Evolutionary Architecture Search for Generative Adversarial Networks Based on Weight Sharing

Yu Xue, *Member, IEEE*, Weinan Tong, Ferrante Neri, *Senior Member, IEEE*,
Peng Chen, Tao Luo, *Member, IEEE*, Liangli Zhen, Xiao Wang, *Senior Member, IEEE*

***Abstract*—Generative adversarial networks (GANs) are a powerful generative technique, but GANs often face the challenge of training instability. Network architecture plays a significant role in determining the final performance of GANs, but designing a fine architecture requires in-depth domain knowledge. This paper aims to address this issue by searching for high-performance generator architectures through neural architecture search (NAS). The proposed approach, called EWSGAN, is based on weight sharing and consists of two steps. Firstly, we train a supernet of generator based on weight sharing. Then, a multi-objective evolutionary algorithm is employed to extract subnets from the supernet, with fitness evaluation carried out by directly inheriting weights from the supernet and the search is performed for the Pareto front of candidate network structures. Two strategies are used to stabilize the training of the supernet of the generator: a fair single-path sampling strategy and a discarding strategy. Experimental results show that the architecture designed by our approach achieved a Fréchet Inception Distance (FID) of 9.09 and an Inception Score (IS) of 8.99 on CIFAR-10, which is the new state-of-the-art in the NAS-GANs field. Competitive results were also obtained on STL-10 (IS=10.51, FID=21.89).**

***Index Terms*—Neural architecture search, generative adversarial networks, evolutionary computation, generative model.**

I. Introduction

GENERATIVE adversarial networks (GANs) [1] are a powerful technology in the field of generative models, achieving impressive results in areas such as image super resolution [2], [3], image generation [4], and object detection [5], [6]. GANs are made up of two neural networks that collaborate throughout an adversarial learning method: a generator and a discriminator. As the discriminator network attempts to identify real samples from synthesized samples, the generator network learns to mimic the distribution of real samples. Despite the successes of GANs, they face two common challenges during training: gradient vanishing and mode collapse. To address the challenges and enhance the GANs' training stability, researchers have explored various approaches such as different loss functions, e.g., Least-Square [7], Wasserstein distance [8], Hinge loss [9], gradient regularization techniques (e.g., gradient penalty [10], spectral normalization [11], gradient normalization [12]), and modified network architectures, e.g., deep convolutional neural networks [13], deep residual networks [11].

Neural architecture search (NAS) has achieved successful applications in supervised tasks, particularly in the field of image classification [14], [15], [16]. This success could be credited to the loss function's ability to effectively reflect the training status of networks in these tasks, and the loss value on the validation set can act as a trustworthy proxy of network performance. However, the development of NAS for unsupervised tasks such as generative adversarial networks has been limited because of the non-convex and non-concave characteristics of its complex minimax optimization function, which makes it difficult to find Nash Equilibrium. Additionally, the loss function in GANs does not provide a good reflection of the training state, and the calculation of commonly used assessment indicators for GANs such as Inception Score (IS) [17] and Fréchet Inception Distance (FID) [18] are computationally expensive and non-differentiable. The work presented in [19] used reinforcement learning to search for GANs' architecture, training a recurrent neural network (RNN) as a controller to generate network architectures and using the IS metric as the reward function. However, this approach requires evaluating many structures, which is too time-consuming and requires more than 1000 GPU days. AutoGAN [20] improved upon this method by using parameter sharing and dynamic-reset strategies to speed up structure training and stepwise search strategies to improve RNN controller efficiency, but this can result in suboptimal structure searching. Gradient-based architecture search methods, such as AdversarialNAS [21] and AlphaGAN [22], relax the discrete search space to be continuous and alternately optimize the weights and architectural parameters of the generator and discriminator using gradient optimization. While this method can lead to fast improvement upon an initial design, the joint optimization of weights and architecture further complicates the relationship between them and gradient-based methods can cause deviation and unfairness due to their greedy nature.

In this work, we present a novel method to the architecture search for generative adversarial networks using a weight-

This work was supported by National Natural Science Foundation of China (NO.61876089, NO.61876185, NO.61902281, NO.61403206), by Natural Science Foundation of Jiangsu Province (NO.BK20141005), by Natural Science Foundation of the Jiangsu Higher Education Institutions of China (NO.14KJB520025), and partially supported by the project of Distinguished Professors of Jiangsu Province.

Yu Xue and Weinan Tong are with Nanjing University of Information Science and Technology, Nanjing, China (e-mail: xueyu@nuist.edu.cn; e-mail:tong-weinan@nuist.edu.cn).

Ferrante Neri is with NICE Research Group, Department of Computer Science, University of Surrey, Guildford, UK (e-mail: f.neri@surrey.ac.uk)

Peng Chen is with National Institute of Advanced Industrial Science and Technology (AIST), Japan; RIKEN-CCS, Japan (e-mail: chin.hou@aist.go.jp).

Tao Luo and Liangli Zhen are with Agency for Science, Technology and Research (A*STAR), Singapore (e-mail: {luo_tao, zhen_liangli}@ihpc.a-star.edu.sg).

Xiao Wang is with Oak Ridge National Laboratory, USA (e-mail: wangx2@ornl.gov).

sharing strategy. To achieve this goal, we put forward a two-stage approach, including: (1) Training a supernet of the generator with a fixed discriminator. (2) Conducting an architecture search phase using the trained supernet as a performance evaluator. In the second stage, we sample subnetworks from the supernet and inherit their weights for evaluation. The effectiveness of the architecture search depends on the quality of the trained supernet. Therefore, it is crucial to invest sufficient effort in this stage. For the supernet, we only activate one path at a time while conducting adversarial training with the discriminator. This allows the subnetworks to work independently while reducing memory and computational requirements.

This paper investigates the use of a weight-sharing strategy in the training of a supernet, a type of neural network. While this strategy speeds up the training, it can also lead to instability in the generator and even the collapse of the model. To alleviate this problem, we employ a fair choice strategy where each operation in the choice blocks of the supernet is trained an equal number of times in each static discriminator environment. This helps to ensure that the training progress of each operation is consistent, preventing inadequately trained operations from having a negative influence on the overall supernet and affecting the training of the discriminator. This paper explains a potential issue in the training of the supernet model, which is the presence of "bad" operations. Due to the large search space, the choice block in the supernet may contain some operations that are either poor or inappropriate for the network structure. These "poorly performing" operations can have a negative influence on the overall stability of the supernet. To solve this problem, we employ a "discard" strategy, which removes these operations from the supernet. This helps to stabilize the training of the supernet and improve efficiency by reducing the search space to more promising network structures. After training the supernet, we formulate the NAS as a multi-objective optimization problem whose objectives are the minimization of the FID and maximization of the IS and we use the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [23] to solve the problem formulated in this way.

The contributions of this work are summarized as follows:

1) We propose a novel approach to automatically design architectures for GANs. The supernet training method based on single-path sampling and the search method of multi-objective evolutionary algorithm for GANs has low requirements for GPU and high search efficiency.

2) We investigate the single-path sampling training method of GANs and present two strategies to ensure the stability of the supernet generator, including the strictly fair single-path sampling strategy and the commonality-based discarding strategy.

3) We investigate the correlation between the number of evaluation samples and evaluation indicators for GANs, and propose a low-fidelity evaluation strategy to effectively evaluate the performance of subnetworks. Our method can evaluate hundreds of network architectures in a few GPU hours because of the implemented weight-sharing strategy and low-fidelity evaluation.

4) Our experiments confirm the performance and stability of our method, yielding the best-performing model under consistent regularization conditions on the CIFAR-10 and STL-10 datasets.

II. RELATED WORKS

A. Generative Adversarial Networks

Generative adversarial networks [1] consist of two parts, i.e., a generator and a discriminator. The generator generates realistic samples to deceive the discriminator, while the discriminator learns to distinguish between generated samples and real samples. The main objective of the generator is to generate samples that have a distribution consistent with that of the real samples. Typically, GANs optimize a minimax function that can be expressed as:

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

where x is the real samples subject to a true distribution p_{real} while z represents noise that follows a normal distribution. $G(z)$ is the generated samples while $D(x)$ is the discriminator's prediction for x.

Although GANs' training is notoriously difficult and unstable, researchers have made significant progress in recent years. With extensive effort, GANs can now produce images that are indistinguishable from real ones. Various methods have been proposed to improve GANs, including modifications to objective functions [7], [8], gradient regularization [11], [12], self-attention mechanisms [24], evolutionary algorithms [25], [26], and more. The architecture of GANs is also a critical factor in achieving stable training, and studies have shown that effective architecture can significantly enhance GANs' performance [27], [28]. This paper focuses on the automated design of GANs' architectures.

B. Neural Architecture Search

The emergence of high-performance neural network architectures has promoted the use of deep learning in various fields. Typical architectures such as ResNet [29] and Transformer [30] have significantly improved the performance of neural networks. However, designing high-performance and user-friendly neural networks remains challenging because it requires extensive domain knowledge and a complex trial-and-error process. This limitation hinders the practical application of deep learning to many problems. Neural architecture search (NAS) is a powerful technology that automates the design of network architectures, which is expected to reduce the time-consuming process of manual design by researchers [31]. NAS involves defining a search space, using a predefined search strategy to discover promising network architectures within the space, and evaluating their performance. If the target performance is not achieved, the architectures are updated using the search strategy until the goal is reached.

Popular NAS methods are generally classified into three distinct groups: 1) Reinforcement learning-based approaches, such as the strategy proposed by Zoph *et al.*[32], [33], which

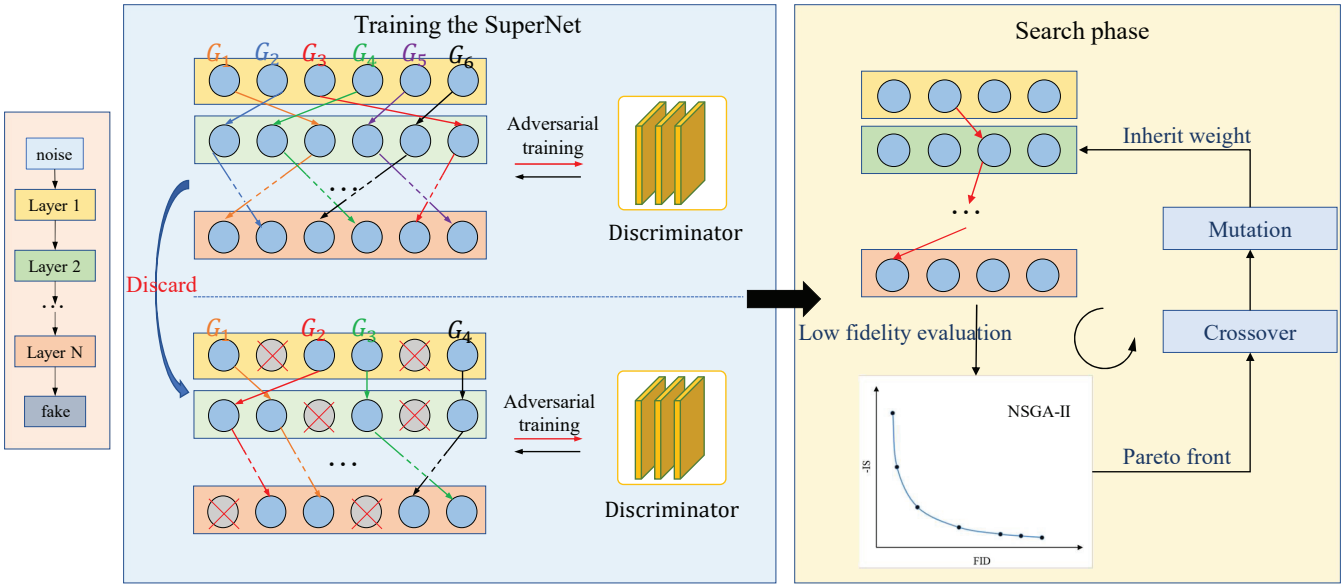


Fig. 1. The overall framework of EWSGAN.

trains an RNN controller to produce network architectures. 2) Gradient-based methods, such as the method put forth by Liu *et al.*[34], which uses the softmax function to convert the discrete search space into a continuous space with gradient guidance for updating network architecture parameters. Xue *et al.*[35] further improved search efficiency and reduced memory overhead by applying channel attention mechanisms to it. 3) Evolutionary algorithm-based approaches, such as the multi-objective evolutionary algorithm proposed by Lu *et al.*[36], which generates high-performance networks on the Pareto frontier. Sun *et al.*[15] use evolutionary algorithms to generate network architectures and propose a connection weight initialization method and a variable-length gene coding strategy.

C. NAS for GANs

Due to the unique challenges of training GANs, architecture search techniques for GANs are an active area of research. The non-differentiability and time-consuming nature of popular GANs evaluation metrics (IS and FID) has led to the use of reinforcement learning, with IS as a reward, to train an RNN controller to generate generator networks. However, this approach can be very time-consuming, as demonstrated by AGAN [19], which searched a space of 20,000 architectures at a cost of 1200 GPU days. To reduce search time, AutoGAN [20] used a layer-by-layer search strategy, parameter sharing, and dynamic-resetting to speed up network structure training. Although this approach improves search efficiency, it may limit the exploration of GANs' structures and lead to local optima. To address this issue, E^2GAN [37] transformed the GANs' structure search into a Markov decision-making process, which improved the efficiency of the controller. Both evolutionary computation-based and gradient-based methods have been proposed for architecture search in GANs. EAGAN [38] decouples the search of the discriminator and the generator network architectures into two steps using a continuous evolution method. AdversarialNAS [21] uses a differentiable architecture search approach to search for GANs' architectures, relaxing the discrete search space into a continuous one and optimizing both weight and architecture parameters using minimax functions. AlphaGAN [22] further modifies the optimization of architectural parameters by using a game-theoretic "duality gap" as the goal, which can make GANs more inclined to reach Nash Equilibrium.

III. METHODOLOGY

In this section, we introduce the EWSGAN framework we propose. We first outline the overall process of EWSGAN. Next part is the search space and coding strategy of the generator, followed by the training method and discard strategy for the supernet of the generator. Lastly, we discuss the evolutionary strategy employed during the search phase.

A. Overall Framework

The EWSGAN algorithm can be summarized as follows: Firstly, a supernet generator is defined and trained using a single path sampling strategy against a fixed discriminator, which has an architecture consistent with that used in AutoGAN [20]. To ensure fairness and reduce deviation, a strategy of sampling without replacement is adopted, which ensures that different operations in the supernet are trained for the same duration under the same discriminator environment. After a certain number of training epochs, a commonality-based strategy is adopted to identify potentially good operations or to remove poorly-performing ones, which improves training efficiency and reduces the negative impact of poor paths on other good paths. The trained supernet is then used as a performance evaluator, and subnets are collected from the supernet to evaluate and search for potentially powerful architectures. To improve search efficiency and satisfy multiple objectives, the

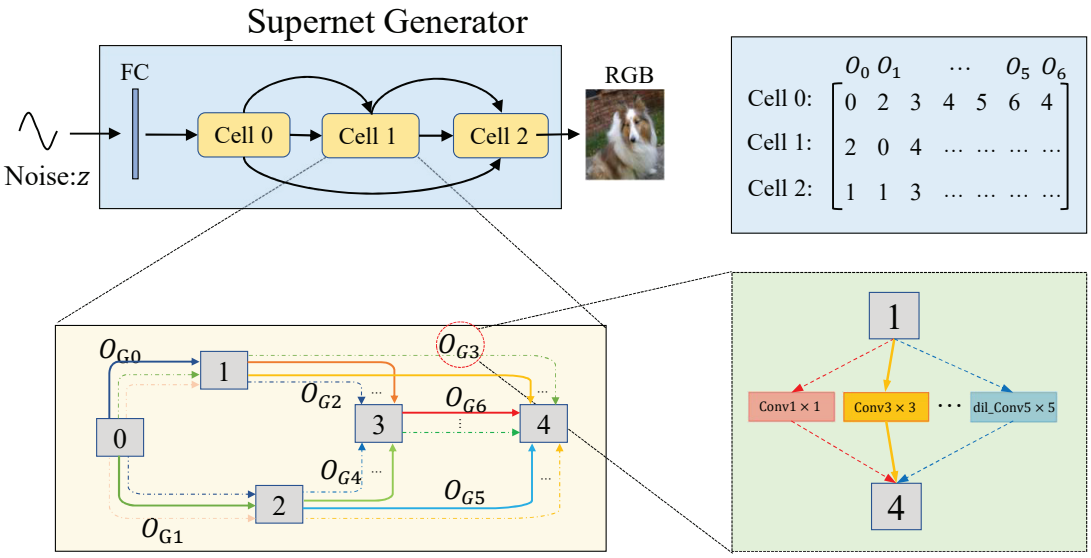


Fig. 2. Schematic diagram of search space and an example of cell.

NSGA-II algorithm is used. The overall framework of the method is shown in Fig 1. Our ultimate objective can be formulated as follows:

$$\begin{aligned} \alpha^* &= \arg\min_{\alpha} F\left(\alpha \mid \omega_G^*\right) \\ \text{s.t.} \quad \omega_G^* &= \arg\min_{\omega_G} E_{z\sim p_z}[\log(1 - D(G(z,\alpha))]] \\ \omega_D^* &= \arg\max_{\omega_D} E_{x\sim p_{\text{data}}}[\log D(x)] \\ &\quad + E_{z\sim p_z}[\log(1 - D(G(z,\alpha))]] \end{aligned} \tag{2}$$

where $F\left(\alpha \mid \omega_G^*\right) = \left(f_1\left(\alpha \mid \omega_G^*\right), f_2\left(\alpha \mid \omega_G^*\right), \ldots, f_n\left(\alpha \mid \omega_G^*\right)\right)$ is a multi-objective optimization function. ω_G^* is the weight of the supernet generator and ω_D^* is the weight of the discriminator. α represents the architecture parameter.

B. Search Space and Coding Strategy

Our generator's search space is based on cells, which is consistent with previous work [21], [22], [38]. Specifically, we use three cells for stacking in the supernet generator, each consisting of two up-sampling operations and five normal operations. The up-sampling operations include 'nearest neighbour sampling', 'bilinear interpolation sampling', and '3 × 3 transposed convolution'. The normal operations include seven alternatives, i.e., 'none', 'skip connect', and five convolution operations. To represent an active path or a subnetwork, we use a 3 × 7 matrix, where the first two numbers of each line (in the range 0-2) represent the corresponding up-sampling operation, and the last five numbers (in the range 0-6) represent the normal operation. Fig 2 shows the composition of the supernet and an example of a cell.

C. Training of Supernet Generator

To ensure stability during the architecture search stage, the training of the supernet generator is critical. The supernet training is performed in two stages. In the first stage, all the paths in the supernet are strictly and fairly trained, while in the second stage, only the potentially good paths are fairly trained. For stable and effective training of the supernet, it is crucial to ensure that each path has equal training times under the same discriminator environment. Therefore, we sample the supernet's paths in each minibatch without replacement and update all paths through gradient accumulation every time the supernet is updated. The strategy of uniform sampling does not guarantee an equal number of training times for each operation in the choice blocks. This can lead to uneven training, where poorly trained paths can mislead well-trained ones [39]. This creates a "bucket effect" in the generator subnetwork, where the weak path is imbalanced with the discriminator's ability, ultimately leading to unstable GANs' training. This will lead to fluctuations in the training of various operations in the supernet, ultimately causing the training of the supernet to fail.

In addition, since the weight of paths in the supernet is heavily shared, it is easy for poorly trained paths to mislead well-trained ones, which can have a fatal impact on the GANs' supernet and cause the supernet training to fail. To mitigate this issue, we discard poorly performing paths in the supernet in a timely fashion. Once all paths have undergone strict and fair training for a specified duration, we further enhance the training speed and minimize the effect of bad paths on good paths by pruning the supernet. This approach allocates limited computing resources to well-performing paths, thereby improving training efficiency. In order to identify potentially bad and good paths, we employ a discard strategy based on commonality. Specifically, we evaluate and sort a certain number of subnetworks, and calculate the proportion of each operation in the top 50% subnetworks. We then select operations to discard based on their proportion in the subnetworks. For example, if we sample one hundred subnetworks and sort them. In a choice block of the top fifty subnetworks, the 'nearest neighbour sampling' operation occurs 10 times,

and the remaining two operations occur 15 and 25 times, respectively. Then, we will discard the 'nearest neighbour sampling' operation. In our experiments, we discard one alternative operation in the up-sampling operation and three alternative operations in the normal operation. The training process of the supernet is outlined in Algorithm 1.

Algorithm 1 Supernet Training

Input: training iterations N, the discriminator's updating steps per iteration n_D, the batch size of supernet generator M_G, the batch size of discriminator M_D, Adam optimizer parameters $\alpha_{lr}, \beta_1, \beta_2$, iterations of warm-up N_w, initial discriminator's and generator's parameters w_D, w_G
Output: Supernet
Create $model_m = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ by fair sampling
for $i = 1 \to N$ **do**
  for $k = 1 \to n_D$ **do**
    Sample a batch of noise $\mathbf{z}$ and real data $\mathbf{x}$, model α_k
    $g_{w_D} \leftarrow \nabla_{w_D}\left[\frac{1}{m}\sum_{i=1}^{m}\log D\left(x^{(i)}\right) + \frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D\left(G\left(z^{(i)}, \alpha_k\right)\right)\right)\right]$
    $w_D \leftarrow \mathrm{Adam}\left(g_{w_D}, w_D, \alpha_{lr}, \beta_1, \beta_2\right)$
  end for
  for $j = 1 \to m$ **do**
    Sample a batch of noise $\mathbf{z}$
    Calculate gradients for model α_j and accumulate gradients
    $g_{w_G} \leftarrow \frac{1}{m}\sum_{i=1}^{m}\log D\left(G\left(z^{(i)}, \alpha_j\right)\right)$
  end for
  $w_G \leftarrow \mathrm{Adam}\left(g_{w_G}, w_G, \alpha_{lr}, \beta_1, \beta_2\right)$
  Recreate $model_m = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ by fair sampling
  while $i = N_w$ **do**
    Discard poor-performing operations
  end while
end for

D. Search Strategy

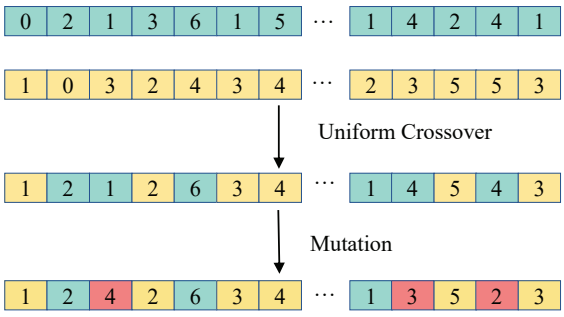


Fig. 3. Schematic diagram of crossover and mutation

Once the training of the supernet generator is complete, the supernet will serve as a performance evaluator. We sample subnetwork in a way that directly inherits the weight of the supernet. Benefiting from the weight sharing and single path sampling strategies, we can obtain a well-performing generator by sampling subnetworks. However, evaluating hundreds of subnetworks in the search stage using the standard evaluation indicators (IS and FID) is time-consuming, as it requires generating 50,000 images. To improve the efficiency of the search process, we use a low-fidelity evaluation strategy, where only 5,000 images are used for evaluation. This strategy not only saves a lot of time but also ensures a high evaluation accuracy. Since the search space is large and we consider multiple objectives such as FID and IS, we employ a multi-objective evolutionary algorithm, NSGA-II, to ensure the correct ranking of performance.

Regarding crossover and mutation, we apply uniform crossover where the offspring's gene point has a 50% chance of being inherited from each parent. Additionally, in order to enhance global search and avoid local optima, we randomly select 1-4 gene points in the offspring for mutation. The search process is further illustrated in Algorithm 2.

Algorithm 2 Search Strategy

Input: the number of generations T, the trained supernet S, the number of population N, initial population P_1
Output: K individuals on the Pareto front.
for $i = 1 \to T$ **do**
  $Q_1 = uniform_crossover_and_mutation(P_1)$
  $R_i = P_i \cup Q_i$
  for all $p \in R_i$ **do**
    Calculate fitness values of architecture p with inherited weights from the supernet S.
  end for
  $F = non_dominated_sorting(R_i)$
  Choose N individuals as P_{i+1} by ranks and the crowding distance.
end for
Pick K architectures from P_{T+1} on the Pareto front to train.

IV. EXPERIMENTS

We evaluated the performance of EWSGAN on unsupervised generation tasks using the CIFAR-10 and STL-10 datasets, which are commonly used in the NAS for GANs field. To validate the effectiveness and stability of EWSGAN, we conducted extensive comparative and ablation experiments.

A. Datasets and Evaluation Metrics

CIFAR-10 and STL-10 are datasets commonly used in the NAS for GANs field [20], [21], [37], [38]. CIFAR-10 contains 50,000 training samples and 10,000 testing samples in 10 categories with resolutions of 32×32. STL-10 contains 105,000 samples with resolutions of 96×96. In the experiment, we cut the original image size to 48×48. We solely employed spectrum normalization for the discriminator to verify the efficacy of our method and for a fair comparison.

In addition to visual results, we also used two commonly used evaluation metrics, the Inception Score (IS) and Fréchet Inception Distance (FID), to assess the quality of our results. To calculate these metrics, we randomly generated 50,000 samples from our models.

TABLE I
RESULTS OF DIFFERENT METHODS ON CIFAR-10 AND STL-10.

Method	GPU Days	Search space	Search method	CIFAR-10		STL-10	
				IS↑	FID↓	IS↑	FID↓
DCGAN [13]	-	-	Manual	6.64±0.14	37.70	-	-
WGAN-GP [10]	-	-	Manual	7.86±0.07	29.30	-	-
SNGAN [11]	-	-	Manual	8.22±0.05	21.70	9.16±0.12	40.10
GNGAN [12]	-	-	Manual	8.72±0.11	9.55	9.74±0.15	23.62
Progressive GAN [40]	-	-	Manual	8.80±0.05	18.33	-	-
Improv MMD GAN [41]	-	-	Manual	8.29	16.21	9.34	37.63
ProbGAN [42]	-	-	Manual	7.75	24.60	8.87±0.09	46.74
BigGAN [9]	-	-	Manual	9.22	14.73	-	-
AGAN [19]	1200	20000	RL	8.29±0.09	30.50	9.23±0.08	52.70
AutoGAN [20]	2	$\sim 10^5$	RL	8.55±0.10	12.42	9.16±0.12	31.01
E^2GAN [37]	0.3	$\sim 10^5$	RL	8.51±0.13	11.26	9.51±0.09	25.35
DGGAN [43]	580	-	Heuristic	8.64±0.06	12.10	-	-
DEGAS [44]	1.2	$\sim 10^8$	Gradient	8.37±0.08	12.01	9.71±0.11	28.76
AdversarialNAS [21]	1	$\sim 10^{38}$	Gradient	8.74±0.07	10.87	9.63±0.19	26.98
AlphaGAN [22]	0.13	$\sim 10^{11}$	Gradient	8.98±0.09	10.35	10.12±0.13	22.43
EAGAN [38]	1.2	$\sim 10^{38}$	EA	8.81±0.10	9.91	10.44±0.08	22.18
EWSGAN(ours)	1	$\sim 10^{15}$	EA	**8.99±0.11**	**9.09**	**10.51±0.13**	**21.89**

B. Implementation Details

The parameters used for the training of the supernet are different from those used for the complete training of the searched architecture. For the training of the supernet: the batch size of the generator is 80, and the batch size of the discriminator is 40. A total of 200 epochs are trained. After training 50 epochs, we discard part of the operations in the supernet. The number of operations in the normal operation candidate pool is decrease from seven to four, and the number of operations in the up-sampling operation candidate pool is reduced from three to two. The learning rate for the supernet and discriminator is 0.0001. The Adam optimizer is used to update parameters, while β_1 and β_2 are set to 0 and 0.9. We use spectrum normalization for the discriminator during training. For the complete training stage, the parameters are set as follows: the batch size of the generator is 256, and the batch size of the discriminator is 128. The learning rate is 0.0002. A total of 600 epochs are trained. The settings of other parameters are consistent with those in the supernet training stage. Parameter settings in the architecture search stage are set as follows: we keep a population of 20 individuals. The number of evolution iterations is 20. A total of 800 subnetworks are evaluated. We used a uniform crossover with variable intensity of 1-4 gene point mutations. The low-fidelity evaluation method for individuals is to generate 5,000 samples for evaluation. The goal of the NSGA-II is to maximize IS and minimize FID.

C. Experimental Results and Analysis

Fig 5 shows the searched generator architecture on CIFAR-10. In the up-sampling operation, our generator prefers to use the non-parametric up-sampling method in the shallow layer of the network. The possible reason is that the input of the generator is random noise, which contains limited information, so the shallow layer of the network does not need very strong learning ability. In the deep layer of the network, it prefers to use the transposed convolution with parameters and learning ability, because, after the processing of the shallow layer of the network, it already has more information to learn from the network. In normal operation, our network tends to use larger convolution cores and dilated convolutions, i.e., four "5 × 5 cov" operations and four "3 × 3 dil_cov" operations, because larger convolution cores have larger receptive fields and stronger information processing capabilities. The network does not use dilated convolution on two consecutive layers, which is logical because continuous dilated convolution is easy to cause information loss, which is fatal to the generation task. In addition, the network does not choose a 5 × 5 dilated convolution. The possible reason is that the dataset is relatively small, and the large receptive field leads to complex computation. Our network architecture has obtained the best IS and FID values in the NAS-GAN field under the same regularization conditions. The final results are weaker than some manual-designed methods because they use more complex methods and cannot be compared together.

D. Low fidelity evaluation experiment

In the search stage, our method needs to evaluate many individuals, so we need a more efficient method to evaluate the individual generator. In order to calculate the IS and FID values, a large number of samples need to be generated. The more samples generated, the more representative the results are. However, the real performance of individuals is not important, what matters is their relatively correct ranking. Therefore, we conducted experiments to generate different samples. Based on the results of 50,000 samples, we compare the Kendall correlation and time consumption of different calculation samples, and the experimental results are shown in



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

Fig. 4. The generated images by discovered generator in random. *Left*: CIFAR-10. *Right*: STL-10.

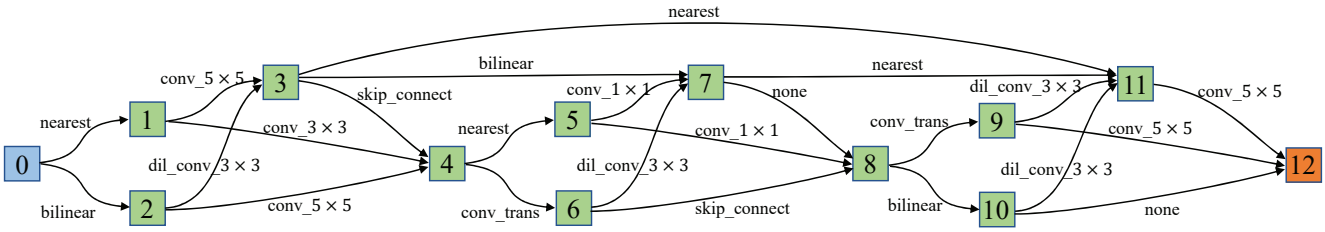


Fig. 5. The searched architecture of the generator.

Fig 6. The abscissa represents a different number of samples, the principal axis represents Kendall correlation coefficient, and the secondary axis represents the time spent. Line and histogram charts represent the Kendall correlation coefficient and time consumption, respectively. The calculation accuracy of 5,000 samples to 50,000 samples varies slightly, but the time varies greatly. The calculation time from 1,000 samples to 5,000 samples is similar, but the accuracy rate is significantly improved. In order to comprehensively consider the impact of time and sorting performance, we generated 5,000 samples to evaluate the individual generators.

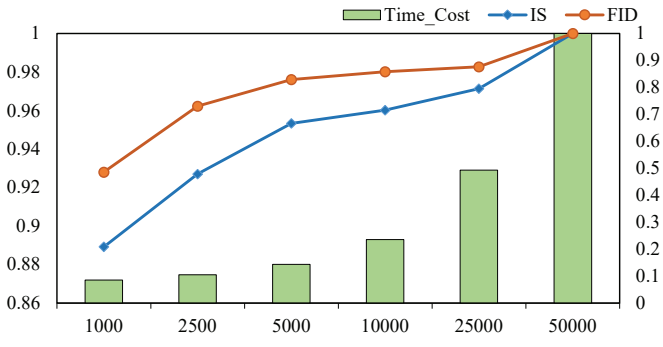


Fig. 6. The time required to calculate IS and FID with a different number of samples and their fidelity.

E. Time Cost

EWSGAN includes two main steps. The first is the training of the supernet, which typically takes around 20 hours. In the second step, an evolutionary search is performed, which takes roughly 4 hours, equivalent to about 1 GPU day. The experimental setup used for testing is the same as in prior research studies [21], [20], [37], using a single Nvidia RTX2080Ti GPU.

F. Ablation Studies

In the searching stage, network individuals are assessed based on the weight inheritance of the supernet, making it crucial to train the supernet effectively. However, the single-path sampling training method is often affected by weight coupling, causing instability and difficulties in training. To validate the effectiveness of our fair single-path sampling training method and discarding strategy, we monitored supernet performance under various training approaches, as depicted in Fig 7. Uniform sampling fails to balance all supernet operations, resulting in unstable GANs training. The learning curve in Fig 7 confirms this instability. Poorly performing operations at different network layers in the vast search space adversely impact network performance and consume computing resources. Our fair sampling and model pruning

strategies effectively enhance supernet training stability and performance, which is crucial for applying NAS.



Fig. 7. Learning curves of the supernet under various training methods. The supernet performance is monitored once every five epochs, and the learning rate and other parameters are the same. *Top*: IS learning curves. *Bottom*: FID learning curves.

G. Correlation Analysis

To explore the relationship between the performance of subnets that inherit weight from supernet and the actual performance of stand-alone training, we separately trained the searched network architecture on the Pareto frontier and calculated the Kendall correlation coefficient to test their correlation. The Kendall correlation coefficient [45] measures the degree of correlation between two variables, with a higher coefficient indicating a more positive correlation. We can be observe from Fig. 8 that there is a moderate positive correlation between the performance of the directly inherited subnetwork and its actual performance (Kendall correlation coefficient: $\tau = 0.524$). Although the correlation in our method is not very strong, it effectively improves the likelihood of discovering excellent structures, consistent with the original intention of NAS.

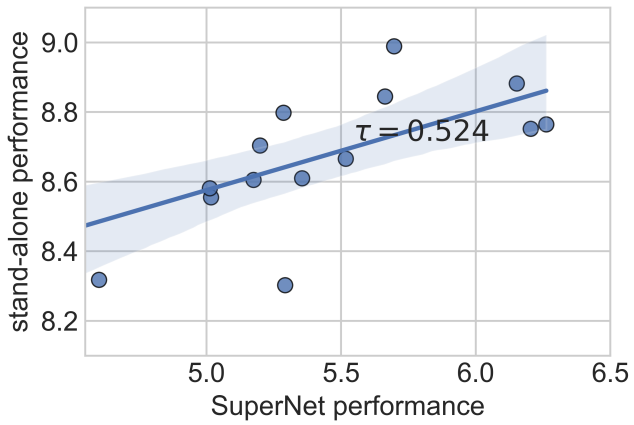


Fig. 8. The subnet's performance of inheriting weight and performance of stand-alone training.

V. CONCLUSION

In this paper, we propose a novel approach to search architectures for GANs, referred to as EWSGAN. The EWSGAN framework consists of two steps: the first is the training of a supernet generator and the second step is the architecture search for the high-performance generator using a multi-objective evolutionary algorithm. The two strategies proposed by us can effectively improve the training stability of the supernet generator, namely the fair single-path sampling strategy and the commonality-based discarding strategy. The experiments showed that our approach achieved the best performance on CIFAR-10 and STL-10 datasets in the field of NAS-GANs and the architecture search was completed in just 1 GPU day on CIFAR-10. However, our method currently only focuses on the generator's architecture search and doesn't consider the discriminator, which is also a crucial component in the performance of GANs. Improving the correlation between the performance of the subnetworks that inherit weights from the supernet and the performance of stand-alone training is another area of future work. Additionally, we will extend our method to higher-resolution datasets for the generation task to meet more practical needs.

VI. ACKNOWLEDGMENT

This manuscript has been co-authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher acknowledges the US government license to provide public access under the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[2] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *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*, 2017, pp. 4681–4690.

[3] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1905–1914.

[4] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[5] M. Zhang, K. T. Ma, J. H. Lim, Q. Zhao, and J. Feng, "Anticipating where people will look using adversarial networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1783–1796, 2018.

[6] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual generative adversarial networks for small object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1222–1230.

[7] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 214–223.

[9] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[11] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018.

[12] Y.-L. Wu, H.-H. Shuai, Z.-R. Tam, and H.-Y. Chiu, "Gradient normalization for generative adversarial networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6373–6382.

[13] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations*, 2016.

[14] Y. Bi, B. Xue, P. Mesejo, S. Cagnoni, and M. Zhang, "A survey on evolutionary computation for computer vision and image analysis: Past, present, and future trends," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1, pp. 5–25, 2023.

[15] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2020.

[16] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[17] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[19] H. Wang and J. Huan, "AGAN: Towards automated design of generative adversarial networks," *arXiv preprint arXiv:1906.11080*, 2019.

[20] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "AutoGAN: Neural architecture search for generative adversarial networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3224–3234.

[21] C. Gao, Y. Chen, S. Liu, Z. Tan, and S. Yan, "Adversarialnas: Adversarial neural architecture search for GANs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5680–5689.

[22] Y. Tian, L. Shen, G. Su, Z. Li, and W. Liu, "AlphaGAN: Fully differentiable architecture search for generative adversarial networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6752–6766, 2021.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7354–7363.

[25] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 921–934, 2019.

[26] S. Chen, W. Wang, B. Xia, X. You, Q. Peng, Z. Cao, and W. Ding, "CDE-GAN: Cooperative dual evolution-based generative adversarial network," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 986–1000, 2021.

[27] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[28] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5907–5915.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[31] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[32] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[33] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[34] H. Liu, K. Simonyan, and Y. Yang, "DARTs: Differentiable architecture search," in *International Conference on Learning Representations*, 2018.

[35] Y. Xue and J. Qin, "Partial connection based on channel attention for differentiable neural architecture search," *IEEE Transactions on Industrial Informatics*, 2022.

[36] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-NET: Neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

[37] Y. Tian, Q. Wang, Z. Huang, W. Li, D. Dai, M. Yang, J. Wang, and O. Fink, "Off-policy reinforcement learning for efficient and effective GAN architecture search," in *European Conference on Computer Vision*. Springer, 2020, pp. 175–192.

[38] G. Ying, X. He, B. Gao, B. Han, and X. Chu, "EAGAN: Efficient two-stage evolutionary architecture search for GANs," in *European Conference on Computer Vision*. Springer, 2022, pp. 37–53.

[39] X. Chu, B. Zhang, and R. Xu, "FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 239–12 248.

[40] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[41] W. Wang, Y. Sun, and S. Halgamuge, "Improving MMD-GAN training with repulsive loss function," in *International Conference on Learning Representations*, 2019.

[42] H. He, H. Wang, G.-H. Lee, and Y. Tian, "ProbGAN: towards probabilistic GAN with theoretical guarantees," in *International Conference on Learning Representations*, 2018.

[43] L. Liu, Y. Zhang, J. Deng, and S. Soatto, "Dynamically grown generative adversarial networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8680–8687.

[44] S. Doveh and R. Giryes, "DEGAS: Differentiable efficient generator search," *Neural Computing and Applications*, vol. 33, no. 24, pp. 17 173–17 184, 2021.

[45] P. K. Sen, "Estimates of the regression coefficient based on kendall's tau," *Journal of the American Statistical Association*, vol. 63, no. 324, pp. 1379–1389, 1968.