CycleGANAS: Differentiable Neural Architecture Search for CycleGAN

Taegun An
Korea University
Department of Computer Science
antaegun20@korea.ac.kr

Changhee Joo
Korea University
Department of Computer Science
changhee@korea.ac.kr

Abstract

We develop a Neural Architecture Search (NAS) framework for CycleGAN that carries out unpaired image-to-image translation task. Extending previous NAS techniques for Generative Adversarial Networks (GANs) to CycleGAN is not straightforward due to the task difference and greater search space. We design architectures that consist of a stack of simple ResNet-based cells and develop a search method that effectively explore the large search space. We show that our framework, called CycleGANAS, not only effectively discovers high-performance architectures that either match or surpass the performance of the original CycleGAN, but also successfully address the data imbalance by individual architecture search for each translation direction. To our best knowledge, it is the first NAS result for CycleGAN and shed light on NAS for more complex structures.

1. Introduction

Generative Adversarial Networks [7] (GANs) is a unsupervised generative modeling for diverse and realistic data generation with a generator and discriminator trained in an adversarial manner. It can deal with the domain with insufficient amounts of data or high labeling costs and also has a great advantage over the domain where the generation of diverse and high-quality data is critical. In consequence, many variants and extensions of GANs have been proposed for a variety of tasks such as conditional generation, style transfer, machine translation, and anomaly detection.

CycleGAN is one of the powerful extensions of GANs and was developed for the image-to-image translation *without image pairing* [44]. Successfully removing the expensive laboring cost of building paired datasets, CycleGAN has been intensively applied to many translation applications, e.g., style transfer, medical diagnosis, voice conversion, etc. However, CycleGAN requires thorough fine-tuning of multiple neural networks, which has to consider the task objectives and possible imbalances in the dataset. Without the fine-tuning, it is likely to suffer from instability as GANs [21, 22]. This often becomes an obstacle when applying CycleGAN to new translation tasks [37] or designing a versatile architecture suitable for multiple tasks.

As a result, for CycleGAN, it is common to have a manual optimization process to tailor its architecture to specific applications [8, 13, 18]. Such manual optimization is not only very costly but also arbitrarily restricts the scope of the architecture search.

To our best knowledge, there is no priori NAS work for CycleGAN. However, there have been several interesting NAS works for unconditional GANs including AGAN [33], AutoGAN [6], and AdversarialNAS [4]. AGAN and AutoGAN adopted Reinforcement Learning (RL) to explore architectures through trial-and-error. AdversarialNAS has a more complex cell structure than the others, which motivates it to exploit the gradient-based approach introduced in DARTs [20] to effectively search on a large search space (10^{38}). Although they are an effective NAS method for GANs, their extension to CycleGAN is neither straightforward nor fruitful due to essential differences in the task and structure. In general, an image-to-image translation task is more complex than an unconditional image generation task, and CycleGAN has twice as many neural networks as GANs. Further, unlike unconditional GANs that can be trained with any random inputs, CycleGAN work only with inputs from finite unpaired datasets and thus should be more sample efficient. The larger search space and the limited inputs are one of the key factors that differentiate the NAS for CycleGAN and the NAS for GANs, and motivate us to develop a novel NAS framework for CycleGAN.

In this paper, we propose CycleGANAS, a multi-network architecture search framework for CycleGAN under data imbalance. We design a simple cell inspired by the design of residual cells and build the supernetworks of generators and discriminators of CycleGAN by stacking many simple cells. Accounting for the task nature of CycleGAN and the required search efficiency, we optimized the architecture and neural network weights with CycleGAN objectives *simultaneously*, which is different from the previous NAS for GANs that takes an iterative bi-level optimization method. Finally, during the search, we let the two generators have a different architecture, allowing them to be better tailored to each subtask and dataset. Through the experiments on various unpaired datasets, we show that CycleGANAS searches for good architectures not only efficiently but also in a stable manner, even under the data imbalance.

Our contributions can be summarized as follows.

- We develop CycleGANAS, a novel framework of multi-network architecture search for CycleGAN under data imbalance. We stack up many simple ResNet-based cells and take the gradient-based approach along with single-level joint optimization of neural network architecture and weights. Our framework admits asymmetric architectures that take into account the data imbalance.
- We show the performance of CycleGANAS through extensive experiments with various unpaired datasets. We investigate the effect of CycleGAN architecture and model size, demonstrating that manual balancing or naive asymmetric models are not effective for data imbalance. In contrast, we observe that CycleGANAS successfully searches for good architectures in a stable manner and achieve high performance under the data imbalance.

2. Related Works

2.1. Neural Architecture Search (NAS)

Neural architecture search (NAS) is a big branch of automated machine learning (Auto-ML) to search for the best neural network design, taking into consideration the task and dataset. NAS studies are commonly classified using three criteria: the search space, search method, and evaluation method [3, 9]. Among these, the search method is the key element for identifying various NAS algorithms. In the following, we provide a concise overview of NAS algorithms in the literature from the perspective of the search method.

Several search methods have been developed for NAS including RL, evolutionary algorithms (EA), differentiable methods, and other optimization techniques. Since RL was successfully adopted for NAS in [46], it has been exploited to optimize many architectures including CNNs [1, 2, 26]. Although they outperform the handcrafted neural networks in performance, they demand a significant amount of computation. Although recent NAS frameworks with Bayesian optimization (BO) [12, 29, 30, 35, 43] or evolutionary algorithms (EA) [19, 25, 27] substantially reduce the computation cost, their applications are still limited to a problem with small search space. Gradient-based NAS method, first appeared in DARTS [20], enables NAS with a large search space through continuous relaxation of architectures. It has been reported that in certain problems, the architecture converges to the optimal one under the gradient-based NAS [16].

2.2. Generative Adversarial Networks (GANs)

Since the GANs framework has been developed for the unconditional image generation via adversarial training of generator and discriminator, the framework has been extended to many computer vision tasks including super-resolution [34, 41], image inpainting [38], natural language processing (NLP) [5, 39], etc.

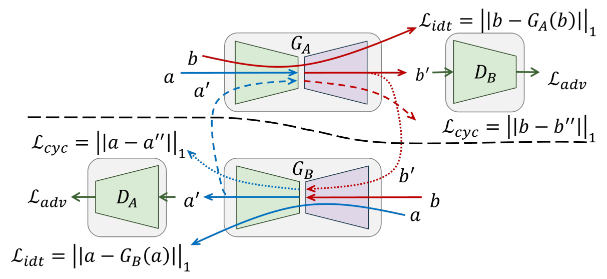
CycleGAN is also an extension of the GANs framework to the multi-network system for image-to-image translation task. Pix2pix [10] is the first work that performs the translation task with two pairs of GANs and paired data. CycleGAN [44] and DiscoGAN [14] remove the requirement of the dataset pairing by introducing the cycle-consistency objective, and enable the translation with *unpaired data*. The technique has been now widely used for medical image translation [42], frame prediction [15], inter-domain translation [28], and multi-modal learning [45].

2.3. NAS for GANs

Since the success of NAS for convolutional neural networks (CNNs), NAS for GANs has attracted much attention. As in CNNs, RL-based search methods were adopted in NAS for GANs [6, 33, 40], in which the architecture of GANs is divided into a predetermined number of cells and optimized through an LSTM agent. A common challenge of these RL-based approaches is the computational complexity to obtain Inception Score (IS) or Frechet inception distance (FID) score, which is used as the reward feedback. Some works introduce score predictors to mitigate the computation burden [36].

The most relevant to our work among previous NAS frameworks for GANs is AdversarialNAS [4] that adopts the gradient-based, differentiable architecture search method. Using the adversarial loss, AdversarialNAS could achieve state-of-the-art performance on a large search space at the cost of 1 GPU day. However, the extension of AdversarialNAS to CycleGAN is not straightforward, since it has a large and complicated cell structure, which makes it hard to scale, and its bi-level optimization is not sufficiently efficient for the task with a small amount of data, i.e., the image-to-image translation task of CycleGAN [31].

While not directly related to NAS, there are a few studies that focus on compressing the architecture of CycleGAN via combinatorial optimization [17] or evolutionary algorithm [32]. Given good reference architectures, they find small-size architectures with comparable performance. Although they are doing a sort of architecture search, their approach is quite different from NAS since they have clear reference models.



$\mathcal{L}_{idt} = ||b - G_A(b)||_1$
G_A
b
a
a'
b'
D_B
$\mathcal{L}_{adv}$
$\mathcal{L}_{cyc} = ||b - b''||_1$
$\mathcal{L}_{cyc} = ||a - a''||_1$
G_B
b'
$\mathcal{L}_{adv}$
D_A
a'
b
a
$\mathcal{L}_{idt} = ||a - G_B(a)||_1$

Figure 1. Overall process of CycleGAN and its losses. CycleGAN uses two generators G_A, G_B to translate image $a \in \mathcal{A}$ to image $b' \in \mathcal{B}$ and image $b \in \mathcal{B}$ to image $a' \in \mathcal{A}$, and two discriminators D_A, D_B to distinguish the generated images. It computes several losses using the images from different sources.

3. CycleGANAS – NAS for CycleGAN

3.1. Preliminaries

CycleGAN has two generators (G_A, G_B) and two discriminators (D_A, D_B) as shown in Fig. 1, and translate images in dataset, which consists of two subdatasets, each from a different domain denoted by $\mathcal{A}, \mathcal{B}$. We slightly abuse the notation and also denote the subdataset of each domain by $\mathcal{A}$ and $\mathcal{B}$, respectively. CycleGAN aims to learn the mapping of optimal generators $G_A^* : \mathcal{A} \to \mathcal{B}$, $G_B^* : \mathcal{B} \to \mathcal{A}$, and optimal discriminators $D_A^* : \Omega \to \{0, 1\}$, $D_B^* : \Omega \to \{0, 1\}$ that distinguish between translated samples and real samples, where Ω denotes the set of all possible image samples. For CycleGAN, we relax the output of discriminators to a real number in the range $[0, 1]$, and consider G_A, G_B, D_A, D_B as a function that takes an input image and outputs an image or a number in $[0, 1]$. The objective of CycleGAN consists of three loss functions, known as the adversarial, cycle-consistency, and identity loss [44].

- The adversarial objective makes each pair of generator and discriminator engage in a two-player mini-max game. The *adversarial loss* $L_{adv}(G_A, D_B)$ for G_A, D_B can be written as

$$L_{adv}(G_A, D_B) = \mathbb{E}_{b\sim\mathcal{B}}[\log D_B(b)] + \mathbb{E}_{a\sim\mathcal{A}}[\log(1 - D_B(G_A(a)))]. \quad (1)$$

  The adversarial loss $L_{adv}(G_B, D_A)$ for G_B and D_A can be defined similarly.
- It is claimed that CycleGAN should be cycle-consistent, i.e., $G_B(G_A(a)) \approx a$ for $a \in \mathcal{A}$ and $G_A(G_B(b)) \approx b$ for $b \in \mathcal{B}$. The cycle consistency objective is imposed only on the generators and couples them under the cooperative framework to generate better output images for each other. The *cycle-consistency loss* $L_{cyc}(G_A, G_B)$ can be written as

$$L_{cyc}(G_A, G_B) = \mathbb{E}_{a\sim\mathcal{A}}[|G_B(G_A(a)) - a|] + \mathbb{E}_{b\sim\mathcal{B}}[|G_A(G_B(b)) - b|]. \quad (2)$$

- It has been shown that, if the input image does not belong to the target domain, the identity objective helps the generator to preserve the identity of an input image, e.g., color composition. The *identity loss* $L_{idt}(G_A)$ for G_A can be written as

$$L_{idt}(G_A) = \mathbb{E}_{b\sim\mathcal{B}}[|G_A(b) - b|]. \quad (3)$$

  The identity loss for G_B can be written similarly.

The full objective of CycleGAN is formed by linearly combining the three loss functions as

$$\begin{aligned} &L(G_A, G_B, D_A, D_B) \\ &= \lambda_1 L_{adv}(G_A, D_B) + \lambda_2 L_{adv}(G_B, D_A) \\ &+ \lambda_3 L_{cyc}(G_A, G_B) + \lambda_4 L_{idt}(G_A) + \lambda_5 L_{idt}(G_B), \end{aligned} \quad (4)$$

where $\lambda_1, \ldots, \lambda_5$ are a weight. Through this work, we set $\lambda = (\lambda_1, \ldots, \lambda_5) = (1, 1, 10, 5, 5)$ unless otherwise stated. Note that the discriminators are involved only in the adversarial losses.

3.2. Search space

We develop CycleGANAS that simultaneously searches neural network architectures for G_A, G_B, D_A, D_B. Considering the significant challenge in conducting architecture search from scratch, many NAS approaches restrict their search space to a combination of operations referred to as a *cell*, and construct a neural network architecture by stacking a few cells [4, 6]. We also follow the cell-based NAS approach, but different from the previous works, we use a much simpler cell structure and stack many of them to construct an architecture.

Motivated by the neural network architectures of CycleGAN, we design a ResNet-based cell for CycleGANAS. Our cell has only two operations, each of which followed by a normalization layer and an activation layer. We restrict the cell's operations by *cell type*, which can be either encoding e, residual r, or decoding d. Letting $\mathcal{S}^T$ be the set of possible operations for the cell type $T \in \{e, r, d\}$, we define the operation sets as

- $\mathcal{S}^e = \{$max pooling, avg pooling, Conv3x3, Conv4x4, Conv5x5, Conv7x7, DilConv3x3, DilConv5x5$\}$,
- $\mathcal{S}^r = \{$Conv3x3, Conv5x5, Conv7x7, DilConv3x3, DilConv5x5$\}$,
- $\mathcal{S}^d = \{$Nearest neighbor interpolation, Bi-linear interpolation, Transposed Conv3x3$\}$.

A cell C^T of type T has two operations from $\mathcal{S}^T$.

We build a generator G of N cells; starting from one encoding cell, followed by $N - 2$ residual cells, and ending

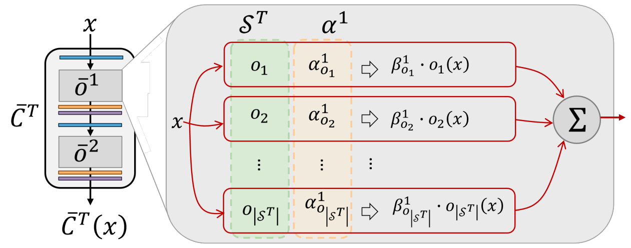


Figure 2. A super-network cell with two mixed operations $\bar{o}^1, \bar{o}^2$. The weights α of the mixed operations are trained by the gradient descent. After the architecture search, each cell is converted to an ordinary cell with two discrete operations.

with one decoding cell. Thus, we can represent a generator as a sequence of cells: $G = (C_1^e, C_2^r, \ldots, C_{N-1}^r, C_N^d)$. For a decoder D, we have it with a much simpler structure of 2 encoding cells: $D = (C_1^e, C_2^e)$. Note that a generator has the search space of size $|\mathcal{S}^e|^2 \cdot |\mathcal{S}^r|^{2(N-2)} \cdot |\mathcal{S}^d|^2$, where $|\cdot|$ denotes the cardinality. We will use $N = 11$ as a default setting for each generator, resulting in $8^2 \times 5^{18} \times 3^2 \approx 2.2 \times 10^{15}$. Similarly, each discriminator has the search space of size 8^4. Since there are two generators and two discriminators, the total search space size of CycleGANAS is 8.1×10^{37}, which is comparably large considering those of AutoGAN (10^5) and AdversarialNAS (10^{38}).

Element-wise searching in the huge space will demand prohibitively large computational cost. To avoid excessive computation, we adopt the idea of *mixed operation* from DARTS [20] and take the approach of *differentiable neural architecture search*. Basically, we convert the discrete search space into a continuous one by replacing an operation with a mixed one of multiple operations that admits the gradient-based search. To elaborate, consider a cell $\bar{C}^T$ that has two mixed operations $\bar{o}^1, \bar{o}^2$ in order, where each $\bar{o}^i$ with $i \in 1, 2$ is a combination of all possible operations, i.e.,

$$\bar{o}^i = \sum_{o \in \mathcal{S}^T} \beta_o^i \cdot o,$$

where $\beta_o^i = \frac{\exp(\alpha_o^i)}{\sum_{p \in \mathcal{S}^T} \exp(\alpha_p^i)}$ and α_o^i is the weight of operation $o \in \mathcal{S}^T$, as shown in Fig. 2. Also, suppose that we build a super-network by stacking the cells with mixed operations. Then we can optimize all α^i's of the super-network's cells through gradient descent. When the search finishes, we construct a discrete architecture by selecting the highest-weight operations in the super-network, i.e., each cell $\bar{C}^T = \{\bar{o}^1, \bar{o}^2\}$ of the super-network is converted to an ordinary cell $C^T = \{o^1, o^2\}$, where $o^i = \arg\max_{o \in \mathcal{S}^T} \alpha_o^i$.

Architecture search with the super-network demands a substantial amount of memory and time to take into account all the possible operations, which can be burdensome in practice. To this end, it is common that, for the search, one uses the super-network with reduced hidden dimension, and after the search, builds the discrete architecture with restored hidden dimension [4]. We also apply the technique of the *hidden dimension reduction* to CycleGANAS for the architecture search. For example, during the search, a (super-network) generator of our CycleGANAS takes an image input of $256 \times 256 \times 3$, and encodes it to a tensor of $64 \times 64 \times 64$, whose hidden dimension size is smaller than that of the original CycleGAN's encoder output ($64 \times 64 \times 256$). Passing through the residual blocks, the generator decodes it back to the shape of $256 \times 256 \times 3$. Once the search completes, from the trained super-network, we construct a discrete architecture, whose encoder output has the shape of $64 \times 64 \times H$, where the hidden dimension H is a hyperparameter.

Algorithm 1 One-step CycleGANAS.

```
Initialize weights: $\theta_{G_A}, \theta_{G_B}, \theta_{D_A}, \theta_{D_B}$
Initialize differentiable architecture: $G, G, D, D$
Input: Unpaired dataset $\mathcal{A}, \mathcal{B}$
Output: architecture of $G_A, G_B, D_A, D_B$
 1: for each search epoch do
 2:     for each iteration do
 3:         sample $a \in \mathcal{A}$ and $b \in \mathcal{B}$
 4:         /* Forward path */
 5:         Compute $L_{adv}(G_A, D_B), L_{adv}(G_B, D_A)$
 6:         Compute $L_{cyc}(G_A, G_B), L_{cyc}(G_B, G_A)$
 7:         Compute $L_{idt}(G_A), L_{idt}(G_B)$
 8:         Compute $L(G_A, G_B, D_A, D_B)$
 9:         /* Backward path */
10:         Update $\theta_{G_A}, \theta_{G_B}$ from $L(\cdot)$
11:         Update $\theta_{D_A}, \theta_{D_B}$ from corresponding $L_{adv}(\cdot)$
12:     end for
13: end for
14: return architecture of $G_A, G_B, D_A, D_B$
```

3.3. Optimization process

Let α denote the vector of all architecture weights, and let w denote the vector of neural network weights. Note that if α changes, the optimal w also changes, and vice versa. Typically, a NAS for GANs takes a bi-level optimization process to optimize α and w, which is equivalent to iteratively apply the following two equations in sequence,

$$\begin{aligned} \alpha^* &= \arg\min_\alpha L_{val}(\alpha, w^*), \\ w^* &= \arg\min_w L_{train}(\alpha^*, w), \end{aligned} \tag{5}$$

where L_{val} is a loss with the validation dataset and L_{train} is a loss with the training dataset. For example, AutoGAN has the FID score for L_{val} and the adversarial loss for L_{train}, and AdversarialNAS has the adversarial loss for both L_{val} and L_{train}.

We highlight that the bi-level optimization ([5]) requires the computation of L_{train} and L_{val} from two exclusive datasets for stability. AutoGAN and AdversarialNAS can easily accomplish this since the input to the GANs generator is a random sample from Gaussian noise. For CycleGAN, however, the input to a generator is not a random sample but an image from a subdataset, and thus each input subdataset has to be divided into two for the bi-level optimization. Further, rotating the role of divided subdatasets will be necessary; otherwise, the architecture optimizer and weight optimizer will only observe a portion of the entire subdataset, leading to a performance degradation. As a result, the bi-level optimization process of CycleGANAS requires a more intricate approach to dividing the dataset and rotating their roles.

We simplify the optimization process by developing a single-level optimization process of α and w. We use the entire input subdataset for the joint optimization of

$$(\alpha^*, w^*) = \arg\min_{\alpha,w}\ L_{train}(\alpha, w), \tag{6}$$

where $L_{train} = L(G_A, G_B, D_A, D_B)$ is the CycleGAN loss ([4]). Note that the single-level optimization not only eliminates the need for dividing the input subdataset but also reduces the number of optimization steps by half, in comparison to the bi-level optimization. This results in an accelerated learning process. We denote CycleGANAS with the single-level optimization by *one-step* CycleGANAS, whose detailed process is shown in Algorithm [1], where $\theta = (\alpha, w)$.

4. Experiments

We evaluate CycleGANAS with several unpaired datasets, e.g. maps, facades, apple2orange, horse2zebra, summer2winter, and iphone2dslr-flower.Each dataset has two subdatasets, each of which may have a different number of images, and all the images are of the same shape $3 \times 256 \times 256$. For certain datasets, there exists an imbalance in the data, and the image translations between the two subdatasets have different levels of difficulty. For example, in the horse2zebra dataset, the number of pixels taken by zebra is more than twice as many as those taken by horse [23]. This data imbalance demands asymmetric capability of neural networks, in particular, of the two generators. In general, a more challenging translation task, e.g., zebra-to-horse, is likely to necessitate a larger generator model[1]. We demonstrate that CycleGANAS not only successfully search high-performance architectures, but also, in the presence of dataset imbalance, naturally adopts asymmetric architectures, effectively mitigating the issue.

Throughout our experiments, we mostly use the configuration of CycleGAN [44], e.g., batch size 1 and instance

[1]We remark that the original CycleGAN is symmetric – its two generators have the same architecture, and the same for the two discriminators.

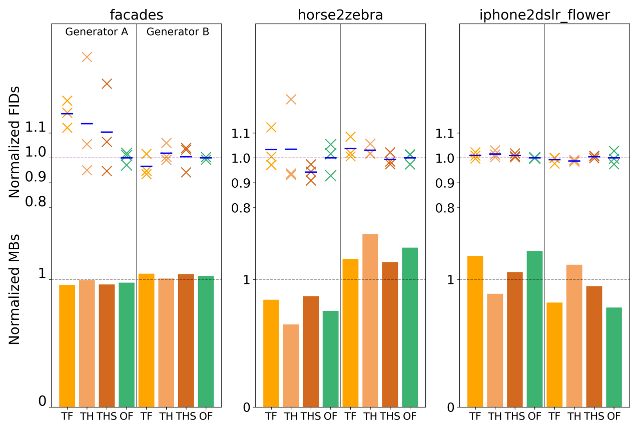


Figure 3. Performance of the four variants of CycleGANAS with three datasets, in terms of the generator model size and FID scores. The model sizes are normalized w.r.t. that of the original CycleGAN's generator, and the FIDs w.r.t. the average FIDs of one-step CycleGANAS (OF). For each architecture outcome, we repeat the weight training 3 times, and mark the FIDs by cross and the average by blue dash.

normalization. For one-step CycleGANAS of Algorithm [1], we use Adam optimizer with learning rate of $\alpha = 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, and set the maximum search epoch to 400. All our experiments are based on Python 3.8, CUDA 11.3, CuDNN 8.2.0, and the learning frameworks are implemented with PyTorch. To evaluate architectures, we use the FID score estimation provided by the clean-fid project [24]. Since the FID score is stochastic for an architecture [11], we repeat the weight training several times and select the one with the best (lowest) FID score.

We present the impact of the optimization method on performance, and evaluate the performance of CycleGANAS using diverse datasets and illustrate its response to data imbalance.

4.1. Bi-level vs single-level optimization

Besides the single-level optimization, we can incorporate our CycleGANAS framework with the previous bi-level optimization, which is denoted by *two-step* CycleGANAS. Numerous variations of the two-step CycleGANAS can emerge based on how the subdataset is divided and rotated. We consider the following three variants of two-step CycleGANAS.

- (TF) Two-step CycleGANAS with the full subdataset: Without dividing the subdatasets, we use the bi-level optimization of w using $\mathcal{A}$ and α using $\mathcal{B}$.
- (TH) Two-step CycleGANAS with evenly halved subdatasets: For $\mathcal{A}$, we assign all its images to $\mathcal{A}_1$ or $\mathcal{A}_2$ at random such that $|\mathcal{A}_1| = |\mathcal{A}_2|$. Similarly, we have $\mathcal{B} \rightarrow \mathcal{B}_1, \mathcal{B}_2$. Then, we use the bi-level optimization of w

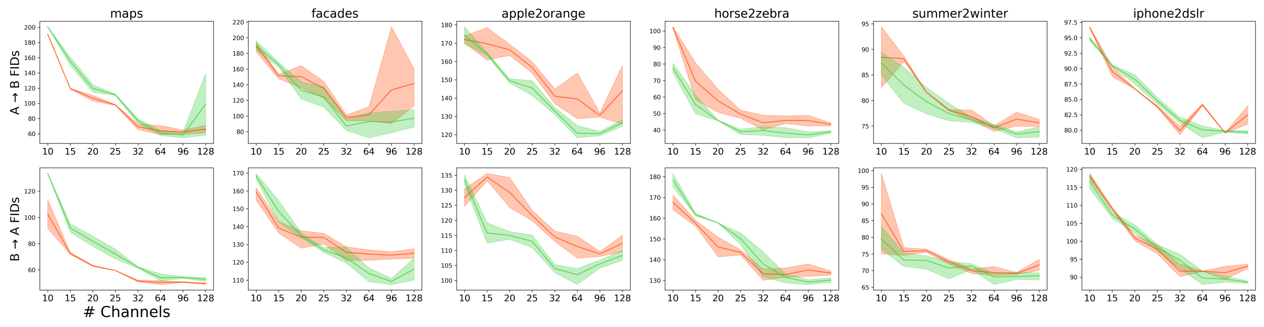


Figure 4. Performance of CycleGAN (red) and the architecture outcomes of CycleGANAS (green) with different hidden dimension sizes. We evaluate each architecture 3 times. The mean (thick line) and range (light shade) of FIDs are shown. The outcomes of CycleGANAS achieve comparable performance or even outperforms CycleGAN.

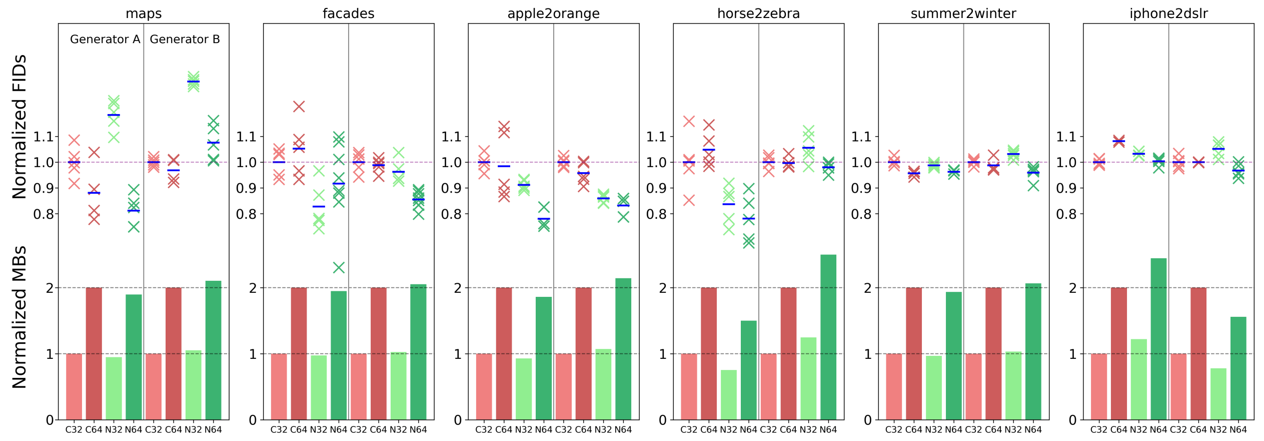


Figure 5. Performance of CycleGAN (red) and the architecture outcomes of CycleGANAS (green). Two CycleGAN architectures with $H = 32$ ($C32$) and $H = 64$ ($C64$) are used. We also use two versions of CycleGANAS architecture outcomes ($N32$ and $N64$), such that their total model sizes are roughly the same as $C32$ and $C64$, respectively. Comparing their performance over 6 datasets, we can observe that the architectures searched by CycleGANAS outperform CycleGAN, in particular, for the larger model size.

using $\mathcal{A}_1, \mathcal{B}_1$ and α using $\mathcal{A}_2, \mathcal{B}_2$ throughout the search.

- (THS) Two-step CycleGANAS with evenly halved subdatasets and swapping: It is similar to Two-half, except that after certain epoch, we swap the role of the halved subdatasets. Our intention is for the optimizers of w and α to have visibility over all images within the entire subdatasets. In our experiments, we do the bi-level optimization of w using $\mathcal{A}_1, \mathcal{B}_1$ and α using $\mathcal{A}_2, \mathcal{B}_2$ for the first 200 epochs, and then do the optimization of w using $\mathcal{A}_2, \mathcal{B}_2$ and α using $\mathcal{A}_1, \mathcal{B}_1$ afterward.

Unlike the two-step CycleGANAS, our one-step CycleGANAS features a simpler optimization process and necessitates only half the iterations.

- (OF) One-step CycleGANAS with the full subdatasets. We jointly optimize w and α using $\mathcal{A}, \mathcal{B}$.

We search CycleGAN architectures using the above four schemes with three different datasets of facades, horse2zebra, and iphone2dslr. Fig. 3 shows the model size of the generators and FIDs of their searched architectures, which are normalized by the model size of the original CycleGAN's generator, and by the minimum FIDs of one-step OF CycleGANAS, respectively.

In the facades dataset, the model sizes are all similar and the best (lowest) FID is achieved by two-step THS. On the other hand, one-step OF achieves the least variation of FIDs and the lowest average, which implies that one-step OF performs well in a stable manner. In horse2zebra with data imbalance, all schemes effectively discover asymmetric architectures, with a focus on enhancing the generator for zebra-to-horse translation, although the extent of asymmetry varies among the schemes. Also we can observe that two-step THS and one-step OF outperform two-step TH. We conjecture that it is due to the limited visibility of two-step TH's optimizers to the subdatasets. In iphone2dslr, CycleGAN is asked to increase or decrease the image resolution, and it is clear that the resolution-increasing task is more difficult than the other. Similar to the horse2bebra case, most schemes except two-step TH provide asymmetric architectures enhancing the generator for iphone-to-dslr translation. Further, one-step OF achieves the best FID scores. Another interesting observation is that the reversely asymmetric architectures searched by two-step TH also achieve comparable performance.

Overall, two-step TF/TH/THS schemes exhibit a high variation depending on the translation task, while one-step OF achieves a competitive, close-to-best result in a stable manner. Further it is worth highlighting that one-step OF completes the optimization process in half the iterations of two-step methods. Given its performance and simplicity, we establish it as the default choice for CycleGANAS. Henceforth, when referring to CycleGANAS, we are referring to one-step CycleGANAS.

4.2. Performance evaluation

We demonstrate the search capability of CycleGANAS. Over 6 datasets, we evaluate the architectures searched by CycleGANAS in comparison with the original CycleGAN. We vary the hidden dimension size H for both CycleGAN and the search outcomes of CycleGANAS to see their achievable performance. For each architecture, we repeat its evaluation 3 times.

Fig. 4 shows the mean (thick line) and range (light shade) of the FID scores of CycleGAN (red) and the architecture outcomes of CycleGANAS (green). Overall, in terms of the lowest FIDs, the outcomes of CycleGANAS demonstrate comparable performance (for maps and iphone2dslr) or even outperform CycleGAN (for facades, apple2orange, horse2zebra, and summer2winter). Another interesting observation is that a larger H does not imply a better performance, and there is an optimal value, usually between 32 and 96, depending on the dataset and architecture.
Remark: the architecture of CycleGAN and the outcomes of CycleGANAS have convolutions of different filter sizes, leading to differences in their model size in bytes even when they have the same hidden dimension size H.

Next we conduct a more direct performance comparison between CycleGANAS outcomes and CycleGAN by ensuring their model sizes are equal. From the previous experiment results, we train two CycleGAN architectures with $H = 32$ and 64, denoted by $C32$ and $C64$, respectively. For the architectures searched by CycleGANAS, we adjust the hidden dimension sizes accordingly such that the total model size roughly equals that of $C32$ or $C64$. Note that we configure the two generators searched by CycleGANAS to have the same hidden dimension size H, and as a result, depending on their chosen convolution operations, they will have a different model size, leading to asymmetric architectures.

Fig. 5 shows the experiment results over 6 datasets in terms of generator model sizes and FIDs. All the model sizes are normalized with respect to that of $C32$ generator (11.378 MB), and the FIDs with respect to $C32$'s mean FIDs. $N32$ and $N64$ denote the architectures searched by CycleGANAS with normalized H with respect to $C32$ and $C64$, respectively. Overall, CycleGANAS successfully finds good architectures for most datasets (except maps). Its architectures have comparable (for summer2winter and iphone2dslr) or better performance (for facades, apple2orange, horse2zebra) than the CycleGAN counterpart. In particular, with the larger model size, the outcomes of CycleGANAS outperform the original CycleGAN, by up to 30% in $\mathcal{A}$-to-$\mathcal{B}$ and 10% in $\mathcal{B}$-to-$\mathcal{A}$. Further we can also observe that, for the datasets with imbalance (horse2zebra and iphone2dslr), CycleGANAS provides asymmetric generator architectures accordingly, and for the others, it yields the generators of similar model size.

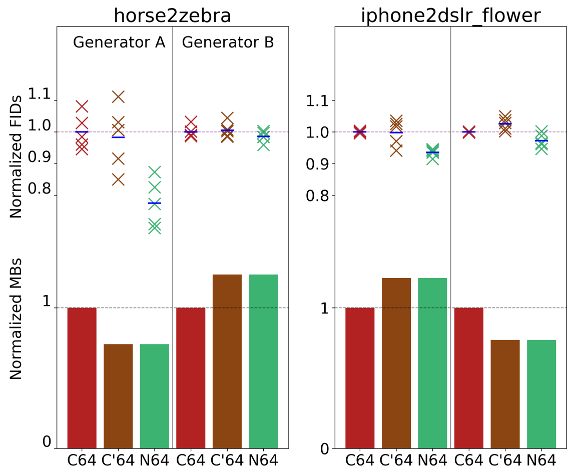


Figure 6. Performance of the original CycleGAN with $H = 64$ ($C64$), its asymmetric variant by scaling the hidden dimension ($C'64$), and the architecture outcome of CycleGANAS ($N64$). Scaling the hidden dimension does not improve the performance, while the architecture search does.

4.3. Architecture search vs hidden dimension scaling

The two generators of the original CycleGAN have the same architecture. In contrast, the architectures searched by CycleGANAS may have asymmetric structure that naturally comes from the dataset when it selects appropriate operations in $\mathcal{S}^e, \mathcal{S}^r, \mathcal{S}^d$ for each of the two generators. One may argue that the crucial factor to the performance is the asymmetric model size of the two generators, rather than the selection of their operations. We show that this is not the case, and the operation selection (i.e., architecture search) is of great importance.

For the datasets of horse2zebra and iphone2dslr, we consider the symmetric architectures of $C64$ (CycleGAN) and the asymmetric architectures $N64$ searched by CycleGANAS. Note that their total model sizes are the same. Then we build another asymmetric architectures $C'64$ by scaling the hidden dimension of $C64$ generators. To elaborate, we scale the hidden dimensions of two generators of $C64$ such that their model sizes equal to those of $N64$'s two generators, respectively. As a result, each generator of $C'64$ has the same operations as the $C64$ counterpart and the same model size as the $N64$ counterpart.

Fig. 6 shows the model sizes and the FIDs of 5 evaluations, normalized w.r.t. the generator model size and mean FIDs of $C64$. It is confirmed that the generators of $C'64$ have the same model sizes as those of $N64$. We can observe that the mean FIDs ($\mathcal{A}$-to-$\mathcal{B}$, $\mathcal{B}$-to-$\mathcal{A}$) of $C'64$ are $(45.19, 132.87)$ for horse2zebra and $(84.03, 93.41)$ for iphone2dslr-flower, which are similar to $C64$'s $(45.78, 133.42)$ and $(84.12, 91.70)$, respectively. The searched asymmetric architectures of $N64$ achieve the lowest FIDs of $(38.05, 131.44)$ and $(80.07, 89.75)$, respectively. This suggests that, when dealing with data imbalance, simply scaling the hidden dimension might not yield substantial benefits; instead, identifying suitable operations becomes of paramount importance.

5. Conclusion

We develop a NAS framework for CycleGAN that carries out unpaired image-to-image translation task. Compared to NAS for GANs, NAS for CycleGAN is more challenging due to the limited samples from dataset, multiple neural networks and their involvement in learning, and data imbalance.

We design a framework, called CycleGANAS, that can search CycleGAN architectures of two generators and two discriminators, simultaneously. For flexible and practical search, we build architectures by stacking many simple ResNet-based cells, take the approach of differentiable search through super-networks, and apply the hidden dimension reduction. We further reduce the computational complexity and stabilize the search with the single-level optimization, enabling CycleGANAS to effectively explore a vast search space of size 8.1×10^{36}.

Our experiments demonstrate that CycleGANAS effectively discovers high-performance architectures that either match or surpass the performance of the original CycleGAN. Furthermore, CycleGANAS successfully addresses the data imbalance by individually searching for distinct generator architectures for each translation direction, thereby regulating the model ratio.

References

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2017. 2

[2] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. 2

[3] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. 2

[4] Chen Gao, Yunpeng Chen, Si Liu, Zhenxiong Tan, and Shuicheng Yan. Adversarialnas: Adversarial neural architecture search for gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3, 4

[5] Yang Gao, Rita Singh, and Bhiksha Raj. Voice impersonation using generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2506–2510, 2018. 2

[6] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 3

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014. 1

[8] David Haws and Xiaodong Cui. Cyclegan bandwidth extension acoustic modeling for automatic speech recognition. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6780–6784, 2019. 1

[9] Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212: 106622, 2021. 2

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[11] Haesung Jo and Changhee Joo. Autogan-dsp: Stabilizing gan architecture search with deterministic score predictors. Forthcoming, 2023. 5

[12] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. 2

[13] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019. 1

[14] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon. Kim. Learning to discover cross-domain rela-

tions with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 1857–1865. JMLR.org, 2017. 2

[15] Yong-Hoon Kwon and Min-Gyu Park. Predicting future frames using retrospective cycle gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[16] Liam Li, Mikhail Khodak, Nina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search. In *International Conference on Learning Representations*, 2021. 2

[17] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[18] Yu Li, Sheng Tang, Rui Zhang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Asymmetric gan for unpaired image-to-image translation. *IEEE Transactions on Image Processing*, 28(12):5881–5896, 2019. 1

[19] Tingting Liang, Yongtao Wang, Zhi Tang, Guosheng Hu, and Haibin Ling. Opanas: One-shot path aggregation network architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10195–10203, 2021. 2

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1, 2, 4

[21] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 1

[22] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *Proceedings of the 35th International Conference on Machine Learning*, pages 3481–3490. PMLR, 2018. 1

[23] Taesung Park. *Machine Learning for Deep Image Synthesis*. PhD thesis, EECS Department, University of California, Berkeley, 2021. 5

[24] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022. 5

[25] Wei Peng, Xiaopeng Hong, Haoyu Chen, and Guoying Zhao. Learning graph convolutional network for skeleton-based human action recognition by neural searching. *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, 2020. 2

[26] Hieu Pham, Melody Y. Guan, Barret Zoph, and Jeff Le, Quoc V. andDean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 2

[27] AJ Piergiovanni, Anelia Angelova, Alexander Toshev, and Michael S. Ryoo. Evolving space-time neural architectures for videos. In *ICCV*, 2019. 2

[28] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. Mirrorgan: Learning text-to-image generation by redescription. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[29] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *International Conference on Learning Representations*, 2021. 2

[30] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *International Conference on Learning Representations*, 2021. 2

[31] Divya Saxena, Jiannong Cao, Jiahao Xu, and Tarun Kulshrestha. Re-gan: Data-efficient gans training via architectural reconfiguration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16230–16240, 2023. 2

[32] Han Shu, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunjing Xu, Qi Tian, and Chang Xu. Co-evolutionary compression for unpaired image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[33] Hanchao Wang and Jun Huan. Agan: Towards automated design of generative adversarial networks. *ArXiv*, abs/1906.11080, 2019. 1, 2

[34] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018. 2

[35] Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10293–10301, 2021. 2

[36] Yun Yi, Haokui Zhang, Wenze Hu, Nannan Wang, and Xiaoyu Wang. Nar-former: Neural architecture representation learning towards holistic attributes prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7715–7724, 2023. 2

[37] H. You, Y. Cheng, T. Cheng, C. Li, and P. Zhou. Bayesian cycle-consistent generative adversarial networks via marginalizing latent sampling. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020. 1

[38] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[39] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017. 2

[40] Tian Yuan, Wang Qin, Huang Zhiwu, Li Wen, Dai Dengxin, Yang Minghao, Wang Jun, and Fink Olga. Off-policy reinforcement learning for efficient and effective gan architecture search. In *The European Conference on Computer Vision (ECCV)*, 2020. 2

[41] 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 *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[42] Zizhao Zhang, Lin Yang, and Yefeng Zheng. Translating and segmenting multimodal medical volumes with cycle- and shape-consistency generative adversarial network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9242–9251, 2018. 2

[43] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. BayesNAS: A Bayesian approach for neural architecture search. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7603–7613. PMLR, 2019. 2

[44] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 3, 5

[45] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems 30*, pages 465–476. Curran Associates, Inc., 2017. 2

[46] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. 2