This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

AdversarialNAS: Adversarial Neural Architecture Search for GANs

Chen Gao[1,2,4], Yunpeng Chen[4], Si Liu[3]*, Zhenxiong Tan[5], Shuicheng Yan[4]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]School of Computer Science and Engineering, Beihang University
[4]Yitu Technology [5]Beijing Forestry University
gaochen@iie.ac.cn, liusi@buaa.edu.cn, yunpeng.chen@yitu-inc.com

Abstract

Neural Architecture Search (NAS) that aims to automate the procedure of architecture design has achieved promising results in many computer vision fields. In this paper, we propose an AdversarialNAS method specially tailored for Generative Adversarial Networks (GANs) to search for a superior generative model on the task of unconditional image generation. The AdversarialNAS is the first method that can search the architectures of generator and discriminator simultaneously in a differentiable manner. During searching, the designed adversarial search algorithm does not need to comput any extra metric to evaluate the performance of the searched architecture, and the search paradigm considers the relevance between the two network architectures and improves their mutual balance. Therefore, AdversarialNAS is very efficient and only takes 1 GPU day to search for a superior generative model in the proposed large search space (10^{38}). Experiments demonstrate the effectiveness and superiority of our method. The discovered generative model sets a new state-of-the-art FID score of 10.87 and highly competitive Inception Score of 8.74 on CIFAR-10. Its transferability is also proven by setting new state-of-the-art FID score of 26.98 and Inception score of 9.63 on STL-10. Code is at: `https://github.com/chengaopro/AdversarialNAS`.

1. Introduction

Image generation is a fundamental task in the field of computer vision. Recently, GANs [10] have attracted much attention due to their remarkable performance for generating realistic images. Previous architectures of GANs are designed by human experts with laborious trial-and-error testings (Fig. 1 a)) and the instability issue in GAN training extremely increases the difficulty of architecture design. Therefore, the architecture of the generative model in GAN literature has very few types and can be simply divided into two styles: DCGANs-based [32] and ResNet-based [14]. On the other hand, the benefits of specially designing the network architecture have been proven through lost of discriminative networks, such as ResNet [14], DenseNet [17], MobileNet [34], ShuffleNet [46], EfficientNet [36] and HRNet [35]. Therefore, the research about the backbone architecture of GANs needs more attention to further improve the performance of the generative model.

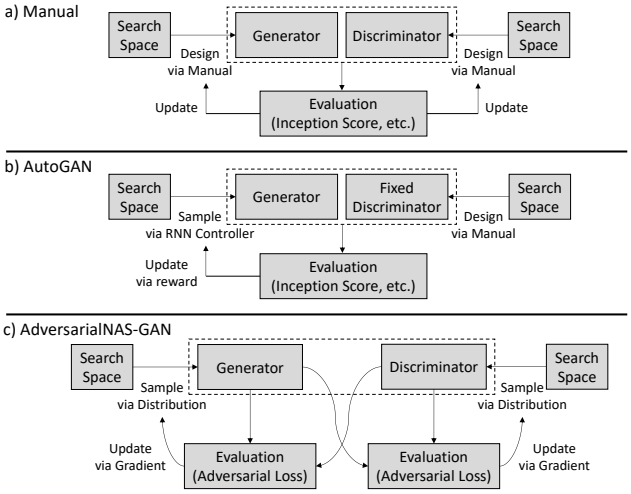

Figure 1. Comparisons of different ways of designing GAN architectures. a) The previous hand-crafted GAN architectures depend on the experience of human experts. b) AutoGAN [9] adopts IS or FID as reward to update the architecture controller via reinforcement learning. c) The proposed AdversarialNAS searches architecture in a differentiable way with an adversarial search mechanism, which achieves better performance with higher efficiency.

Recently, Neural Architecture Search (NAS) has been studied heatedly owing to its ability to automatically discover the optimal network architecture, which significantly reduces human labor. However, on generation tasks, specifically GANs-based generation, only AutoGAN [9] and

*Corresponding author

AGAN [38] have explored the application of NAS.

To design a NAS algorithm specially tailored for GANs on the unconditional image generation task, there are two main challenges. First, it is expected to utilize an efficient supervision signal to guide the search process in this unsupervised task. However, the existing works [9, 38] both adopt the Inception Score (IS) [33] or FID to evaluate the architecture performance and take IS or FID as a reward to update the architecture controller via reinforcement learning strategy. Obtaining IS or FID needs to generate hundreds of images and use the statistics produced by an Inception network to calculate the final score. Thus it is extremely time-consuming, e.g. 200 GPUs over 6 days [38]. Second, the relevance and balance between generator and discriminator need to be considered during searching since the training process of GANs is a unique competition. However, AutoGAN search for a generator with a pre-defined growing discriminator (Fig. 1 b)), where the architecture of the discriminator can be regarded as fixed and may limit the algorithm to search for an optimal architecture of generator.

In this work, we propose an **Adversarial N**eural **A**rchitecture **S**earch (AdversarialNAS) method to address the above challenges (Fig. 1 c)). First, we design a large search space (10^{38}) for fragile GAN and relax the search space to be continuous. Thus the architecture can be represented by a set of continuous variables obeying certain probability distribution and searched in a differentiable manner. Second, we propose to directly utilize the existing discriminator to evaluate the architecture of generator in each search iteration. Specifically, when searching for the generator architecture, the discriminator provides the supervision signal to guide the search direction, which is technically utilized to update the architecture distribution of generator through gradient descent. Therefore, our method is much more efficient since the extra computation cost for calculating evaluation metric is eliminated. Third, in order to consider the relevance and balance between the generator and discriminator, we propose to dynamically change the architecture of discriminator simultaneously during searching. Accordingly, we adopt the generator to evaluate the architecture of discriminator and comput the loss to update the discriminator architecture through ascending the stochastic gradient. The two architectures play against each other in a competition to continually improve their performance, which is essentially an adversarial searching mechanism. Therefore, the AdversarialNAS gets rid of calculating extra evaluation metric and solves the unsupervised task through an adversarial mechanism. It adequately considers the mutual balance between the two architectures, which benefits for searching a superior generative model.

To sum up, our main contributions are three-fold.

- We propose a novel AdversarialNAS method, which is the first gradient-based NAS method in GAN field and achieves state-of-art performance with much higher efficiency. We design a large architecture search space (10^{38}) for GAN and make it feasible to search in. Our AdversarialNAS can only tasks 1 GPU day for searching an optimal architecture in the large search space.
- Considering GAN is an unique competition between two networks, the proposed AdversarialNAS alternately searches the architecture of both of them under an adversarial searching strategy to improve their mutual balance, which is specifically tailored for GAN.
- The searched architecture has more advanced transferability and scalability while achieving state-of-the-art performance on both CIFAR-10 and STL-10 datasets.

2. Related Work

2.1. Generative Adversarial Networks

Although Restricted Boltzmann Machines [15] and flow-based generative models [6] are all capable of generating natural images, GANs [10] are still the most widely used methods in recent years due to their impressive generation ability. GANs based approaches have achieved advanced results in various generation tasks, such as image-to-image translation [18, 5, 19, 48], text-to-image translation [43, 45] and image inpainting [29]. However, the potential of GANs has not been fully explored since there is rare work [32] studying the impact of architecture design on the performance of GANs. In this work, we aim to search for a powerful and effective network structure specifically for the generative model via an automatic manner.

2.2. Neural Architecture Search

Automatic Machine Learning (AutoML) has attracted lots of attention recently, and Neural Architecture Search (NAS) is one of the most important direction. The goal of NAS is to automatically search for an effective architecture that satisfies certain demands. The NAS technique has applied to many computer vision tasks such as image classification [2, 25, 26, 31, 49], dense image prediction [24, 47, 3] and object detection [8, 30].

Early works of NAS adopt heuristic methods such as reinforcement learning [49] and evolutionary algorithm [41]. Obtaining an architecture with remarkable performance using such methods requires huge computational resources, e.g., 2000 GPUs days [41]. Therefore, lots of works design various strategies to reduce the expensive costs including weight sharing [31], performance prediction [1], progressive manner [25] and one-shot mechanism [26, 42]. The DARTS [26] in one-shot literature is the first approach that relaxes the search space to be continuous and conducts searching in a differentiable way. The architecture parameters and network weights can be trained simultaneously

in an end-to-end fashion by gradient descent. Thus it extremely compresses the search time.

However, all of these methods are designed for recognition and supervision tasks. To the best of our knowledge, there have been limited works [9] exploring applying NAS to unsupervised or weakly supervised tasks. In this work, we present the first gradient-based NAS method in GAN field and achieve state-of-the-art performance with much higher efficiency in the unsupervised image generation task.

2.3. NAS in GANs

Recently, a few works have attempted to incorporate neural architecture search with GANs. AutoGAN [9] adopts the reinforcement learning strategy to discover the architecture of generative models automatically. However, it only searches for the generator with a fixed architecture of discriminator. This mechanism limits the performance of the searched generator since the stability of GANs training is highly dependent on the balance between these two players. Besides, the search space is relatively small (10^5), thus its randomly searched architecture can achieve acceptable results, e.g., FID (lower is better): 21.39 (random) and 12.42 (search) in CIFAR-10 dataset. The AGAN [38] enlarges the search space specifically for the generative model, but the computational cost is expensive (1200 GPUs days) under the reinforcement learning framework. The performance of the discovered model is slightly worse, e.g., FID: 30.5 in CIFAR-10. Moreover, the reward used to update the weights of the network controller during evaluation stage is Inception Score, which is not a suitable supervisory single to guide the architecture search since it is time-consuming.

Instead, we search the architecture in a differentiable way and discard the evaluation stage. The reward of previous methods is obtained after a prolonged training and evaluation process, while our signal (loss) for guiding the search direction is given instantly in each iteration. Thus our method is more efficient. The designed adversarial search algorithm improves the mutual balance of the two networks for stabling and optimizing the search process.

3. Method

In this section, we first introduce the proposed search space of GANs and the way for relaxing it to be continuous. Then we describe the AdversarialNAS method.

3.1. Search Space for GANs

The goal of the proposed AdversarialNAS is to automatically search for an superior architecture of generative model through an adversarial searching mannr. Specifically, we aim to search for a series of cells, including Up-Cell and Down-Cell, as the building blocks to construct the final architecture of GAN. Three Up-Cells and four Down-Cells are stacked to form a generator and discriminator respectively. Since the convolution neural network has a natural hierarchical structure and each layer has unique function, we search for the cells each with a different architecture.

We represent a cell as a Directed Acyclic Graph (DAG) consisting of an ordered sequence of N nodes (Fig. 2). The cell takes image features as input and outputs processed features, where each node x_i in DAG indicates an intermediate feature and each edge $f_{i,j}$ between two nodes x_i, x_j is a specific operation. Since we aim to search for an optimal architecture of generator that is actually an upsampling network, we design a search space for specific Up-Cell that is almost fully connected topology, as given in the left of Fig. 2. The Up-Cell consists of 4 nodes, and each node can be obtained by its previous nodes through selecting an operation from a candidate set according to the search algorithm. The search space of generator $\mathbb{F}_G$ includes a candidate set of normal operations, which is designed as below.

- None
- Identity
- Convolution 1x1, Dilation=1
- Convolution 3x3, Dilation=1
- Convolution 3x3, Dilation=2
- Convolution 5x5, Dilation=1
- Convolution 5x5, Dilation=2

The 'None' means there is no operation between two corresponding nodes, which is used to change the topology of the cell. The 'Identity' denotes the skip connection operation that provides multi-scale features. The stride of these operations is 1 so that they will keep the spatial resolution. The search space of generator also contains a subset of upsampling operations, which is devised as below.

- Transposed Convolution 3x3
- Nearest Neighbor Interpolation
- Bilinear Interpolation

Note that, these operations can only be searched by edge $0 \rightarrow 1$ and $0 \rightarrow 2$ in a specific Up-Cell. To search for the generator in an adversarial way, we simply invert the Up-Cell to form a Down-Cell (shown in the right of Fig. 2) ensuring their balance. The search space of discriminator $\mathbb{F}_D$ also contains a candidate set of normal operations, which is the same as the one of Up-Cell. However, the candidate set of downsampling operations is achieved by

- Average Pooling
- Max Pooling
- Convolution 3x3, Dilation=1
- Convolution 3x3, Dilation=2
- Convolution 5x5, Dilation=1
- Convolution 5x5, Dilation=2

With stride equaling 2, the downsampling operations can only be searched in edge $2 \rightarrow 4$ and $3 \rightarrow 4$. Therefore, during searching, there are totally 10^{38} different network architectures for GANs.

3.2. Continuous Relaxation of Architectures

The goal of the search algorithm is to select a specific operation from the pre-defined candidate set for each edge. Therefore, the intermediate node $x_{n,j}$ in the n-th cell can be calculated through the selected functions and its previous connected nodes as $x_{n,j} = \sum_{i<j} f_{n,i,j}(x_{n,i})$. For RL-

based NAS algorithms, the function $f_{n,i,j}$ is directly sampled from the candidate set according to the learnable architecture controller. Inspired by Gradient-based NAS algorithm [26], we relax the function $f_{n,i,j}$ to a soft version through Gumbel-Max trick [27]:

$$f^{soft}_{n,i,j}(x) = \sum_{f\in\mathbb{F}_G} \frac{\exp((p^f_{n,i,j} + o^f)/\tau)}{\sum_{f'\in\mathbb{F}_G} \exp((p^{f'}_{n,i,j} + o^{f'})/\tau)} f(x), \tag{1}$$

where o^f is the noise sampled from the Gumbel (0,1) distribution, and the τ is the softmax temperature. The $p^f_{n,i,j}$ is the probability of selecting a specific function f in edge $i \rightarrow j$ of n-th cell. The Gumbel version softmax is applied to follow the learned probability distribution more strictly. Therefore, each edge will contain a probability vector $[p^{f_1}, ..., p^{f_m}], m = |\mathbb{F}_G|$. This discrete probability distribution is calculated through a simple softmax function as $p^f = \frac{\exp(\alpha^f)}{\sum_{f\in\mathbb{F}_G}\exp(\alpha^{f'})}$, where the α is the learnable parameter. Therefore, the goal of searching for an architecture is converted to learning an optimal set of probability vectors for every edge, and the architecture can be derived from the learned probability distribution. Besides, in order to dynamically change the architecture of discriminator simultaneously, we also conduct a set of continuous parameters β for calculating the probability of each function in discriminator as $q^f = \frac{\exp(\beta^f)}{\sum_{f\in\mathbb{F}_D}\exp(\beta^{f'})}$. Therefore, the soft version of the function can be achieved like the generator as

$$f^{soft}_{n,i,j}(x) = \sum_{f\in\mathbb{F}_D} \frac{\exp((q^f_{n,i,j} + o^f)/\tau)}{\sum_{f'\in\mathbb{F}_D} \exp((q^{f'}_{n,i,j} + o^{f'})/\tau)} f(x). \tag{2}$$

Then, the proposed AdversarialNAS aims to learn a set of continuous parameters α and β in a differentiable manner and obtain the final architecture of generator by simply preserving the most likely operations in the search space. Note that, we term the networks with all operations softly combined by the architecture parameters as Super-G and Super-D. The topology of the network would be changed by the learned high probability 'None' operation, and the 'Identity' operation would provide multi-scale fusion.

3.3. Adversarial Architecture Search

Before introducing the optimization strategy of the proposed AdversarialNAS, we first briefly revisit the optimization function in the classification literature. The searching process is formulated as a bilevel optimization problem:

$$\begin{aligned} &\min_{\alpha} \quad L_{val}(w^*(\alpha), \alpha) \\ &s.t. \quad w^*(\alpha) = \arg\min_{w} L_{train}(w, \alpha), \end{aligned} \tag{3}$$

where L_{val} and L_{train} denote the loss function on the validation and training set respectively. The goal of the search

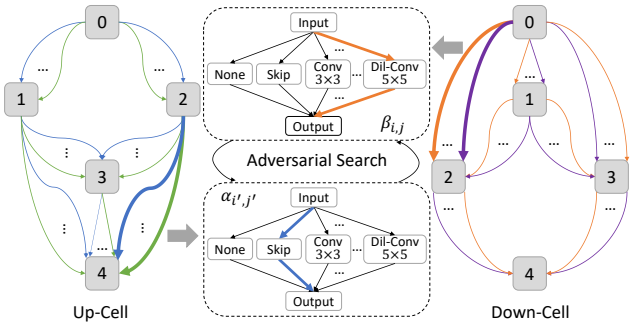

Figure 2. The search space of Up-cell and Down-Cell. The architectures of both Up-Cell and Down-Cell will continuously promote each other in an adversarial manner.

algorithm is to discover an optimal architecture α^* by calculating and minimizing the validation loss $L_{val}(w^*, \alpha)$, where w^* is the optimal weights of the current architecture α and is obtained by calculating and minimizing the training loss $L_{train}(w, \alpha)$. Both the weight and architecture are optimized by ascending its gradient descent.

However, in the task of unconditional image generation, there are no labels to supervise the searching procedure. AutoGAN [9] and AGAN [38] apply IS to evaluate the architecture performance and optimize the architecture by RL strategy. Computing IS requires generating hundreds of images and adopts Inception model to infer the result offline after a prolonged training trajectory of each discrete architecture, which is extremely time consuming. Therefore, we propose to make the architectures of generator and discriminator compete with each other to improve both of their performance, i.e., utilizing discriminator to guide the generator search and vice versa. AdversarialNAS leverages an adversarial optimization strategy that is inspired by the formulation of original GANs [10] for optimizing the architecture in a differentiable way. Thus the optimization process is defined as a two-player min-max game with value function $V(\alpha, \beta)$ where the weight of each network must be current optimal. The formulation of the introduced algorithm is given in Eqn.(4):

$$\begin{aligned} &\min_{\alpha}\max_{\beta} V(\alpha,\beta) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x \mid \beta, W^*_D(\beta))] \\ &+ \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z \mid \alpha, W^*_G(\alpha)) \mid \beta, W^*_D(\beta)))] \\ &s.t. \\ &W^*_D(\beta) = \arg\max_{W_D(\beta)} \mathbb{E}_{x\sim p_{data}(x)}[\log D(x \mid \beta, W_D(\beta))] \\ &+ \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G^*_{D_\beta}(z) \mid \beta, W_D(\beta)))] \\ &W^*_G(\alpha) = \arg\min_{W_G(\alpha)} \mathbb{E}_{z\sim p_z(z)}[\log(1 - D^*_{G_\alpha}(G(\alpha \mid W_G(\alpha))))], \end{aligned} \tag{4}$$

where p_{data} means true data distribution and p_z is a prior distribution. In the up-level stage the $W^*_D(\beta)$ denotes the

optimal weights of discriminator under the specific architecture β and $W_G^*(\alpha)$ represents the optimal weights of generator under the architecture α. In the low-level stage, the two optimal weights $\{W_G^*(\alpha), W_D^*(\beta)\}$ for any particular pair of architectures $\{\alpha, \beta\}$ can be obtained through another min-max game between W_G and W_D:

$$\min_{W_G(\alpha)} \max_{W_D(\beta)} V(W_G(\alpha), W_D(\beta)) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x \mid \beta, W_D(\beta))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z \mid \alpha, W_G(\alpha)) \mid \beta, W_D(\beta)))]. \quad (5)$$

However, this inner optimization (Eq. 5) is time-consuming. For NAS in the classification task [26, 7, 4], the inner optimization (Eq. 3) is normally approximated by one step training as $\nabla_\alpha L_{val}(w^*(\alpha), \alpha) \approx \nabla_\alpha L_{val}(w - \xi\nabla_w L_{train}(w, \alpha), \alpha)$. Inspired by this technique, for a given pair of architectures $\{\alpha, \beta\}$, the corresponding optimal weights $\{W_G^*(\alpha), W_D^*(\beta)\}$ can be obtained by single step of adversarial training (Eq. 5) as vanilla GANs.

Algorithm 1 Minibatch stochastic gradient descent training of Adversarial Neural Architecture Search.

1: **for** number of training iterations **do**
2: **for** k step **do**
3: Sample minibatch of $2m$ noise samples $\{z^{(1)}, ..., z^{(2m)}\}$ from noise prior.
4: Sample minibatch of $2m$ examples $\{x^{(1)}, ..., x^{(2m)}\}$ from real data distribution.
5: Update the architecture of discriminator by ascending its stochastic gradient: $\nabla_\beta \frac{1}{m}\sum_{i=1}^{m}\left[\log(x^i) + \log(1 - D(G(z^i)))\right]$
6: Update the weights of discriminator by ascending its stochastic gradient: $\nabla_{W_D} \frac{1}{m}\sum_{i=m+1}^{2m}\left[\log(x^i) + \log(1 - D(G(z^i)))\right]$
7: **end for**
8: Sample minibatch of $2m$ noise samples $\{z^{(1)}, ..., z^{(2m)}\}$ from noise prior.
9: Update the architecture of generator by descending its stochastic gradient: $\nabla_\alpha \frac{1}{m}\sum_{i=1}^{m}\left[\log(1 - D(G(z^i)))\right]$
10: Update the weights of generator by descending its stochastic gradient: $\nabla_{W_G} \frac{1}{m}\sum_{i=m+1}^{2m}\left[\log(1 - D(G(z^i)))\right]$
11: **end for**

Moreover, the min-max game between two architectures can also be searched in an alternative way. Specifically, the currently optimal architecture of generator for the given discriminator can be achieved through single step of adversarial training, which has been proven by Goodfellow in [10]. The proposed AdversarialNAS algorithm is shown in Alg. 1, and optimal architectures or weights in each iteration can be achieved by ascending or descending the corresponding stochastic gradient. Note that, the order of the updating strategy is architecture first in each training iteration, which guarantees the weights for updating the corresponding architecture to be currently optimal. For example, the discriminator used in ninth line of Alg. 1 is D^* with optimal architecture and weights for the current generator.

The proposed AdversarialNAS method can be plug-and-play to the original training procedure of GANs to search the architecture more naturally, which is specifically tailored for GANs.

4. Experiments

4.1. Experimental Setup

Datasets. Following [9, 38], we adopt CIFAR-10 [22] and STL-10 to evaluate the effectiveness of our approach. The CIFAR-10 contains 60,000 natural images including 10 different classes in 32×32 spatial resolution. Concretely, we use its training set that consists of 50,000 images without any data augmentation technique to search for the optimal architecture of the generator. We also apply this training set to train the discovered architecture. To further evaluate the transferability of the architecture, we also adopt totally 105,000 images in STL-10 dataset to directly train the searched architecture without any data augmentation for a fair comparison with previous works.

Implementation. We use Adam optimizer [21] and hinge loss to train the shared weights of Super-GAN and provide the supervision signal for updating the architectures. Specifically, the hyper-parameters of optimizers for training the weights of both generator and discriminator are set to $\beta_1 = 0.0$, $\beta_2 = 0.9$ and learning rate is set to 0.0002. The hyper-parameters of optimizers for optimizing both architectures are set to $\beta_1 = 0.5$, $\beta_2 = 0.9$ and the learning rate is 0.0003 with the weight decay of 0.0001. When searching, the batch size is set to 100 for both generator and discriminator, and we search for about 2,500 iterations. When training the derived generator, we directly adopt the discriminator used in AutoGAN [9] for a fair comparison, which is similar to the one in SNGAN [28]. The batch size is set to 40 for generator and 20 for discriminator, respectively. We train the network for about 500 epochs, and the hyper-parameters of the optimizer are the same as the ones in searching. Besides, the same as all other methods, we randomly generate 50,000 images for calculating the Inception Score and FID to evaluate the network performance.

Computational Costs. The proposed AdversarialNAS takes about 12 hours to converge for searching for an optimal architecture on two NVIDIA RTX 2080Ti GPUs. It requires only 1 GPU day to achieve the final architecture in a large search space (about 10^{38}), while AutoGAN [9] requires 2 GPU days in a quite small search space (about 10^5)

and AGAN [38] needs even 1200 GPU days for searching in a comparable space. Note that we directly use the released code of AutoGAN to search on the same hardware 2080Ti GPU and the searching time of AGAN is from their original paper (running on NVIDIA Titan X GPU).

4.2. Compared with State-of-the-Art Approaches

In this section, we discuss the searched architecture and compare its performance with state-of-the-art approaches including hand-crafted and auto-discovered ones. To explore the transferability of the discovered architecture, we directly apply it to another dataset and retrain its weights for comparing with other methods. Besides, we further study the scalability of the searched architecture and prove its superiority to other methods.

4.2.1 Results on CIFAR-10

At the end of the searching program, we directly sample the architecture from the search space by picking the operations with maximum weights α. The optimal architecture searched on CIFAR-10 is shown in Tab. 1 and some valuable observations can be received from this table.

- The searched generator prefer 'Bilinear' operation for upsampling features although it has no learnable parameters. Besides, the 'Bilinear Interpolation' provides more accurate expanded features than simple 'Nearest ' operation, which is discovered by the searching algorithm.
- Surprisingly, there is no dilated convolution in this architecture. It seems that, for low-resolution images (32×32), simply stacking normal convolutions may already satisfy and achieve the optimal Effective Receptive Field (ERF) of the generator.
- We can also observe that the deeper cell tends to be more shallow since more 'None' operations are preferred. The shallow cell has more multi-scale feature fusion operation, which is represented by the discovered parallel 'Identity' connection of convolution.

The quantitative comparisons with previous state-of-the-art methods are given in Tab. 2. From the table, we can see that the proposed AdversarialNAS is the first gradient-based approach that can search in a large search space with affordable cost. The designed search space has 10^{38} different architectures of GANs, which is several orders of magnitude larger than the search space (10^5) of AutoGAN [9]. Moreover, the proposed method only takes about 1 GPU day for searching for an optimal architecture while the AGAN [38] spends 1200 GPU days under a comparable search space. In the CIFAR-10 dataset, our discovered 'AdversarialNAS-GAN' achieves new state-of-the-art FID score (10.87), which is quite encouraging. It also obtains

Up-Cell	Edge	Operation	Num	Resolution
Cell-1	$0 \to 1$	Bilinear	1	$4 \to 8$
	$0 \to 2$	Bilinear	1	$4 \to 8$
	$1 \to 3$	Identity	1	$8 \to 8$
	$1 \to 4$	Conv 3×3	256	$8 \to 8$
	$2 \to 3$	None	–	–
	$2 \to 4$	Conv 3×3	256	$8 \to 8$
	$3 \to 4$	Identity	1	$8 \to 8$
	$3 \to c_2$	Bilinear	1	$8 \to 16$
	$3 \to c_3$	Nearest	1	$8 \to 32$
Cell-2	$0 \to 1$	Bilinear	1	$8 \to 16$
	$0 \to 2$	Bilinear	1	$8 \to 16$
	$1 \to 3$	None	–	–
	$1 \to 4$	Conv 3×3	256	$16 \to 16$
	$2 \to 3$	Identity	1	$16 \to 16$
	$2 \to 4$	Conv 3×3	256	$16 \to 16$
	$3 \to 4$	Conv 3×3	256	$16 \to 16$
	$3 \to c_3$	Nearest	1	$16 \to 32$
Cell-3	$0 \to 1$	Nearest	1	$16 \to 32$
	$0 \to 2$	Bilinear	1	$16 \to 32$
	$1 \to 3$	None	–	–
	$1 \to 4$	Conv 3×3	256	$32 \to 32$
	$2 \to 3$	Conv 3×3	256	$32 \to 32$
	$2 \to 4$	None	–	–
	$3 \to 4$	Conv 3×3	256	$32 \to 32$

Table 1. The searched optimal architecture of generator by the proposed AdversarialNAS on CIFAR-10 with no category labels used. The 'Num' indicates the number of operations.

an Inception Score (8.74 ± 0.07) that is highly competitive with state-of-the-art Progressive GAN [20] (8.80 ± 0.05) and superior to AutoGAN [9] (8.55 ± 0.10). It is worth noting that the Progressive GAN applies a well-designed progressive training strategy that is time-consuming, while we directly train the discovered generator as vanilla GANs.

Besides, we randomly generate 50 images without cherry-picking, which are given in the Fig. 3. These qualitative results demonstrate that our searched generator can create diverse images that contain realistic appearance and natural texture without any clue of model collapse.

Figure 3. The CIFAR-10 images generated by discovered generator in random without cherry-picking.

Method	Search Method	Search Space	Search Cost	Size (MB)	IS↑ on C-10	FID↓ on C-10	IS↑ on S-10	FID↓ on S-10
DCGANs [32]					6.64 ± 0.14	–	–	–
Improved GAN [33]					6.86 ± 0.06	–	–	–
LRGAN [44]					7.17 ± 0.17	–	–	–
DFM [40]					7.72 ± 0.13	–	8.51 ± 0.13	–
ProbGAN [13]					7.75	24.6	8.87 ± 0.09	46.74
WGAN-GP, ResNet [12]	Manual	–	–	–	7.86 ± 0.07	–	–	–
Splitting GAN [11]					7.90 ± 0.09	–	–	–
MGAN [16]					8.33 ± 0.10	26.7	–	–
Dist-GAN [37]						17.61	–	36.19
Progressive GAN [20]					$\mathbf{8.80 \pm 0.05}$	–	–	–
Improving MMD-GAN [39]					8.29	16.21	9.23 ± 0.08	37.64
SN-GAN [28]				4.3	8.22 ± 0.05	21.7	9.16 ± 0.12	40.1
AGAN [38]	RL	–	1200	20.1	8.29 ± 0.09	30.5	9.23 ± 0.08	52.7
Random Search [23]†	Random	10^5	2	–	8.09	17.34	–	–
AutoGAN [9]	RL	10^5	2	4.4	8.55 ± 0.10	12.42	9.16 ± 0.12	31.01
Random Search [23]††	Random	10^{38}	1	12.5	6.74 ± 0.07	38.32	7.66 ± 0.08	53.45
AdversarialNAS-GAN	Gradient	10^{38}	1	8.8	8.74 ± 0.07	**10.87**	$\mathbf{9.63 \pm 0.19}$	**26.98**

Table 2. The quantitative comparisons with state-of-the-art approaches. † indicates the results are achieved in the search space of AutoGAN and †† denotes the results in our search space.

4.2.2 Transferability of the Architectures

Following the setting of AutoGAN [9] and AGAN [38], we directly apply the generator searched on CIFAR-10 to STL-10 dataset for evaluating the transferability of the architecture. Specifically, we adopt totally 105,000 images with no labels used to train this network. The number of training epochs is the same as the one on CIFAR-10 and we also randomly generate 50,000 images for calculating the Inception Score and FID. We alter the resolution of input noise to 6×6 for generating the image with the size of 48×48, as the AutoGAN and AGAN do.

The quantitative results are shown in Tab. 2. We can observe that our network suffers no overfitting on the CIFAR-10 dataset and has a superior ability of generalization. Specifically, it achieves the state-of-the-art Inception Score (9.63) and FID (26.98) on STL-10, which are far better than all hand-crafted and auto-discovered methods. The qualitative results are also given in Fig. 4 to prove its ability to generate diverse and realistic images.

Figure 4. The STL-10 images randomly generated without cherry-picking by the generator discovered on CIFAR-10.

4.2.3 Scalability of the Architectures

In this section, we further explore the scalability of the discovered architecture on the CIFAR-10 dataset.

We compare our searched generator with two representative works, manual-designed SNGAN [28] and auto-discovered AutoGAN [9]. We scale the parameter size of these generators from 1 MB to 25 MB through channel dimension, which is a large scope. Note that, for a fair comparison, we use the same discriminator with a fixed size in all experiments to observe the impact of generator capacity changes. The qualitative comparisons are illustrated in Fig. 5 and Fig. 6. The x-axis in both figures denotes the parameter size (MB) of the specific generator. The y-axis is IS in Fig. 5 and is FID in Fig. 6. These experiments demonstrate that our searched architecture is more stable and almost unaffected when scaling the model size. When the size is extremely compressed to only 1 MB, the SNGAN and AutoGAN all suffer from the disaster of performance degradation, while the performance of 'AdversarialNAS-GAN' is almost unchanged. Notably, the performance will continually drop when expanding the generator size because the enlarged generator will not be balanced with the fixed-size discriminator any more. However, both Fig. 5 and Fig. 6 demonstrate that our discovered architecture will not suffer from performance drop, which means it is more robust and has superior inclusiveness for discriminators.

Methods	Discriminator		CIFAR-10		STL-10	
	Architecture	Type	IS↑	FID↓	IS↑	FID↓
Random Search	Fixed	AutoGAN-D	6.74 ± 0.13	38.32	7.66 ± 0.11	53.45
SingalNAS	Fixed	SNGAN-D	7.72 ± 0.03	27.79	6.56 ± 0.12	84.19
SingalNAS	Fixed	AutoGAN-D	7.86 ± 0.08	24.04	8.52 ± 0.05	38.85
SingalNAS	Fixed	Super-D	7.77 ± 0.05	23.01	8.62 ± 0.03	41.57
AdversarialNAS	Dynamic	Searched-D	$\mathbf{8.74 \pm 0.07}$	**10.87**	$\mathbf{9.63 \pm 0.19}$	**26.98**

Table 3. We search the generative model on CIFAR-10 with different methods and retain the weight of these searched architectures to evaluate their performance on both CIFAR-10 and STL-10.

4.3. Ablation Studies

To further evaluate the effectiveness of the proposed AdversarialNAS, we conduct a series of ablation studies.

First, we conduct a random search strategy [23] to search for the generator where we adopt the fixed-architecture discriminator of AutoGAN for a fair comparison. The performance of the searched generative model is shown in Tab. 3. Second, we propose 'SingalNAS' to search the optimal generator with different types of fixed architecture of discriminator, while the weights of discriminator can still be trained. Accordingly, the supervision signal for updating the generator architecture comes from the fixed architecture of discriminator, and the discriminator architecture does not dynamically change according to generator during searching. We adopt the discriminator architecture of SNGAN and AutoGAN, respectively. In addition, to verify the influences of our search space, we also conduct 'SingalNAS' with the fixed Super-D. Third, we use the proposed 'AdversarialNAS' to search the generator and discriminator simultaneously. Note that, the time consuming of both searching and training in all experiments is constrained to be consistent.

The effectiveness of our adversarial searching strategy can be observed from the comparisons in Tab. 3.

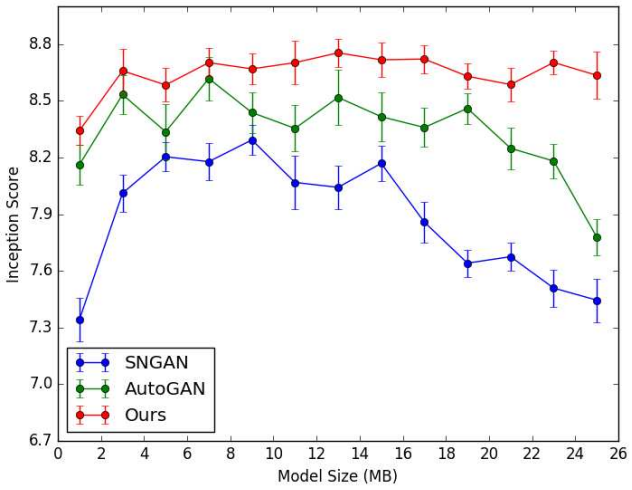

Figure 5. Inception Score curves of different methods.

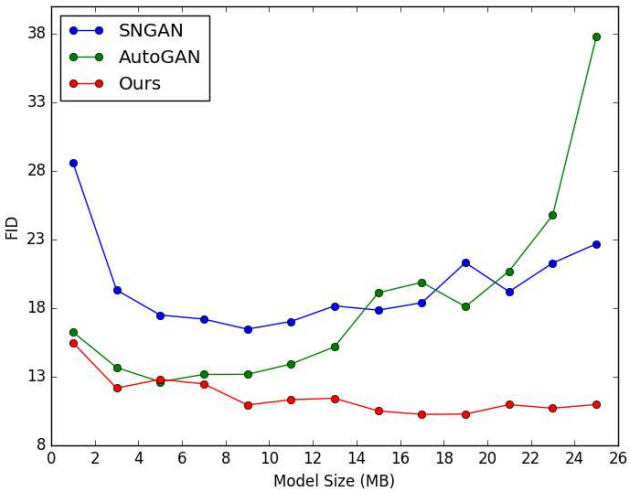

Figure 6. FID curves of different methods.

5. Conclusion

In this work, we propose a large search space for GANs and a novel AdversarialNAS method to search for a superior generative model automatically. The proposed searching algorithm can directly be inserted to the original procedure of GAN training and search the architecture of generator in a differentiable manner through an adversarial mechanism, which extremely reduces the search cost. The discovered network achieves state-of-the-art performance on both CIFAR-10 and STL-10 datasets, and it also has advanced transferability and scalability.

Furthermore, we believe the idea behind our AdversarialNAS is not only specific to NAS-GAN and may benefit other potential field where there are multiple network architectures requiring mutual influence, such as network architecture distillation, pruning and mutual learning.

Acknowledgement This work is partially supported by the National Natural Science Foundation of China (Grant 61572493, Grant 61876177), Beijing Natural Science Foundation (L182013, 4202034) and Fundamental Research Funds for the Central Universities. This work is also sponsored by Zhejiang Lab (No.2019KD0AB04).

References

[1] Andrew Brock, Theodore Lim, James Millar Ritchie, and Nicholas J Weston. Smash: One-shot model architecture search through hypernetworks. In *ICLR*, 2018.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[3] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NIPS*, 2018.

[4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *arXiv preprint arXiv:1904.12760*, 2019.

[5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[6] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[7] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *ICCV*, 2019.

[8] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019.

[9] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *ICCV*, 2019.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[11] Guillermo L. Grinblat, Lucas C. Uzal, and Pablo M. Granitto. Class-splitting generative adversarial networks. *ArXiv*, abs/1709.07359, 2017.

[12] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NIPS*, 2017.

[13] Hao He, Hao Wang, Guang-He Lee, and Yonglong Tian. Probgan: Towards probabilistic gan with theoretical guarantees. In *ICLR*, 2019.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[15] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[16] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Q. Phung. Mgan: Training generative adversarial nets with multiple generators. In *ICLR*, 2018.

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[19] Wentao Jiang, Si Liu, Chen Gao, Jie Cao, Ran He, Jiashi Feng, and Shuicheng Yan. Psgan: Pose and expression robust spatial-aware gan for customizable makeup transfer. *arXiv preprint arXiv:1909.06956*, 2019.

[20] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[23] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019.

[24] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pages 82–92, 2019.

[25] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018.

[26] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[27] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *NIPS*, 2014.

[28] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[29] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[30] Junran Peng, Ming Sun, Zhaoxiang Zhang, Tieniu Tan, and Junjie Yan. Efficient neural architecture transformation searchin channel-level for object detection. *arXiv preprint arXiv:1909.02293*, 2019.

[31] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

[32] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[33] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[35] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[36] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[37] Ngoc-Trung Tran, Tuan-Anh Bui, and Ngai-Man Cheung. Dist-gan: An improved gan using distance constraints. In *ECCV*, 2018.

[38] Hanchao Wang and Jun Huan. Agan: Towards automated design of generative adversarial networks. *arXiv preprint arXiv:1906.11080*, 2019.

[39] Wei Wang, Yuan Sun, and Saman K. Halgamuge. Improving mmd-gan training with repulsive loss function. *ArXiv*, abs/1812.09916, 2018.

[40] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. In *ICLR*, 2017.

[41] Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, 2017.

[42] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[43] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

[44] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *arXiv preprint arXiv:1703.01560*, 2017.

[45] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[46] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[47] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *CVPR*, 2019.

[48] Defa Zhu, Si Liu, Wentao Jiang, Chen Gao, Tianyi Wu, and Guodong Guo. Ugan: Untraceable gan for multi-domain face translation. *arXiv preprint arXiv:1907.11418*, 2019.

[49] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.