Knowledge-Based Systems 276 (2023) 110757

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

AutoInfo GAN: Toward a better image synthesis GAN framework for high-fidelity few-shot datasets via NAS and contrastive learning

Jiachen Shi [a], Wenzhen Liu [a], Guoqiang Zhou [a,*], Yuming Zhou [b]

[a] College of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China
[b] Department of Computer Science and Technology, Nanjing University, 210093, Nanjing, China

A R T I C L E   I N F O

Article history:
Received 9 April 2023
Received in revised form 21 June 2023
Accepted 24 June 2023
Available online 30 June 2023

Keywords:
Few-shot learning
High-fidelity image synthesis
Generative adversarial network
Neural architecture search
Contrastive learning

A B S T R A C T

Background: Generative adversarial networks (GANs) are vital techniques for synthesizing high-fidelity images. Recent studies have applied them to generation tasks under small-data scenarios. Most studies do not directly train GANs on few-shot datasets, which have small data samples; instead, they borrow methods to transfer knowledge from large datasets to GANs with small ones. Partial fine-tuning of GANs is difficult to ensure the transfer performance, especially when the image domains are of great difference. FastGAN firstly trains GAN with small data samples by a carefully designed skip layer exception (SLE) connection to improve synthesis and an unsupervised discriminator to avoid overfitting. Problem. However, in FastGAN, different designs of SLE connections and operation settings would lead to great differences in synthesis performance. It is necessary to find the most appropriate ways of architecture design. Meanwhile, FastGAN merely improves discriminator learning, but ignores that the generator learning process is also insufficient due to small data samples.
Objective: Based on FastGAN, this study aims to find the best generator designs and then improve the training process of it via unsupervised learning. Methods. This work applies a reinforcement learning neural architecture search method to find the optimal GAN architecture and an unsupervised contrastive loss function assisted by a discriminator to optimize generator learning. These two methods constitute our AutoInfoGAN.
Results: Experiments were conducted on 11 datasets using AutoInfoGAN, covering a wide range of image domains, achieving better results than state-of-the-art (SOTA) models.
Conclusion: The experimental results demonstrate the SOTA performance of our proposed AutoInfo GAN on few-shot datasets, and we are cautioned that although instance normalization (IN) improves synthesized image quality, it performed poorly in our mode-collapse test.

1. Introduction

The emergence of the deep learning generative adversarial network (GAN) [1] has shown great potential in image synthesis and translation, photo editing, and artistic creation. When using real-world datasets, such as those that contain portraits of real people or the artworks of specific artists, the availability of training samples is scarce, especially high-fidelity (i.e. 256 $\times$ 256 or greater) ones. Typically, high-fidelity few-shot datasets have 1k or fewer usable samples. This has led to interesting research that has generated novel image synthesis methods [2–7]. Extant research has broadened the scope of data augmentation and promoted the industrialization of deep learning. GAN training under high-resolution (i.e. high-fidelity) requirements with small data samples is too difficult for most real-world applications. The GAN learning problem caused by small datasets and the subsequent poor performance when tackling fine-grained high-fidelity image tasks must be overcome while preventing mode collapse and overfitting.

Owing to the difficulty of training GANs on small datasets, data-transference techniques have arisen that leverage large datasets to augment smaller ones. This large-to-small (L2S) method pretrains the model using large-scale sets in advance, followed by transferring the knowledge into few-shot datasets using fine-tuning methods [8–10] or the transference penalty [11–14]. However, when the inter-domain gap is large, the transfer performance will drop sharply, which is often inferior to GANs directly trained on the few-shot datasets. L2S models are pre-trained on large datasets such as the Visual Geometry Group (VGG) Face set(2394 categories, 100 pieces each)[1], Animal Faces

* Corresponding author.
E-mail address: zhougq@njupt.edu.cn (G. Zhou).

1 https://www.robots.ox.ac.uk/~vgg/software/vgg_face/

(149 categories, 100 pieces each)[2], and Flowers (102 categories, 40 pieces each)[3].

Furthermore, InsGAN [15] and FastGAN [16] models directly train using few-shot datasets by introducing contrastive learning methods [17] and unsupervised discriminators, respectively, to augment small datasets. FastGAN is a progressively architecture design that could generates high-fidelity images and improves training efficiency. However, it does not take optimization measures for generator to overcome the insufficient learning caused by only a few training samples and does not find the optimal architecture to better extract internal data feature. InsGAN also does not conduct more research on overfitting and mode-collapse problems.

In this paper, in the face of high-fidelity few-shot datasets, we propose AutoInfo GAN based on the FastGAN, leveraging two key technologies to improve image synthesis from the perspective of architecture design and generator training. First, we employ the neural architecture search (NAS) algorithm to find the optimal generator architecture for better extracting the feature information from images. Second, we provide a novel contrastive representation learning method for the generator to mitigate the problem of small training samples and ensures effective network learning. Third, we conduct comprehensive experiments to investigate the effectiveness of our AutoInfo GAN in comparison to the state-of-the-art (SOTA) models. Furthermore, we conduct ablation studies to provide a deeper understanding of the performance contributions of our architectural components and loss function. This study makes the following major contributions based on the FastGAN [16] model:

1. A hybrid two-stage NAS method with dynamic reset and two-stage searching capabilities that quickly finds the optimal generator architecture.
2. A discriminator-assisted contrastive optimization function that makes full use of few-shot image datasets to improve learning.
3. An improved fine-grained feature extraction method for few-shot datasets.
4. An improved GAN fine-tuning method with data-transfer training techniques for simple real-world applicability.
5. An open source code for this study, which can be easily used in future high-fidelity few-shot image synthesis studies[4] and can facilitate researchers to use AutoInfo GAN as a baseline model, thus helping develop more effective approaches.

The organization of the remainder of this paper is as follows. Section 2 introduces important works and results related to our approach. Section 3 reviews the baseline GAN architecture. Section 4 details our methodology, and Section 5 details our experimental results and provides analytical interpretations via an ablation study. Section 6 to Section 8 provide discussions of hyper-parameters, implications for researchers and practitioners, and potential validity threats of our study. Finally, Section 9 provides a conclusion and recommendations for future research.

2. Related works

The most critical problems in the generation task of high-fidelity few-shot datasets are how to solve the problem of insufficient training samples and how to make GAN sufficiently extract the fine-grained features hidden in the high-resolution images. Most of the existing works employ transfer learning to avoid small samples. There is little work to train GANs directly on small data sets. Furthermore, we could improve the feature extraction ability of GANs by carefully designing the architecture and hence better extracting the fine-grained information. Therefore, this paper introduces the related work from the perspective of existing few-shot image generation and GAN architecture design, and analyzes their problems.

2.1. Few-shot high-resolution image synthesis

This paper defines that the few-shot generation problem is directly employing small datasets, usually less than 1k samples, to train GANs to generate high-resolution images. While some research use transferring methods to solve the few-shot generation problems, which are as follows:

2.1.1. Large-to-small dataset transference

Fine-tuning methods. Noguchi et al. [8] examined the effects of fine-tuning performance based on the correlation between source and target domains on pre-trained weights, resulting in new L2S transfer methods. Zhao et al. [9] conducted migration research on different model layers and discriminator and generator parameters to illuminate the significant correlation between training methods and effects, which inspired additional studies on the intelligent leveraging of small data to train GANs. Wang et al. [10] introduced a simple knowledge mining network that strongly benefitted specific target domains by leveraging prior knowledge, and the generation performance of the model was improved in the target domain.

Regularization methods. Ojha et al. [12] provided transfer methods that leveraged cross-domain distance loss methods to preserve the relative similarities and differences of source-domain instances. An anchor-based strategy was applied to encourage latent space regional diversity. Li et al. [11] regularized weight changes during transference to better preserve the source information while simultaneously fitting the GAN to the target domain. Xiao et al. [13] designed a cross-domain spatial structural consistency loss that helped align the spatial structural information between synthetic image pairs in both source and target domains. Zhao et al. [14] discovered that slowing the degradation of diversity could enhance adaptation and proposed a dual contrastive learning method that retains the source domain's multilevel diversity information for the target domain generator. Furthermore, Zhao et al. [18] proposed knowledge truncation to mitigate the incompatible knowledge transfer problem. Li et al. [19] addressed the extremely imbalanced data augmentation problem by designing a new penalty function and a new evaluation approach to assess the availability of generated instances.

Generally, fine-tuning methods only improve a subsection of a model's parameters. Regularization-based methods normalize the transfer process based on prior knowledge, usually by imposing a penalty on the parameter- and feature-change processes. However, the modification of network parameters does not apply to dataset samples with significant style differences, and applying such modifications can make things worse. Hence, we still need to change the architecture or find a generalized design that can achieve powerful generative effects on divergent small datasets.

2.1.2. Only small datasets

Besides that, there are some work carefully design GANs and directly train them on the small datasets. Jeong et al. [17] and Zhao et al. [15] successively applied contrastive learning to few-shot image generation, causing the discriminator to distinguish real and generated images via infoNCE loss, which better guided

2 https://github.com/NVlabs/FUNIT
3 https://www.robots.ox.ac.uk/~vgg/data/flowers/102/
4 https://github.com/shijiachen1/AutoInfoGAN

the generator. Based on this, Liu et al. [16] successfully generated high-fidelity images using a carefully designed architecture with a skip-layer exception (SLE) module and a self-supervised discriminator with a decoder to reconstruct images, achieving SOTA performance. Yang et al. [20] aligned the prototypes and features of real and generated images in the embedding space to improve the fidelity of the generated images. Tseng et al. [21] proposed the Lecam regularization of the discriminator to solve the overfitting problem, demonstrating that its regularization was theoretically effective in few-shot dataset scenarios. For extreme cases, Sushko et al. [22] and Shaham et al. [23] conducted experiments on exceedingly small samples (1–10). However, the generation model repeatedly processed patterns of single images, resulting in biased learning results. These methods pay more attention to the discriminator constraints while overlooking the improvement of the generator. Therefore, the current study provides more attention to generator networks.

2.2. GAN architecture

2.2.1. Manual design for high-resolution images

The self-attention (SA) GAN [24] adds an attention mechanism to a convolution-based GAN to provide a non-local model that assists the generator and discriminator in building relationships among different feature areas. The SA mechanism contains three 1×1 convolutions for the SoftMax and multiple feature maps to acquire attention feature maps, resulting in a good performance. BigGAN [25] greatly improves the generation fidelity of SAGAN by increasing the batch size, introducing truncation techniques, controlling the gradients of the generator and discriminator, and sending noise vector z to multiple layers rather than just the initial one to directly affect features of different resolutions and hierarchies. The ProgressiveGAN model [26] generates and extracts fine-grained feature information hidden in high-resolution images by changing the size of the feature maps and the number of channels step-by-step. This type of architectural design has been widely applied owing to its practical convenience and excellent performance in high-resolution image synthesis tasks. StyleGAN [27] provides a feature decoding network based on ProgressiveGAN to decouple the latent space, gradually adding it back to the different generator network layers to control the generated high-quality image content.

These GANs are manually designed for large-scale datasets and require significant human resources, so applying them directly to few-shot datasets for image synthesis would not guarantee the model convergence. Therefore, the necessary time-consuming manual design of the GAN architecture is needed to adjust the operation order and hyperparameters to achieve better performance.

2.2.2. Automatic design

Gong et al. [28] first proposed a NAS framework (i.e. AutoGAN) for a GAN. They used an on-policy reinforcement learning (RL) algorithm and policy gradient (PG) to search the optimal generator architecture and achieved SOTA performance at that time on the CIFAR-10 and STL-10 datasets. Subsequently, Tian et al. [29] introduced an off-policy RL soft actor critical network to improve search efficiency and effects. Gao et al. [30] used a gradient optimization algorithm to search the generator and discriminator networks simultaneously to achieve SOTA results. Several studies (e.g., StableAutoGAN [31], Efficient AutoGAN [32], and Multi-Self GAN [33]) further improved search efficiency and stability. Besides that, more research have been proposed to improve the architecture design [34,35] and apply them into other fields as image segmentation [36]. However, these algorithms require large-scale datasets (e.g. CIFAR-10 and STL-10). Therefore, real-world few-shot dataset problems remain underrepresented, and finding the optimal public architecture remains a problem.

Although NAS-driven GANs have resulted in significant achievements, they still lack performance analysis capability with high-fidelity synthesis requirements using few-shot datasets. Moreover, these architectures are overly complex and require carefully designed search spaces and search algorithms.

3. Preliminaries

This section reviews the baseline GAN architecture (including generator and discriminator) used in searching optimal GAN and the baseline loss function used in training derived GAN from scratch.

3.1. Generator network

FastGAN's generator progressively combines two different upsampling blocks: UpblockComp and Upblock (see Fig. 1(a)). From there, it is notable that noise injection and GLU are two operations used to stabilize the GAN training and their detailed explanations could be seen in FastGAN [16]. It also exhibits an SLE mechanism to connect the entire architecture. We can also observe that one of the most obvious characteristics of SLE connection is the data multiplication of two feature maps, which could greatly reduce the computational burden. Inside the upsampling blocks, normalization and convolution operations are also conducted. The general architecture is illustrated in Fig. 1(b). This paper introduces more combination ways of upsamplings, normalizations, and SLE connections to fine-tune the generator architecture.

3.2. Discriminator architecture

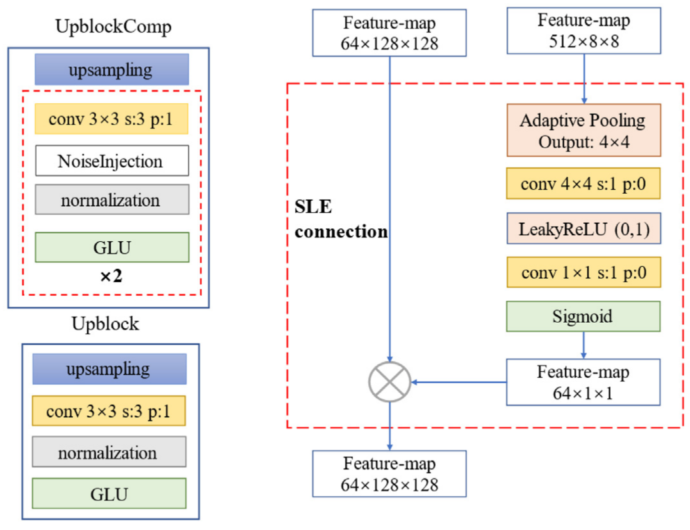
The discriminator network includes an additional decoder network to reconstruct intermediate feature maps from the discriminator. Generally, a discriminator is used to judge the input images, and the decoder network is used to realize the reconstruction task to better guide generator training. The details are shown in Fig. 2. In Fig. 2(a), we can see that the DownblockComp in the discriminator conducts two different operations on input feature maps and then get the addition value as the final output. This kind of operation is more conducive to fine-grained feature extraction in high-fidelity images. In Fig. 2, we can find that discriminator also has SLE connections to build the entire architecture, which is aimed to correspond to the generator architecture and be beneficial to the stable training of GAN.

3.3. Original loss function

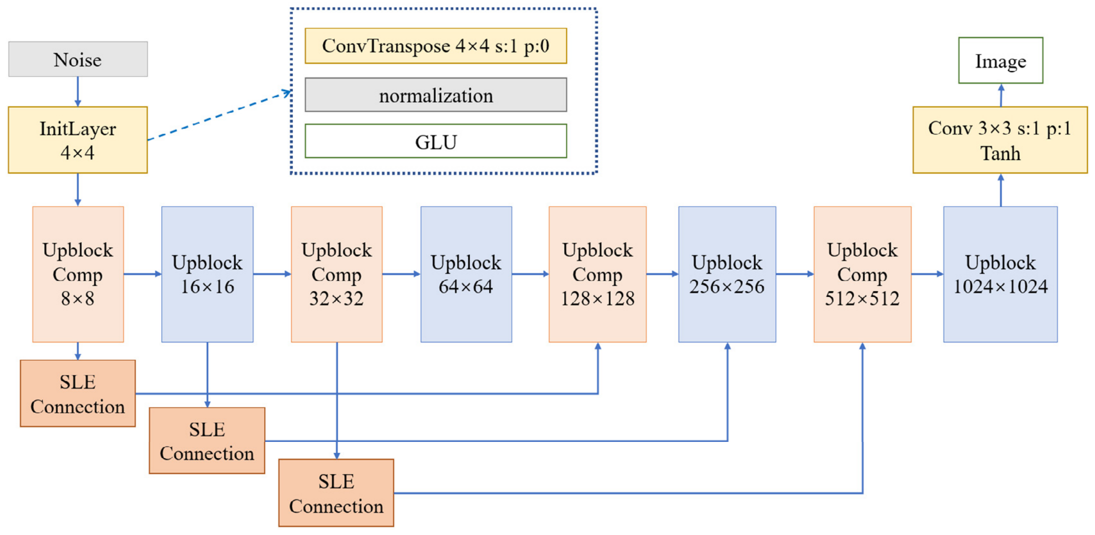
The original loss function of fastGAN includes the hinge version of the adversarial and reconstruction losses for the discriminator. Practically, the learned perceptual image patch similarity (LPIPS) [37] is used as the reconstruction loss to measure the L2 distance between real and reconstructed images. The applied equations are as follows:

$$\begin{aligned} LP\left(x, x_0\right) &= \sum_{l} \frac{1}{H_l W_l} \sum_{h,w} \left\| w_l \odot \left(\hat{y}^l_{hw} - \hat{y}^l_{0hw}\right) \right\|_2^2 \\ \hat{y}^l, \hat{y}^l_0 &\in \mathbb{R}^{H_l \times W_l \times C_l} \\ \mathcal{L}_{\text{recons}} &= \mathbb{E}_{\mathbf{f} \sim D_{(x)}, x \sim P_{\text{real}}} \left[LP(\mathcal{G}(\mathbf{f}), \mathcal{F}(x)) \right] \end{aligned} \tag{1}$$

where $\hat{y}^l$ and $\hat{y}^l_0$ are the extracted feature maps of x, x_0 from the pre-trained model (such as the VGG [38]), where w_l is the weight coefficient, H_l and W_l are the weight and height of the feature maps, respectively. f is the intermediate feature map from the



(a) GeneratorComponents



(b) Generator

Fig. 1. The architecture of Generator. In Fig. 1(a), the right block shows the skip-layer excitation connection, which multiplies two sizes of feature maps to implement the skip connection mechanism. The left two blocks realize the up-sampling function in FastGAN. Fig. 1(b) is the entire architecture of the generator.

discriminator, $\mathcal{G}$ represents the decoder network, and $\mathcal{F}$ represents processing, including random cropping and down-sampling on real images.

$$\begin{aligned}\mathcal{L}_D = &-\mathbb{E}_{x\sim P_{\text{real}}}\left[\min(0,-1+D(x))\right]\\ &-\mathbb{E}_{\hat{x}\sim G(z)}\left[\min(0,-1-D(\hat{x})\right]+\mathcal{L}_{\text{recons}}\\ \mathcal{L}_G = &-\mathbb{E}_{z\sim\mathcal{N}}[D(G(z))]\end{aligned} \tag{2}$$

On the basis of generator loss, this paper introduces and modifies the infoNCE loss to improve the generator training by using synthesized images.

4. Methods

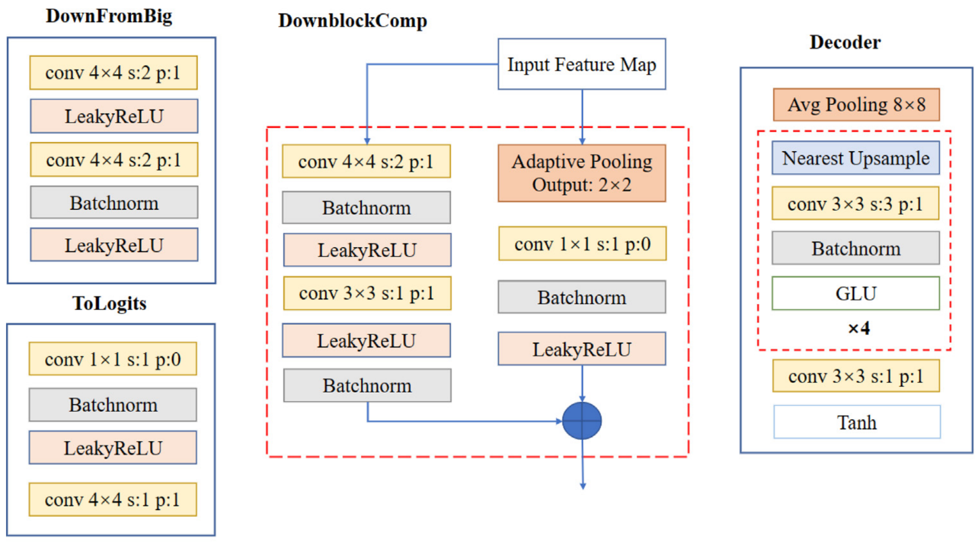
This paper proposes two technologies (i.e. a hybrid two-stage NAS and a discriminator-assisted contrastive loss function) to improve GAN architecture design and training process respectively. RL-based NAS is widely accepted by researchers who are devoted to GAN automatic design, and this section would first detail the search space and optimization strategy used in our RL-based NAS. In order to solve the problem of over-large search space and find the public optimal architecture appropriate for multi-datasets, we design a hybrid two-stage search strategy. When we get the optimal GAN, a contrastive loss function assisted by the discriminator is proposed to optimize the GAN re-training process.

4.1. Reinforcement learning based NAS

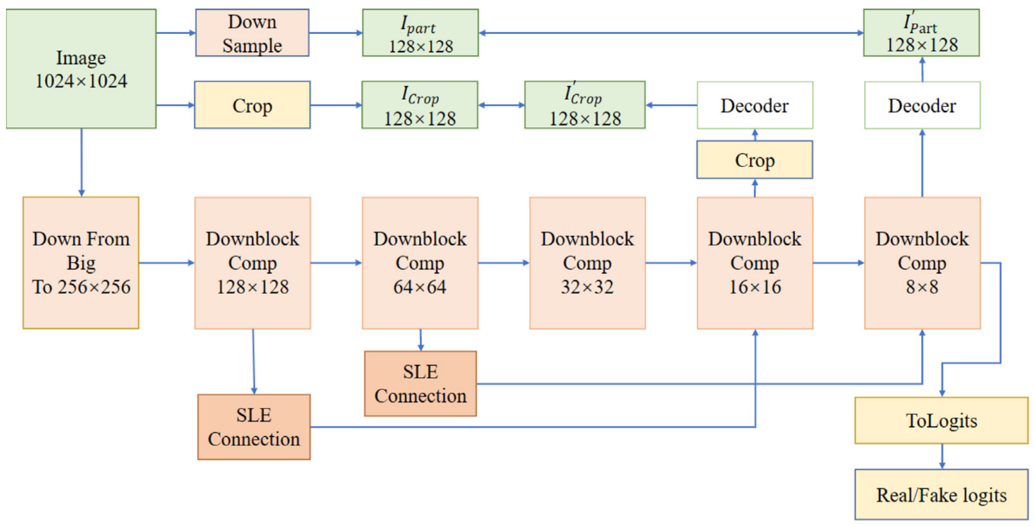
RL-based NAS includes three modules to handle search spaces, search algorithms, and performance evaluations. We initially train shared GAN models with the gradients of policy network fixed. Then, we train the controller (policy) network with the fixed shared GAN. The policy network first samples candidate architectures and evaluates their performance to achieve rewards, which guides policy updates. This study employs a lower fidelity method [39] based on short-term training to evaluate architectural performance.

4.1.1. Search space

In this study, UpblockComp and Upblock (see Fig. 1(a)) are combined into one cell, and the up-sampling and normalization



(a) DiscriminatorComponents



(b) Discriminator

Fig. 2. Architecture of the discriminator. The main body of the discriminator is composed of several down-sampling blocks. Furthermore, it also includes a decoder to reconstruct the intermediate feature maps from the discriminator. These network components could be seen in Fig. 2(a). The entire architecture could be seen in Fig. 2(b).

operations per cell are the same. In FastGAN, the SLE connections are limited to several blocks, and we expand these to the overall architecture to find the optimal connection methods. Additionally, the up-sampling and normalization operation types in FastGAN include nearest up-sampling and batch normalization (BN), respectively. Thus, we increase the number of candidate types to find more combinations and the first-stage search is used to find the optimal SLE connections over the entire architecture. Fig. 3 shows the search space of SLE connections in one cell. Specifically, SLE connections in our study are classified into short-cut SC_i and skip connection $Skip_i$ types, where i is the index of the current cell. Skip type means the SLE connections between the previous cells' intermediate output and the current cell's intermediate output, and short-cut type means the SLE connection between the previous cells' output and the current cell's output. Stage 2 searches for up-sampling and normalization operations in the cell. up-sampling operations include deconvolutions, bilinear, and nearest types. Normalization operations include BN [40], instance normalization (IN) [41], and 'none' types (i.e. without normalization). Table 1 lists the specific symbols and their explanations. Therefore, the search space of a single search cell is recorded as $2^i \cdot 3 \cdot 3 \cdot 2^{i-1}$, where $i \in \{1, \ldots, N\}$ is the index of the current search block, N is the number of upper sample blocks, and the search space of the overall architecture is $9^i \cdot 2^{i^2}$.

4.1.2. Optimization strategy

The PG optimization is an on-policy RL algorithm used in our study to update the policy network. The global objective is to maximize the expected rewards, $maxE_{m\sim\pi(m;\theta)}\mathcal{R}(m)$. First, we sample M architectures using a controller, in which all architectures share the same weights. Then, we evaluate their performance. The derived estimations are then sent back to the controller, which is updated by the PG algorithm [42] with a moving average baseline to reduce variance. Thus, $B = E_{(1:j-1)}[\mathcal{R}(m)]$. The mathematical definition is as follows:

$$\nabla_\theta \mathbb{E}_{\mathbf{m}\sim\pi(\mathbf{m};\theta)}[\overline{\mathcal{R}}(m)] = \frac{1}{M}\sum_{j=1}^{M}\left(\mathcal{R}^j - B\right)\cdot\nabla_\theta \log\pi\left(m_j;\theta\right) \tag{3}$$

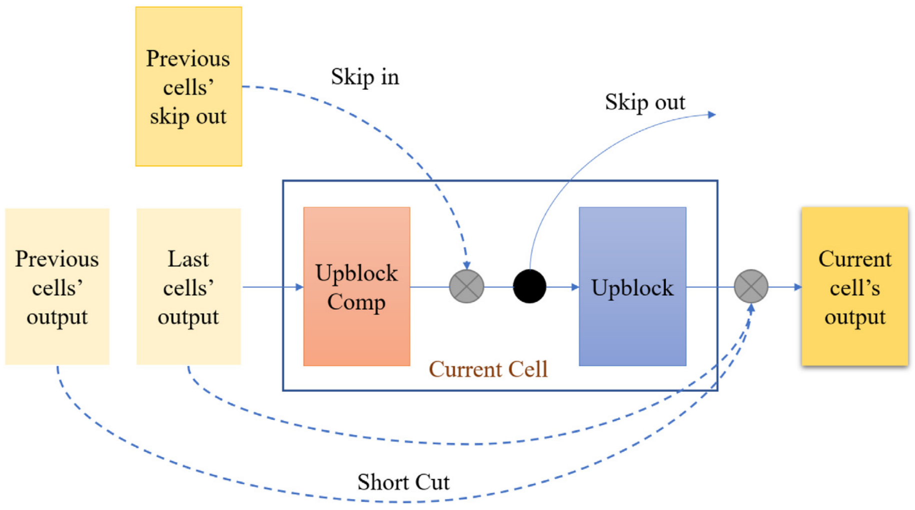


Fig. 3. Single cell is composed of UpblockComp and Upblock. The SLE connection of previous cells' skip out and the output of UpblockComp is considered as the skip out of current cell. And the shortcut is the SLE connection of the previous cells' output and the current cell's output.

Table 1
Explanations of hyperparameters.

Types	Description	Content
U	Basic types of up-sampling operation	0: Bilinear-Upsample 1: Nearest-Upsample 2: Deconvolution
N	Basic types of normalization	0: None 1: Batch-Norm 2: Instance-Norm
SC_i	Whether having SLE connections with previous cells' output	0: No 1: Yes
$Skip_i$	Whether having SLE connections with previous cells' intermediate skip out	0: No 1: Yes

where m_j represents the jth architecture, and R^j represents the reward value of the jth architecture. π denotes the policy network used to select candidate architectures. Furthermore, we reconvert the Frechet inception distance (FID) scores [43] to reward the $\mathcal{R}$ values to guide the search process. The computations of FID and reward ($\mathcal{R}$) are as follows:

$$\mathrm{FID} = \left\|\mu_r - \mu_g\right\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{\frac{1}{2}}\right) \tag{4}$$

$$\mathcal{R} = \alpha \cdot \exp^{-(FID/\tau_r)} \tag{5}$$

where μ is the feature mean of the real and generated images. Tr refers to the trace operation of the matrix, and Σ represents the covariance matrix. α and τ_r are the hyperparameters set to stabilize policy network training (i.e., 5 and 100, respectively). Furthermore, we search only for the generator architecture.

4.2. Hybrid architecture search

4.2.1. Two-stage search

As we analyzed in Section 4.1.1, the search space of the entire architecture is too large, with a near-$O^{(a^{n^2+n})}$ time complexity. Therefore, we propose a two-stage search scheme to address this problem. At the first stage, we determine the optimal macro-architecture (i.e. how the skip and short-cut mechanisms connect network cells). At the second stage, we search for the micro-operations within the architecture (i.e. normalization and up-sampling types). Concrete explanations of these operations can be found in Section 4.1.1.

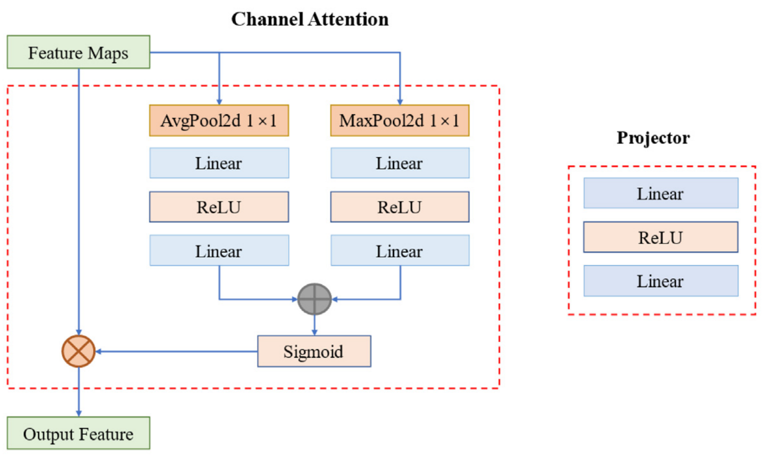
Algorithm 1 Hybrid two-stage NAS method

Input: Shared GAN, Discriminator, Controller Macro, Controller Micro, Few-Shot Datasets.
Settings: Shared epochs1 = 15, shared epochs2 = 30, change step = 15, max iter = 60, ctrl epochs = 30, switch step = 30.
Output: Top **k** architectures;
Search process:

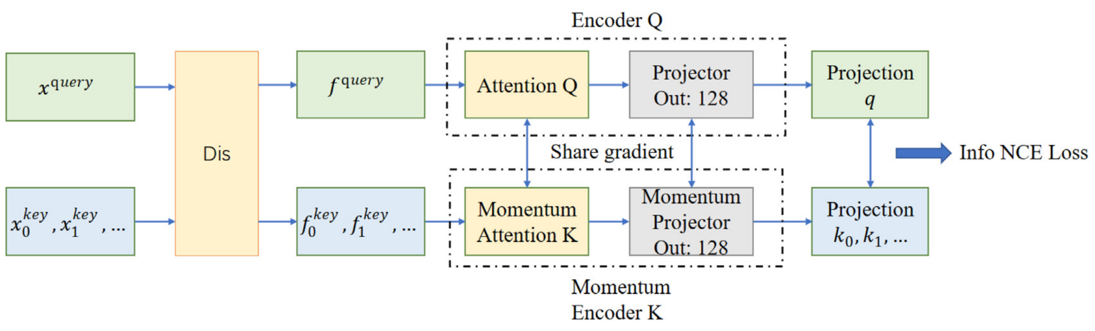
```
1: while iterations < max iter do
2:   if search_iter == switch step then
3:     Sampling M candidate skip types and compute their FID scores.
4:     Selecting top-k skip types based on FID.
5:   end if
6:   if search_iter in change step then
7:     Changing and resizing few-shot datasets.
8:     Delete Shared GAN, Discriminator.
9:     New Shared GAN, Discriminator.
10:  end if
11:  if search_iter < switch step then
12:    Training shared GAN in shared epochs1.
13:    Training controller macro during the control epochs.
14:  else
15:    Training shared GAN with top-k skip types in shared epochs2.
16:    Training controller micro in ctrl epochs.
17:  end if
18: end while
19: Get topK architectures and choose the optimal one.
```

4.2.2. Dynamic reset

We then face many different few-shot datasets. Thus, the optimal architecture must have excellent transferability. Therefore, we propose a proxy task for the dynamic reset. Specifically, we first search the architecture based on one dataset and change it to another after a few epochs with the initialized shared GAN gradients. Meanwhile, the policy network does not need to be reset. The pseudo-code of our entire search method can be viewed in Algorithm 1. Specifically, in Algorithm 1, lines 2–4 denote the ending of search stage one and getting top-k skip types for search stage two. Lines 6–10 denote the beginning of dynamic reset, which is used to change another few-shot dataset for search and re-initialize GAN. Lines 11–17 are the search body,



(a) Network components



(b) Computing process of contrastive loss

Fig. 4. The computation process of infoNCE Loss. Fig. 4(a) is the network components of the encoder, left part is the attention network used in this paper. The right block is the projector used to map the attention features. Fig. 4(b) is the progress of computing our InfoNCE Loss, i.e. the similarity loss. The encoders are responsible for encoding the feature maps from the discriminator and matching the projection query (q) to a dictionary of projection keys (k) by infoNCE loss. The dictionary $\{k_0, k_1, \ldots\}$ are defined by the ongoing changed data samples. Note that encoder K does not have gradients and is driven by a momentum update with encoder Q.

where lines 11–14 are the search stage one to find optimal skip types and lines 15–17 are the search stage two to find optimal normalization and up-sampling operations.

4.3. Generator loss function based on contrastive learning

We modify the infoNCE loss in MoCo [44] to regularize the generator network. Concretely, we add discriminator network and attention network based on the infoNCE loss in MoCo. Discriminator is used to pre-process the input data, which is more conducive to distinguishing the differences between positive examples and negative examples. The attention networks are used to focus on the fine-grained differences. The computation process of our proposed discriminator-assisted contrastive loss is shown in Fig. 4. We first employ two different data augmentations on false images to obtain images x^{query} and x^{keys}. Then, x^{query} and x^{keys} are sent to the discriminator to obtain intermediate feature maps f^{query} and f^{keys}. They are separately processed by two encoder networks, both have an attention network and projection layer to obtain the target projections (i.e. q and k). Current projection keys are en-queued and regarded as positive samples. Finally, the projection keys in the dictionary queue are viewed as negative samples to calculate the infoNCE loss between them. The mathematical equation of infoNCE loss for q and k is as follows:

$$\mathcal{L} = -\log \frac{\exp\left(q \cdot k_{+}/\tau\right)}{\sum_{i=0}^{N} \exp\left(q \cdot k_i/\tau\right)}, \tag{6}$$

where k_i is the projection key in the dictionary from previous batches, and N is the total number of candidate keys. The product of k_i and q is regarded as a negative sample. k_+ is the image of the current batch that is viewed as a positive sample. Thus, this loss function allows encoders to extract unique image features at the current batch size with the assistance of the discriminator. τ is the temperature coefficient used to control the discrimination of the generator toward negative samples, which is set to 0.2 in this paper. In fact, larger temperature coefficient means that the data distribution will be smoother, resulting in a lack of importance learning. Therefore, we need to provide larger learning gradient and after several attempts, we choose 0.2 as the suitable value. Note that the discriminator outputs two different feature maps for reconstruction: $Image^{big}$ and $Image^{small}$. Therefore, we calculate the loss functions of these maps using the following equation:

$$\mathcal{L}_{info} = \lambda_1 \cdot \mathcal{L}_{small} + \lambda_2 \cdot \mathcal{L}_{big}. \tag{7}$$

and the final loss function of the generator in this study is as follows:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim\mathcal{N}}[D(G(z))] + \lambda \cdot \mathcal{L}_{info}. \tag{8}$$

Where λ is set to 0.1 in this paper. In summary, the training process after finding the optimal GAN is explained with Algorithm 2. Specifically, training GAN within an iteration includes two stages: training discriminator (D) with generator (G) fixed and training G with D fixed. In Algorithm 2, code blocks 2–6 describe the training stage of D in an iteration. In block 3, we first acquire real images x from the source domain I_{real} and map the random sampled noise z into fake images $\hat{x}$ in the target generation domain by the fixed G. Then, we get the judgement $pred_D$ and $pred_G$ to x and $\hat{x}$, and

Algorithm 2 Contrastive GAN training

Input: Optimal generator (G), Discriminator (D), Encoder big (EB), Encoder small (ES), Few-shot datasets.
Settings: $MAX\_ITER = 100k$, Batch size = 8.
Training process:

```
1:  while $iterations < MAX\_ITER$ do
2:      Training discriminator
3:      G.eval(), D.train();
        Sampling $x \in I_{real}$;
        Sampling $\hat{x} \in G(z), z \in noise$;
        Getting $pred_D, f_1, f_2 = D(x)$;
        Getting $pred_G = D(\hat{x})$;
        Getting $I_{part}, I_{down} = Crop(x), Down(x)$;
        Get $I^{'}_{part}, I^{'}_{down} = Decoder(f_1, f_2)$;
4:      Calculating $\mathcal{L}_{recons}$ using eq. (1) with ($I_{part}, I_{down}, I^{'}_{part}, I^{'}_{down}$).
5:      Getting $\mathcal{L}_D$ using eq. (2) with ($\mathcal{L}_{recons}$, $pred_D$, and $pred_G$).
6:      Updating discriminator
7:      Training generator
8:      G.train(), D.eval();
        Sampling $\hat{x} \in G(z), z \in noise$;
        Getting $pred_g = D(\hat{x})$;
        Getting $query, key$ = Data augmentation ($\hat{x}$);
        Getting $Q_{big}, Q_{small} = D(query)$;
        Getting $K_{big}, K_{small} = D(key)$;
        Getting $q_{big}, k_{big} = EB(Q_{big}, K_{big})$;
        Getting $q_{small}, k_{small} = ES(Q_{small}, K_{small})$;
9:      Calculating $\mathcal{L}_{small}$ and $\mathcal{L}_{big}$ with ( $q_{big}, k_{big}, q_{small}, k_{small}$) using Eq. (6).
10:     Getting $\mathcal{L}_G$ using Eq. (7) and (8) with ($pred_g$, $\mathcal{L}_{small}$ and $\mathcal{L}_{big}$).
11:     Updating generator
12: end while
```

intermediate feature maps f_1, f_2 made by D. Finally, we randomly crop and down-size x to get I_{part}, I_{down}, and reconstruct f_1, f_2 by *Decoder* to get the corresponding I'_{part}, I'_{down}. Block 4 describes using I_{part}, I_{down}, I'_{part}, I'_{down} to calculate reconstruct loss. Block 5 describes using $pred_D$, $pred_G$ to calculate original D loss, which is used to measure the distance between source domain target domain, and combining these loss functions to get $\mathcal{L}_D$. In block 6, we update the D using $\mathcal{L}_D$. Blocks 7–11 denote the G training stage in an iteration. In block 3, we first generate fake images $\hat{x}$ in the target domain and get the judgment $pred_g$ of them made by fixed D. Then, we conduct data augmentations on $\hat{x}$ to get *query* and *key*. Finally, we get the contrastive representation of *query* and *key* via D and encoders. Block 9 calculates the contrastive loss function and Block 10 combines the original G loss, which is used to shorten the distance between target domain and source domain, and contrastive loss to get our $\mathcal{L}_G$ loss. Block 12 updates the G using $\mathcal{L}_G$.

5. Experiments

In this section, we elaborate on the experimental details and compare the results of AutoInfo GAN with the SOTA models. Furthermore, we conduct another ablation studies to demonstrate the effectiveness of our proposed two technologies respectively and to study whether AutoInfo GAN would have mode collapse.

5.1. Experiment details

5.1.1. Datasets

Our experiments were conducted on the few-shot datasets consistent with FastGAN (i.e. 256^2 resolution images from Animal-Face Dog and Cat [45]), 100-Shot-Obama, Panda, and Grumpy-cat [46], and 1024^2 resolution images from Oxford-flowers [47], art paintings, Pokemon, anime face, skulls, and shells, which can also be found in FastGAN[5]. We did not use Flickr-Faces-High-Quality (FFHQ) [27] as our experimental dataset, as we could not control the sampled 1000 FFHQ images to be equivalent to FastGAN.

5.1.2. Experiment settings

We used FID to measure the synthesis performance and LPIPS [37] to provide perceptual distances between images, both are lower and better. LPIPS is used to report the reconstruction quality when performing latent-space back-tracking on G, given real images. Moreover, we find that the inception score (IS) is not suitable as a metric, as it is also used to evaluate the generation diversity, which is contrary to the scenarios of the few-shot datasets. Additionally, the hyperparameter settings remained equivalent during the searching and re-training stages. Specifically, we used the Adam [48] optimizer with a learning rate of 0.0002, β_1 of 0.5, and β_2 of 0.999 to optimize the generator and discriminator. For the controller, the learning rate was $3.5e-4$, β_1 was 0.0 and β_2 was 0.9.

5.1.3. Compared models

The compared models in this paper included the unconditional SOTA model of public datasets, StyleGAN2, the fine-tuned StyleGAN2 trained on FFHQ using the Freeze-D method from [49], and the SOTA GAN model of the few-shot image synthesis task, FastGAN. StyleGAN2 is based on recent studies [50,51], including the model configurations and differentiable data augmentation.

5.2. Comparison results

The optimal generator architecture is illustrated in Fig. 5, and the search cost is two GPU days with a 11G memory RTX-2080Ti. In Fig. 5, we can see that our optimal generator slightly differs from the FastGAN. Specifically, the optimal generator preferred IN and none normalization. Besides that, black lines represent the skip types between cells 1 and 3, cells 2 and 3, and the orange line represents the shortcut connection between cell 1 and cell 3. The entire connection ways are similar to FastGAN. A concrete analysis of the searched architectures is found in Sections 5.3.1 and 5.3.3. To compare the practical training results, we generated 5000 images, which is consistent with FastGAN, to calculate the FID scores. FID is an index used to evaluate the distance between two distributions (a lower value indicates a better performance), which is also the evaluation index in FastGAN. The calculation of it could be seen at Eq. (4). We retrained the optimal GAN with 100k iterations and recorded the FID scores for every 10k iterations. The lowest FID score was selected as the best performer. The average relative error of the final best results was less than 5.

The results are shown in Tables 2 and 3, where the bold numbers represent the lowest FID scores among different datasets. Note that, these results for comparison are all recorded in the FastGAN. Due to the instability of instance normalization, which has been illustrated in Section 5.3.3, we only exhibit the FID results of AutoInfo GAN with searched skip types and info loss function. We can see that our model achieves the best results on most datasets. The listed FID results of other GANs are publicly available in FastGAN [16]. Surprisingly, we find that it did not no matter that training our model required six batches of 1024 × 1024 resolution images because the final results were still good. Furthermore, some 1024 × 1024 resolution images can achieve better performance if resized and trained with 512 × 512 resolutions over eight batches.

[5] https://github.com/odegeasslbc/FastGAN-pytorch

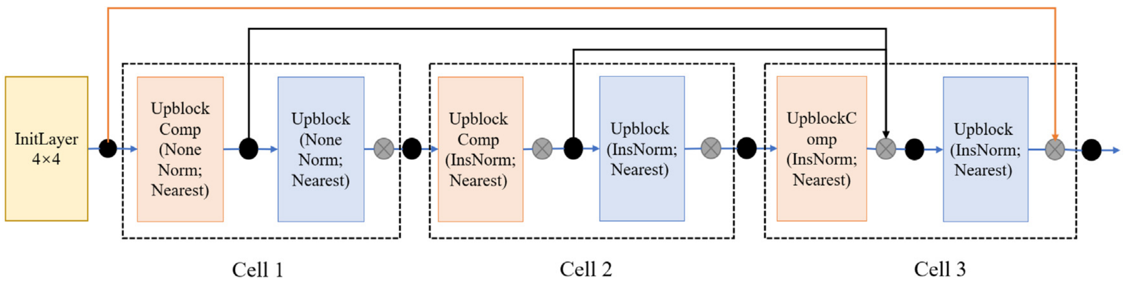

InitLayer 4×4
Upblock Comp (None Norm; Nearest)
Upblock (None Norm; Nearest)
Upblock Comp (InsNorm; Nearest)
Upblock (InsNorm; Nearest)
UpblockC omp (InsNorm; Nearest)
Upblock (InsNorm; Nearest)
Cell 1
Cell 2
Cell 3

Fig. 5. Our selected optimal generator architecture. The optimal SLE connections are: short-cut types (orange line) between cells 1 and 3 and skip types (black lines) between cells 1 and 3, cells 2 and 3. The first cell has no normalization and the other cells have IN. All up-sampling operations are nearest.

Table 2
FID comparison with 256^2 resolution datasets.

Image number	Animal face - Dog 389	Animal face - Cat 160	Obama 100	Panda 100	Grumpy-cat 100
StyleGAN2	58.85	42.44	46.87	12.06	27.08
StyleGAN2 finetune	61.03	46.07	35.75	14.5	29.34
FastGAN (baseline)	50.66	35.11	41.05	10.03	26.65
AutoInfo GAN (skip + info)	**49.72**	**33.33**	**35.54**	**9.36**	**24.83**

Table 3
FID comparison with 1024^2 resolution datasets.

Image number	Art paintings 1000	Flower 1000	Pokemon 800	Anime face 120	Skull 100	Shell 60
StyleGAN2	74.56	45.23	190.23	152.73	127.98	241.37
StyleGAN2 finetune	N/A	36.72	60.12	61.23	107.68	220.45
FastGAN (baseline)	45.08	25.66	57.19	59.38	130.05	155.47
AutoInfo GAN (skip + info)	**42.93**	**24.05**	**46.21**	**52.62**	**106.10**	**143.74**

Table 4
Ablation study with 256^2 resolution datasets.

	Animal face - Dog	Animal face - Cat	Obama	Panda	Grumpy-cat
FastGAN	50.66	35.11	41.05	10.03	26.65
AutoInfo GAN (skip only)	51.32	**33.59**	**35.58**	9.80	25.65
AutoInfo GAN (norm only)	**47.95**	34.24	36.60	**9.10**	25.66
AutoInfo GAN (info only)	49.97	34.69	36.24	9.41	**25.01**

In Figs. 6 and 7, We select 4 (256 × 256) datasets and 4 (1024 × 1024) datasets respectively. Within each dataset, we randomly sampled 9 real images from the source domain and randomly generated 9 fake images in the target generation domain. Considering the generated 256 × 256 images in Fig. 6, it can be observed that although we can distinguish the categories of these images, some of them still have a mode collapse situation, especially in the dataset of 100-shot Obama, where some images have serious deformation. In Fig. 7, we exhibit 1024 × 1024 images generated by FastGAN and AutoInfoGAN, and compare them with real images. In fact, high resolution image synthesis is of great difficulty, it is difficult for both FastGAN and AutoInfoGAN to generate clear and complete images. For example, we cannot even distinguish the image categories of the synthesized pokemon and shells. However, based on the pokemon dataset, we can find that the objection deformation in the FastGAN is more serious than that in the AutoInfoGAN. Considering the Skulls dataset, we can also find that some skulls in the FastGAN have a more serious mode collapse. Therefore, it is obvious that AutoInfoGAN can achieve better synthesis performance than FastGAN, while it still needs a lot of work to further develop high-fidelity few-shot image synthesis.

5.3. Ablation study

5.3.1. Architecture search

In this study, we independently tested the efficacy of our search combinations of skip types and normalization operations. These techniques were separately trained from scratch using the original FastGAN. We conducted ablation experiments on 256 × 256 resolution datasets, and the FID results are shown in Table 4. The first row lists the results in original FastGAN and the 2nd - 3rd rows list the results of AutoInfo GAN trained only with the searched skip types and normalization operations respectively. Bold numbers represent the best FID results in each dataset. We found that our architecture connections achieve better performance than the baseline model on most datasets. Surprisingly, our algorithm prefers IN rather than BN, and in practice, IN tends to achieve the best performance. We think it may be the case that BN changes the original well-learned distribution, which then influences the 1-Lipschitz continuity of spectral normalization [52] in the generator. Hence, the fidelity of the generated images is affected. In the next subsection, 5.3.3, we explain that the IN-based generator seems more prone to mode collapse. This may be related to the loss of BN that would otherwise normalize the distribution of every layer's output. While considering the up-sampling operation, the nearest is also the optimal choice as FastGAN.

5.3.2. Contrastive loss function

We independently tested the efficacy of our proposed generation function, noting significant performance improvements, as shown in Table 4. The 4th row shows the comparison results. As can be seen, merely using our proposed info loss function can also achieve a great or even the best performance improvement on most few-shot datasets.

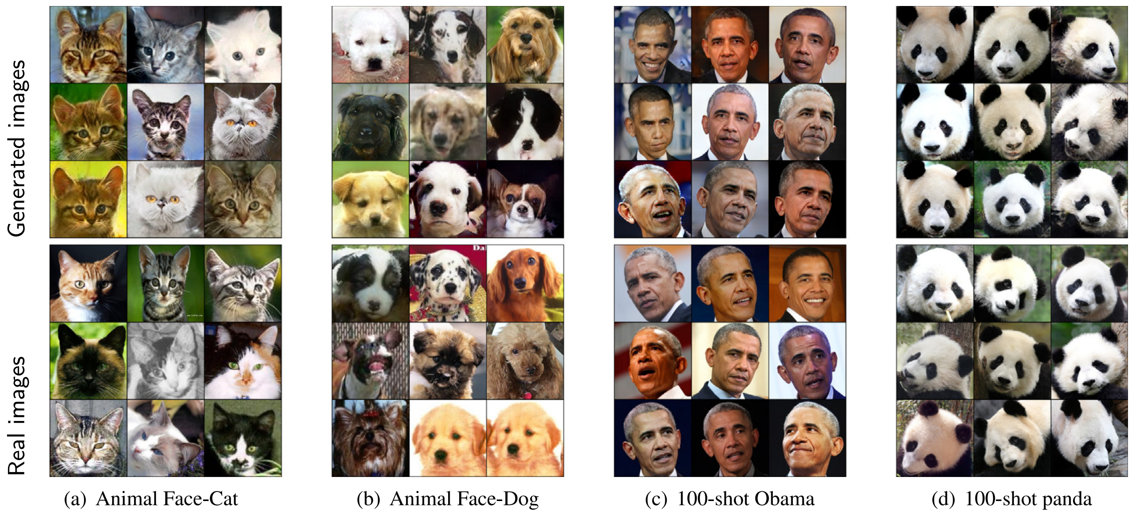


Fig. 6. This figure lists the comparison of some generated images and their original images with the resolution of 256 × 256. The above line shows the generated images by AutoInfoGAN and the below shows the original images.

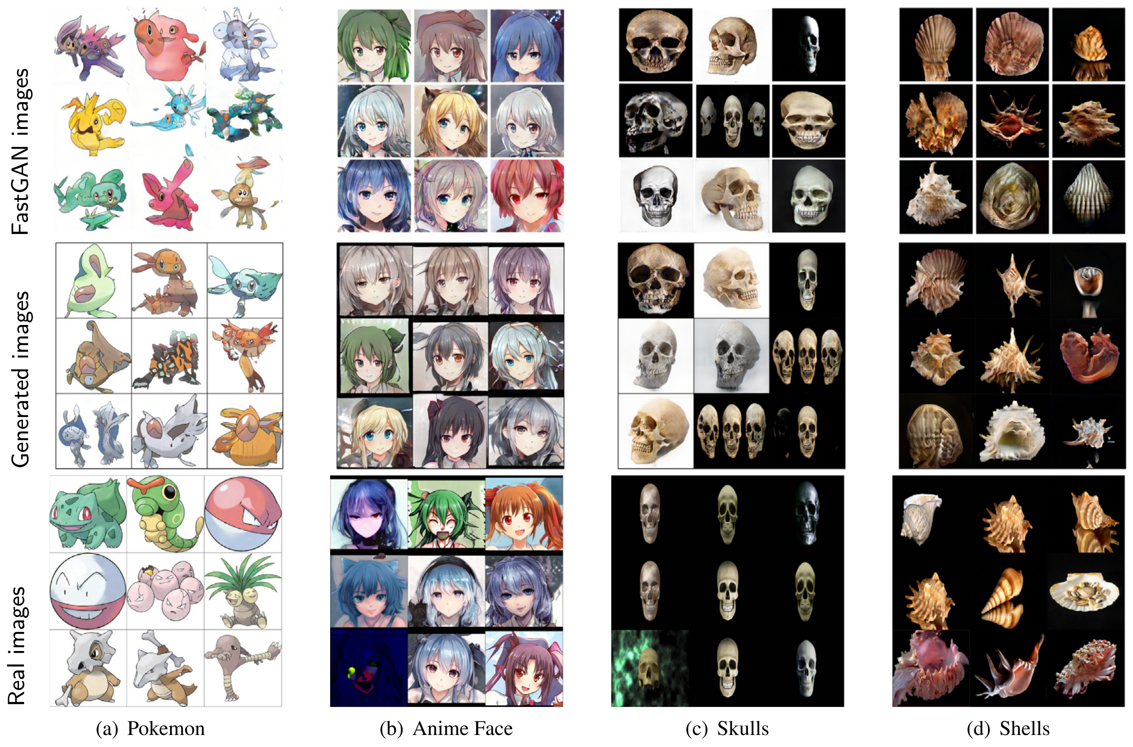


Fig. 7. This figure lists the comparison of FastGAN generated images, AutoInfoGAN generated images and their original images with the resolution of 1024 × 1024. The first line shows the images generated FastGAN, second line shows the images generated AutoInfoGAN, and the last line shows the original images.

5.3.3. *Mode collapse test with back-tracking*

Based on FastGAN, we tested the back-tracking ability of our model. Generally, a well-trained generator can convert a real image back to a vector in the latent space and change the content of images by altering the back-tracked vector. Specifically, we updated the latent vectors and inverted them into synthesis images using a well-trained generator after 1k iterations. We computed the average mean LPIPS between the inversions and real images at every 100 iterations to verify whether the generator experienced mode collapse. The LPIPS results are listed in Table 5, where the 1st row lists the results of FastGAN and the 2nd to 4th rows list the results of AutoInfo GAN trained only with our searched normalization operations, skip types, and info loss function respectively. The bold numbers represent the best LPIPS of each dataset. From Table 5, we can see that although IN-based GAN does not achieve good back-tracking performance, our searched skip types and infoNCE loss can still provide excellent results.

6. Discussion

In previous sections, we compared our AutoInfo GAN with existing SOTA models to gain an understanding of the detailed performance improvements. The AutoInfo GAN model is very simple while being sufficient for our purposes. However, in practice, it may be expected that AutoInfo GAN should achieve higher searching and training efficacy in terms of model optimization. In

Table 5
LPIPS of back-tracking with generator.

Datasets	Animal-Dog	Animal-Cat	Obama	Panda	Grumpy-Cat
			256		
FastGAN	1.918	1.821	–	–	–
AutoInfo GAN (norm)	2.434	2.323	2.152	1.633	2.065
AutoInfo GAN (skip)	1.905	1.814	**1.196**	1.536	1.571
AutoInfo GAN (info)	**1.891**	**1.809**	1.285	**1.528**	**1.532**

Table 6
Hyper-parameters of controller.

Search epoch	5	10	15	20	25	30	35	40	45	50	55	60
Search space			512						729			
Capacity						150						
0.5, 0.999	142	110	82	55	26	9	143	117	94	65	39	11
0.0, 0.9	143	122	91	64	35	14	145	124	102	79	53	19

Table 7
Lower batch size comparison.

Models	Art	Pokemon	Anime	Shell	Skull	GPU
			1024			Memory
AutoInfo 8 Batch	42.93	46.21	52.62	143.74	106.10	12G
AutoInfo 6 Batch	44.86	51.26	55.26	153.33	123.47	9G
FastGAN 8 Batch	45.08	57.19	59.38	155.47	130.05	9G

this section, we describe additional experimental results to provide a more comprehensive understanding of the characteristics of AutoInfo GAN to facilitate its adaptation for future research and tasks.

6.1. How do the hyperparameters affect the search efficacy of AutoInfo GAN?

We set β_1 and β_2 in the controller optimizer to 0.0 and 0.9. Setting this hyperparameter lower would help the controller explore more possible architectures and would be useful for finding the global optimal architecture. This subsection discusses the effect of hyperparameters on search convergence and overfitting. Practically, at each controller training stage, we sampled 30 candidate GANs and evaluated their performance. To reflect the search strategy of the controller, we counted the number of different sampled architectures every five search epochs. That is, at each statistical interval, the number of candidate architectures with different types did not exceed 150. The concrete change process is presented in Table 6. We provide the original search space of two search stages. As can be seen, lower settings help the controller explore more candidate architectures at first and shorten the search range. Besides that, when choosing the final 10 candidate architectures, the lower hyperparameter has seven different architectures and the higher one has only four. This phenomenon illustrates that lower architectures may help uncover new possibilities and alleviate search overfitting.

6.2. How do the hyperparameters affect the generation efficacy of AutoInfo GAN?

As discussed, different hyperparameter settings (e.g. β_1 and β_2) in the generator optimizer will lead to differences in the final performance. Furthermore, setting β_1 and β_2 lower can stabilize the training process and acquire better performance, although the corresponding cost is training time. Generally, if we can quickly acquire a reasonably trained GAN, we can set β_1 and β_2 as the default values of 0.5 and 0.999, respectively. However, if we want a better-trained GAN, we can set β_1 and β_2 0.2 and 0.9, respectively. A comparison of the FID-changing processes with different hyperparameters is presented in Fig. 8. We trained AutoInfo GAN with two aforementioned hyperparameters (i.e. 0.5, 0.999, and 0.2, 0.9) on four 256×256 datasets and reported the FID value every 10 training epochs. In Fig. 8, we find that lower hyperparameters can achieve better performance (i.e. lower FID) at last, although they usually perform poorly at the beginning. Furthermore, considering the FID-changing process on the panda dataset, it is notable that lower hyper-parameters can also stabilize the training process and avoid mode collapse. Additionally, we found that setting a lower batch size does not significantly influence the final results. We conducted some experiments on several 1024×1024 resolution datasets, as shown in Table 7, and observed that setting a batch size of eight enabled AutoInfo GAN to perform similarly with FastGAN. Larger batch sizes will result in more improvements at the cost of computing power.

7. Implications

This study has important implications for practitioners and researchers.

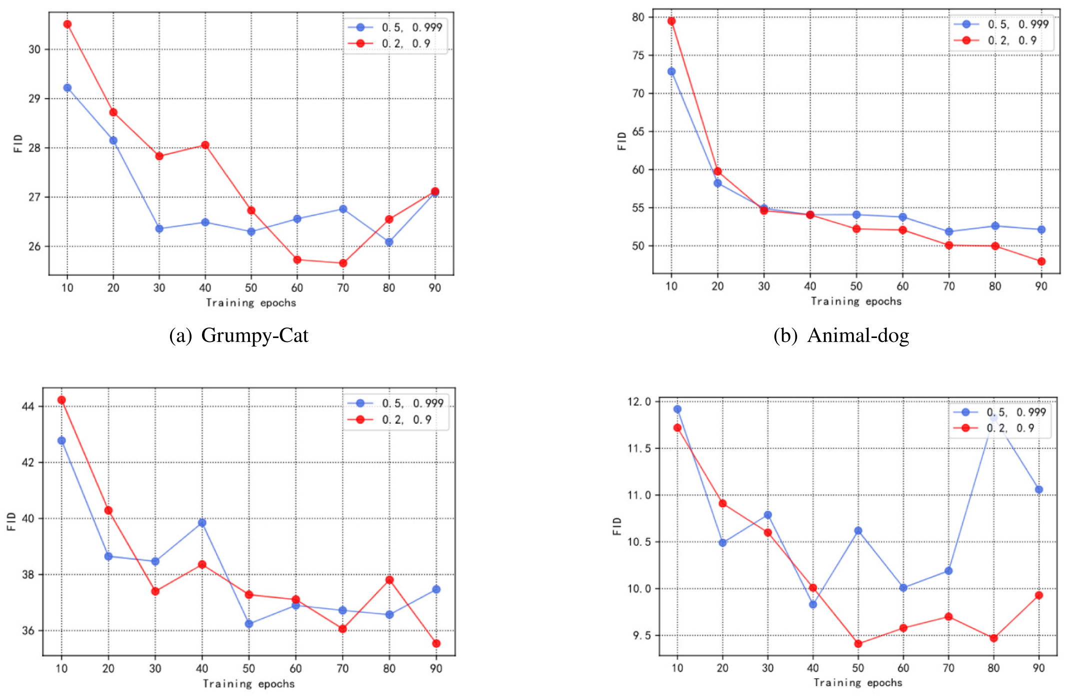
7.1. For practitioners

In practice, AutoInfo GAN can be easily used by practitioners for high-quality image generation when there are few data samples. Therefore, we provide useful guidance for further improving and applying AutoInfo GAN in practice. Practitioners can use AutoInfo GAN very flexibly by altering the default settings while stabilizing the training process via hyperparameter modifications reducing GPU memory use reduction by changing the batch size without lowering performance results. Meanwhile, we analyzed the limitations of our model so that practitioners can be aware of the deficiencies of IN and take actions accordingly.

7.2. For researchers

With AutoInfo GAN, we provide a simple yet strong baseline for few-shot high-fidelity image synthesis. In the future, if a new corresponding approach is proposed, it should be compared with AutoInfo GAN to demonstrate its practical value. This will help the community to develop more effective approaches. The detailed implications are as follows.

1. **Our work highlights the importance of fine-tuning architectures.** In the ablation study of Section 5.3.1, we conducted experiments to validate the use of NAS to fine-tune our GAN architecture to achieve better performance results. Notably, our designed search space and corresponding algorithms are still in their early stages of refinement. Designing more efficient NAS algorithms alongside more exploratory research will be crucial to developing better GANs.



(a) Grumpy-Cat (b) Animal-dog

(c) Obama (d) Panda

Fig. 8. The change process of FID during training with different hyper-parameters using AutoInfo GAN. We conduct experiments on four different datasets to observe the results.

2. **Our work provides useful suggestions for improving generator optimization.** In this study, to solve the problem of insufficient training samples and optimize generator learning, we introduced a new contrastive loss function to optimize learning the instance differences and borrowed a discriminator to project the generated images. These methods are demonstrably effective according to our ablation study, discussed in Section 5.3.2. Therefore, we strongly recommend that future studies consider the discriminator when using contrastive learning to perform few-shot high-fidelity image synthesis.

8. Threats to validity

We considered the most important threats to the internal and external construct validity of our study.

8.1. Internal validity

Internal validity concerns the causal relationship between the treatment and the observed results and how confident we are that the treatment caused the result. The first threat to internal validity is the selection of extant data-limited generation approaches. Because this study was devoted to the current research progress of data-limited image synthesis, it is important to select the most representative works. To this end, although there were few generation approaches to choose from, this study considered all SOTA models from top international conferences and journals. To the best of our knowledge, these approaches are the most widely cited works in the field of few-shot dataset synthesis and continue to attract much attention. Therefore, this threat was minimized.

The second threat to internal validity is the selection of few-shot datasets. Our FastGAN backbone was also tested with the Flickr-Faces-HQ (FFHQ) high-resolution dataset (1024×1024) of human face images. However, it was necessary to sample 1k images to satisfy the limitations of data scenarios, and we could not ensure the same samples. Therefore, we only calculated the FID results of AutoInfo GAN for other existing public few-shot datasets. In the future, we should also consider the FFHQ dataset. However, this threat is presently minimized.

8.2. External validity

External validity is the degree to which the research results can be generalized to the population under study and other research settings.

The major threat to external validity is that our proposed method may not be directly generalizable to other data-limited GAN models. We proposed a hybrid two-stage NAS method for generator- and discriminator-assisted contrastive loss functions to improve synthesis quality. We chose FastGAN as our baseline, which achieved SOTA results on the given few-shot datasets. However, FastGAN has a supervised discriminator that is designed to avoid overfitting. It is necessary to consider the discriminator design to avoid overfitting when we employ GAN models in the future. However, AutoInfo GAN can be directly employed by other practitioners and researchers to complete few-shot high-fidelity image synthesis and achieve SOTA results.

9. Conclusion and future work

In this paper, we introduced two methods to improve generator learning. The synthesis fidelity of our recommendation was assessed on fewer than 1000 high-resolution images. On 11 public datasets, we showed that the auto-searched generator and contrastive optimization can boost the synthesis performance of GANs. We expect that this work will provide new perspectives for future research. Thus, we now list several areas that should be improved:

1. Our search space should be further expanded, so that we can find more useful operations, convolution layers, or layer placements to further improve GAN performance and avoid mode collapse.
2. FID values vary greatly under different datasets, which may directly influence the training of the policy network. Ways to balance the FID gap among different datasets should be examined.
3. Because our RL-based PG algorithm does not outperform the SOTA models, we should further introduce off-policy RL search methods. NAS also has different PG algorithms that we should apply to few-shot learning scenarios to evaluate the search cost savings.
4. From the GAN perceptive, high-resolution demand requires GANs with larger channels and feature maps, which will generate more high-dimensional noise. Attention mechanisms and other methods of noise reduction should be studied. Some methods [53,54] that can be used to improve the quality of image generation in GAN can also be referenced.

CRediT authorship contribution statement

Jiachen Shi: Writing – original draft, Project administration, Methodology. **Wenzhen Liu:** Conceptualization. **Guoqiang Zhou:** Supervision. **Yuming Zhou:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

The authors are grateful to the College of Computer Science, Nanjing University of Posts and Telecommunications, for kindly providing the required computational resources. This work was financially supported by the National Program on Key Basic Research Project (2020YFA0713600), the National Natural Science Foundation of China (62272214). This work is supported by Jiangsu Provincial Key Research & Development Plan, China under Grants No. BE2022065-5.

References

[1] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A.C. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014, pp. 2672–2680.
[2] L. Clouâtre, M. Demers, FIGR: few-shot image generation with reptile, 2019, CoRR abs/1901.02199 arXiv:1901.02199.
[3] Y. Hong, L. Niu, J. Zhang, L. Zhang, Matchinggan: Matching-based few-shot image generation, in: IEEE International Conference on Multimedia and Expo, ICME 2020, London, UK, July (2020) 6-10, IEEE, 2020, pp. 1–6.
[4] Y. Hong, L. Niu, J. Zhang, W. Zhao, C. Fu, L. Zhang, F2GAN: fusing-and-filling GAN for few-shot image generation, in: C.W. Chen, R. Cucchiara, X. Hua, G. Qi, E. Ricci, Z. Zhang, R. Zimmermann (Eds.), MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October (2020) 12-16, ACM, 2020, pp. 2535–2543.
[5] Z. Gu, W. Li, J. Huo, L. Wang, Y. Gao, Lofgan: Fusing local representations for few-shot image generation, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October (2021) 10-17, IEEE, 2021, pp. 8443–8451.
[6] G. Ding, X. Han, S. Wang, S. Wu, X. Jin, D. Tu, Q. Huang, Attribute group editing for reliable few-shot image generation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, la, USA, June (2022) 18-24, IEEE, 2022, pp. 11184–11193.
[7] Y. Hong, L. Niu, J. Zhang, L. Zhang, Few-shot image generation using discrete content representation, in: J. Magalhães, A.D. Bimbo, S. Satoh, N. Sebe, X. Alameda-Pineda, Q. Jin, V. Oria, L. Toni (Eds.), MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October (2022) 10-14, ACM, 2022, pp. 2796–2804.
[8] A. Noguchi, T. Harada, Image generation from small datasets via batch statistics adaptation, in: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 2750–2758.
[9] M. Zhao, Y. Cong, L. Carin, On leveraging pretrained gans for generation with limited data, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 2020, Virtual Event, 119 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 11340–11351.
[10] Y. Wang, A. Gonzalez-Garcia, D. Berga, L. Herranz, F.S. Khan, J. van de Weijer, Minegan: Effective knowledge transfer from gans to target domains with few images, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June (2020) 13-19, Computer Vision Foundation / IEEE, 2020, pp. 9329–9338.
[11] Y. Li, R. Zhang, J. Lu, E. Shechtman, Few-shot image generation with elastic weight consolidation, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December (2020) 6-12, Virtual, 2020, pp. 15885—-15896.
[12] U. Ojha, Y. Li, J. Lu, A.A. Efros, Y.J. Lee, E. Shechtman, R. Zhang, Few-shot image generation via cross-domain correspondence, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June (2021) 19-25, Computer Vision Foundation / IEEE, 2021, pp. 10743–10752.
[13] J. Xiao, L. Li, C. Wang, Z. Zha, Q. Huang, Few shot generative model adaption via relaxed spatial structural alignment, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, la, USA, June (2022) 18-24, IEEE, 2022, pp. 11194–11203.
[14] Y. Zhao, H. Ding, H. Huang, N. Cheung, A closer look at few-shot image generation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, la, USA, June (2022) 18-24, IEEE, 2022, pp. 9130–9140.
[15] S. Zhao, Z. Liu, J. Lin, J. Zhu, S. Han, Differentiable augmentation for data-efficient GAN training, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December (2020) 6-12, Virtual, 2020, pp. 7559–7570.
[16] B. Liu, Y. Zhu, K. Song, A. Elgammal, Towards faster and stabilized GAN training for high-fidelity few-shot image synthesis, in: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May (2021) 3-7, OpenReview.Net, 2021, pp. 15885—-15896.
[17] J. Jeong, J. Shin, Training gans with stronger augmentations via contrastive discriminator, in: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May (2021) 3-7, OpenReview.Net, 2021.
[18] Y. Zhao, C. Du, M. Abdollahzadeh, T. Pang, M. Lin, S. Yan, N. Cheung, Exploring incompatible knowledge transfer in few-shot image generation, 2023, CoRR abs/2304.07574.
[19] W. Li, J. Chen, J. Cao, C. Ma, J. Wang, X. Cui, P. Chen, EID-GAN: generative adversarial nets for extremely imbalanced data augmentation, IEEE Trans. Ind. Informatics 19 (3) (2023) 3208–3218.
[20] M. Yang, Z. Wang, Z. Chi, W. Du, Protogan: Towards high diversity and fidelity image synthesis under limited data, Inform. Sci. 632 (2023) 698–714.
[21] H. Tseng, L. Jiang, C. Liu, M. Yang, W. Yang, Regularizing generative adversarial networks under limited data, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021 virtual June (2021) 19-25, Computer Vision Foundation / IEEE, 2021, pp. 7921–7931.
[22] V. Sushko, J. Gall, A. Khoreva, One-shot GAN: learning to generate samples from single images and videos, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021 virtual June (2021) 19-25, Computer Vision Foundation / IEEE, 2021, pp. 2596–2600.
[23] T.R. Shaham, T. Dekel, T. Michaeli, Singan: Learning a generative model from a single natural image, in: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 4569–4579.
[24] H. Zhang, I.J. Goodfellow, D.N. Metaxas, A. Odena, Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 2019, long beach, california, USA, 97 of proceedings of machine learning research, in: K. Chaudhuri, R. Salakhutdinov (Eds.), PMLR, 2019, pp. 7354–7363.
[25] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May (2019) 6-9, OpenReview.net, 2019.

[26] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

[27] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, IEEE Trans. Pattern Anal. Mach. Intell. 43 (12) (2021) 4217–4228.

[28] X. Gong, S. Chang, Y. Jiang, Z. Wang, Autogan: Neural architecture search for generative adversarial networks, in: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 3223–3233.

[29] Y. Tian, Q. Wang, Z. Huang, W. Li, D. Dai, M. Yang, J. Wang, O. Fink, Off-policy reinforcement learning for efficient and effective GAN architecture search, in: A. Vedaldi, H. Bischof, T. Brox, J. Frahm (Eds.), Computer Vision - ECCV 2020-16th European Conference, Glasgow, UK, August (2020) 23-28, Proceedings, Part VII, 12352 of Lecture Notes in Computer Science, Springer, 2020, pp. 175–192.

[30] C. Gao, Y. Chen, S. Liu, Z. Tan, S. Yan, Adversarialnas: adversarial neural architecture search for gans, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June (2020) 13-19, Computer Vision Foundation / IEEE, 2020, pp. 5679–5688.

[31] Y. Fan, G. Zhou, W. Zhang, S. Bao, J. Shen, Determining learning direction via multi-controller model for stably searching generative adversarial networks, Neurocomputing 464 (2021) 37–47.

[32] Y. Fan, X. Tang, G. Zhou, J. Shen, Efficientautogan: Predicting the rewards in reinforcement-based neural architecture search for generative adversarial networks, IEEE Trans. Cogn. Dev. Syst. 14 (1) (2022) 234–245.

[33] J. Shi, G. Zhou, S. Bao, J. Shen, Multi-selfgan: a self-guiding neural architecture search method for generative adversarial networks with multi-controllers, IEEE Trans. Cogn. Dev. Syst. (2022).

[34] D. Pang, X. Le, X. Guan, RL-DARTS: differentiable neural architecture search via reinforcement-learning-based meta-optimizer, Knowl. Based Syst. 234 (2021) 107585.

[35] R. Shang, S. Zhu, J. Ren, H. Liu, L. Jiao, Evolutionary neural architecture search based on evaluation correction and functional units, Knowl. Based Syst. 251 (2022) 109206.

[36] Z. Fan, G. Hu, X. Sun, G. Wang, J. Dong, C. Su, Self-attention neural architecture search for semantic image segmentation, Knowl. Based Syst. 239 (2022) 107968.

[37] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June (2018) 18-22, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 586–595.

[38] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May (2015) 7-9, Conference Track Proceedings, 2015.

[39] F. Runge, D. Stoll, S. Falkner, F. Hutter, Learning to design RNA, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May (2019) 6-9, OpenReview.net, 2019.

[40] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: F.R. Bach, D.M. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 2015, 37 of JMLR Workshop and Conference Proceedings, JMLR.org, 2015, pp. 448–456.

[41] D. Ulyanov, A. Vedaldi, V.S. Lempitsky, Instance normalization: the missing ingredient for fast stylization, 2016, CoRR abs/1607.08022 arXiv:1607.08022.

[42] W. Chen, K.K.L. Wong, S. Long, Z. Sun, Relative entropy of correct proximal policy optimization algorithms with modified penalty factor in complex environment, Entropy 24 (4) (2022) 440.

[43] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: D. Precup, Y.W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 2017, 70 of Proceedings of Machine Learning Research, PMLR, 2017, pp. 214–223.

[44] K. He, H. Fan, Y. Wu, S. Xie, R.B. Girshick, Momentum contrast for unsupervised visual representation learning, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June (2020) 13-19, Computer Vision Foundation / IEEE, 2020, pp. 9726–9735.

[45] Z. Si, S. Zhu, Learning hybrid image templates (HIT) by information projection, IEEE Trans. Pattern Anal. Mach. Intell. 34 (7) (2012) 1354–1367.

[46] S. Zhao, Z. Liu, J. Lin, J. Zhu, S. Han, Differentiable augmentation for data-efficient GAN training, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December (2020) 6-12, virtual, 2020.

[47] M. Nilsback, A. Zisserman, A visual vocabulary for flower classification, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 2006, New York, NY, USA, IEEE Computer Society, 2006, pp. 1447–1454.

[48] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May (2015) 7-9, Conference Track Proceedings, 2015.

[49] S. Mo, M. Cho, J. Shin,

[50] S. Zhao, Z. Liu, J. Lin, J. Zhu, S. Han, Differentiable augmentation for data-efficient GAN training, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December (2020) 6-12, virtual, 2020.

[51] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, T. Aila, Training generative adversarial networks with limited data, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December (2020) 6-12, virtual, 2020.

[52] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018.

[53] W. Li, Z. Liang, P. Ma, R. Wang, X. Cui, P. Chen, Hausdorff GAN: improving GAN generation quality with hausdorff metric, IEEE Trans. Cybern. 52 (10) (2022) 10407–10419.

[54] W. Li, J. Chen, Z. Wang, Z. Shen, C. Ma, X. Cui, Ifl-gan: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation, IEEE Trans. Neural Netw. Learn. Syst. (2022).