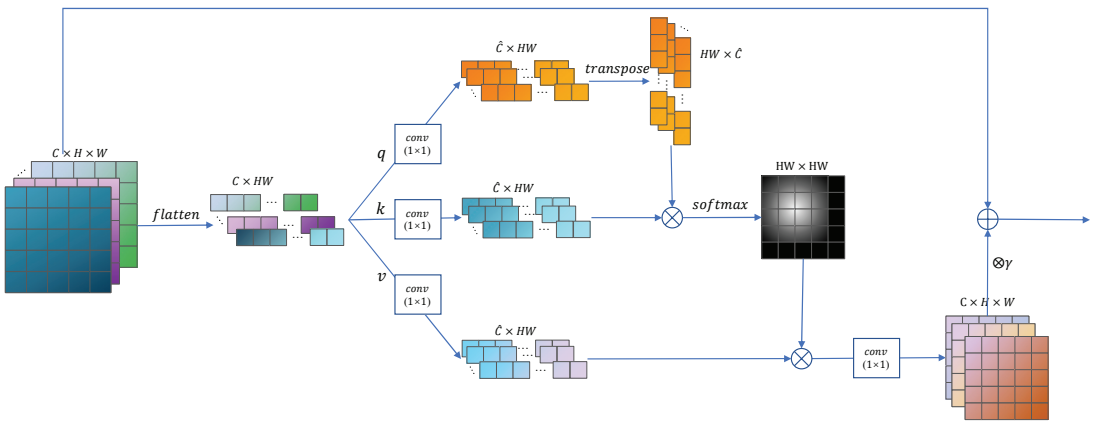

Figure 3. Self-attention module. The tensor shapes of the intermediate variable are annotated next to them. "⊕" represents the element-wise sum, and "⊗" represents matrix multiplication.

First flatten the input feature map $\boldsymbol{x}$ as $\overline{\boldsymbol{x}}$ and map it to three feature spaces, namely the query space $\boldsymbol{q}$, the key space $\boldsymbol{k}$, and the value space $\boldsymbol{v}$, where $q(\overline{\boldsymbol{x}}) = \boldsymbol{W_q}\overline{\boldsymbol{x}}, k(\overline{\boldsymbol{x}}) = \boldsymbol{W_k}\overline{\boldsymbol{x}}, v(\overline{\boldsymbol{x}}) = \boldsymbol{W_v}\overline{\boldsymbol{x}}$.

The attention weight matrix $\boldsymbol{\beta}$ is calculated by:

$$\boldsymbol{\beta} = \text{softmax}\Big(\boldsymbol{q}(\overline{\boldsymbol{x}})^T \boldsymbol{k}(\overline{\boldsymbol{x}})\Big) \tag{8}$$

where $\overline{\boldsymbol{x}} \in \mathbb{R}^{C \times HW}$, C, H, and W represent the number of channels, height, and width of the feature map, $\boldsymbol{\beta} \in \mathbb{R}^{HW \times HW}$.

The output of self-attention module is formulated as:

$$\boldsymbol{o} = \boldsymbol{W_h}\boldsymbol{\beta} v(\overline{\boldsymbol{x}}) \tag{9}$$

In the above formulation, $\boldsymbol{W_q} \in \mathbb{R}^{\hat{C} \times C}$, $\boldsymbol{W_k} \in \mathbb{R}^{\hat{C} \times C}$, $\boldsymbol{W_v} \in \mathbb{R}^{\hat{C} \times C}$ and $\boldsymbol{W_h} \in \mathbb{R}^{C \times \hat{C}}$.

The output of the self-attention module is then multiplied by a learnable scale parameter γ, and it is initialized as 0, and then added to the original input. Thus, the final output is formulated as:

$$\boldsymbol{y} = \gamma \boldsymbol{o} + \boldsymbol{x} \tag{10}$$

4. Results

In this section, we conduct unconditional generative task experiments on some datasets. The performance of our method is verified by qualitative and quantitative methods. Compared with some previous GANs methods, experimental results show that our proposed PEGANs is competitive in generating performance.

4.1. Implementation Details

Experiments were conducted on two datasets with different resolutions: CIFAR-10 [27] and STL-10 [28]. The generator and discriminator architectures in the experiment use the standard CNN architecture to keep consistent with [18,19]. We used the Adam optimizer [29], and the parameters settings are shown in Algorithm 1. It took about 12 h to train a model on CIFAR-10 using an RTX3080Ti.

4.2. Evaluation Metrics

In addition to visually showing the generated images, we also quantitatively evaluated our proposed methods using two popular evaluation methods, Inception Score (IS) [30] and Fréchet Inception Distance (FID) [31]. Inception Score is positively correlated with the performance of the generated model. In contrast to Inception scores, FID is inversely proportional to the generation quality. Please note that FID is more responsive to the quality of the generated sample than IS, and FID is closer to the human senses. For the sake of fair comparison, all evaluation metrics were calculated using the official version of 50K randomly generated samples.

4.3. Experimental Results

The methods in Table 1 all use the standard CNN architecture, the difference is our optimization method and the use of the self-attention module. Please note that our experimental results are the average training results using five different random seeds. Our method outperformed other methods on most metrics. Although the Inception Score of MGAN on CIFAR-10 was slightly higher than our method, our FID was lower. Compared with FID, IS reflects the authenticity of the image, while FID can better reflect the diversity of images, and FID is closer to the human senses as an evaluation metric for generated images [32]. Moreover, the experimental results reported in MGAN may not be the average results of multiple experiments. Therefore, we believe that our method has stronger performance. The randomly generated images are shown in Figure 4.

Table 1. Comparison of IS and FID with state-of-the-art methods on CIFAR-10 and STL-10 (unsupervised image generation). "-" indicates that no results were reported in the paper.

Method	CIFAR-10		STL-10	
	Inception Score ↑	FID ↓	Inception Score ↑	FID ↓
Real data	11.24 ± 0.12	7.8	26.08 ± 0.26	0
Standard CNN				
DCGAN [12]	6.40 ± 0.05	36.9	7.54	-
SNGAN [18]	7.58 ± 0.12	25.5	8.79	43.2
GNGAN-CR [19]	8.04 ± 0.19	22.8 ± 1.5	9.00 ± 0.15	30.18 ± 0.82
WGAN-GP [17]	6.68 ± 0.06	40.2	8.42 ± 0.13	55.1
EGAN-GP [20]	7.34 ± 0.07	27.3	-	-
EASGAN [23]	7.45 ± 0.08	22.1	-	38.84
MGAN [13]	$\mathbf{8.33 \pm 0.1}$	26.7	9.22 ± 0.11	-
PEGANs (ours)	8.30 ± 0.09	$\mathbf{17.15 \pm 0.64}$	$\mathbf{9.52 \pm 0.11}$	$\mathbf{26.85 \pm 0.91}$

(a) CIFAR-10 (b) STL-10

Figure 4. Samples randomly generated by our PEGANs (not cherry-picked).

4.3.1. Training Stability on CIFAR-10

On CIFAR-10, we plotted the FID curves of different methods in training to evaluate the effectiveness of our proposed PEGANs and the training stability. As shown in Figure 5, our method exhibits stronger stability and can achieve lower FID scores. The training speeds of different methods are shown in Figure 6. The abscissa is the training time, the number of iterations for different methods is the same, and the end of the curve represents the end of the training. It is obvious that our method is the most time-consuming; because we use evolutionary computing technology, we need to train more generator individuals and need to compute attention weights, but we can achieve better IS and FID. In other words, training is offline, and the inference speed of different methods is almost the same.

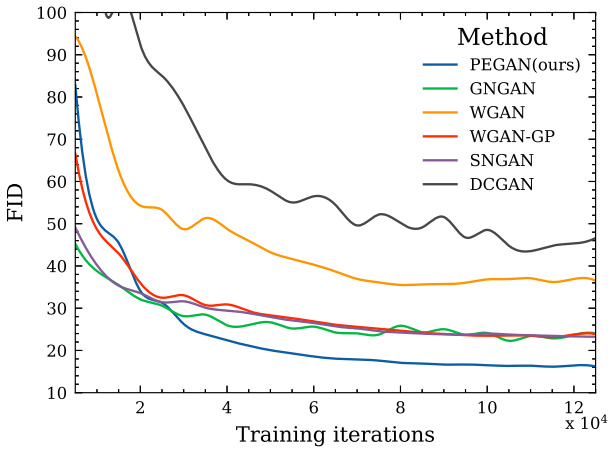

Figure 5. Learning curves of FID for different methods on CIFAR-10 (over iteration).

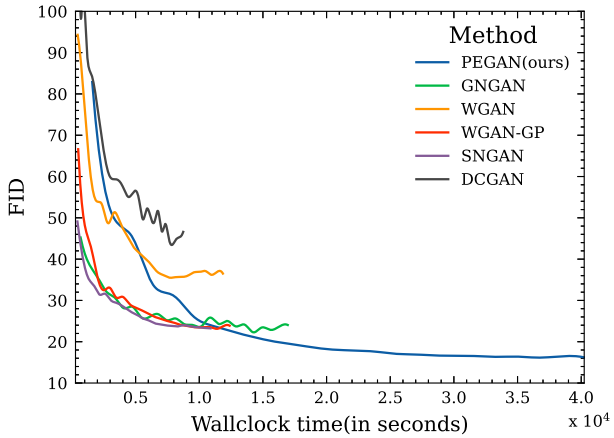

Figure 6. Learning curves of FID for different methods on CIFAR-10 (over time).

4.3.2. Hyperparameter Analysis

Since our method introduces a new hyperparameter, the evaluation interval p, we conducted experiments using different p values to test the effect of different evaluation intervals on the results, and the experimental results are shown in Figure 7. The results of each evaluation interval p come from five different random seeds. When $p = 1$, the individual generators need to be evaluated in each iteration, and the advantages of different objective functions are difficult to reflect due to the small magnitude of each update. As the evaluation interval increases, the training is more stable and the FID value becomes smaller. However, when the evaluation interval is too large, the difference between different offspring generators will be too large, which will easily cause optimization imbalance. When the evaluation interval p is in a suitable range, the training is more stable. Different evaluation interval p values will also have an impact on the training speed. The more frequent the evaluation, the slower the training speed, as shown in Table 2 . When the evaluation interval p is greater than 5, the improvement of training speed is not obvious, because the evaluation time of generator offspring only accounts for a small part of the training time of generator and discriminator, and with the increase of evaluation interval, this proportion is almost negligible. However, according to Figure 7, the training is more stable when p is about 10. Considering the training speed and performance, we recommend that the hyperparameter evaluation interval p be 10.

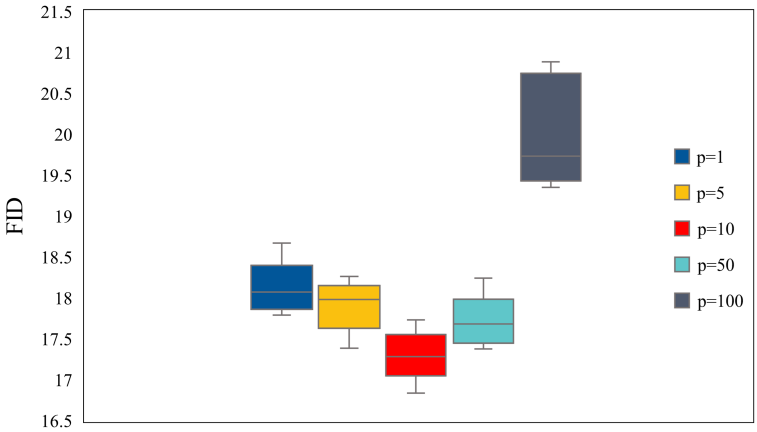

Figure 7. Comparison of FID for different evaluation intervals p on CIFAR-10.

Table 2. Training speed for different values of hyperparameter p.

p	1	5	10	50	100
iterations/s	2.70	3.08	3.16	3.19	3.19

4.3.3. Ablation Experiment

For the sake of verifying the role of the phased evolutionary algorithm and self-attention module we proposed, we set GNGAN (no phased evolution and self-attention module) as baseline, and gradually added the phased evolution algorithm and the self-attention module to it. Their IS and FID scores are shown in Table 3. The phased evolutionary algorithm effectively combines the advantages of different objective functions and can improve the IS and FID score of the model. Compared with the traditional evolutionary GANs algorithm, phased evolution also improves the training speed. In addition, the self-attention module can further improve the performance of the model at little cost (few additional parameters). Since the self-attention module processes the entire feature map, the global and local relationships can be obtained.

Table 3. The ablation experiments on CIFAR-10 (the data in the table are only for the generator).

Method	Params (Million)	GFLOPs	IS	FID
baseline	3.813	0.406	7.71 ± 0.14	23.52 ± 0.80
+phased evolution	3.813	0.406	8.01 ± 0.11	18.69 ± 0.84
PEGANs with self-attention	3.816	0.408	8.30 ± 0.09	17.15 ± 0.64

5. Conclusions and Future Work

In this paper, we improve a method for optimizing generative adversarial networks, namely phased evolutionary generative adversarial networks. Our proposed PEGANs is the further development of evolutionary GANs, which further improves the performance and training speed. Essentially using evolutionary algorithms, the discriminator plays against multiple generators at the same time, while the individual generators use different objective functions as mutation strategies. Every time after the specified number of training iterations, the generator offspring are evaluated, and a strategy of survival of the fittest is adopted. Therefore, the generator can continuously adjust the adversarial strategy during adversarial training to take advantage of different objective functions and achieve the goal of stable training. Additionally, we adopt a self-attention module to improve the shortcomings of convolution operations, and succeed in obtaining long-range dependencies. Experimental results show that our proposed PEGANs can obtain convincing performance in improving generation quality and stability.

However, our method has the problem of being time-consuming, which is inherent in evolutionary algorithms, and it has not been tested on higher resolutions datasets. In future work, we will focus on improving training speed and try generative tasks on high-resolution or complex datasets, such as face generation tasks. In addition, it seems interesting to study the properties of other different objective functions and combine them to maximize their potential.

Author Contributions: Y.X. conceived the ideas and methodologies, provided funding and equipment support, and supervised research and reviewed manuscripts. W.T. conducted the experiment and wrote the manuscript of this paper. F.N. conducted supervision research and project management, and edited the manuscripts. Y.Z. managed experimental data and revised the manuscripts. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (61876089, 61876185, and 61902281), and the Natural Science Foundation of Jiangsu Province (BK20141005).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: In this paper, we use two public datasets, CIFAR-10 can be obtained at https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 5 May 2022) and STL-10 can be obtained at https://cs.stanford.edu/~acoates/stl10/ (accessed on 5 May 2022).

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

References

1. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; Volume 27.
2. Qiao, T.; Zhang, J.; Xu, D.; Tao, D. Mirrorgan: Learning text-to-image generation by redescription. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 1505–1514.
3. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Honolulu, HI, USA, 21–26 July 2017; pp. 2223–2232.
4. Choi, Y.; Uh, Y.; Yoo, J.; Ha, J.W. Stargan v2: Diverse image synthesis for multiple domains. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8188–8197.
5. Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; Aila, T. Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8110–8119.
6. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4681–4690.
7. Wang, X.; Yu, K.; Wu, S.; Gu, J.; Liu, Y.; Dong, C.; Qiao, Y.; Change Loy, C. Esrgan: Enhanced super-resolution generative adversarial networks. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
8. Arjovsky, M.; Bottou, L. Towards principled methods for training generative adversarial networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017; pp. 1–17.
9. Mao, X.; Li, Q.; Xie, H.; Lau, R.Y.; Wang, Z.; Paul Smolley, S. Least squares generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2794–2802.
10. Nowozin, S.; Cseke, B.; Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Volume 29.
11. Denton, E.L.; Chintala, S.; Fergus, R. Deep generative image models using a laplacian pyramid of adversarial networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
12. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
13. Hoang, Q.; Nguyen, T.D.; Le, T.; Phung, D. Mgan: Training generative adversarial nets with multiple generators. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
14. Albuquerque, I.; Monteiro, J.; Doan, T.; Considine, B.; Falk, T.; Mitliagkas, I. Multi-objective training of generative adversarial networks with multiple discriminators. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 202–211.
15. Nguyen, T.; Le, T.; Vu, H.; Phung, D. Dual discriminator generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
16. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 214–223.
17. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved training of wasserstein gans. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
18. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y. Spectral normalization for generative adversarial networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
19. Wu, Y.L.; Shuai, H.H.; Tam, Z.R.; Chiu, H.Y. Gradient normalization for generative adversarial networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual Conference, 11–17 October 2021; pp. 6373–6382.
20. Wang, C.; Xu, C.; Yao, X.; Tao, D. Evolutionary generative adversarial networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 921–934. [CrossRef]
21. Chen, S.; Wang, W.; Xia, B.; You, X.; Peng, Q.; Cao, Z.; Ding, W. Cde-gan: Cooperative dual evolution-based generative adversarial network. *IEEE Trans. Evol. Comput.* **2021**, *25*, 986–1000. [CrossRef]
22. Ghosh, A.; Kulharia, V.; Namboodiri, V.P.; Torr, P.H.; Dokania, P.K. Multi-agent diverse generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, Utah, USA, 18–22 June 2018; pp. 8513–8521.
23. Lin, Q.; Fang, Z.; Chen, Y.; Tan, K.C.; Li, Y. Evolutionary architectural search for generative adversarial networks. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, *6*, 783–794. [CrossRef]
24. Li, J.; Li, J.; Zhou, W.; Lü, S. Evolutionary generative adversarial networks based on new fitness function and generic crossover operator. *arXiv* **2021**, arXiv:2109.11078.
25. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7794–7803.
26. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. In Proceedings of the International Conference on Machine Learning. PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 7354–7363.

27. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features From Tiny Images*; University of Toronto; Toronto, ON, Canada, 2009.
28. Coates, A.; Ng, A.; Lee, H. An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 215–223.
29. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
30. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved techniques for training gans. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Volume 29.
31. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
32. Brock, A.; Donahue, J.; Simonyan, K. Large scale gan training for high fidelity natural image synthesis. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.