Off-Policy Reinforcement Learning for Efficient and Effective GAN Architecture Search (Supplementary Material)

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
[2] UESTC
[3] Navinfo Europe
[4] University College London

In this supplementary material, we present additional information of E^{2}GAN.

1 Network Implementation Details

Policy network and Q Network For E^{2}GAN, there are two networks: the policy network and the Q-network. Since we proposed to formulate the architecture design as an MDP and ensured a stable state representation, the agent can now make the decision on the next cell architecture based solely on the input state s. It is important to note that this alleviated the need of using an RNN to model the architecture states (as previously proposed in the AutoGAN's framework [2]). In this research, we adopt a fully-connected MLP with two hidden layers of 128 units as our policy network to provide action distributions. We utilize the same invertible squashing function technique as proposed in [4] to the output layer of the policy network. Similarly, for the Q-network, we use a fully-connected MLP with two hidden layers of 128 units, predicting the Q-value. ReLU activation function is used in all the hidden layers. The other hyper-parameters could be found in Table 1. The choices of SAC hyperparameters have been well studied in [3], we use the default values from the paper.

Hyperparameters	Value
Minibatch size	Full memory
Actor learning rate	3e-4
Critic learning rate	3e-4
Target smoothing coefficient(τ)	0.005
Discount(γ)	1
α	0.1
Optimizer	Adam [5]

Table 1. SAC Hyperparameters

* Equal contribution.

Discriminator Network We follow AutoGAN and only search for the generator. We use the same discriminator setup which adopts the multi-level architecture search (MLAS), where the corresponding discriminator grows progressively. This discriminator was manually designed by the authors, and adopted many techniques discovered by SOTA GANs to stabilize the training, such as spectral norm [6], Hinge Loss [7], and residual components [1]. Please find the details in Figure 1 and also in the AutoGAN paper.

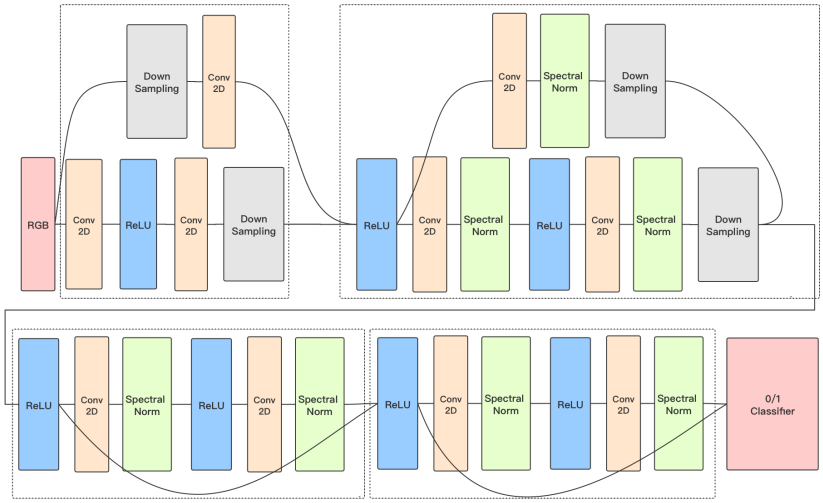

Fig. 1. The discriminator architecture used by both AutoGAN and E^{2}GAN.

2 Training Details

GAN training For GAN training, we follow the same training strategy and hyper-parameters as AutoGAN [2], which follows the training setting of spectral normalization GAN [6]. The learning rate of both generator and discriminator is set to 2×10^{-4}, using the hinge loss, Adam optimizers [5], 1:5 as generator/discriminator training ratio. The spectral normalization is only enforced on the discriminator. Searching and training are conducted on a RTX2080TI GPU.

RL training As mentioned in the main paper, the entire RL training process contains two periods: the exploration period and the exploitation period. During the exploration period, the agent will explore and learn the possible architectures. While in the exploitation period, the agent will always choose the best architecture, in order to quickly stabilize the policy.

Architecture Number	Cell 1	Cell 2	Cell 3
Seed 1 - Top 1	[0, 1, 0, 1]	[0, 1, 2, 1, 0]	[0, 1, 0, 1, 2]
Seed 2 - Top 1	[0, 1, 0, 0]	[0, 1, 2, 0, 0]	[0, 1, 0, 0, 2]
Seed 3 - Top 1	[0, 1, 2, 0]	[0, 1, 2, 0, 1]	[0, 1, 0, 0, 3]

Table 2. Discovered architectures by E^{2}GAN in 3 different seeds from scratch.

The exploration period is set to be 70% of iterations, and the rest as exploitation iterations. This ratios is chosen to ensure a balanced trade-off between sufficient exploration and successful convergence. As a guideline, most of the iterations should be used for exploration, but sufficient iterations should also be kept for the second phase, such that the algorithm can output stable architectures in the end.

3 Visualization of Discovered Architectures

3.1 Multiple random seeds

We train our agent from scratch with three different seeds, our agent steadily converged the policy in the exploitation period and discovered similar architectures. The discovered architectures are listed in Table 2 , and visualized in Fig 2,3,4.

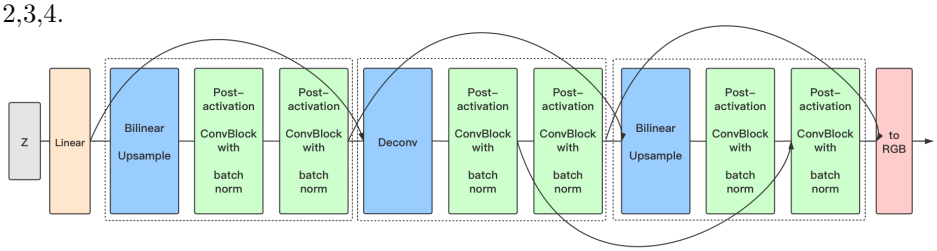

Fig. 2. The generator architecture discovered by E^{2}GAN on CIFAR-10 in Seed 1.

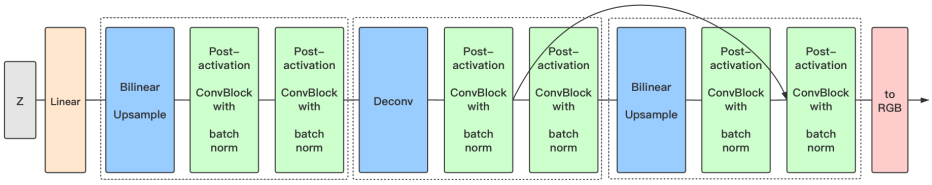

Fig. 3. The generator architecture discovered by E^{2}GAN on CIFAR-10 in Seed 2.

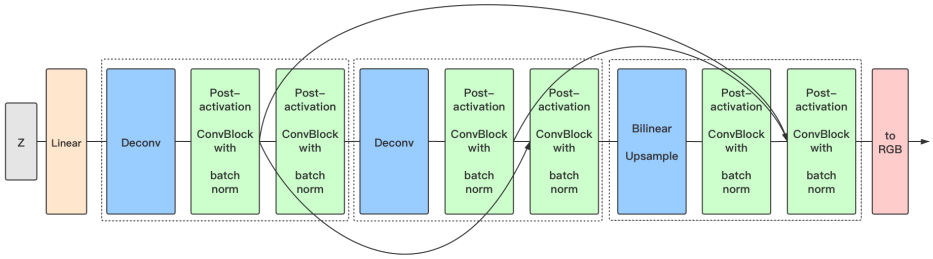

Fig. 4. The generator architecture discovered by E^{2}GAN on CIFAR-10 in Seed 3.

4 Ablation Study

4.1 MDP vs Non-MDP

In previous methods such as AutoGAN, the GAN architecture of an earlier layer is decided greedily without considering future layers. However, a better architecture may have a lower reward for the first layer but a higher final reward

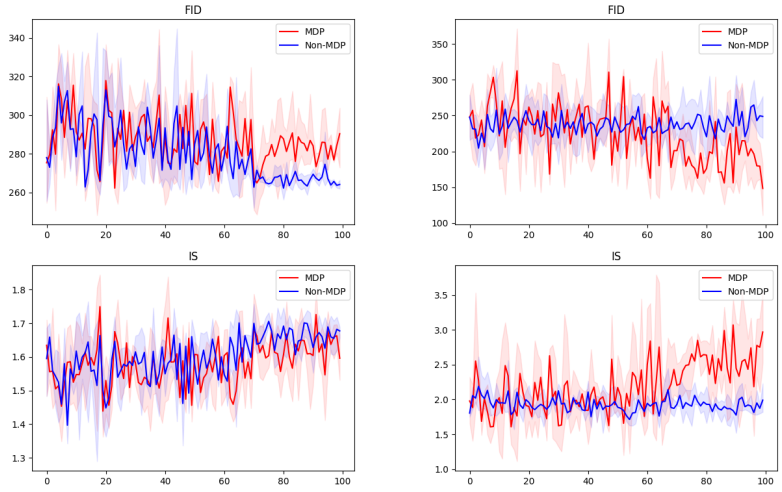

Fig. 5. Training curves on architecture searching. (Left) Performance of the first cell. (Right) Performance of the final output of the last cell. Although Non-MDP setting found better shallow architecture due to it greedy strategy, the MDP setting shows better final results for the entire architecture in terms of both IS and FID. The X-axis indicates the time steps, while the Y-axis is the performance on the proxy task.

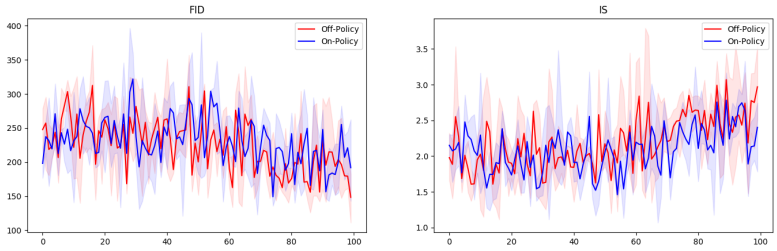

Fig. 6. Training curves on architecture searching. Off-Policy setting shows better results on both IS and FID. The X-axis indicates the total time steps,while the Y-axis is the performance on the proxy task.

for the entire architecture. By formulating NAS as a MDP process, we search for the architecture with the highest cumulative reward, which considers all layers and in theory allows us to target this 'more global' optima. This property is also supported empirically by an additional ablation study. Even though the non-MDP formulation found a first layer with 5% higher IS score, the final score of it in the last layer is 32 % lower than our proposed MDP formulation.

4.2 On-policy vs Off-policy

Our MDP framework works for both on/off-policy RL algorithms, the reason we choose SAC is that its effectiveness and good data efficiency have been widely acknowledged in many tasks [3].

To find out the difference between the on/off-policy, we reset the memory buffer size after every update to approximate an on-policy SAC. With all the other settings the same, both the IS and FID of off-policy RL results are better than on-policy RL.

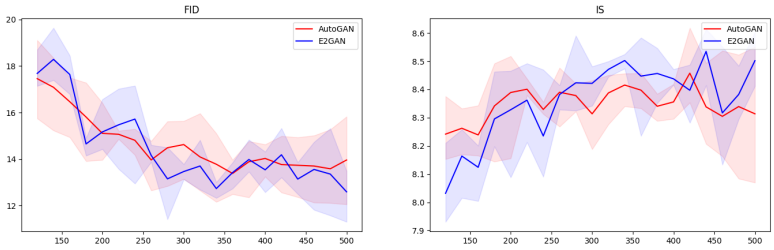

Fig. 7. Performance curves of E^{2}GAN and AutoGAN. The X-axis indicates the training epochs, while the Y-axis is the min/max/average performance. The evaluation frequency is 20 epochs.

5 Evaluation of the Discovered GAN

The evaluation protocol provided by AutoGAN uses the best model found in a 20 epoch interval. Due to the unstable nature of GAN training, this protocol may be insufficient to compare different methods. We thus additionally compare the GAN performance curve against training epochs for our best architecture and AutoGAN's best architecture. We run the training of both models for three times and report the average/min/max performance. Our model shows highly competitive performance.

References

1. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
2. 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)
3. 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)
4. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al.: Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905 (2018)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
7. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: International Conference on Machine Learning. pp. 7354–7363 (2019)