# 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 frequently face challenges with training stability. Network architecture plays a significant role in determining the final output of GANs, but designing a fine architecture demands extensive domain expertise. This paper aims to address this issue by searching for high-performance generator's architectures through neural architecture search (NAS). The proposed approach, called evolutionary weight sharing generative adversarial networks (EWSGAN), is based on weight sharing and comprises two steps. First, a supernet of the generator is trained using weight sharing. Second, a multi-objective evolutionary algorithm (MOEA) is employed to identify optimal subnets from the supernet. These subnets inherit weights directly from the supernet for fitness assessment. Two strategies are used to stabilise the training of the generator supernet: a fair single-path sampling strategy and a discarding strategy. Experimental results indicate that the architecture searched by our method achieved a new state-of-the-art among NAS-GAN methods with a Fréchet inception distance (FID) of 9.09 and an inception score (IS) of 8.99 on the CIFAR-10 dataset. It also demonstrates competitive performance on the STL-10 dataset, achieving FID of 21.89 and IS of 10.51.**

***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]. In essence, GANs are composed of two neural networks: a generator and a discriminator, which achieve their predefined goals through an adversarial learning strategy. As the discriminator network attempts to distinguish real samples from synthesised samples, the generator network learns to mimic the distribution of real samples. Despite the successes of GANs, they are prone to two common issues during training: gradient vanishing and mode collapse. To address these issues and enhance the training stability, researchers have explored various approaches, including different loss functions, e.g., least squares [7], Wasserstein distance [8], hinge loss [9] and gradient regularisation techniques (including gradient penalty [10], spectral normalisation [11] and gradient normalisation [12]), as well as modified network architectures, such as deep convolutional neural networks [13] and deep residual networks [11]. While these manual adjustments have been proven useful for stabilizing training and improving performance, they demand a substantial amount of expertise and computational resources for experimentation.

Transitioning from manual fine-tuning, neural architecture search (NAS) emerges as a promising innovation designed to automate the process of optimising network architecture. NAS has been applied successfully in supervised tasks, particularly in the field of image classification [14], [15], [16]. This could be attributed to the loss function's ability to effectively reflect the training status of networks in these tasks, whereby the loss value in the validation set may act as a reliable proxy for network performance [17]. Nevertheless, the development of NAS for unsupervised tasks, such as GANs, has been limited due to the non-convex and non-concave characteristics of its complex min-max optimisation function, which makes it difficult to determine the Nash equilibrium. Additionally, the loss function in GANs does not provide an adequate reflection of the training state, and the calculation of commonly used assessment indicators, such as the inception score (IS) [18] and Fréchet inception distance (FID) [19], are computationally expensive and non-differentiable. The research presented in [20] employed reinforcement learning to search for GAN architectures, training a recurrent neural network (RNN) as a controller to generate network architectures and adopting the IS metric as the reward function. However, this approach requires the evaluation of multiple structures over more than 1,000 graphics processing unit (GPU) days, making it far too time-consuming. AutoGAN [21] improved upon this method by using parameter sharing and dynamic-reset strategies to accelerate structure training and stepwise search strategies and thus enhance RNN controller efficiency, but this can result in suboptimal structure searching. Gradient-based architecture search methods, such as AdversarialNAS [22] and AlphaGAN [23], relax the discrete search space to make it continuous and alternately optimise the weights and architectural

This work was supported by National Natural Science Foundation of China (NO.62376127, NO.61876089, NO.61876185, NO.61902281, NO.61403206), Natural Science Foundation of Jiangsu Province (NO.BK20141005), 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, Weinan Tong and Ferrante Neri 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 the 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).

Yu Xue and Ferrante Neri are both corresponding authors.

parameters of the generator and discriminator using gradient optimisation. While this method can rapidly improve upon an initial design, the joint optimisation of the weights and architecture further complicates their interrelationship. Moreover, due to their greedy nature, gradient-based methods result in operations wherein the supernet is void of learning parameters, such as skip connections, which can easily gain larger weights, leading to bias. Ultimately, the searched network structure is prone to a significant number of skip connections [24].

In this paper, we present a novel method of searching for GAN architectures using a weight-sharing strategy, called evolutionary weight-sharing generative adversarial networks (EWSGAN). The motivation behind this study is to efficiently and automatically search for GAN architectures that can achieve stable training and generate high-quality samples. The primary objective is to address the complexity and time-consuming nature of manual GAN architecture design. To attain this goal, we propose a two-stage approach, including the following: (1) training a supernet of the generator with a fixed discriminator and (2) conducting an architecture search phase using the trained supernet as a performance evaluator. In the second stage, we sample subnets from the supernet and inherit the weights of the supernet 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 when conducting adversarial training with the discriminator. This enables the subnets to function independently while reducing the memory and computational requirements.

This paper investigates the application of a weight-sharing strategy in the training of a supernet. While this strategy speeds up the training, it can also lead to instability in the generator and even the collapse of the model. To remedy this problem, we employ a fair choice strategy whereby each operation in the supernet's choice blocks is trained an equal number of times in each static discriminator environment. This helps to ensure consistency in the training progress of each operation, preventing insufficiently trained operations from exerting a negative influence on the overarching supernet and affecting the training of the discriminator. This paper highlights a potential issue in the training of the supernet model, which is the presence of 'bad' operations. Due to the large search space, the supernet's choice block may contain certain operations that are either poor or inappropriate for the network structure. Such poorly performing operations can negatively impact the overall stability of the supernet. To resolve this problem, we employ a 'discard' strategy, which removes these operations from the supernet, helping to stabilise its training and improve its efficiency by limiting the search space to more promising network structures. It must be noted that our commonality-based discard strategy is not to be confused with the dropout method [25], as our objective is to discard alternative operations in the search space.

After training the supernet, we define the NAS as a multi-objective optimisation problem, the objectives of which involve the minimisation of the FID and maximisation of the IS, and we accordingly utilise the non-dominated sorting genetic algorithm II (NSGA-II) [26] to solve it.

The novelty and contributions of this study are summarised as follows:

- We propose a novel approach to automatically designing GAN architectures. The supernet training method based on single-path sampling and the multi-objective evolutionary algorithm (MOEA) search method have low GPU requirements and high search efficiency.
- We investigate the single-path sampling GAN training method and present two strategies to ensure the stability of the supernet generator: the fair single-path sampling strategy and the commonality-based discarding strategy.
- We investigate the correlation between the number of GAN evaluation samples and evaluation indicators and propose a low-fidelity evaluation strategy to effectively assess subnets' performance. In conjunction with our weight-sharing strategy and low-fidelity evaluation, our method can assess hundreds of network architectures in just a few GPU hours.
- The experimental results confirm the performance and stability of our method, which, under consistent normalization conditions, yields the best-performing model on the CIFAR-10 and STL-10 datasets.

## II. RELATED WORKS

### A. Generative Adversarial Networks

As noted, GANs [1] consist of two parts – a generator and a discriminator. Both are implemented using artificial neural networks, and the parameterisation of GANs mainly involves defining the network structure of these components and initialising their weights and biases. By continuously adjusting and optimising these weights and biases through the backpropagation of adversarial training, high-performance generator and discriminator are ultimately obtained. The GAN network parameters have a significant impact on the quality and diversity of generated samples. During adversarial training, the generator produces realistic samples to deceive the discriminator, while the discriminator learns to distinguish between generated and real samples. The principal objective of the generator is to output samples with distributions consistent with those of the real samples. The success of GANs depends on maintaining a balance between these two networks, and parameterisation of the networks and the training process are crucial for generating high-quality samples. Typically, GANs optimise a min-max function that can be expressed as follows:

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

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

Although obtaining good network parameters via adversarial training is difficult and leads to high instability in GANs, researchers have made significant progress in recent years. Following extensive efforts, GANs can now produce images

that are indistinguishable from their real-life counterparts. Furthermore, various methods have been proposed for improving GANs, including the application of different objective functions according to the actual requirements of the training process [7], [8], evolutionary algorithms [27], [28] and self-attention mechanisms [29]. Simultaneously, gradient regularisation technology can be used to stabilise GAN training [11], [12]. The architecture of GANs is also a critical factor in achieving stable training, with studies revealing that effective architectures can significantly enhance their performance [30], [31]. Thus, this paper focuses on the automated design of GAN architectures.

### B. Neural Architecture Search

The emergence of high-performance neural network architectures has promoted the use of deep learning in various fields; ResNet [32] and Transformer [33] are typical examples of such architectures. However, designing high-performance, user-friendly (i.e., explainable [34], scalable and robust) neural networks remains a challenge because it requires extensive domain knowledge and a complex trial-and-error process. This limitation hinders the practical application of deep learning in many problems. NAS is a powerful technology that automates the design of network architectures and is aimed at reducing the time-consuming manual design process [35]. For instance, Sun *et al*. [36] employed NAS technology to automatically design network architectures and succeeded in creating an effective method for detecting COVID-19 in CT images. In brief, NAS involves defining a search space, using a predefined search strategy to discover promising network architectures within that space then evaluating their performances. The search strategy is used to update the architectures until the target performance is achieved.

Popular NAS methods can generally be classified into three distinct groups: 1) reinforcement learning-based approaches, such as the strategy proposed by Zoph *et al*. [37], [38], which trains an RNN controller to produce network architectures; 2) gradient-based methods, such as that put forth by Liu *et al*. [39], which uses the softmax function to convert the discrete search space into a continuous space and uses gradient guidance to update network architectural parameters (Xue *et al*. [40] further improved search efficiency and reduced memory overhead by applying channel attention mechanisms) and 3) evolutionary algorithm-based approaches, such as the MOEA method, proposed by Lu *et al*. [41], which generates high-performance networks on the Pareto frontier. Sun *et al*. [15] adopted evolutionary algorithms to generate network architectures and proposed a connection weight initialisation method and variable-length gene encoding strategy. Moreover, with respect to unsupervised classification, Sun *et al*. [42] proposed using variable-length gene encoding mechanisms to search for the optimal network architecture for variational autoencoders based on different block designs and the concept of asymmetry.

### C. Neural Architecture Search for Generative Adversarial Networks

Due to their unique training challenges, GAN architecture search techniques are an active area of research. The non-differentiability and time-consuming nature of popular GAN evaluation metrics (e.g., IS and FID) has led to the use of reinforcement learning, with IS as a reward, to train an RNN controller to output generator networks. However, this approach can be extremely time consuming, as demonstrated by AGAN [20], which searched a space of 20,000 architectures for 1,200 GPU days. To reduce the search time, AutoGAN [21] uses a layer-by-layer search strategy, parameter sharing and a dynamic resetting technique to speed up the network structure training. Although this approach improves the search efficiency, it disregards the coupling nature of deep and shallow networks, which limits the exploration of GAN structures and leads to a higher probability of generating local optimal structures. To counter this issue, $E^2$GAN [43] transformed the GAN structure search into a Markov decision-making process, which enhanced the efficiency of the controller. AdversarialNAS [22] uses a differentiable approach to search for GAN architectures, relaxing the discrete search space into a continuous one and optimising both the weights and architectural parameters using min-max functions. AlphaGAN [23] further modifies the optimisation of the architectural parameters by using a game-theoretic 'duality gap' as the goal, which can make GANs more liable to reach the Nash equilibrium. However, the gradient-based architecture search method has an unfair issue [44], where the network favours skip connections.

GAN compression [45] uses architecture search to identify the optimal network channel configuration for achieving model compression. The most critical difference is that GAN compression aims to reduce the model's computational complexity and storage space while compromising the performance as little as possible. The main purpose of our search is to find the optimal GAN network architecture for a given task to improve the model's performance. Due to the significant impact of the model's network structure on its performance and efficiency, we focus on identifying better network structures. In addition, GAN compression typically targets pre-trained models whose search spaces are fixed, while our search technology can potentially discover new architectures by searching in a wider space.

Two evolutionary architecture search methods COEGAN [46] and EAGAN [47] – are most relevant to our approach. The COEGAN method uses an evolutionary algorithm based on neuro-evolution of augmented topologies to divide the generator and discriminator into two independent populations for structural evolution. Nevertheless, this method has low search efficiency and the maximum depth of the offspring is only four layers, meaning it has difficulty coping with more complex practical requirements. This method was only tested on two small datasets: MNIST and Fashion-MNIST. The EAGAN approach adopts a continuous evolution method to generate network architectures by alternately optimising the evolutionary architecture individuals and gradient update weight parameters. However, this method highly couples the architectural and weight parameters, which may exacerbate the training instability of GANs. As such, we decouple weight parameter optimisation from architecture parameter optimisation; that is, we initially optimise the weight parameters

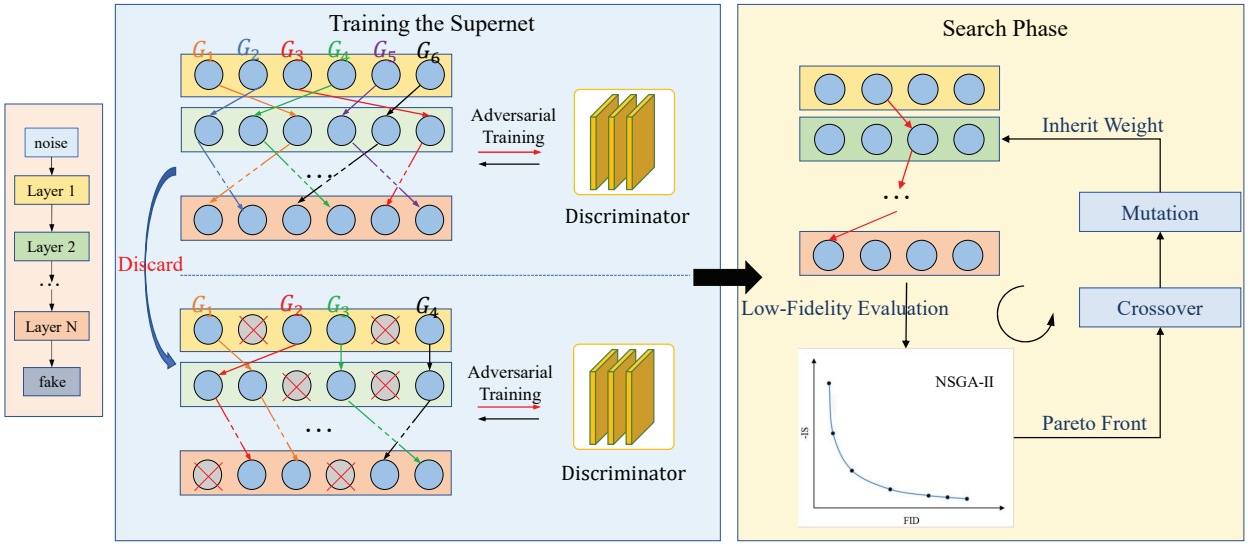


Fig. 1. The framework of our proposed EWSGAN. The content in the box with the blue background depicts the supernet training process: the upper portion portrays the use of strict fair training networks, while the lower portion portrays the potential good-path stage of fair training and the blue circles represent the operations in the choice blocks. The content in the box with the yellow background depicts the search phase of the multi-objective network architecture.

of the supernet then maintain them and gather excellent subnets through evolutionary algorithms. In addition, EAGAN inherently has a potential issue with small-model traps [48] caused by its continuous evolutionary process. Since some operations with small parameter quantities are easier to train, they are more likely to survive during evolution and progress to the next round of training. This permits operations with small parameters to undergo more training, leading to the algorithm's preference for small models. In contrast to existing methods, our fair training method can provide consistent training opportunities for supernet operations, which, in turn, provides ample training opportunities for the operations that are difficult to train but have stronger feature processing capabilities.

## III. EVOLUTIONARY WEIGHT SHARING GENERATIVE ADVERSARIAL NETWORKS

In this section, we introduce the framework of EWSGAN. First, we outline its overall process before we describe the search space and coding strategy of the generator followed by the training method and discard strategy for the supernet of the generator. Finally, we discuss the evolutionary computation technique employed during the search phase.

### A. Overall Framework

The EWSGAN algorithm can be summarised as follows. First, 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 [21]. To ensure fairness and reduce deviation, a strategy of sampling without replacement is adopted, which ensures that different supernet operations are trained for identical durations in identical discriminator environments. Following a certain number of training epochs, a commonality-based strategy is employed to identify potentially good operations or to withdraw poorly performing ones, which improves the training efficiency and reduces the negative impact of poor paths on good paths. Due to the fact that the supernet training method uses single-path sampling strategy, subnets can directly inherit weights from the supernet without retraining to achieve a good performance. Consequently, we use the trained supernet as an auxiliary performance evaluator, wherein the subnet evaluation process requires no retraining but directly inherits the weight of the supernet to evaluate and search for potentially powerful architectures. To improve the search efficiency and satisfy multiple objectives, we employ the NSGA-II algorithm. The overall framework of the method is depicted in Fig. 1. The 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 $F\left(\alpha \mid \omega_G^*\right) = \left(f_1\left(\alpha \mid \omega_G^*\right), f_2\left(\alpha \mid \omega_G^*\right), \ldots, f_n\left(\alpha \mid \omega_G^*\right)\right)$ and is a multi-objective optimisation function, $\omega_G^*$ is the weight of the supernet generator, $\omega_D^*$ is the weight of the discriminator and $\alpha$ represents the architectural parameters.

### B. Search Space and Coding Strategy

The generator's search space is consistent with previous studies [22], [23], [47]. Specifically, for stacking in the supernet generator, we use three cells, each consisting of five nodes and seven choice blocks. The five nodes are connected by the seven choice blocks, which can be subdivided into two upsampling and five normal choice blocks. We use $\mathcal{A}$ to denote the entire search space, $\alpha$ to represent a subnet within the search space and $\alpha \in \mathcal{A}$. $\alpha$ denotes a $3 \times 7$ matrix, where the element in the $i$-th row and $j$-th column, $a_{ij}$, represents the operation of the $j$-th choice block for the $i$-th

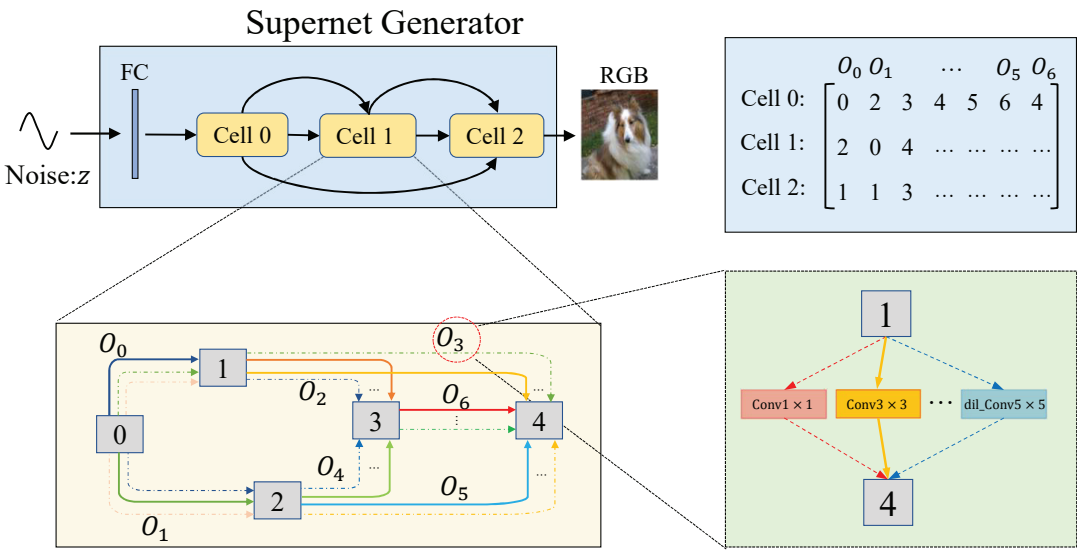


Fig. 2. Schematic of the search space and an example of a cell. The supernet consists of three cells, each comprising five nodes connected by seven edges. Each edge contains different alternative operations, which are activated one at a time. The solid line represents the activation of a certain operation, as shown in the bottom-right image. Solid line 3 × 3 convolutions are activated and participate in network forward and backward propagation. The matrix (upper right) shows an example of encoding where each entry $a_{ij}$ indicates the operation (identified by a number) occurring between two nodes of a cell. In the cell diagram (lower left), the operation number is identified by a colour. For example $O_0$ is Cell 1 is the operation between the node 0 and the node 1. The type of this operation is represented as a 2 in the matrix and as a dark blue arrow in the cell diagram.

cell. The operations in the upsampling choice blocks include 'nearest neighbour sampling', 'bilinear interpolation sampling' and '3 × 3 transposed convolution'. The operations in the normal choice blocks include seven alternatives, i.e., 'none', 'skip connect' and then five convolution operations. Selecting a single operation from among all the choice blocks in the supernet to form a subnet can also be considered as activating this path. In a subnet, $\alpha$, the first two numbers of each line (in the range of 0–2) denote the corresponding upsampling choice blocks, and the last five numbers (in the range of 0–6) denote the normal choice blocks. Fig. 2 shows the composition of the supernet and an example of a cell.

### C. Supernet Generator Training

To ensure stability during the architecture search stage, the supernet generator training is critical and is performed in two stages. In the first stage, all the supernet paths are fairly trained, while in the second stage, only the potentially good paths are fairly trained. For stable and effective supernet training, it is essential to ensure that each path undergoes an equal number of training times in the same discriminator environment. Therefore, we sample the supernet paths in each minibatch without replacement and update all the paths through gradient accumulation every time the supernet is updated. The uniform sampling strategy does not guarantee an equal number of training times for each operation in the choice blocks, which can lead to uneven training, whereby poorly trained paths can mislead well-trained ones [49]. This creates a 'bucket effect' in the generator subnet, where the weak path is unbalanced with the discriminator's ability, leading to unstable GAN training and, ultimately, fluctuations in the training of various operations in the supernet, causing the training to fail.

Since the weights of the paths in the supernet are heavily shared, it is easy for poorly trained paths to mislead well-trained ones, which can fatally impact the GAN supernet and cause the training to fail. To solve this issue, we discard any poorly performing paths in the supernet timely, and then once all the paths have undergone fair training for a specified duration, we further enhance the training speed and minimise the effect of bad paths on good ones by pruning the supernet. This approach allocates limited computing resources to well-performing paths, thereby improving the training efficiency. To identify potentially bad and good paths, we employ a commonality-based discard strategy, whereby we evaluate and sort a certain number of subnets and calculate the proportion of each operation in the top 50% subnets. We then select operations to be discarded according to their proportions in the subnets. For instance, if we sample 100 subnets and sort them to obtain a choice block containing the top 50, the nearest neighbour sampling operation occurs 10 times, and the remaining two operations occur 15 and 25 times, respectively. Following this, we discard the nearest neighbour sampling operation. In our experiments, we discard one alternative operation in the upsampling operation and three alternative operations in the normal operation. When discarding, we randomly sample 42 subnets for evaluation. The supernet training process is outlined in Algorithm 1.

### D. Search Strategy

Once the supernet generator training is complete, the supernet will serve as a performance evaluator. We sample the subnets in a manner that directly inherits the weights of the supernet. Benefiting from both the weight sharing and single-path sampling strategies, we can obtain a high-performance generator by sampling subnets. However, evaluating hundreds

**Algorithm 1** Supernet Training

**Input:** training iterations $N$, the discriminator's updating steps per iteration $n_D$, the supernet generator batch size $M_G$, the discriminator batch size $M_D$, Adam optimiser parameters $\alpha_{lr}, \beta_1$and$\beta_2$, warm-up iterations $N_w$ and the initial discriminator and generator 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 $\alpha_k$
    $g_{w_D} \leftarrow \nabla_{w_D}\left[\frac{1}{m}\sum_{i=1}^{m}\log D\left(x^{(i)}\right) + \frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D\left(G\left(z^{(i)}, \alpha_k\right)\right)\right)\right]$
    $w_D \leftarrow \mathrm{Adam}\left(g_{w_D}, w_D, \alpha_{lr}, \beta_1, \beta_2\right)$
  **end for**
  **for** $j = 1 \to m$ **do**
    Sample a batch of noise $\mathbf{z}$
    Calculate gradients for model $\alpha_j$ 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**
    Fair sample $k$ subnets and calculate their IS values
    Sort the $k$ subnets and retain the top 50%
    Calculate the proportion of each operation in the choice blocks
    Discard poor operations based on the proportion of each
  **end while**
**end for**

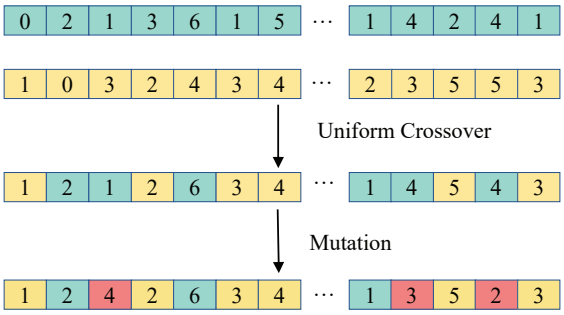


Fig. 3. Schematic of crossover and mutation. Randomly select two individuals from the population (represented by green and yellow). Generate individuals using a uniform crossover method and mutate the genes of the new individuals (red indicates variation).

**Algorithm 2** Search Strategy

**Input:** the number of generations $T$, the trained supernet $S$, the population number $N$ and initial population $P_1$
**Output:** $K$ individuals on the Pareto front.
**for** $i = 1 \to T$ **do**
  $Q_i = uniform\_crossover\_and\_mutation(P_i)$
  $R_i = P_i \cup Q_i$
  **for** all $p \in R_i$ **do**
    Calculate the 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 rank and crowding distance.
**end for**
Pick $K$ architectures from $P_{T+1}$ on the Pareto front to train.

of subnets in the search stage using the standard evaluation indicators (IS and FID) is time consuming, as it requires the generation of 50,000 images. To improve the efficiency of the search process, we use a low-fidelity evaluation strategy, whereby we assess only 5,000 images. This strategy not only saves a great deal of time but also guarantees high evaluation accuracy. Since the search space is large and we consider multiple objectives, that is the FID and IS metrics, we employ an MOEA, i.e., NSGA-II, to ensure the correct performance ranking.

Regarding crossover and mutation, we apply uniform crossover where the offspring's gene has a 50% chance of being inherited from each parent. Furthermore, to enhance the global search and avoid local optima, we set the probability of individual mutation in offspring to a value of 1. Specifically, we randomly select between 1 and 4 genes in the offspring for mutation. Fig. 3 provides a crossover and mutation example. The search process is further illustrated in Algorithm 2.

## IV. EXPERIMENTS

We evaluate the performance of EWSGAN on unsupervised generation tasks using the CIFAR-10 and STL-10 datasets, which are commonly used in the NAS–GAN field. To validate the effectiveness and stability of EWSGAN, we conducted extensive comparative and ablation experiments. The code is available at https://github.com/weinnaTT/EWSGAN. This experimental study is carried out in accordance with the checklist published in [50].

### A. Benchmark Datasets and Evaluation Metrics

The CIFAR-10 and STL-10 datasets are commonly used in the NAS–GAN field [21], [22], [43], [47]. The CIFAR-10 dataset contains 50,000 training samples and 10,000 testing samples in 10 categories with resolutions of 32 $\times$ 32, while the STL-10 contains 105,000 samples with resolutions of 96 $\times$ 96. In our experiment, we downsized the original image to 48 $\times$ 48 and solely employed spectrum normalisation for the discriminator to verify the efficiency of our method and ensure a fair comparison.

In addition to the visual results, we use two common evaluation metrics, i.e., the IS and FID, to evaluate the quality of our results. To calculate these metrics, we randomly generate 50,000 samples from our models.

### B. Implementation Details

The parameters used in the supernet training differ from those used in the complete training of the searched architecture. For the supernet training, the generator batch size is 80

TABLE I
RESULTS OF DIFFERENT METHODS ON THE CIFAR-10 AND STL-10 DATASETS. WE PRESENT THE RESULTS AS REPORTED IN THE ORIGINAL PAPERS, SELECTING THE BEST OUTCOMES WHEN MULTIPLE RUNS ARE PRESENTED.

| Method | GPU Days | Search space | Search method | Size | 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 [51] | - | - | Manual | - | 8.80±0.05 | 18.33 | - | - |
| Improv MMD GAN [52] | - | - | Manual | - | 8.29 | 16.21 | 9.34 | 37.63 |
| ProbGAN [53] | - | - | Manual | - | 7.75 | 24.60 | 8.87±0.09 | 46.74 |
| BigGAN [9] | - | - | Manual | - | 9.22 | 14.73 | - | - |
| AGAN [20] | 1200 | 20000 | RL | 20.1 | 8.29±0.09 | 30.50 | 9.23±0.08 | 52.70 |
| AutoGAN [21] | 2 | $\sim 10^5$ | RL | 5.19 | 8.55±0.10 | 12.42 | 9.16±0.12 | 31.01 |
| $E^2$GAN [43] | 0.3 | $\sim 10^5$ | RL | - | 8.51±0.13 | 11.26 | 9.51±0.09 | 25.35 |
| DGGAN [54] | 580 | - | Heuristic | - | 8.64±0.06 | 12.10 | - | - |
| DEGAS [55] | 1.2 | $\sim 10^8$ | Gradient | - | 8.37±0.08 | 12.01 | 9.71±0.11 | 28.76 |
| AdversarialNAS [22] | 1 | $\sim 10^{38}$ | Gradient | 8.8 | 8.74±0.07 | 10.87 | 9.63±0.19 | 26.98 |
| AlphaGAN [23] | 0.13 | $\sim 10^{11}$ | Gradient | 2.95 | 8.98±0.09 | 10.35 | 10.12±0.13 | 22.43 |
| EAGAN [47] | 1.2 | $\sim 10^{38}$ | EA | 7.1 | 8.81±0.10 | 9.91 | 10.44±0.08 | 22.18 |
| **EWSGAN (ours)** | 1 | $\sim 10^{15}$ | EA | 13.4 | **8.99±0.11** | **9.09** | **10.51±0.13** | **21.89** |

and the discriminator batch size is 40. The supernet is trained for 200 epochs in total. After training 50 epochs, we discard a portion of the supernet operations. The number of operations in the normal operation candidate pool is decreased from seven to four, while the number in the upsampling candidate pool is reduced from three to two and the supernet and discriminator learning rate is 0.0001. The Adam optimiser is then used to update the parameters, and $\beta_1$ and $\beta_2$ are set to 0 and 0.9. We employ spectrum normalisation for the discriminator during training. For the complete training stage, the parameters are set as follows: the generator batch size is 256, the discriminator batch size is 128, the learning rate is 0.0002 and a total of 600 epochs are trained. The remaining parameter settings are consistent with those in the supernet training stage. The parameter settings in the architecture search stage are set as follows: a population of 20 individuals is retained and the number of evolution iterations is 20. A total of 420 subnets need to be evaluated. We apply a uniform crossover and a variable intensity mutation that exchanges between 1 and 4 genes. The low-fidelity evaluation method for individuals is to generate 5,000 samples for evaluation. The goal of the NSGA-II is to maximise the IS and minimise the FID.

### C. Experimental Results and Analysis

The searched generator architecture on the CIFAR-10 dataset is shown in Fig. 4. The results of different algorithms are shown in Table I. The results displayed for all the algorithms in Table I represent the best results from multiple runs (when multiple runs are available). Regarding upsampling operations, our generator prefers to use the non-parametric method in the shallow layer of the network. This is perhaps because the generator input is random noise, which contains limited information; thus, the shallow layer of the network does not need a particularly strong learning ability. In the deep layer of the network, it prefers to employ transposed convolution with parameters and learning ability, since after processing the shallow layer of the network, it already has additional information to learn from the network. In terms of normal operations, 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 since 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 highly likely to cause information loss, which is fatal to the generation task. Moreover, the network does not choose a 5 × 5 dilated convolution. This may be because the dataset is relatively small, and the large receptive field leads to complex computation. The network architecture we searched for has fewer skip connections and none operations compared to the continuous evolution-based method of EAGAN and the gradient-based method of AdversarialNAS. The reason for this is that our fair training provides fair training opportunities for each operation, allowing certain operations with larger parameters but stronger feature processing capabilities to receive sufficient training time. Although this leads to a larger number of parameters in our model, it results in a stronger image-generation capability. The generated images we searched for are depicted in Fig. 5. Our network architecture obtained the best IS and FID values in the NAS—GAN field under the same normalization conditions. The final results were weaker than those of some manually designed methods since they use more complex procedures and are not comparable. Furthermore, to explore the scalability of our model, we duplicated the third cell of the searched network structure to generate a fourth cell. We

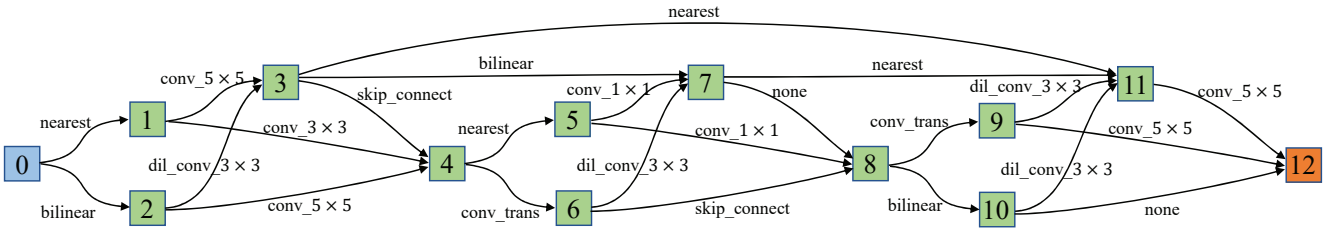


Fig. 4. The searched architecture of the generator, where 0 and 12 are the input and output nodes, respectively.

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

Fig. 5. Randomly selected images among those generated by the generator designed by EWSGAN.

then conducted experiments on the CelebA dataset [56] and produced various images, as shown in Fig. 6. The generated facial images are clear and demonstrate the model's good scalability.

Fig. 6. The facial images created by the searched generator on CelebA.

### D. Low-Fidelity Evaluation Experiment

In the search stage, our method requires the evaluation of numerous individuals; thus, we need a more efficient approach for evaluating individual generators. To calculate the IS and FID values, a large number of samples must be generated. The more samples generated, the more representative the results will be. Nevertheless, the real performance of individuals is unimportant; what matters is the accuracy of their ranking. As such, we conducted experiments to generate different samples — note: all samples are generated randomly. 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 provided in Fig. 7. The abscissa represents the number of samples, the principal axis represents the Kendall correlation coefficient and the secondary axis represents the time spent, while the line and histogram charts depict the Kendall correlation coefficient and time consumption, respectively. The calculation accuracy of 5,000 samples to 50,000 samples shows a slight variation, whereas the time required varies significantly. 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 trade-off between time and sorting performance, we generated 5,000 samples to evaluate the individual generators.

### E. Time Cost and Analysis

The EWSGAN algorithm essentially comprises two steps: supernet training and evolutionary search. We use $T_{total}$ to represent the total time required, $T_{train}$ to represent the time required to train the supernet and $T_{search}$ to denote the time required for the search. The total can be expressed as follows:

$$\begin{aligned} T_{total} &= T_{train} + T_{search} \\ &= N_w \times t_1 + (N - N_w) \times t_2 \\ &\quad + (T_{evo} + 1) \times t_{eval} \times P, \end{aligned} \tag{3}$$

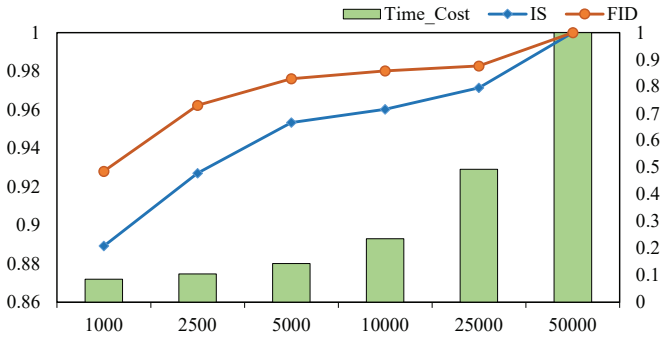


Fig. 7. The time required to calculate the IS and FID with different numbers of samples and their fidelities.

where $N$ represents the number of epochs in the supernet training, $N_w$ represents the number of epochs for warm-up, $t_1$ is the time required to train one epoch in the original supernet, $t_2$ is the time required to train one epoch in a pruned supernet, $P$ is the number of individuals in the population, $T_{evo}$ is the number of generations and $t_{eval}$ is the time required to evaluate an individual.

The experimental equipment used for testing is the same as in prior research [22], [21], [43], using a single Nvidia RTX2080Ti GPU. The training time for the supernet is approximately 20 hours, and the evolutionary search time is roughly four hours, which totals one GPU day. It takes approximately one GPU day to retrain on the CIFAR-10 dataset and approximately 3.4 GPU days on the STL-10 dataset.

### F. Ablation Studies

*1) The effectiveness of fair training and discard strategies:* In the search stage, network individuals are assessed based on the weights inherited from the supernet, making it crucial to train the supernet effectively. However, the single-path sampling training method tends to be affected by weight coupling, causing instability and training difficulties. To validate the effectiveness of our fair single-path sampling training method and discarding strategy, we monitored the supernet's performance when subjected to various training approaches, as depicted in Fig. 8. Uniform sampling fails to balance all supernet operations, resulting in unstable GAN training. Even if a strategy is implemented to discard some operations, it cannot improve training stability. This is because the uniform sampling method cannot guarantee the fair training of each operation, making it difficult to accurately evaluate each one and identify potentially bad operations in the network. The learning curve in Fig. 8 confirms the instability of both training methods. With the fair sampling training method, all the operations can undergo the same number of training sessions simultaneously. In this scenario, a discard strategy can identify potentially bad operations more effectively than the uniform sampling method, allowing the supernet to train more stably. It is clear from the learning curve for fair training without discarding that the network becomes unstable and performance decreases in the latter stages of training because poorly performing operations in 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.

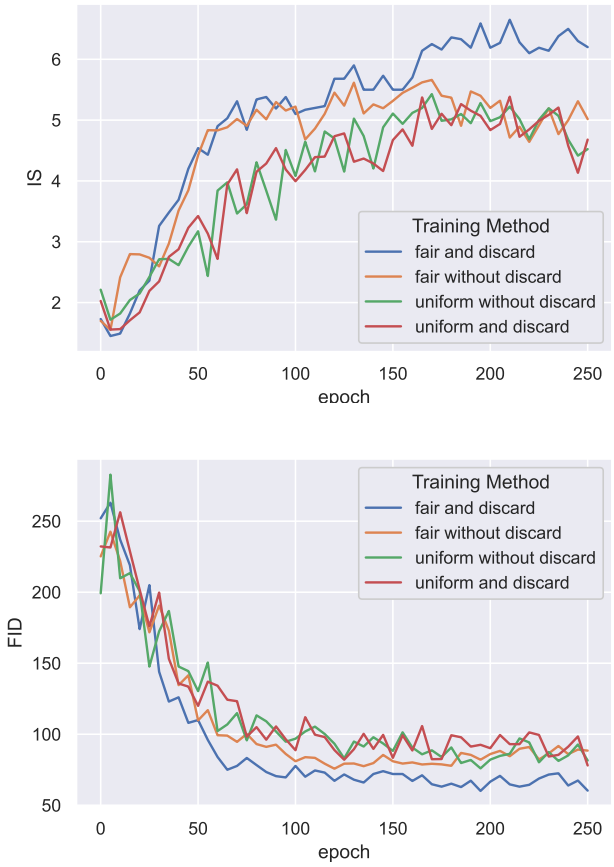


Fig. 8. Supernet learning curves under various training methods. The supernet's performance is monitored once every five epochs, while the learning rate and other parameters remain the same. *Top*: IS learning curves. *Bottom*: FID learning curves.

*2) The need for a multi-objective approach:* To investigate the effectiveness of a multi-objective approach in this case, we tested five different search algorithms. The first method, used here to provide reference results, randomly samples 20 subnets from the search space. The second method is a single objective version of the EWSGAN model whose objective function is the IS (to be maximised). We express this method as EWSGAN (IS). Analogously, the third method is a single objective version of the EWSGAN model with the goal of minimising the FID and is indicated as EWSGAN (FID). The fourth method is a single objective version of the EWSGAN model with scalarised (and normalised) fitness $0.5 \times \mathrm{IS}/5 - 0.5 \times \mathrm{FID}/90$ as its goal. We indicate this method as EWSGAN (Weigh). Each method is run with a population size of 20, and all individual networks are trained for 200 epochs. The results in the objective space of these methods (final populations) are plotted in Fig. 9 alongside the final EWSGAN population. To enhance its readability, the scatter plot has been zoomed and some random samples excluded from the image. We conducted a Wilcoxon test [57] to evaluate

TABLE II
AVERAGE IS AND FID SCORES ± STANDARD DEVIATION (CALCULATED OVER THE FINAL POPULATION OF CANDIDATE NETWORKS) FOR THE EWSGAN METHOD AGAINST ITS SINGLE-OBJECTIVE VARIANTS.

| | Random | EWSGAN (IS) | EWSGAN (FID) | EWSGAN (Weigh) | EWSGAN |
|---|---|---|---|---|---|
| IS | 7.79 ± 0.77 | 8.27±0.10 | 8.21±0.09 | 8.28±0.11 | 8.36±0.10 |
| FID | 22.23±11.40 | 15.98±1.21 | 14.83±1.23 | 15.76±1.25 | 13.87±0.98 |
| $p_{\text{IS}}$ | 1.1E-4 | 4.1E-2 | 2.1E-4 | 3.9E-2 | - |
| $p_{\text{FID}}$ | 2.5E-6 | 7.1E-6 | 1.8E-2 | 3.7E-5 | - |

TABLE III
EXPERIMENTS ON THE STATISTICAL SIGNIFICANCE OF EWSGAN VS EAGAN AND EAGAN (G), A MODIFIED VERSION OF EAGAN THAT RUNS IN THE SAME SEARCH SPACE AS EWSGAN. WE PRESENT THE RESULTS FOR EACH RUN, INCLUDING MEAN VALUES, STANDARD DEVIATION, AND $p$ VALUES.

| Method | Metrics | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | mean±std | $p$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| EAGAN (G) | IS | 8.53±0.1 | 8.69±0.09 | 8.76±0.09 | 8.55±0.08 | 8.61±0.11 | 8.66±0.11 | 8.69±0.09 | 8.56±0.1 | 8.63±0.08 | 4.5E-3 |
| | FID | 11.87 | 14.79 | 11.09 | 11.19 | 11.09 | 10.08 | 10.64 | 12.15 | 11.61±1.34 | 6.3E-3 |
| EAGAN [47] | IS | 8.81±0.1 | 8.63±0.09 | 8.69±0.1 | 8.73±0.1 | 8.85±0.11 | 8.62±0.17 | 8.79±0.1 | 8.63±0.1 | 8.72±0.08 | 2.1E-1 |
| | FID | 9.91 | 12.84 | 10.53 | 10.73 | 10.32 | 11.68 | 10.45 | 11.18 | 10.96±0.87 | 4.6E-2 |
| EWSGAN | IS | 8.81±0.09 | 8.99±0.11 | 8.78±0.11 | 8.89±0.09 | 8.83±0.07 | 8.61±0.11 | 8.73±0.18 | 8.78±0.13 | 8.8±0.1 | - |
| | FID | 9.89 | 9.09 | 10.38 | 9.57 | 10.29 | 10.99 | 9.49 | 10.9 | 10.1±0.64 | - |

the significance of different versions of EWSGAN, where when $p$-value is less than 0.05, it represents a significant difference between the two distributions. The corresponding statistical results are reported in Table II and emphasise that, although correlated, the IS and FID are different metrics and their employment as objective functions in an optimisation algorithm leads to different results. The scalarised method of EWSGAN (Weigh) also produces results that are less satisfactory than those detected by the EWSGAN. The results provided clearly indicate that this NAS approach benefits from multi-objective problem formulation.

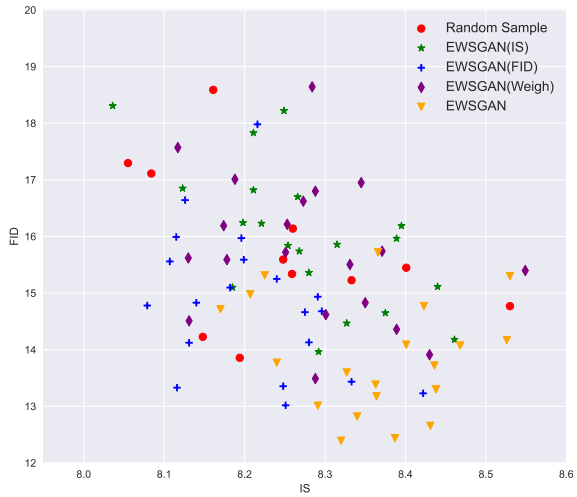


Fig. 9. Final populations in the objective space under various scenarios. The closer the points in the graph are to the bottom-right corner, the better the network. Some randomly sampled solutions are not depicted as they would fall outside the represented window.

*3) Statistical significance of the results:* To test the statistical significance of the results, we conducted eight independent experiments on CIFAR-10. The outcomes are displayed in Table III, where *#i* denotes the *i*-th test. The values shown as mean±std represent the mean value plus the standard deviation over the eight runs available, and related $p$-values. Table III presents a comparison between the proposed EWSGAN and both the original version of EAGAN [47] and a modified version (EAGAN (G)). EAGAN is responsible for designing both the generators and discriminators of the GAN. On the other hand, EWSGAN designs the generator only. To ensure direct comparability, we implemented EAGAN (G) to use EAGAN's search logic for the design of the generator, thus reproducing the same search space and experimental conditions as EWSGAN [50].

The experimental results demonstrate the superiority of our method over EAGAN (G). When compared to EAGAN, there is no notable difference in IS metrics, yet EWSGAN achieves notably improved results in FID indicators. Additionally, it becomes evident that aligning EAGAN with the specifications of EWSGAN has seemingly resulted in a decline in its performance. Although EAGAN (G) is more directly comparable to the proposed EWSGAN, it is also the outcome of altering the original design, potentially accounting for its performance degradation. The results show that EWSGAN is superior also to the original EAGAN. However, we may observe that while the proposed EWSGAN significantly outperforms EAGAN (G) in terms of both IS and FID, it outperforms significantly EAGAN only with respect to FID.

### G. Correlation Analysis

To explore the relationship between the performance of subnets that inherit weight from the supernet and the actual performance of standalone 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 [58] measures the degree that two variables correlate, with a higher coefficient indicating a more positive correlation. It is evident from Fig. 10 that there is a moderately positive correlation between the performance of the directly inherited subnet and its actual performance (Kendall correlation coefficient: $\tau = 0.524$). Although the correlation in our method is not particularly powerful, it effectively increases the likelihood of

discovering excellent structures, which is consistent with the original intention of NAS.

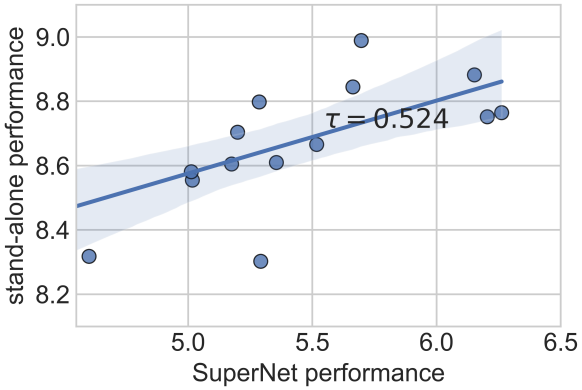


Fig. 10. The subnet's performance when inheriting weight *vs* stand-alone training performance.

## V. DISCUSSION

In this section, we discuss the limitations of our method and propose potential areas for future research. First, we acknowledge the need to evaluate the efficacy of our commonality-based discarding strategy in identifying underperforming operations. We should ensure that all operations in the supernet receive sufficient training time and avoid premature discarding. Finding ways to avert early dismissal due to insufficient training is an important requirement that warrants further investigation. Second, while our method focuses on optimising the IS and FID as multiple objectives, we may have overlooked the importance of network computing efficiency. Thus, future research should seek to strike a better balance between performance and efficiency when defining architecture search objectives; thus, exploring additional metrics or objectives that capture network efficiency would be valuable. Lastly, it is worth noting that our proposed framework does not cover the search for discriminator network structures, which significantly influence GAN performance. Extending our approach to include the search for optimal discriminator architectures is crucial for more comprehensive GAN improvement. Future studies should, therefore, investigate techniques for effectively searching discriminator network structures and determining their impact on overall GAN performance. In summary, addressing these limitations and incorporating advancements in our method would lead to a more robust and comprehensive architecture search for GAN generators and discriminators.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel approach to searching architectures for GANs, known as EWSGAN. The EWSGAN framework consists of two steps: (a) the training of a supernet generator and (b) an architecture search for high-performing generator using an MOEA. The two strategies proposed by us – fair single-path sampling and commonality-based discarding, can effectively improve the training stability of supernet generator. The experiments revealed that our method achieved the best performance on the CIFAR-10 and STL-10 datasets compared with peer method, and the architecture search can complete in just one GPU day on the CIFAR-10 dataset. Our method also demonstrates good scalability, and the searched structure can be directly extended to the CelebA dataset, exhibiting promising potential.

In future research, we plan to explore the possibility of applying our method to discriminator architecture search and investigating more efficient network architectures. Additionally, we will delve into the relationship between supernet-based architecture search methods and separately trained networks. Furthermore, we intend to extend our method's application to higher-resolution datasets for generation tasks, addressing more practical needs and enabling broader application scenarios. Lastly, we will explore other evaluation metrics for GANs and consider them as objectives for the architecture's design [59], [60].

## 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 *Proceedings of Advances in Neural Information Processing Systems*, 2014.

[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," in *Proceedings of Advances in Neural Information Processing Systems*, 2017, p. 5767–5777.

[11] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018, pp. 1–26.

[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 *Proceedings of International Conference on Learning Representations*, 2016, pp. 1–16.

[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] D. Zhou, X. Zhou, W. Zhang, C. C. Loy, S. Yi, X. Zhang, and W. Ouyang, "EcoNAS: Finding proxies for economical neural architecture search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2020, pp. 11 396–11 404.

[18] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proceedings of Advances in Neural Information Processing Systems*, 2016, p. 2234–2242.

[19] 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," in *Proceedings of Advances in Neural Information Processing Systems*, 2017, p. 6629–6640.

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

[21] 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.

[22] 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.

[23] 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.

[24] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair DARTs: Eliminating unfair advantages in differentiable architecture search," in *Proceedings of European Conference on Computer Vision*. Springer, 2020, pp. 465–480.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] R. Saleem, B. Yuan, F. Kurugollu, A. Anjum, and L. Liu, "Explaining deep neural networks: A survey on the global interpretation methods," *Neurocomputing*, vol. 513, pp. 165–180, 2022.

[35] 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.

[36] Y. Gong, Y. Sun, D. Peng, P. Chen, Z. Yan, and K. Yang, "Analyze covid-19 ct images based on evolutionary algorithm with dynamic searching space," *Complex & Intelligent Systems*, vol. 7, pp. 3195–3209, 2021.

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

[38] 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.

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

[40] Y. Xue and J. Qin, "Partial connection based on channel attention for differentiable neural architecture search," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 5, pp. 6804–6813, 2023.

[41] 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.

[42] X. Chen, Y. Sun, M. Zhang, and D. Peng, "Evolving deep convolutional variational autoencoders for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 815–829, 2020.

[43] 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 *Proceedings of European Conference on Computer Vision*. Springer, 2020, pp. 175–192.

[44] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "DARTs+: Improved differentiable architecture search with early stopping," *arXiv preprint arXiv:1909.06035*, 2019.

[45] M. Li, J. Lin, Y. Ding, Z. Liu, J.-Y. Zhu, and S. Han, "GAN compression: Efficient architectures for interactive conditional GANs," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5284–5294.

[46] V. Costa, N. Lourenço, J. Correia, and P. Machado, "COEGAN: Evaluating the coevolution effect in generative adversarial networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 374–382.

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

[48] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7105–7114.

[49] 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.

[50] M. Lindauer and F. Hutter, "Best practices for scientific research on neural architecture search," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 9820–9837, 2020.

[51] 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.

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

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

[54] 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.

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

[56] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3730–3738.

[57] D. Rey and M. Neuhäuser, *Wilcoxon-Signed-Rank Test*. Springer Berlin Heidelberg, 2011, pp. 1658–1659.

[58] 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.

[59] A. Borji, "Pros and cons of GAN evaluation measures: New developments," *Computer Vision and Image Understanding*, vol. 215, p. 103329, 2022.

[60] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7176–7185.

[61] S. You, T. Huang, M. Yang, F. Wang, C. Qian, and C. Zhang, "Greedy-NAS: Towards fast one-shot nas with greedy supernet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1999–2008.

[62] K. R. Traoré, A. Camero, and X. X. Zhu, "Fitness landscape footprint: A framework to compare neural architecture search problems," *arXiv preprint arXiv:2111.01584*, 2021.

**Yu Xue** received the Ph. D. degree from School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China, in 2013. He is a professor at the School of Software, Nanjing University of Information Science and Technology. He was a visiting scholar at the Victoria University of Wellington, New Zealand and at the Michigan State University, USA. His research interests include Deep Learning, Evolutionary Computation, Machine Learning, and Computer Vision.

**Weinan Tong** is currently pursuing the M.Sc. degree, with a focus on deep learning, image generation, and evolutionary neural architecture search, with the School of Software, Nanjing University of Information Science and Technology, Nanjing, China.

**Ferrante Neri** (M'03-SM'19) received the Laurea and Ph.D. degrees in electrical engineering from the Technical University of Bari, Bari, Italy, in 2002 and 2007, respectively, and the second Ph.D. degree in scientific computing and optimization and the D.Sc. degree in computational intelligence from the University of Jyväskylä, Jyväskylä, Finland, in 2007 and 2010, respectively. He is a Full Professor of machine learning and artificial intelligence and the Head of the Nature Inspired Computing and Engineering (NICE) Research Group, University of Surrey, Guildford, United Kingdom. He is also a Jiangsu Distinguished Professor, China. His research focuses on metaheuristic optimisation with applications in the context of machine learning

**Peng Chen** is a researcher at National Institute of Advanced Industrial Science and Technology (AIST) and a visiting scientist at RIKEN Center for Computational Science (RIKEN-CCS), Japan. He received the B.E. degree in navigation from Dalian Maritime University, China, in 2005; the M.E. degree in traffic information engineering and control from Shanghai Maritime University, China, in 2007; the Ph.D. from Tokyo Institute of Technology, Japan, in 2020. His research interests include parallel computing, image processing, and machine learning.

**Tao Luo** received his bachelor's degree from the Harbin Institute of Technology, Harbin, China, in 2010, his master's degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013, and his Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2018. He is currently a senior research scientist with the Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research, Singapore (A*STAR), Singapore. His current research interests include high-performance computing, efficient and green AI, quantum computing, and hardware–software co-exploration.

**Liangli Zhen** received his PhD degree in computer science from Sichuan University in 2018. He is a senior scientist and group manager at the Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), Singapore. His current research interests include machine learning and computer vision. He has published more than 30 papers in top tier journals and conferences, including IEEE TPAMI, TIP, TNNLS, ICLR, CVPR, and ICCV.

**Xiao Wang** is currently a research staff scientist at Oak Ridge National Laboratory. He received Bachelor's degrees in both Mathematics and Computer Science with honor from Saint John's University, MN, in 2012, and Master's degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2016. In 2017, he received a PhD degree in electrical and computer engineering from Purdue University. Then, he pursued postdoctoral research training at Harvard Medical School. His research interest lies in the intersection among machine learning, image processing and high performance computing.

# APPENDIX

## A. Parameter settings

Table IV and Table V respectively show the parameter settings used in our experiments.

TABLE IV
PARAMETER SETTINGS ASSOCIATED WITH THE NETWORK TRAININGS CONDUCTED IN THIS STUDY.

| Parameters | Training the Supernet | Re-Training | |
|---|---|---|---|
| | | CIFAR-10 | STL-10 |
| Epoch | 200 | 600 | 600 |
| Batch_Size G/D | 80/40 | 256/128 | 128/64 |
| Learning Rate $\alpha_G$ | 1e-4 | 2e-4 | 3e-4 |
| Learning Rate $\alpha_D$ | 1e-4 | 2e-4 | 2e-4 |
| $\beta_1$ | 0.0 | 0.0 | 0.0 |
| $\beta_2$ | 0.9 | 0.9 | 0.9 |
| n_critic | 5 | 5 | 5 |
| Latent Dimension | 120 | 120 | 120 |
| Channels of G/D | 256/128 | 256/128 | 256/128 |
| Loss Function | Hinge-Loss | Hinge-Loss | Hinge-Loss |
| ema decay | - | 0.999 | 0.999 |
| Warm-up Epoch | 50 | - | - |

TABLE V
PARAMETER SETTING ASSOCIATED WITH THE EVOLUTIONARY OPERATORS EMPLOYED IN THIS STUDY.

| Evolutionary Parameters | Value |
|---|---|
| Iteration Number | 20 |
| Population Size | 20 |
| Crossover Operation | uniform crossover |
| Mutation Probability | 1 |
| Number of Mutation Genes | 1-4 |
| Number of Evaluation Samples | 5000 |
| Optimisation Objective 1 | IS |
| Optimisation Objective 2 | FID |

## B. Alternative discard strategy

The purpose of our model pruning is to identify and remove potentially bad operations in the supernet to improve the stability and efficiency of supernet training. Therefore, determining how to effectively identify which operations are good and which are bad is the key to effective model pruning. In addition to our commonality-based discard method, the greedy discard method can be adopted, see [61]. The first step is to randomly sample a fixed number of subnet architectures and then evaluate and sort them. Subsequently, according to the sorting order, the next step is to merge the subnet operations into an empty operation pool until the latter meets the predefined goals. For instance, we need to reduce the number of choice block operations from seven to four, so we sample 42 network architectures for evaluation and sorting. The subnets are measured by calculating the IS. Starting with the top-ranked subnet, its operations must then be merged into new empty choice blocks until each choice block contains four different operations, and subsequent operations will no longer be added.

We compared the two discard strategies, and the experimental results are shown in Fig. 11. There is a definite gap between the two methods, with the commonality-based method proving superior, due to the fact that these types of methods focus more on the contribution of operations to the overall performance of the supernet. In other words, when an operation $a$ performs well when combined with other operations, it contributes significantly to the supernet's overall performance. However, when another operation $b$ only performs well in a specific combination, its contribution to the overall performance of the subnet is relatively minimal. As such, operation $a$ should be more conducive to stable supernet training; therefore, we prefer to use commonality-based discard methods.

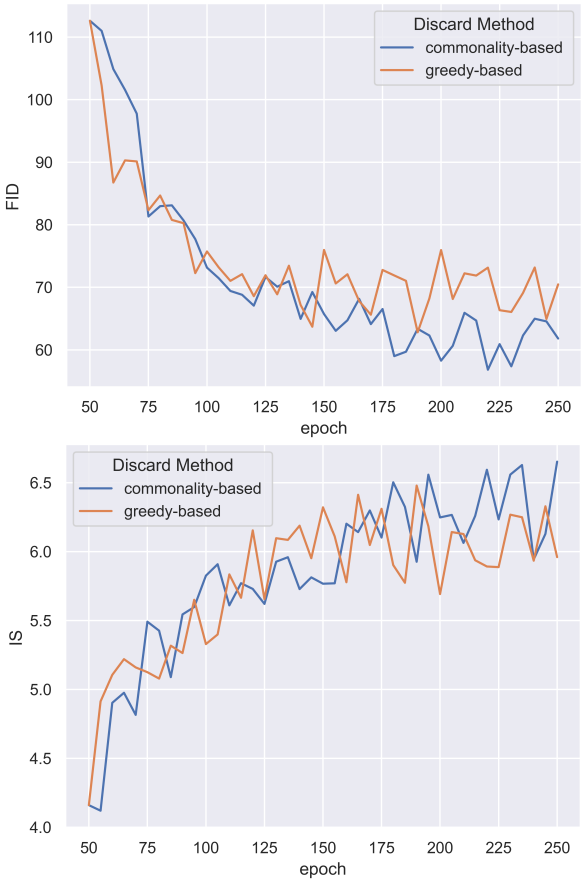


Fig. 11. The impact of different discard strategies on supernet training.

## C. Other crossover operators

Crossover operators affect search performance. Consequently, we investigate the performance of different crossover operators – uniform crossover, single-point crossover and two-point crossover – to determine the most suitable method for solving our evolutionary problem. We apply the same parameter settings, running them five times, and we use the average value and standard deviation of fitness for the 20 individuals

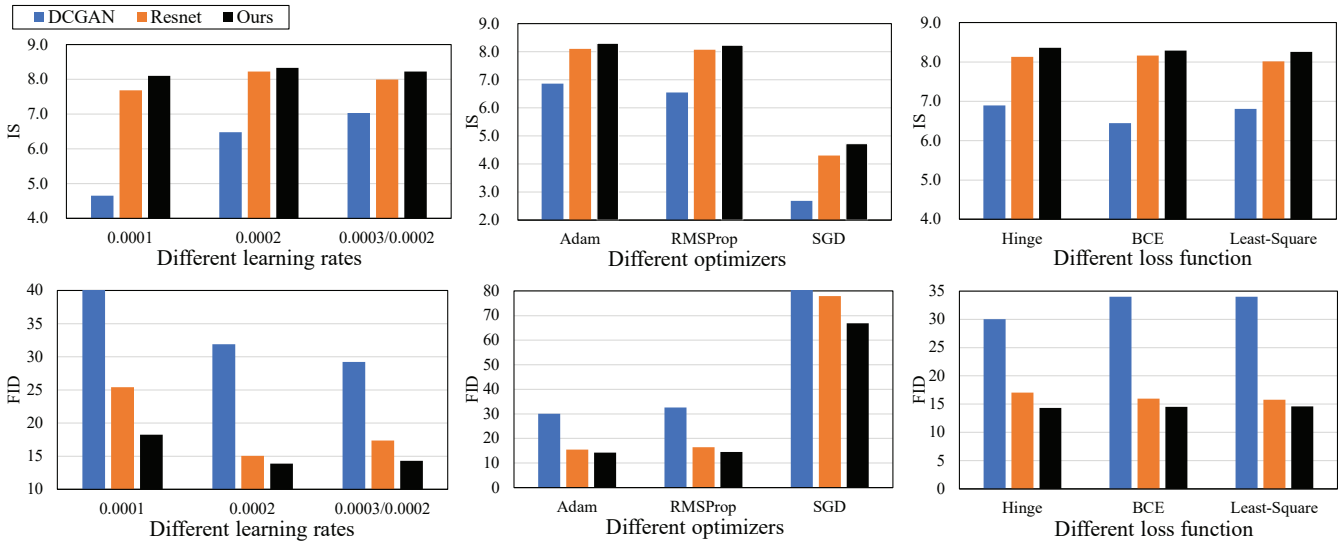


Fig. 12. The performance of different network structures under different parameter settings.

TABLE VI
THE IMPACT OF DIFFERENT CROSSOVER OPERATORS ON THE PERFORMANCE OF THE DESIGNED GENERATIVE ADVERSARIAL NETWORKS (GANS).

| | Uniform crossover | Single-point crossover | Two-point crossover |
|---|---|---|---|
| IS↑ | 5.57±0.20 | 5.59±0.21 | 5.37±0.16 |
| FID↓ | 80.83±1.16 | 80.96±2.68 | 81.55±1.82 |
| $p_{\text{IS}}$ | - | 9.2E-1 | 1.7E-1 |
| $p_{\text{FID}}$ | - | 7.5E-1 | 4.6E-1 |

in the last generation for comparison. The experimental results are shown in Table VI.

When considering the IS, uniform crossover is marginally less effective than single-point crossover. However, in terms of the FID, uniform crossover demonstrates a slight advantage; while the difference between the two methods is insignificant, the overall performance favours uniform crossover, which exhibits greater stability with a smaller standard deviation. Conversely, two-point crossover does not appear to be suitable for our approach when compared to the other two methods.

In summary, the uniform crossover is more suited to our evolutionary algorithm, as this crossover method endows offspring with more changes. The $p$-values confirm that the impact of the crossover operators on the EWSGAN performance is not significant.

### D. Parameter sensitivity

Different parameter settings have the potential to impact GAN performance. Therefore, we conducted experiments using various parameter configurations and compared them to two commonly used GAN network structures: the standard convolutional neural network proposed by DCGAN [13] and the ResNet architecture based on residual networks [11]. To evaluate their performances, we employed spectrum normalisation and analysed the effects of different parameter settings, focusing specifically on the learning rate, loss function and optimiser, which are frequently used in GANs. Each network is trained for 200 epochs by broadly following the indications in [62]. The results of our analysis, presented in Fig. 12, indicate that our network architecture consistently achieved stable performance across diverse parameter settings.