Conference Paper

Off-Policy Reinforcement Learning for Efficient and Effective GAN Architecture Search

Author(s):
Tian, Yuan; Wang, Qin; Huang, Zhiwu; Li, Wen; Dai, Dengxin; Yang, Minghao; Wang, Jun; Fink, Olga

ETH Library

Off-Policy Reinforcement Learning for Efficient and Effective GAN Architecture Search

Yuan Tian[1*], Qin Wang[1*], Zhiwu Huang[1], Wen Li[2], Dengxin Dai[1],
Minghao Yang[3], Jun Wang[4], and Olga Fink[1]

[1] ETH Zürich
{yutian, qwang, ofink}@ethz.ch {zhiwu.huang, dai}@vision.ee.ethz.ch
[2] UESTC   [3] Navinfo Europe   [4] University College London
liwenbnu@gmail.com minghao.yang@navinfo.eu junwang@cs.ucl.ac.uk

Abstract. In this paper, we introduce a new reinforcement learning (RL) based neural architecture search (NAS) methodology for effective and efficient generative adversarial network (GAN) architecture search. The key idea is to formulate the GAN architecture search problem as a Markov decision process (MDP) for smoother architecture sampling, which enables a more effective RL-based search algorithm by targeting the potential global optimal architecture. To improve efficiency, we exploit an off-policy GAN architecture search algorithm that makes efficient use of the samples generated by previous policies. Evaluation on two standard benchmark datasets (i.e., CIFAR-10 and STL-10) demonstrates that the proposed method is able to discover highly competitive architectures for generally better image generation results with a considerably reduced computational burden: 7 GPU hours. Our code is available at https://github.com/Yuantian013/E2GAN.

Keywords: Neural Architecture Search, Generative Adversarial Networks, Reinforcement Learning, Markov Decision Process, Off-policy

1   Introduction

Generative adversarial networks (GANs) have been successfully applied to a wide range of generation tasks, including image generation [12,3,1,53,15], text to image synthesis [44,62,40] and image translation [25,7], to name a few. To further improve the generation quality, several extensions and further developments have been proposed, ranging from regularization terms [4,14], progressive training strategy [26], utilizing attention mechanism [59,61], and to new architectures [27,3].

While designing favorable neural architectures of GANs has made great success, it typically requires a large amount of time, effort, and domain expertise. For instance, several state-of-the-art GANs [27,3] design appreciably complex

---

[*] Equal contribution.

generator or discriminator backbones for better generating high-resolution images. To alleviate the network engineering pain, an efficient automated architecture searching framework for GAN is highly needed. On the other hand, Neural architecture search (NAS) has been applied and proved effective in discriminative tasks such as image classification [30] and segmentation [33]. Encouraged by this, AGAN [52] and AutoGAN [11] have introduced neural architecture search methods for GAN based on reinforcement learning (RL), thereby enabling a significant speedup of architecture searching process.

Similar to the other architecture search tasks (image classification, image segmentation), recently proposed RL-based GAN architecture search method AGAN [52] optimized the entire architecture. Since the same policy might sample different architectures, it is likely to suffer from noisy gradients and a high variance, which potentially further harms the policy update stability. To circumvent this issue, multi-level architecture search (MLAS) has been used in Auto-GAN [52], and a progressive optimization formulation is used. However, because optimization is based on the best performance of the current architecture level, this progressive formulation potentially leads to a local minimum solution.

To overcome these drawbacks, we reformulate the GAN architecture search problem as a Markov decision process (MDP). The new formulation is partly inspired by the human-designed Progressive GAN [26], which has shown to improve generation quality progressively in intermediate outputs of each architecture cell. In our new formulation, a sequence of decisions will be made during the entire architecture design process, which allows state-based sampling and thus alleviates the variance. In addition, as we will show later in the paper, by using a carefully designed reward, this new formulation also allows us to target effective global optimization over the entire architecture.

More importantly, the MDP formulation can better facilitate off-policy RL training to improve data efficiency. The previously proposed RL-based GAN architecture search methods [11,52] are based on on-policy RL, leading to limited data efficiency that results in considerably long training time. Specifically, on-policy RL approach generally requires frequent sampling of a batch of architectures generated by current policy to update the policy. Moreover, new samples are required to be collected for each gradient step, while the previous batches are directly disposed. This quickly becomes very expensive as the number of gradient steps and samples increases with the complexity of the task, especially in the architecture search tasks. by comparison, off-policy reinforcement learning algorithms make use of past experience such that the RL agents are enabled to learn more efficiently. This has been proven to be effective in other RL tasks, including legged locomotion [32] and complex video games [38]. However, exploiting off-policy data for GAN architecture search poses new challenges. Training the policy network inevitably becomes unstable by using off-policy data, because these training samples are systematically different from the on-policy ones. This presents a great challenge to the stability and convergence of the algorithm [2]. Our proposed MDP formulation can make a difference here.

By allowing state-based sampling, the new formulation alleviates this instability, and better supports the off-policy strategy.

The contributions of this paper are two-fold:

1. We reformulate the problem of neural architecture search for GAN as an MDP for smoother architecture sampling, which enables a more effective RL-based search algorithm and potentially more global optimization.
2. We propose an efficient and effective off-policy RL NAS framework for GAN architecture search (E^{2}GAN), which is 6 times faster than existing RL-based GAN search approaches with competitive performance.

We conduct a variety of experiments to validate the effectiveness of E^{2}GAN. Our discovered architectures yield better results compared to RL-based competitors. E^{2}GAN is efficient, as it is able to to find a highly competitive model within **7 GPU hours**.

2   Related Work

Reinforcement Learning Recent progress in model-free reinforcement learning (RL) [49] has fostered promising results in many interesting tasks ranging from gaming [37,48], to planning and control problems [23,31,58,18,6,19] and even up to the AutoML [65,41,34]. However, model-free deep RL methods are notoriously expensive in terms of their sample complexity. One reason of the poor sample efficiency is the use of on-policy reinforcement learning algorithms, such as trust region policy optimization (TRPO) [46], proximal policy optimization(PPO) [47] or REINFORCE [56]. On-policy learning algorithms **require new samples generated by the current policy for each gradient step**. On the contrary, off-policy algorithms aim to reuse past experience. Recent developments of the off-policy reinforcement learning algorithms, such as soft Actor-Critic (SAC) [17], have demonstrated substantial improvements in both performance and sample efficiency in previous on-policy methods.

Neural architecture search Neural architecture search methods aim to automatically search for a good neural architecture for various tasks, such as image classification [30] and segmentation [33], in order to ease the burden of hand-crafted design of dedicated architectures for specific tasks. Several approaches have been proposed to tackle the NAS problem. Zoph and Le [65] proposed a reinforcement learning-based method that trains an RNN controller to design the neural network [65]. Guo et al. [16] exploited a novel graph convolutional neural networks for policy learning in reinforcement learning. Further successfully introduced approaches include evolutionary algorithm based methods[43], differentiable methods [35] and one-shot learning methods [5,35]. Early works of RL-based NAS algorithms [57,41,65,34] proposed to optimize the entire trajectory (i.e., the entire neural architecture). To the best of our knowledge, most of the previously proposed RL-based NAS algorithms used on-policy RL algorithms, such as REINFORCE or PPO, except [63] which uses Q-learning algorithm for NAS, which is a value-based method and only supports discrete state

space problems. For on-policy algorithms, since each update requires new data collected by the current policy and the reward is based on the internal neural network architecture training, the on-policy training of RL-based NAS algorithms inevitably becomes computationally expensive.

GAN Architecture Search Due to the specificities of GAN and their challenges, such as instability and mode collapse, the NAS approaches proposed for discriminative models cannot be directly transferred to the architecture search of GANs. Only recently, few approaches have been introduced tackling the specific challenges of the GAN architectures. Recently, AutoGAN has introduced a neural architecture search for GANs based on reinforcement learning (RL), thereby enabling a significant speedup of the process of architecture selection [11]. The AutoGAN algorithm is based on on-policy reinforcement learning. The proposed multi-level architecture search (MLAS) aims at progressively finding well-performing GAN architectures and completes the task in around 2 GPU days. Similarly, AGAN [52] uses reinforcement learning for generative architecture search in a larger search space. The computational cost for AGAN is comparably very expensive (1200 GPU days). In addition, AdversarialNAS [10] and DEGAS [9] adopted a different approach, i.e., differentiable searching strategy [35], for the GAN architecture search problem.

3   Preliminary

In this section, we briefly review the basic concepts and notations used in the following sections.

3.1   Generative Adversarial Networks

The training of GANs involves an adversarial competition between two players, a generator and a discriminator. The generator aims at generating realistic-looking images to 'fool' its opponent. Meanwhile, the discriminator aims to distinguish whether an image is real or fake. This can be formulated as a min-max optimization problem:

$$\min_G \max_D \mathbb{E}_{x \sim p_{real}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))], \qquad (1)$$

where G and D are the generator and discriminator parametrized by neural networks. z is sampled from random noise. x are the real and $G(z)$ are the generated images.

3.2   Reinforcement Learning

A Markov decision process (MDP) is a discrete-time stochastic control process. At each time step, the process is in some state s, and its associated decision-maker chooses an available action a. Given the action, the process moves into a new state s' at the next step, and the agent receives a reward.

An MDP could be described as a tuple (S, A, r, P, ρ), where S is the set of states that is able to precisely describe the current situation, A is the set of actions, $r(s, a)$ is the reward function, $P(s'|s, a)$ is the transition probability function, and $\rho(s)$ is the initial state distribution.

MDPs can be particularly useful for solving optimization problems via reinforcement learning. In a general reinforcement learning setup, an agent is trained to interact with the environment and get a reward from its interaction. The goal is to find a policy π that maximizes the cumulative reward $J(\pi)$:

$$J(\pi) = \mathbb{E}_{\tau \sim \rho_\pi} \sum_{t=0}^{\infty} r(s_t, a_t) \tag{2}$$

While the standard RL merely maximizes the expected cumulative rewards, the maximum entropy RL framework considers a more general objective [64], which favors stochastic policies. This objective shows a strong connection to the exploration-exploitation trade-off, and could also prevent the policy from getting stuck in local optima. Formally, it is given by

$$J(\pi) = \mathbb{E}_{\tau \sim \rho_\pi} \sum_{t=0}^{\infty} [r(s_t, a_t) + \beta \mathcal{H}(\pi(\cdot|s_t))], \tag{3}$$

where β is the temperature parameter that controls the stochasticity of the optimal policy.

4 Problem Formulation

4.1 Motivation

Given a fixed search space, GAN architecture search agents aim to discover an optimal network architecture on a given generation task. Existing RL methods update the policy network by using batches of entire architectures sampled from the current policy. Even though these data samples are only used for the current update step, the sampled GAN architectures nevertheless require tedious training and evaluation processes. The sampling efficiency is therefore very low resulting in limited learning progress of the agents. Moreover, the entire architecture sampling leads to a high variance, which might influence the stability of the policy update.

The key motivation of the proposed methodology is to stabilize and accelerate the learning process by step-wise sampling instead of entire-trajectory-based sampling and making efficient use of past experiences from previous policies. To achieve this, we propose to formulate the GAN architecture search problem as an MDP and solve it by off-policy reinforcement learning.

4.2 GAN Architecture Search formulated as MDP

We propose to formulate the GAN architecture search problem as a Markov decision process (MDP) which enables state-based sampling. It further boosts

the learning process and overcomes the potential challenge of a large variance stemming from sampling entire architectures that makes it inherently difficult to train a policy using off-policy data.

Formulating GAN architecture search problem as an MDP provides a mathematical description of architecture search processes. An MDP can be described as a tuple (S, A, r, P, ρ), where S is the set of states that is able to precisely describe the current architecture (such as the current cell number, the structure or the performance of the architectures), A is the set of actions that defines the architecture design of the next cell, $r(s, a)$ is the reward function used to define how good the architecture is, $P(s'|s, a)$ is the transition probability function indicating the training process, and $\rho(s)$ is the initial architecture. We define a cell as an architecture block we are using to search in one step. The design details of states, actions, and rewards is discussed in Section 5.

It is important to highlight that the formulation proposed in this paper has two main differences compared to previous RL methods for neural architecture search. Firstly, it is essentially different to the classic RL approaches for NAS [65], which formulate the task as an optimization problem over the entire trajectory/architecture. Instead, the MDP formulation proposed here enables us to do the optimization based on the disentangled steps of cell design. Secondly, it is also different to the progressive formulation used by AutoGAN [11], where the optimization is based on the best performance of the current architecture level and can potentially lead to a local minimum solution. Instead, the proposed formulation enables us to potentially conduct a more global optimization using cumulative reward without the burden of calculating the reward over the full trajectory at once. It is important to point out that the multi-level optimization formulation used in AutoGAN [11] does not have this property.

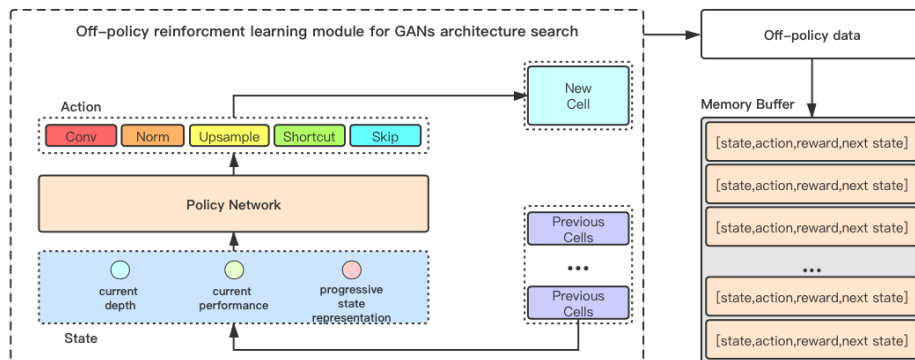## 5 Off-policy RL for GAN Architecture Search

In this section, we integrate off-policy RL in the GAN architecture search by making use of the newly proposed MDP formulation. We introduce several innovations to address the challenges of an off-policy learning setup.

The MDP formulation of GAN architecture search enables us to use off-policy reinforcement learning for a step-wise optimization of the entire search process to maximize the cumulative reward.

5.1 RL for GAN Architecture Search

Before we move on to the off-policy solver, we need to design the state, reward, and action to meet the requirements of both the GAN architecture design, as well as of the MDP formulation.

State MDP requires a state representation that can precisely represent the current network up to the current step. Most importantly, this state needs to be stable during training to avoid adding more variance to the training of the

Fig. 1. Overview of the proposed E^{2}GAN: the off-policy reinforcement learning module for GAN architecture search. The entire process comprises five steps: 1)The agent observes the current state s_t, which is designed as s=[Depth, Performance, Progressive state representation]. 2)The agent makes a decision a_t on how to design the cell added to previous cells according to the state information. a_t includes the skip options, upsampling operations, shortcut options, different types of convolution blocks and a normalization block 3)Progressively train the new architecture, obtain the reward r_t and the new state s_{t+1} information and then loop over it again. 4)Save the off-policy memory tuple $[s_t, a_t, r_t, s_{t+1}]$ into the memory buffer. 5)Sample a batch of data from the memory buffer to update the policy network.

policy network. The stability requirement is particularly relevant since the policy network relies on it to design the next cell. The design of the state is one of the main challenges we face when adopting off-policy RL to GAN architecture search.

Inspired by the progressive GAN [26], which has shown to improve generation quality in intermediate RGB outputs of each architecture cell, we propose a progressive state representation for GAN architecture search. Specifically, given a fixed batch of input noise, we adopt the average output of each cell as the progressive state representation. We down-sample this representation to impose a constant size across different cells. Note that there are alternative ways to encode the network information. For example, one could also deploy another network to encode the previous layers. However, we find the proposed design efficient and also effective.

In addition to the progressive state representation, we also use network performance (Inception Score / FID) and layer number to provide more information about the state. To summarize, the designed state s includes the depth, performance of the current architecture, and the progressive state representation.

Action Given the current state, which encodes the information about previous layers, the policy network decides on the next action. The action describes the architecture of one cell. For example, if we follow the search space used by AutoGAN [11], action will contain skip options, upsampling operations, shortcut
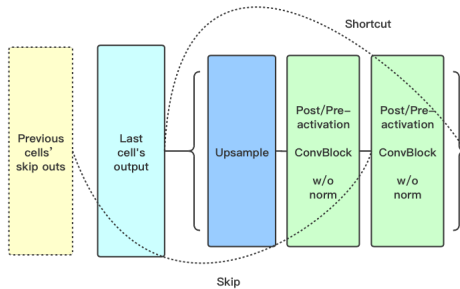
Fig. 2. The search space of a generator cell in one step. The search space is directly taken from AutoGAN [11].

options, different types of convolution blocks, and the normalization option, as shown in Figure 2.

This can then be defined as $a = [conv, norm, upsample, shortcut, skip]$. The action output by the agent will be carried out by a softmax classifier decoding into an operation. To demonstrate the effectiveness of our off-policy methods and enable a fair comparison, in all of our experiments, we use the same search space as AutoGAN [11], which means we search for generator cells, and the discriminator architecture is pre-designed and growing as the generator becomes deeper. More details on the search space are discussed in Section 6.

Reward We propose to design the reward function as the **performance improvement** after adding the new cell. In this work, we use both Inception Score (IS) and Frchet Inception Distance (FID) as the indicators of the network performance. Since IS score is progressive (the higher the better) and FID score is degressive (the lower the better), the proposed reward function can be formulated as:

$$R_t(s, a) = IS(t) - IS(t - 1) + \alpha(FID(t - 1) - FID(t)), \qquad (4)$$

where α is a factor to balance the trade-off between the two indicators. We use $\alpha = 0.01$ in our main experiments. The motivation behind using a combined reward is based on an empirical finding indicating that IS and FID are not always consistent with each other and can lead to a biased choice of architectures. A detailed discussion about the choice of indicators is provided in Section 7.

By employing the performance improvement in each step instead of only using performance as proposed in [11], RL can maximize the expected sum of rewards over the entire trajectory. This enables us to target the potential global optimal structure with the highest reward:

$$J(\pi) = \sum_{t=0} \mathbb{E}_{(s_t, a_t) \sim p(\pi)} R(s_t, a_t) = \mathbb{E}_{architecture \sim p(\pi)} IS_{final} - \alpha FID_{final}, \quad (5)$$

where IS_{final} and FID_{final} are the final scores of the entire architecture.

5.2   Off-policy RL Solver

The proposed designs of state, reward, and action fulfill the criteria of MDPs and makes it possible to stabilize the training using off-policy samples. We are now free to choose any off-policy RL solver to improve data efficiency.

In this paper, we apply the off-the-shelf soft actor-critic algorithm (SAC) [17], an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework, as the learning algorithm. It has demonstrated to be 10 to 100 times more data-efficient compared to any other on-policy algorithms on traditional RL tasks. In SAC, the actor aims at maximizing expected reward while also maximizing entropy. This increases training stability significantly and improves the exploration during training.

For the learning of the critic, the objective function is defined as:

$$J(Q) = \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\frac{1}{2}(Q(s,a) - Q_{target}(s,a))^2 \right] \tag{6}$$

where Q_{target} is the approximation target for Q :

$$Q_{target}(s,a) = Q(s,a) + \gamma Q_{target}(s', f(\epsilon, s')) \tag{7}$$

The objective function of the the policy network is given by:

$$J(\pi) = \mathbb{E}_{\mathcal{D}} \left[\beta[\log(\pi_\theta(f_\theta(\epsilon, s)|s))] - Q(s, f_\theta(\epsilon, s)) \right] \tag{8}$$

where π_θ is parameterized by a neural network f_θ, ϵ is an input vector consisting of Gaussian noise, and the $\mathcal{D} \doteq \{(s, a, s', r)\}$ is the replay buffer for storing the MDP tuples [38]. β is a positive Lagrange multiplier that controls the relative importance of the policy entropy versus the safety constraint.

5.3   Implementation of E^{2}GAN

In this section, we present the implementation details of the proposed off-policy RL framework E^{2}GAN. The training process is briefly outlined in Algorithm 1.

Agent Training Since we reformulated the NAS as a multi-step MDP, our agent will make several decisions in any trajectory $\tau = [(s_1, a_1), ...(s_n, a_n)]$. In each step, the agent will collect this experience $[s_t, a_t, r_t, s_{t+1}]$ in the memory buffer $\mathcal{D}$. Once the threshold of the smallest memory length is reached, the agent is updated using the Adam [28] optimizer via the objective function presented in Eq. 8 by sampling a batch of data from the memory buffer $\mathcal{D}$ in an off-policy way.

The entire search comprises two periods: the exploration period and the exploitation period. During the exploration period, the agent will sample any possible architecture. While in the exploitation period, the agent will choose the best architecture, in order to quickly stabilize the policy.

The exploration period lasts for 70% of iterations, and the exploitation takes 30% iterations. Once the memory threshold is reached, for every exploration step, the policy will be updated once. For every exploitation step, the policy will be updated 10 times in order to converge quickly.

Algorithm 1 Pseudo code for E^{2}GAN search

Input hyperparameters, learning rates α_{ϕ_Q},α_θ
Randomly initialize a Q network $Q(s, a)$ and policy network $\pi(a|s)$ with parameters ϕ_Q, θ and the Lagrange multipliers β,
Initialize the parameters of target networks with $\overline{\phi}_Q \leftarrow \phi_Q, \overline{\theta} \leftarrow \theta$
for each iteration **do**
    Reset the weight and cells of E^{2}GAN
    for each time step **do**
        if Exploration **then**
            Sample a_t from $\pi(s)$, add the corresponding cell to E^{2}GAN
        else if Exploitation **then**
            Choose the best a_t from $\pi(s)$ and add the corresponding cell to E^{2}GAN
        end if
        Progressively train the E^{2}GAN
        Observe s_{t+1}, r_t and store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    end for
    for each update step **do**
        Sample mini-batches of transitions from $\mathcal{D}$ and update Q and π with gradients
        Update the target networks with soft replacement:

$$\overline{\phi}_Q \leftarrow \tau\phi_Q + (1-\tau)\overline{\phi}_Q$$
$$\overline{\theta} \leftarrow \tau\theta + (1-\tau)\overline{\theta}$$

    end for
end for

Proxy Task We use a progressive proxy task in order to collect the rewards fast. When a new cell is added, we train the current full trajectory for one epoch and calculate the reward for the current cell. Within a trajectory, the previous cells' weights will be kept and trained together with the new cell. In order to accurately estimate the Q-value of each state-action pair, we reset the weight of the neural network after finishing the entire architecture trajectory design.

6   Experiments

6.1   Dataset

In this paper, we use the CIFAR-10 dataset [29] to evaluate the effectiveness and efficiency of the proposed E^{2}GAN framework. The CIFAR-10 dataset consists of 50,000 training images and 10,000 test images with a 32×32 resolution. We use its training set without any data augmentation technique to search for the architecture with the highest cumulative return for a GAN generator. Furthermore, to evaluate the transferability of the discovered architecture, we also adopt the STL-10 dataset [8] to train the network without any other data augmentation to make a fair comparison to previous works.

6.2   Search Space

To verify the effectiveness of the off-policy framework and to enable a fair comparison, we use the same search space as used in the AutoGAN experiments [11]. There are five control variables: 1)Skip operation, which is a binary value indicating whether the current cell takes a skip connection from any specific cell as its input. Note that each cell could take multiple skip connections from other preceding cells. 2)Pre-activation [21] and post-activation convolution block. 3)Three types of normalization operations, including batch normalization [24], instance normalization [51], and no normalization. 4)Upsampling operation which is standard in current image generation GAN, including bi-linear upsampling, nearest neighbor upsampling, and stride-2 deconvolution. 5)Shortcut operation.

6.3   Results

The generator architecture discovered by E^{2}GAN on the CIFAR-10 training set is displayed in Figure 3. For the task of unconditional CIFAR-10 image generation (no class labels used), several notable observations could be summarized:

Methods	Inception Score	FID	Search Cost (GPU days)
DCGAN [42]	$6.64 \pm .14$	-	*
Improved GAN [45]	$6.86 \pm .06$	-	*
LRGAN [60]	$7.17 \pm .17$	-	*
DFM [55]	$7.72 \pm .13$	-	*
ProbGAN [20]	7.75	24.60	*
WGAN-GP, ResNet [14]	$7.86 \pm .07$	-	*
Splitting GAN [13]	$7.90 \pm .09$	-	*
SN-GAN [36]	$8.22 \pm .05$	$21.7 \pm .01$	*
MGAN [22]	$8.33 \pm .10$	26.7	*
Dist-GAN [50]	-	$17.61 \pm .30$	*
Progressive GAN [26]	$\mathbf{8.80 \pm .05}$	-	*
Improv MMD GAN [54]	8.29	16.21	*
Random search-1 [11]	8.09	17.34	-
Random search-2 [11]	7.97	21.39	-
AGAN [52]	$8.29 \pm .09$	30.5	1200
AutoGAN-top1 [11]	$8.55 \pm .10$	12.42	2
AutoGAN-top2 [11]	$8.42 \pm .07$	13.67	2
AutoGAN-top3 [11]	$8.41 \pm .11$	13.68	2
E^{2}GAN-top1	$8.51 \pm .13$	**11.26**	0.3
E^{2}GAN-top2	$8.50 \pm .09$	12.96	0.3
E^{2}GAN-top3	$8.42 \pm .11$	12.48	0.3

Table 1. Inception score and FID score of unconditional image generation task on CIFAR-10. We achieve **a highly competitive FID of 11.26** compared to published works. We mainly compare our approach with RL-based NAS approaches: AGAN [52] and AutoGAN [11]. Architectures marked by (*) are manually designed.
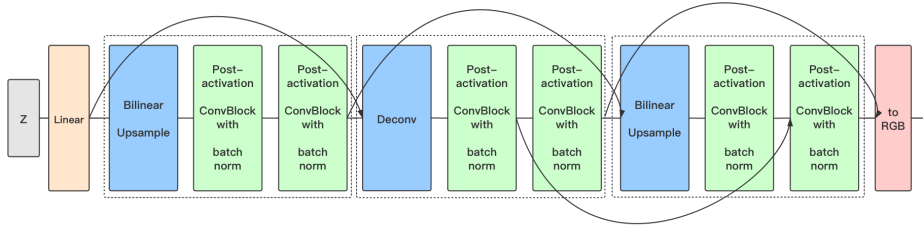
Fig. 3. The generator architecture discovered by E^{2}GAN on CIFAR-10.

* E^{2}GAN prefers post-activation convolution block to pre-activation convolution blocks. This finding is contrary to AutoGAN's preference, but coincides with previous experiences from human experts.
* E^{2}GAN prefers the use of batch normalization. This finding is also contrary to AutoGAN's choice, but is in line with experts' common practice.
* E^{2}GAN prefers bi-linear upsample to nearest neighbour upsample. This in theory provides finer upsample ability between different cells.

Our E^{2}GAN framework only takes about 0.3 GPU day for searching while the AGAN spends 1200 GPU days and AutoGAN spends 2 GPU days.

We train the discovered E^{2}GAN from scratch for 500 epochs and summarize the IS and FID scores in Table 1. On the CIFAR-10 dataset, our model achieves a **highly competitive FID 11.26** compared to published results by Auto-GAN [11], and hand-crafted GAN [42,45,60,55,20,14,13,36,22,54]. In terms of IS score, E^{2}GAN is also highly competitive to AutoGAN [11]. We additionally report the performance of the top2 and top3 architectures discovered in one search. Both have higher performance than the respective AutoGAN counterparts.

We also test the transferability of E^{2}GAN. We retrain the weights of the discovered E^{2}GAN architecture using the STL-10 training and unlabeled set for the

Methods	Inception Score	FID	Search Cost (GPU days)
D2GAN [39]	7.98	-	Manual
DFM [55]	$8.51 \pm .13$	-	Manual
ProbGAN [20]	$8.87 \pm .095$	46.74	Manual
SN-GAN [36]	$9.10 \pm .04$	$40.1 \pm .50$	Manual
Dist-GAN [50]	-	36.19	Manual
Improving MMD GAN [54]	9.34	37.63	Manual
AGAN [52]	$9.23 \pm .08$	52.7	1200
AutoGAN-top1 [11]	$9.16 \pm .13$	31.01	2
E^{2}GAN-top1	$\mathbf{9.51 \pm .09}$	**25.35**	0.3

Table 2. Inception score and FID score for the unconditional image generation task on STL-10. E^{2}GAN uses the discovered architecture on CIFAR-10. Performance is significantly better than other RL-based competitors.
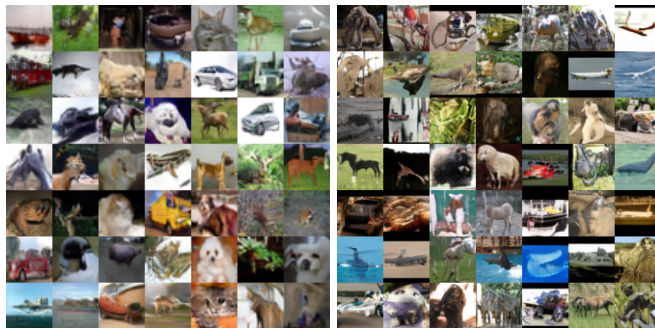
Fig. 4. The generated CIFAR-10(left) and STL-10(right) results of E²GAN which are randomly sampled without cherry-picking.

unconditional image generation task. **E²GAN achieves a highly-competitive performance on both IS (9.51) and FID (25.35)**, as shown in Table 2.

Because our main contribution is the new formulation and using off-policy RL for GAN architecture framework, we compare the proposed method directly with existing RL-based algorithms. We use the exact same searching space as AutoGAN, which does not include the search for a discriminator. As GAN training is an interactive procedure between generator and discriminator, one might expect better performance if the search is conducted on both networks. We report our scores using the exact same evaluation procedure provided by the authors of AutoGAN. The reported scores are based on the best models achieved during training on a 20 epoch evaluation interval. Mean and standard deviation of the IS score are calculated based on the 10-fold evaluation on 50,000 generated images. We additionally report the performance curve against training steps of E²GAN and AutoGAN for three runs in the supplementary material.

7   Discussion

7.1   Reward Choice: IS and FID

IS and FID scores are two main evaluation metrics for GAN. We conduct the ablation study of using different combinations. Specifically, IS only ($\alpha = 0$) and the combination of IS and FID ($\alpha = 0.01$) as the reward. Our agent successfully

Methods	Inception Score	FID	Search Cost (GPU days)
AutoGAN-top1 [11]	$8.55 \pm .1$	12.42	2
E²GAN(IS and FID as reward)	$8.51 \pm .13$	11.26	0.3
E²GAN(IS only as reward)	$\mathbf{8.81 \pm .11}$	15.64	0.1

Table 3. Performance on the unconditional image generation task for CIFAR-10 with different reward choices.
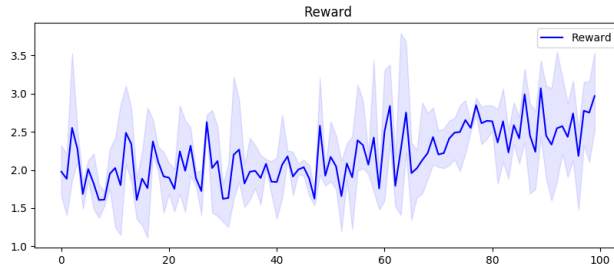
Fig. 5. Training curves on architecture searching. IS score on the proxy task against training time steps. E^2GAN shows relatively good stability and reproducibility.

discovered two different architectures. When only IS is used as the reward, the agent discovered a different architecture using only 0.1 GPU day. The searched architecture achieved state-of-the-art IS score of 8.86, as shown in Table 3., but a relatively plain FID of 15.78. This demonstrates the effectiveness of the proposed method, as we are encouraging the agent to find the architecture with a higher IS score. Interestingly, this shows that, at least in certain edge cases, the IS and FID may not always have a strong positive correlation. This finding motivates us to include FID in the reward. When both IS and FID are used as the reward signal, the discovered architecture performs well in term of both metrics. This combined reward takes 0.3 GPU days (compared to 0.1 GPU days of IS only optimization) because of the relatively expensive cost of FID computation.

7.2   Reproducibility

We train our agent over 3 different seeds. As shown in Figure 5, we observe that our agent steadily converged the policy in the exploitation period. E^2GAN can find similar architectures with relatively good performance on the proxy task.

8   Conclusion

We proposed a novel off-policy RL method, E^2GAN, to efficiently and effectively search for GAN architectures. We reformulated the problem as an MDP process, and overcame the challenges of using off-policy data. We first introduced a new progressive state representation. We additionally introduced a new reward, which allowed us to target the potential global optimization in our MDP formulation. The E^2GAN achieves state-of-the-art efficiency in GAN architecture searching, and the discovered architecture shows highly competitive performance.

Acknowledgement

References

1. Bao, J., Chen, D., Wen, F., Li, H., Hua, G.: Cvae-gan: fine-grained image generation through asymmetric training. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2745–2754 (2017)
2. Bhatnagar, S., Precup, D., Silver, D., Sutton, R.S., Maei, H.R., Szepesvári, C.: Convergent temporal-difference learning with arbitrary smooth function approximation. In: Advances in Neural Information Processing Systems. pp. 1204–1212 (2009)
3. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
4. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Neural photo editing with introspective adversarial networks. arXiv preprint arXiv:1609.07093 (2016)
5. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344 (2017)
6. Chao, M.A., Tian, Y., Kulkarni, C., Goebel, K., Fink, O.: Real-time model calibration with deep reinforcement learning. arXiv preprint arXiv:2006.04001 (2020)
7. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8789–8797 (2018)
8. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth International Conference on Artificial Intelligence and Statistics. pp. 215–223 (2011)
9. Doveh, S., Giryes, R.: Degas: Differentiable efficient generator search. arXiv preprint arXiv:1912.00606 (2019)
10. Gao, C., Chen, Y., Liu, S., Tan, Z., Yan, S.: Adversarialnas: Adversarial neural architecture search for gans. arXiv preprint arXiv:1912.02037 (2019)
11. Gong, X., Chang, S., Jiang, Y., Wang, Z.: Autogan: Neural architecture search for generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3224–3234 (2019)
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680 (2014)
13. Grinblat, G.L., Uzal, L.C., Granitto, P.M.: Class-splitting generative adversarial networks. arXiv preprint arXiv:1709.07359 (2017)
14. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems. pp. 5767–5777 (2017)
15. Guo, Y., Chen, Q., Chen, J., Wu, Q., Shi, Q., Tan, M.: Auto-embedding generative adversarial networks for high resolution image synthesis. IEEE Transactions on Multimedia **21**(11), 2726–2737 (2019)
16. Guo, Y., Zheng, Y., Tan, M., Chen, Q., Chen, J., Zhao, P., Huang, J.: Nat: Neural architecture transformer for accurate and compact architectures. In: Advances in Neural Information Processing Systems. pp. 737–748 (2019)
17. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290 (2018)
18. Han, M., Tian, Y., Zhang, L., Wang, J., Pan, W.: H infinity model-free reinforcement learning with robust stability guarantee. arXiv preprint arXiv:1911.02875 (2019)

19. Han, M., Zhang, L., Wang, J., Pan, W.: Actor-critic reinforcement learning for control with stability guarantee. arXiv preprint arXiv:2004.14288 (2020)
20. He, H., Wang, H., Lee, G.H., Tian, Y.: Probgan: Towards probabilistic gan with theoretical guarantees (2019)
21. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016)
22. Hoang, Q., Nguyen, T.D., Le, T., Phung, D.: Mgan: Training generative adversarial nets with multiple generators (2018)
23. Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., Hutter, M.: Learning agile and dynamic motor skills for legged robots. Science Robotics **4**(26), eaau5872 (2019)
24. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
25. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1125–1134 (2017)
26. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
27. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
28. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
29. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
30. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
31. Kumar, V., Gupta, A., Todorov, E., Levine, S.: Learning dexterous manipulation policies from experience and imitation. arXiv preprint arXiv:1611.05095 (2016)
32. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
33. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 82–92 (2019)
34. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–34 (2018)
35. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=S1eYHoC5FX`
36. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
37. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
38. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)

39. Nguyen, T., Le, T., Vu, H., Phung, D.: Dual discriminator generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2670–2680 (2017)
40. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2337–2346 (2019)
41. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268 (2018)
42. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
43. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 2902–2911. JMLR. org (2017)
44. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: International Conference on Machine Learning. pp. 1060–1069 (2016)
45. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems. pp. 2234–2242 (2016)
46. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning. pp. 1889–1897 (2015)
47. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
48. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms (2014)
49. Sutton, R.S., Barto, A.G., Williams, R.J.: Reinforcement learning is direct adaptive optimal control. IEEE Control Systems Magazine **12**(2), 19–22 (1992)
50. Tran, N.T., Bui, T.A., Cheung, N.M.: Dist-gan: An improved gan using distance constraints. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 370–385 (2018)
51. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
52. Wang, H., Huan, J.: Agan: Towards automated design of generative adversarial networks. arXiv preprint arXiv:1906.11080 (2019)
53. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8798–8807 (2018)
54. Wang, W., Sun, Y., Halgamuge, S.: Improving MMD-GAN training with repulsive loss function. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=HygjqjR9Km
55. Warde-Farley, D., Bengio, Y.: Improving generative adversarial networks with denoising feature matching (2016)
56. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3-4), 229–256 (1992)
57. Xie, S., Zheng, H., Liu, C., Lin, L.: Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926 (2018)

58. Xie, Z., Clary, P., Dao, J., Morais, P., Hurst, J., van de Panne, M.: Iterative reinforcement learning based design of dynamic locomotion skills for cassie. arXiv preprint arXiv:1903.09537 (2019)
59. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., He, X.: Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1316–1324 (2018)
60. Yang, J., Kannan, A., Batra, D., Parikh, D.: Lr-gan: Layered recursive generative adversarial networks for image generation. arXiv preprint arXiv:1703.01560 (2017)
61. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: International Conference on Machine Learning. pp. 7354–7363 (2019)
62. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 5907–5915 (2017)
63. Zhong, Z., Yan, J., Wu, W., Shao, J., Liu, C.L.: Practical block-wise neural network architecture generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2423–2432 (2018)
64. Ziebart, B.D.: Modeling purposeful adaptive behavior with the principle of maximum causal entropy (2010)
65. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. International Conference on Learning Representations (2017)