Evolutionary Design of Generative Adversarial Networks using Deep Evolution

Hana Derouiche[1], Maha Elarbi[1], Slim Bechikh[1,2], Carlos A. Coello Coello[3]
[1]SMART Lab, Computer Science Department, ISG, University of Tunis, Tunis, Tunisia
[2]Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, UK
[3]Cinvestav, Departamento de Computación (Evolutionary Computation Group), Mexico City 07360, MEXICO
Corresponding author email: hana.darouiche@gmail.com

***Abstract*—Designing effective architectures for Generative Adversarial Networks (GANs) remains a challenging task due to the strong interdependence between generator and discriminator structures, the instability of adversarial training, and the sensitivity of performance to architectural choices. Existing evolutionary Neural Architecture Search (NAS) approaches for GANs often optimize a single component or evaluate candidate architectures under a fixed generator–discriminator pairing, in spite of their distinct roles in adversarial training, which can lead to biased evaluations and poor generalizable designs. To address these limitations, we propose here an evolutionary NAS framework for GANs based on a Genetic Algorithm (GA). Each individual represents a full adversarial model and is evaluated at the GAN level using standard image quality metrics. Generator architectures are recombined using a standard crossover operator, while discriminator architectures are optimized via a Decuple Crossover Scheme (DCS), which is a recently introduced structured crossover scheme that generates multiple discriminator candidates through hierarchical recombination. Each offspring generator is paired with all discriminator candidates, and the resulting GANs are jointly evaluated to select the best-performing architectures for the next generation. This strategy enables efficient exploration of generator designs alongside deep, memetic refinement of discriminators, improving stability, robustness, and generative performance. Experimental results on standard image generation benchmarks show that our proposed framework consistently outperforms baseline GANs and existing evolutionary approaches.**

***Index Terms*—Generative Adversarial Networks (GANs), Neural Architecture Search (NAS), Evolutionary Computation (EC), Decuple Crossover Scheme (DCS), Deep evolution, Discriminator**

I. Introduction

GANs [1] have achieved remarkable success in image synthesis and representation learning [2], but their training remains remarkably unstable. A GAN relies on the interaction between a generator G, which synthesizes data, and a discriminator D, which evaluates realism. The quality of this adversarial process depends on maintaining a suitable balance between the two models. While much prior work has focused on G improvements, loss reformulations, and regularization strategies [3]–[6], the design of D architectures remains comparatively underexplored, despite its direct impact on the learning signal received by G. In practice, discriminators are often inherited from standard convolutional designs with limited adaptation, even though both insufficient and overly strong discriminators can harm convergence.

From an optimization perspective, GAN training implicitly assumes that D can track G closely enough to provide informative gradients, an assumption that rarely holds under non-stationary adversarial dynamics. Consequently, architectural design becomes a key factor in controlling training dynamics. Evolutionary algorithms (EAs) [7], [8] provide a suitable alternative to pure gradient-based optimization, as they explore populations of candidate solutions and are less sensitive to non-convexity and instability. However, most existing methods either evolve G and D symmetrically or rely on shallow evolutionary operators, without fully exploiting the hierarchical structure of discriminative models nor reflecting the distinct functional roles of the two components.

In this work, we propose a population-based evolutionary framework for NAS in GANs that explicitly accounts for these role differences. Each individual represents a complete adversarial architecture and is evaluated solely at the GAN level using standard image quality metrics. G architectures are evolved using canonical genetic operators to promote exploration, while D architectures are optimized using a DCS [26], which is a recently introduced deep evolutionary recombination scheme that performs multi-level recombination within a single reproduction event, generating multiple candidate discriminators and enabling structured architectural refinement. For each G offspring, several D variants produced by DCS are paired and evaluated. This one-to-many pairing allows the D to adapt more effectively to evolving generative distributions, reduces adversarial bias, and discourages over-specialization to a single opponent. This results in targeted D refinement while preserving exploratory evolution for generators.

The main contributions of this work are as follows:

- We propose to model the GAN architecture design as an evolutionary NAS problem
- We introduce an evolutionary strategy that applies standard genetic recombination to G and a deep evolutionary scheme to D, reflecting their distinct roles in adversarial learning.
- We embed DCS into D architecture optimization, enabling hierarchical recombination and effective local refinement beyond conventional crossover operators. The proposed GAN designed through DCS is termed Decuple Crossover-based Evolution of GAN (DCE-GAN).
- We achieve improved stability and generation perfor-

mance over baseline GANs and existing evolutionary methods on standard image generation benchmarks.

II. Previous related work

A. Generative Adversarial Networks

GANs, first introduced by Goodfellow et al. [1], formulate data generation as an adversarial game between a G and a D. The G maps latent variables $z \sim p_z(z)$ to synthetic samples, while the D aims to distinguish generated data from real samples drawn from $p_{\text{data}}(x)$. Their joint optimization is normally expressed as [1]:

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

Despite its elegance, this formulation is difficult to optimize in practice. GAN training is often affected by mode collapse, vanishing gradients, and oscillatory behavior, largely because the G depends on the D to provide stable and informative learning signals. Consequently, both model capacity and architectural design have a strong impact on convergence and final image quality. Prior work has addressed these issues through improved objectives [4] and regularization methods [5], [9], as well as through architectural advances such as DCGAN [3] and StyleGAN [11]. However, these architectures remain predominantly handcrafted and generally emphasize G design, whereas D design is more often reused than systematically optimized.

B. Neural Architecture search for GANs

NAS aims to automate network design by exploring a predefined search space using data-driven optimization strategies. Applying NAS to GANs is challenging due to adversarial coupling, unstable training, and the need to jointly evaluate G and D performance. Early approaches such as AutoGAN [12] and E2GAN [13] focused on G search while fixing D, whereas others attempted joint optimization using reinforcement learning or gradient-based methods [14], [15]. Subsequent studies [16] showed that strong generator–discriminator coupling often biases the search and hinders convergence, highlighting the need for more robust NAS strategies that can better tolerate adversarial instability.

C. Evolutionary NAS for GANs

EAs provide a natural framework for NAS in GANs, as they can effectively handle discrete and heterogeneous design spaces, noisy fitness evaluations, and non-differentiable objectives. Several evolutionary NAS-GAN frameworks have been proposed. COEGAN [17] co-evolves generators and discriminators, while EAGAN [18] formulates GAN design as a multi-objective search problem. Nevertheless, these approaches either focus only on G optimization or evolve G and D using identical evolutionary operators, without reflecting their distinct roles in adversarial learning. As a consequence, D architectures are often underexplored, despite their crucial influence on training dynamics and stability.

TABLE I: Summary of representative evolutionary NAS approaches for GAN optimization. Our proposed approach is listed in the last row.

Method	Searched network	Optimization scope	Change operators	Deep Evolution
EvoGAN [27]	G+D	A+H	M	✗
Bi-GAN [33]	G	A+P	M	✗
NSGA-II DCGAN [32]	G	A	C+M	✗
COEGAN [17]	G+D	A	M	✗
EAGAN [18]	G+D	A	C+M	✗
EAS-GAN [28]	G	A	M	✗
EWSGAN [29]	G	A	C+M	✗
DCE-GAN (Ours)	**G+D**	**A+H**	**C+M**	✓

A: Architecture, H: Hyperparameters optimization, P: Parameters; C: Crossover, M: Mutation.

EAs have also been applied directly to GAN training rather than architecture search. Approaches such as EGAN [19] and Evolution-GAN [20] focus on evolving G parameters under multiple objectives or mutation strategies to enhance diversity. Other frameworks, including Lipizzaner [21], Mustangs [22], and CDE-GAN [23], evolve populations of generator–discriminator pairs to improve robustness and to mitigate mode collapse. More recently, GAGAN [24] demonstrated that evolving D parameters using GAs while training a single G can improve stability and generative performance.

These findings highlight the critical role of discriminators in adversarial learning and suggest that targeted evolutionary refinement of discriminators can significantly enhance GAN training dynamics. A comparative summary of representative evolutionary NAS approaches for GANs is provided in Table I.

D. Deep evolution

Beyond conventional EAs, recent works have introduced the notion of deep evolution [25], [26], in which evolutionary operators are applied hierarchically or repeatedly within a single evolutionary step to intensify the exploitation of promising structures. Unlike standard crossover schemes that generate only two offspring per mating event, deep evolution emphasizes multi-level recombination, intermediate offspring reuse, and progressive refinement before selection. This paradigm is motivated by the limitations of canonical crossover operators, which often disrupt meaningful sub-structures and provide limited exploitation capability when optimizing complex representations. Early deep evolution strategies [25] therefore redefined crossover as a multi-stage process in which intermediate offspring are repeatedly recombined before final selection.

A concrete instantiation of deep evolution is the DCS [26], which produces ten offspring from two parents through a structured sequence of recombination steps involving both parents and intermediate solutions, enabling more intensive local exploitation while preserving diversity.

To date, deep evolution has not yet been explored in machine learning or NAS. Existing evolutionary NAS-GAN methods still rely on standard crossover applied once per generation, without exploiting hierarchical recombination or

role-specific evolutionary depth. To the best of the authors' knowledge, this paper introduces deep evolution to GAN architecture design for the first time, by embedding DCS into the D evolution process. This enables deep, multi-level D refinement under fixed G conditions, while preserving adversarial coupling at the GAN level, establishing deep evolution as a promising paradigm for GAN architecture search.

III. Proposed approach

A. Overview of the proposed DCE-GAN

We propose DCE-GAN, an evolutionary NAS framework in which G and D architectures are jointly evolved within a GA. Each individual represents a complete GAN architecture and is evaluated exclusively at the adversarial level using a standard image quality metric. Unlike evolutionary GAN methods that focus on weight optimization or hybrid evolution (i.e., gradient schemes), the proposed method operates purely on architectural level: network weights are randomly initialized, trained only during evaluation, and are not inherited across generations. Consequently, evolution is driven solely by architectural design choices.

The distinction between conventional evolutionary GANs and DCE-GAN is illustrated in Fig. 1. In standard settings, both G and D are recombined using the baseline crossover scheme, producing two offspring GANs per mating event. In contrast, Fig. 1(b) depicts the proposed framework, where G and D architectures are recombined using DCS within the same GA. Specifically, G architectures are evolved using a standard crossover scheme to promote diverse structural exploration. D architectures are optimized using the DCS, which generates ten D candidates through structured recombination, enabling deeper exploitation of the D architecture space while preserving adversarial coupling with the evolving G and maintaining architectural diversity.

Each individual in the population encodes a complete GAN architecture as a structured chromosome

$$\mathcal{C} = [\mathcal{A}_G \mid \mathcal{A}_D],$$

where $\mathcal{A}_G$ and $\mathcal{A}_D$ are fixed-length architectural genomes describing high-level design choices such as the number of layers, channel widths, kernel sizes, normalization strategies, activation functions, and sampling operations. The corresponding search space is summarized in Table II. While inspired by DCGAN-style designs, the search space allows substantial structural variation. Although G and D architectures are manipulated independently during the evolutionary process, they are never evaluated in isolation. All selection decisions are based solely on the performance of the instantiated GAN.

B. Evolutionary search procedure of DCE-GAN

The proposed DCE-GAN performs evolutionary NAS over complete adversarial models. Each individual encodes both G and D architectures and is evaluated solely through adversarial performance.

a) ***GAN encoding****:* At generation t, the population is defined as

$$\mathcal{P}^t = \{(\mathcal{A}^t_{G_i}, \mathcal{A}^t_{D_i})\}_{i=1}^{N},$$

with each individual representing a GAN architecture.

The initial population is generated by randomly sampling architectures from the predefined search spaces (Table II), and weights are randomly initialized prior to evaluation. Despite independent manipulation during variation, G and D are always encoded, selected, and evaluated as a single adversarial unit, preserving the intrinsic coupling of GAN training.

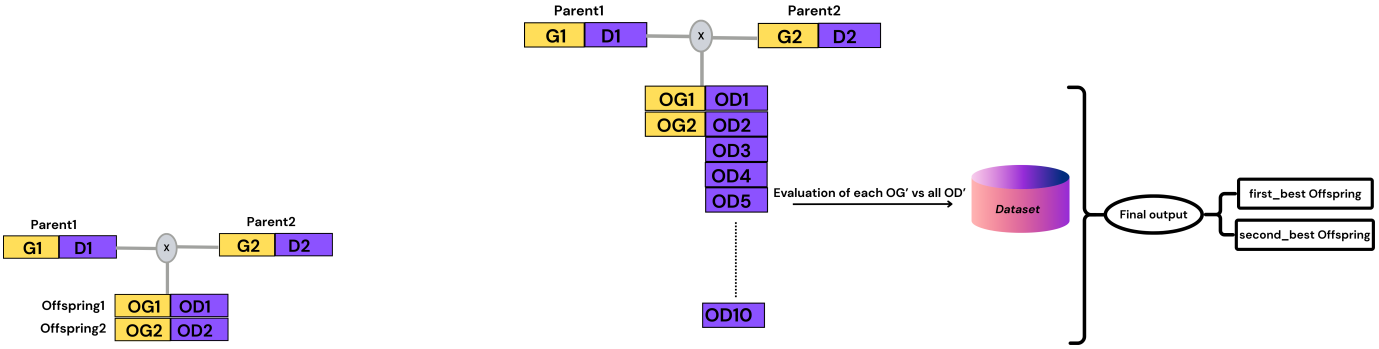
b) ***GAN evaluation****:* Evolution is driven exclusively by GAN-level fitness. Each candidate architecture $(\mathcal{A}_G, \mathcal{A}_D)$ is instantiated, trained for a fixed and limited number of adversarial updates, and evaluated using the the Fréchet Inception Distance (FID) [34] metric. Lower FID indicates better performance. This evaluation protocol is identical for all individuals and across all generations, ensuring a fair comparison and isolating the effect of architectural design from prolonged weight optimization. Parent selection is performed using binary tournament selection at the GAN level, meaning that generators and discriminators are always selected jointly. This preserves adversarial coupling during evolution while maintaining sufficient selection pressure and population diversity. Evolution proceeds until a predefined Number of Fitness Evaluations (NFE) is exhausted.

c) ***GAN deep variation****:* Given two parent GANs $(\mathcal{A}_{G_1}, \mathcal{A}_{D_1})$ and $(\mathcal{A}_{G_2}, \mathcal{A}_{D_2})$, G architectures are first evolved using standard genetic operators. Specifically, the uniform crossover operator is applied to the encoded G design variables, producing two offspring G architectures. Each offspring then undergoes architectural mutation with a fixed probability. In contrast, D architectures are refined using the DCS, which generates a set of ten D candidates through a structured, multi-level recombination process involving both parents and intermediate offspring. Unlike standard crossover, DCS explicitly intensifies exploitation of D search space by preserving and recombining discriminative substructures across multiple recombination stages.

Each offspring G is paired with all D candidates, forming multiple GANs that are evaluated under identical conditions. The two best-performing GAN architectures are selected to populate the next generation, maintaining a constant population size. The process continues until the evaluation budget is reached, and the best-performing GAN is returned. By combining standard evolutionary operators for generators with deep evolutionary recombination for discriminators, DCE-GAN allows for architectural exploration and gradual refinement of D structures.

IV. Experimental Study

In the following subsections, we present the evaluation metric adopted as well as the implementation details, followed by a comparative analysis against twelve state-of-the-art GAN approaches, including manually designed Reinforcement



(a) Baseline crossover

(b) Deep Crossover Scheme

Fig. 1: Illustration of the difference between (a) a baseline evolutionary GAN, where both G and D are recombined using standard crossover to produce two offspring GANs, and (b) the proposed DCE-GAN, where G undergo standard crossover while D are optimized via DCS, yielding multiple candidates evaluated at the GAN level.

TABLE II: List of optimized decision variables related to G and D in a DCGAN-inspired evolutionary GAN framework.

Component	Architectural Element	Symbol	Search Range	Description
Generator	Latent dimension	d_z	$\{64, 100, 128\}$	Dimension of input noise vector z
	Number of layers	L_G	$[4, 7]$	Total number of upsampling layers
	Kernel size	k_G	$\{3 \times 3, 4 \times 4, 5 \times 5\}$	Kernel size for transposed convolutions
	Channel width (initial)	c_G^0	$\{256, 512, 768\}$	Channels at lowest resolution
	Upsampling method	—	{TransConv, UpSample+Conv}	Choice of upsampling operation
	Normalization	—	{BatchNorm, InstanceNorm, None}	Normalization strategy per layer
	Activation function	σ_G	{ReLU, LeakyReLU, SELU}	Nonlinearity for hidden layers
	Output activation	—	{Tanh}	Output image scaling to $[-1, 1]$
Discriminator	Input resolution	—	$\{32, 64, 96, 128\}$	Input image size
	Number of layers	L_D	$[3, 6]$	Total number of downsampling layers
	Kernel size	k_D	$\{3 \times 3, 4 \times 4, 5 \times 5\}$	Kernel size for convolutions
	Channel width (initial)	c_D^0	$\{64, 128, 256\}$	Channels at highest resolution
	Convolution type	—	{Conv, SepConv, GroupConv}	Type of convolution operation
	Normalization	—	{SpectralNorm, BatchNorm, None}	Normalization or regularization method
	Activation function	σ_D	{LeakyReLU, ELU}	Nonlinearity for hidden layers
	Output formulation	—	{Sigmoid, LeastSquares}	Discriminator loss formulation

Learning-based (RL), gradient-based, and evolutionary frameworks. Experiments were conducted on two standard benchmarks: **CIFAR-10** [30], which consists of 60,000 32×32 RGB images across 10 classes and is widely used for low-resolution generation, and **STL-10** [31], which contains higher-resolution images from 10 categories. In our setting, STL-10 images were resized to 64×64, providing a more challenging scenario due to increased visual complexity.

A. Evaluation metric

Generative performance is evaluated using FID [34], which measures the discrepancy between real and generated image distributions in the feature space of a pretrained Inception-V3 network. Let (μ_r, Σ_r) and (μ_g, Σ_g) denote the mean and covariance of real and generated features, respectively. The FID score is defined as [34]:

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}\right). \quad (2)$$

Lower FID values indicate better generative quality. All experiments are repeated over 20 independent runs, and we report the median FID to ensure robustness against stochastic training effects. Following standard practice in the NAS-GAN literature [12], [17], [18], [28], FID is computed using 50,000 generated images for each evaluated architecture.

B. Implementation and evolutionary settings

We adopted a DCGAN-inspired search space for both G and D architectures (Table II), covering key design choices such as depth, kernel sizes, channel widths, normalization, activation functions, and sampling strategies. This design ensures both architectural diversity and compatibility with standard GAN training. Each candidate GAN was trained for a fixed number of mini-batch updates using Adam with a learning rate of 2×10^{-4} and momentum parameters $(\beta_1 = 0.5, \beta_2 = 0.99)$. This budgeted evaluation protocol was applied uniformly across all individuals and compared methods, ensuring that performance differences arise from architectural quality rather than from extended training. No additional post-processing techniques, truncation tricks, or architectural heuristics were employed.

The evolutionary configuration of the proposed DCE-GAN is summarized in Table III. The search is terminated after a fixed number of fitness evaluations (NFEs), providing a computation-aware stopping criterion and enabling a fair comparison across all evolutionary methods. All experiments were

TABLE III: GA configuration for DCE-GAN.

Parameter	Value
Population size	20
Selection	Binary tournament
Crossover	Uniform (prob. 1.0)
Mutation	Uniform (prob. 0.05)
Stopping criterion	4,000 NFEs

implemented in PyTorch and executed on a single NVIDIA GPU. For non-evolutionary baselines, we report the results as published in the original works, following their respective evaluation protocols.

C. Comparative results

Table IV reports the median FID scores across all methods. DCE-GAN consistently outperforms both manually designed and automated approaches.

On CIFAR-10, DCE-GAN achieves the lowest median FID, indicating that the evolved architectures better approximate the real data distribution. This confirms the effectiveness of discriminator-centric deep evolution in improving adversarial learning. On STL-10, which is more complex, due to higher resolution and increased visual complexity, the performance gap further widens. While all methods exhibit higher FID scores compared to CIFAR-10, DCE-GAN maintains a clear advantage over competing NAS-based and evolutionary GAN approaches, demonstrating its ability to scale to more complex data distributions where the discriminative capacity is critical. Unlike existing approaches, where D architectures receive limited focused exploitation, which can lead to weaker or unstable adversarial feedback during training, DCE-GAN explicitly allocates deeper refinement to discriminators via DCS while maintaining exploratory evolution for generators. This role-aware strategy leads to more stable and effective adversarial interactions.

Finally, the results demonstrate that not only joint GAN evolution but also the nature of architectural variation, particularly for discriminators, is crucial for achieving high performance. By combining standard G evolution with deep, structured D recombination, DCE-GAN achieves superior image quality compared to existing evolutionary NAS-GAN baselines.

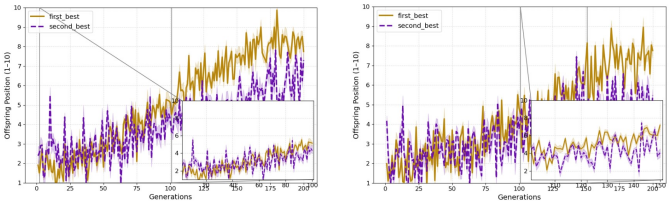
D. Ablation study

1) Impact of deep discriminator recombination: To isolate the contribution of deep D evolution, we compare DCE-GAN with a baseline evolutionary GAN using identical settings, search space, and evaluation protocol, differing only in the D variation mechanism. In the baseline, both G and D are evolved using standard crossover and mutation, producing two offspring GANs per mating event (Fig. 1(a)). This symmetric treatment assumes similar optimization needs for both components, which is inconsistent with GAN dynamics: generators benefit from exploration, whereas discriminators require rapid and fine-grained adaptation.

As shown in Table IV, the baseline achieves FID scores of 12.35 (CIFAR-10) and 25.78 (STL-10), significantly worse

TABLE IV: Median FID scores (lower is better).

Method	Search method	GPU Days	CIFAR-10	STL-10
DCGANs [3]	Manual	—	37.7	—
WGAN-GP	Manual	—	29.3	—
Progressive GAN [10]	Manual	—	18.33	—
AGAN [14]	RL	1200	30.5	52.7
AutoGAN [12]	RL	2	12.42	31.01
E2GAN [13]	RL	0.3	11.26	25.35
DEGAS [35]	Gradient	1.2	12.01	28.76
AlphaGAN [16]	Gradient	1	15.58	25.42
AdversarialNAS [15]	Gradient	1	24.04	38.85
EGAN [19]	EA	1.25	31.6	34.2
EAS-GAN [28]	EA	1	33.2	38.84
EAGAN [18]	EA	1	12.84	26.52
Baseline Evo GAN	EA	1	12.35	25.78
DCE-GAN (Ours)	**EA**	**1**	**9.76**	**21.98**



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

Fig. 2: Evolution of D offspring positions producing the best and second-best GANs. Curves denote the median across runs, with shaded areas indicating the interquartile range (Q_1–Q_3).

than DCE-GAN (9.76 and 21.98). The improvement directly stems from DCS, which enables deeper architectural refinement by generating multiple D candidates. These results demonstrate that standard genetic operators are insufficient for D optimization and that deep recombination is critical for robust adversarial learning.

2) Evolution of best GAN architectures within DCS: To better understand the role of DCS, we analyze where the best-performing GANs originate within the DCS hierarchy. At each generation, we record the D offspring index (OD1–OD10) associated with the best and second-best GANs, aggregated over 20 runs on CIFAR-10 and STL-10.

Figure 2 shows that early generations exhibit high variability, with strong candidates emerging from intermediate offspring positions, reflecting a highly exploratory behavior. As the evolutionary process progresses, the best GANs increasingly originate from deeper levels (OD6–OD10), while variability decreases. This shift indicates that hierarchical recombination progressively refines D architectures, and that deeper DCS levels play a dominant role at later stages. These observations provide empirical support for the effectiveness of deep evolution, confirming that multi-level recombination enables structured refinement that cannot be achieved with shallow operators.

V. Conclusions and Future Work

This paper presented DCE-GAN, an evolutionary NAS framework for GANs in which complete adversarial models

are evolved and evaluated jointly at the GAN level. The proposed approach explicitly differentiates the evolutionary treatment of G and the D. G is optimized using standard genetic recombination to promote exploration, whereas D is refined via DCS, enabling deeper and more competitive architectural exploitation. By evaluating candidate solutions at the GAN level using FID, DCE-GAN preserves adversarial coupling throughout evolution and avoids biases arising from isolated component evaluation. Ablation studies further confirm that the performance gains are primarily driven by deep D recombination. Experimental results on CIFAR-10 and STL-10 show consistent improvements in median FID and a reduced performance variance compared to representative baselines, indicating enhanced stability and robustness, highlighting the importance of targeted D refinement within evolutionary GAN architecture search.

Future work will explore several promising directions. First, DCE-GAN can be extended to conditional settings, where D specialization may play an even more critical role. Second, incorporating multi-objective optimization criteria [36] would enable more flexible trade-off analysis. Finally, adapting DCE-GAN to higher-resolution datasets and more expressive GAN families may further validate its scalability and generality.

REFERENCES

[1] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron and Bengio, Yoshua. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

[2] Shamsolmoali, P., Zareapoor, M., Granger, E., Zhou, H., Wang, R., Celebi, M. E., Yang, J. (2021). Image synthesis with adversarial networks: A comprehensive survey and case studies. Information Fusion, 72, 126-146.

[3] Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv 2015, arXiv:1511.06434.

[4] Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein Generative Adversarial Networks. In Proceedings of the 34th International Conference on Machine Learning, PMLR 70, Sydney, Australia, 6–11 August 2017.

[5] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A. C. (2017). Improved training of wasserstein gans. Advances in neural information processing systems, 30.

[6] Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks. arXiv preprint arXiv:1611.02163.

[7] Goldberg, D. Genetic Algorithms in Search, Optimization and Machine Learning; Addison-Wesley: Reading, MA, USA, 1989.

[8] Michalewitz, Z. Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 1996.

[9] Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957.

[10] Karras, T., Aila, T., Laine, S., Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv:1710.10196.

[11] Karras, T., Laine, S., Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4401-4410).

[12] Gong, X., Chang, S., Jiang, Y., Wang, Z. (2019). Autogan: Neural architecture search for generative adversarial networks. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 3224-3234).

[13] Gong, Y., Zhan, Z., Jin, Q., Li, Y., Idelbayev, Y., Liu, X., ... Ren, J. (2024). E ^{2} GAN: Efficient Training of Efficient GANs for Image-to-Image Translation. arXiv preprint arXiv:2401.06127.

[14] Wang, H., Huan, J. (2019). Agan: Towards automated design of generative adversarial networks. arXiv preprint arXiv:1906.11080.

[15] Gao, C., Chen, Y., Liu, S., Tan, Z., Yan, S. (2020). Adversarialnas: Adversarial neural architecture search for gans. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5680-5689).

[16] Lutz, S., Amplianitis, K., Smolic, A. (2018). Alphagan: Generative adversarial networks for natural image matting. arXiv preprint arXiv:1807.10088.

[17] V. Costa, N. Lourenço, J. Correia, and P. Machado, "Coegan: Evaluating the coevolution effect in generative adversarial networks," in Proceedings The Genetic and Evolutionary Computation Conference (GECCO), 2019, pp. 374–382.

[18] Q. Hoang, T. D. Nguyen, T. Le, and D. Q. Phung, "Mgan: Training generative adversarial nets with multiple generators," in Proceedings International Conference on Learning Representations (ICLR), 2018, pp.1–24.

[19] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," IEEE Transactions on Evolutionary Computation, vol. 23, pp. 921–934, 2019.

[20] Liang, Y., Han, Z., Nie, X., Ohkura, K. (2022). Improving generative adversarial network with multiple generators by evolutionary algorithms. Artificial Life and Robotics, 27(4), 761-769.

[21] T. Schmiedlechner, A. Al-Dujaili, E. Hemberg, and U.-M. O'Reilly, "Towards distributed coevolutionary gans," in Proceedings AAAI 2018 Fall Symposium, 2018, pp. 1–6.

[22] J. Toutouh, E. Hemberg, and U.-M. O'Reilly, "Spatial evolutionary generative adversarial networks," in Proceedings The Genetic and Evolutionary Computation Conference (GECCO), 2019, pp. 472–480.

[23] Chen, S., Wang, W., Xia, B., You, X., Peng, Q., Cao, Z., Ding, W. (2021). CDE-GAN: Cooperative dual evolution-based generative adversarial network. IEEE Transactions on Evolutionary Computation, 25(5), 986-1000.

[24] Konstantopoulou, D., Zacharia, P., Papoutsidakis, M., Leligou, H. C., Patrikakis, C. (2024). GAGAN: Enhancing Image Generation through Hybrid Optimization of Genetic Algorithms and Deep Convolutional Generative Adversarial Networks. Algorithms, 17(12), 584.

[25] Derouiche, H., Elarbi, M., Bechikh, S. (2025). Deep crossover schemes for genetic algorithms: Investigations on the travel salesman problem. Swarm and Evolutionary Computation, 98, 102094.

[26] Derouiche, H., Elarbi, M., Bechikh, S. (2026). A decuple crossover scheme in genetic algorithms: a step toward deep evolution. Evolutionary Intelligence, 19(1), 9.

[27] Liu, F., Wang, H., Zhang, J., Fu, Z., Zhou, A., Qi, J., Li, Z. (2022). EvoGAN: An evolutionary computation assisted GAN.

[28] Lin, Q., Fang, Z., Chen, Y., Tan, K. C., Li, Y. (2022). Evolutionary architectural search for generative adversarial networks. IEEE Transactions on Emerging Topics in Computational Intelligence, 6(4), 783-794.

[29] Xue, Y., Tong, W., Neri, F., Chen, P., Luo, T., Zhen, L., Wang, X. (2023). Evolutionary architecture search for generative adversarial networks based on weight sharing. IEEE transactions on evolutionary computation, 28(3), 653-667.

[30] Krizhevsky, A., Hinton, G. (2009). Learning multiple layers of features from tiny images.

[31] Coates, A., Ng, A., Lee, H. (2011, June). An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics (pp. 215-223). JMLR Workshop and Conference Proceedings.

[32] Du, L., Cui, Z., Wang, L., Ma, J. (2020). Structure tuning method on deep convolutional generative adversarial network with nondominated sorting genetic algorithm II. Concurrency and Computation: Practice and Experience, 32(14), e5688.

[33] Lu, Y., Kakillioglu, B., Velipasalar, S. (2018). Autonomously and simultaneously refining deep neural network parameters by a bi-generative adversarial network aided genetic algorithm. arXiv preprint arXiv:1809.10244.

[34] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 30.

[35] Doveh, S., Giryes, R. (2021). DEGAS: differentiable efficient generator search. Neural Computing and Applications, 33(24), 17173-17184.

[36] Said, R., Bechikh, S., Louati, A., Aldaej, A., Said, L. B. (2020). Solving combinatorial multi-objective bi-level optimization problems using multiple populations and migration schemes. Ieee Access, 8, 141674-141695.