Auto-GAS: Automated Proxy Discovery for Training-free Generative Architecture Search

Lujun Li[1], Haosen Sun[1], Shiwen Li[2], Peijie Dong[3], Wenhan Luo[1], Wei Xue[1], Qifeng Liu[1✉], and Yike Guo[1✉]

[1] The Hong Kong University of Science and Technology
[2] Xidian University
[3] The Hong Kong University of Science and Technology (Guangzhou)
lilujunai@gmail.com;hsunas@connect.ust.hk;shiwenlisw@gmail.com;
pdong212@connect.hkust-gz.edu.cn;whluo.china@gmail.com;weixue@ust.hk;
liuqifeng@ust.hk;yikeguo@ust.hk

Abstract In this paper, we introduce Auto-GAS, the first training-free Generative Architecture Search (GAS) framework enabled by an auto-discovered proxy. Generative models like Generative Adversarial Networks (GANs) are now widely used in many real-time applications. Previous GAS methods use differentiable or evolutionary search to find optimal GAN generators for fast inference and memory efficiency. However, the high computational overhead of these training-based GAS techniques limits their adoption. To improve search efficiency, we explore training-free GAS but find existing zero-cost proxies designed for classification tasks underperform on generation benchmarks. To address this challenge, we develop a custom proxy search framework tailored for GAS tasks to enhance predictive power. Specifically, we construct an information-aware proxy that takes feature statistics as inputs and utilizes advanced transform, encoding, reduction, and augment operations to represent candidate proxies. Then, we employ an evolutionary algorithm to perform crossover and mutation on superior candidates within the population based on correlation evaluation. Finally, we perform generator search without training using the optimized proxy. Thus, Auto-GAS enables automated proxy discovery for GAS while significantly accelerating the search before training stage. Extensive experiments on image generation and image-to-image translation tasks demonstrate that Auto-GAS strikes superior accuracy-speed tradeoffs over state-of-the-art methods. Remarkably, Auto-GAS achieves competitive scores with 110× faster search than GAN Compression. Code at: https://github.com/lliai/Auto-GAS.

Keywords: Generative Model · Generative Architecture Search · AutoML

✉ Corresponding authors.

Table 1: Comparison of search cost for generative architecture search methods.

Method	Algorithm	Cost (GPU-days)	Method	Algorithm	Cost (GPU-days)
AGAN [65]	RL	1200	DGGAN [42]	Heuristic	580
AutoGAN [18]	RL	2	EAS-GAN [40]	EA	1
AlphaGAN [59]	Gradient	1.2	EGAN [64]	EA	1.3
EAGAN [70]	EA	0.8	**Auto-GAS (ours)**	**Training-free**	**0.09**

1 Introduction

Generative models like Generative Adversarial Networks (GANs) [19] have made tremendous strides in visual generation tasks such as image and video synthesis [5,50]. However, their demanding computational footprint presents barriers to deploying these models on edge devices with strict resource constraints [25, 47]. To overcome this, there is growing interest in model compression techniques [7, 12, 14, 49, 67]. Among them, Neural Architecture Search (NAS) [16] automates architecture design, finding compact generative models for edge computing, and enabling flexible structures without pre-trained weights. In contrast, other compression methods are limited to fixed, handcrafted architectures and rely on pre-trained models. The integration of NAS in generative models enables real-time visual creation on edge devices, unlocking new interactive experiences in domains like augmented reality [5, 24, 50].

Conventional Generative Architecture Search (GAS) approaches typically rely on expensive training-based training routes to assess candidate architectures, such as maintaining weight-sharing supernets [13, 22] or conducting multiple training trials from scratch [56, 76]. However, training generative models presents unique obstacles that greatly increase the costs and difficulties of applying these techniques for GAS. The training processes for generative models are generally more complex than discriminative tasks because inherent instabilities (*e.g.*, mode collapse [20]) can occur during optimization. In mode collapse, the model generates only a limited variety of examples from the underlying distribution. These instabilities make it challenging to reliably evaluate architectural candidates with standard NAS training-based evaluations. As shown in Table 1, gradient-based GAS methods like AdversarialNAS [17] and AlphaGAN [59] amplify these problems by jointly searching the generator and discriminator architecture spaces. This coupled search increases the overall complexity and fragility of the optimization procedure. To address these difficulties, DGGAN [42] introduced a progressive search approach where candidates are trained incrementally. However, its process still requires a massive 580 GPU days to complete due to the computationally intensive nature of generative model training.

Recently, training-free NAS approaches have demonstrated remarkable efficiency using zero-cost proxies to predict network performance without training network parameters. These methods are mainly present for encoder-only models for classification tasks. Inspired by their computational efficiency, we comprehensively evaluate existing proxies when applied to generative tasks. As shown

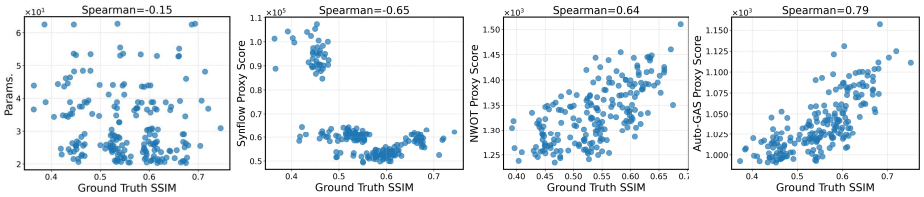

Figure 1: Correlation visualization of proxies (from left to right: Param., Synflow, NWOT, and Auto-GAS) in Generation benchmarks [15].

in Figure 1, we can make some interesting observations: (1) We observe that parameter-level proxies (*e.g.*, Synflow [57], SNIP [28], and GraSP [62]), which estimate performance based on weight importance, generally perform poorer when ranking generator candidates. This indicates their assumptions regarding how architecture impacts learning do not directly translate to generative objectives and training dynamics. (2) Architecture-level proxies (*e.g.*, NWOT [45] and Zen [39]) featuring topological encodings demonstrate stronger ranking ability. However, even the best architecture proxies leave room for improvement to enable performing generative NAS at scale. These findings motivate developing customized proxies that more precisely capture the unique characteristics of visual generation. Thus, a question naturally arises: ***how to design customized proxies for generation tasks?***

To answer this question, we present Auto-GAS, the first automated generative architecture search framework that aims to efficiently utilize custom training-free proxies. Firstly, we derive that the mutual information of latent features within generative models is strongly coupled with task optimization objectives based on information bottleneck theory [3, 54, 61]. Then, we develop proxy search strategies to better estimate the mutual information of these features. Specifically, we build comprehensive operation search spaces and evolutionary algorithms to search optimal information-aware proxies. To better process feature statistics, our search space involves transform-based, encoding-based, reduction-based, and augment-based operators. The search process in Auto-GAS begins with initializing the population of proxies and then evaluates, crosses, and varies them to improve the quality of the proxies. Throughout the search, we directly utilize the ranking correlation with accuracy results in the benchmark as fitting objectives, aiming to find a more suitable proxy for the generation tasks. With this evolutionary process, Auto-GAS outperforms existing training-free NAS approaches by significant margins without manual tuning. Additionally, we search for generator architectures using Auto-GAS and subsequently implement a full training process on the searched generator.

We perform extensive experiments to validate the performance and efficiency strengths of our Auto-GAS in generation tasks like image generation and image-to-image translation. The experiments demonstrate that our Auto-GAS can achieve better-generating performance than other zero-shot proxies and state-of-the-art results on rank consistency. In contrast to handcrafted GAS methods,

our Auto-GAS framework enjoys many benefits: (1) **Automatic.** It enables the automatic discovery of more expressive and efficient proxies that human expert might ignore, which can reduce human bias and ensures that the resulting architectures are optimized for the target problem or dataset. (2) **General.** The search procedure of Auto-GAS improves generalization by discovering proxies that generalize well on different search spaces. Based on the final results for the search objectives, our framework can evolve adaptive proxies to generalize across architectures. (3) **Insightful**. Our framework allows for the automatic discovery of more expressive, efficient, and effective proxies for optimizing and customizing generation architectures, opening up new research directions for automated GAN architecture search. Our contributions can be summarized as follows:

- We present Auto-GAS, the first training-free architecture search framework for generation tasks. Auto-GAS achieves an ultra-efficient and elegant search process with our auto-searched customized proxy.
- We propose a well-organized proxy search space, including latent features as input and various transforms, encoding, reduction, and augment operations as options. We develop evolutionary algorithms for effective & efficient proxy search.
- We conduct extensive experiments on image generation and image-to-image translation tasks. Auto-GAS obtains significant search acceleration and competitive performance on multiple datasets.

2 Related Work

Generative architecture search. The field of Neural Architecture Search (NAS) traditionally encompasses the design of exploration spaces, search methodologies, and assessment techniques to autonomously uncover optimal architectural configurations within specified parameters. Empirical approaches utilize an iterative process of training followed by exploration, either through multiple iterations [4,73] or by employing weight-sharing strategies [41,72]. NAS has also been extensively applied to the domain of GANs Architecture Search (GAS). The primary goal in GAS is to automatically identify superior GAN structures that enhance generative performance. Pioneering reinforcement learning-based GAS techniques, such as AGAN [65] and AutoGAN [18], leverage RNN controllers to construct GAN architectures and employ the Inception Score (IS) as their evaluation criterion. E2GAN [60] and AdversarialNAS [17] implement alternative reward functions. EAGAN [70] conducts discrete searches and enhances the robustness of the GAN exploration process. Nevertheless, these training-dependent search methodologies continue to be hampered by inefficient exploration and substantial computational demands. To tackle this challenge, we propose a training-free approach, grounded in architecture decoupling, to enhance the discovery of competitive GAN structures in a more computationally efficient manner.

Training-free architecture search. Zero-shot architecture exploration [10, 11, 29, 39], alternatively referred to as training-free NAS, offers a swifter alternative to conventional NAS methods. This approach's evaluation process en-

tails only a handful of forward or backward passes with a small batch of input data, rendering it highly efficient. Zero-cost indicators can be categorized into parameter-centric and architecture-centric proxies. Parameter-centric zero-cost proxies (*e.g.*, Synflow [57], SNIP [28], and GraSP [62]) rely on pruning techniques and aggregate the saliency values of each layer's weights as a proxy. For instance, Synflow [58] introduces a modified version of synaptic saliency scores to prevent layer collapse during parameter pruning, while Fisher [1] calculates the sum of all activation gradients in the network, applicable to channel pruning. Grasp [63] ensures gradient flow preservation throughout the network for efficient training. Their detailed formulations are as follows:

$$\rho_{snip} = \left| \frac{\partial \mathcal{L}}{\partial \mathcal{W}} \odot \mathcal{W} \right|, \rho_{synflow} = \frac{\partial \mathcal{L}}{\partial \mathcal{W}} \odot \mathcal{W}, \rho_{fisher} = \left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}} \mathcal{A} \right)^2, \rho_{grasp} = -(H \frac{\partial \mathcal{L}}{\partial \mathcal{W}}) \odot \mathcal{W}, \tag{1}$$

where $\mathcal{L}$, $\mathcal{W}$, $\mathcal{A}$ are loss function, weight and activation. H is the Hessian matrix. However, the complexity of loss function $\mathcal{L}$ in generation tasks renders these gradient-based methods ineffective. Regarding architecture-centric proxies, ZenNAS [39] introduced the novel Zen-Score to assess network expressivity, while NWOT [45] examined correlations between linear maps induced by data points for untrained architectures. Nevertheless, these approaches employ hand-crafted fixed proxies for classification tasks and encoder-only CNN models. In generation tasks, the encoder-decoder generator typically features intricate branches and multi-scale features, posing significant challenges for traditional proxies in predicting final accuracy. To address this issue, we automatically optimize the first customized proxies for GAS. Furthermore, our framework differs from EZNAS-like methods [2, 36, 74, 75] in several aspects: **(1) Tasks.** EZNAS is exclusively designed for classification, whereas our Auto-GAS is tailored for generation tasks. **(2) Search algorithms.** EZNAS merely employs genetic programming, while we propose a proxy evolution search strategy that substantially enhances search efficiency. **(3) Model statistic inputs.** Our method solely utilizes features as input, as we observed poor performance of parameter-level proxies. Conversely, EZNAS selects parameters and gradients as inputs. **(4) Search space.** Our Auto-GAS incorporates numerous feature-related operations absent in EZNAS.

3 Methodology

In this section, we first review existing proxies. Then, we present formulations and insights of our proxy design (see Figure 2). Finally, we illustrate the detailed search process.

3.1 Proxy Search Space

Motivation of our customized proxy design. Generative tasks are relevant to the Information Bottleneck (IB) theory [3, 54, 61], which guides generative models to learn compact representations that retain important characteristics for accurate sample generation. In generative tasks, the IB theory helps understand

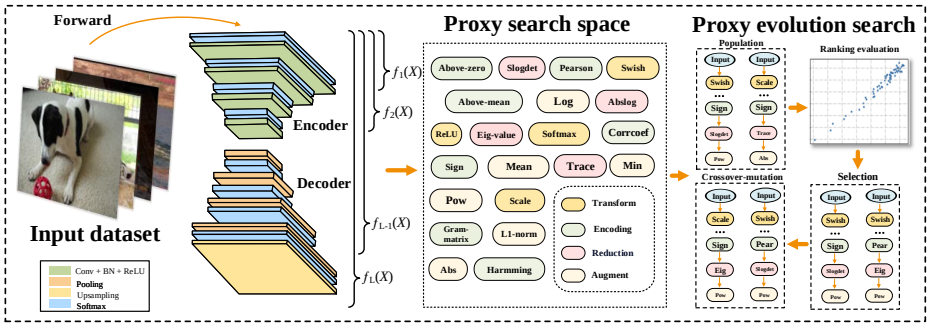

Figure 2: Overall proxy search of Auto-GAS. We extract the latent features of the generator with input data forward-propagation. Then we select various transform, encoding, reduction, and augment operators from the proxy search space to build candidate proxies. Finally, we perform a ranking evaluation to cross and mutate to get the best proxy.

how generative models can balance compression (capturing essential features) and reconstruction (faithful generation of samples) [55, 68]. IB theory suggests maximizing the Mutual Information (MI) $I(\mathbf{X}; \mathbf{F}_i)$ between the input data $\mathbf{X}$ and the corresponding latent representation $\mathbf{F}_i$ can be expressed as:

$$I(\mathbf{X}; \mathbf{F}_i) = H(\mathbf{F}_i) - H(\mathbf{F}_i|\mathbf{X}) \tag{2}$$

where $H(\mathbf{F}_i)$ stands for the entropy of the latent representation $\mathbf{F}_i$ and $H(\mathbf{F}_i|\mathbf{X})$ denotes the conditional entropy. Considering that neural networks are deterministic, and the dataset has a finite number of instances, the conditional entropy becomes zero, simplifying the mutual information (MI) estimation as $I(\mathbf{X}; \mathbf{F}_i) = H(\mathbf{F}_i)$. Consequently, some information coding and compression studies [26, 55] propose some operations (*e.g.*, Entropy and Gram-matrix) to approximate the mutual information on latent representation. These inspired us to develop our train-free method design thought chain: ***assess generative model→ estimate MI→ design expressions via Entropy/Gram-matrix ops → our train-free proxy.*** However, how about designing this potential expressions? Recent Auto-Zero approaches [30, 51] present a promising solution: searching for the best machine learning solution from scratch. In addition, several studies [6] have shown that normalize and activation operations can enhance the formulation. Therefore, we automatically search for optimal proxies based on these operators.

Proxy primitive operations. Our approach evolves the customized proxy in the generation benchmark [15], which consists of U-Net-like encoder-decoder models with various operators and their training results. We extract the latent features of each layer in both the encoder and decoder modules as inputs to the search process and represent the zero-cost proxy as a computation graph. This graph is constructed as ***transform→ encoding→reduction→augment***

parts in processing generator statistics. The available primitive operations can be summarized as follows:

- **Transform operations:** These operations involve different activation functions and feature processing techniques to enhance raw features and represent architectural potentials. Examples of transform operations include *relu*, *swish*, *mish*, *scale*,*norm*, *exp*, and *softmax*.
- **Encoding operations:** Inspired by coding algorithms and feature information estimation, we employ binary and matrix-based operations in the encoding part. These operators help us better estimate the mutual correlation information of samples. Some encoding operations include *sign*, *above-zero*, *above-median*, *gram-matrix*, *person*, and *hamming*.
- **Reduction operations:** Reduction operators bridge the information matrix and the proxy values. They are valuable for extracting useful matrix information from different mathematical perspectives. Reduction operations include *slogdet*, *abslog*, *trace*, and *eigenvalue*.
- **Augment operations:** Augment operators are applied to enhance the proxy values' ability to predict performance results. These operations include arithmetic functions such as *abs*, *pow*, *l2-norm*, *log*, *sum*, and *mean*.

By utilizing these primitive operations, our approach enables the evolution of the customized proxy, facilitating the search for optimal solutions in the generation benchmark. These mathematical operations are essential building blocks to construct a diverse and expressive search space of zero-cost proxy. We provide more details of their formulas and properties in the supplementary material.

3.2 Proxy Evolution Search

Evolutionary process and goal. Utilizing our proxy search space and graph representation, we employ an evolutionary algorithm (EA) outlined in Alg. 1 to discover optimal proxy expressions. Our EA commences with an initial set of candidate proxies and progressively refines the population across generations, employing genetic operators such as selection, crossover, and mutation to yield superior solutions. For fitness evaluation, each proxy is assessed based on the ranking correlation between its output proxy $\mathcal{Q}$ scores and pre-determined performance benchmarks $\mathcal{D}$, enabling efficient identification of the optimal proxy ρ^* within the search space $\mathcal{P}$, as follows:

$$\rho^*_{Auto-GAS} = \arg\max_{\rho \in \mathcal{P}} (\tau(\mathcal{D}, \mathcal{Q})). \tag{3}$$

where Kendall's Tau τ is formulated as the correlation coefficient. Each evolution picks top-k candidates with the highest evaluation score and randomly selects a parent from the candidates for mutation.

Efficient proxy search with existing generation benchmark. It is expensive to derive pre-trained results $\mathcal{D}$, we adopt an alternative solution to achieve proxy search using pre-computed architecture-performance pairs provided in

Algorithm 1 Evolutionary Search for Auto-GAS Proxy

Input: Exploration domain $\mathcal{S}$, candidate set $\mathcal{P}$, maximum iterations $\mathcal{T}$, sampling ratio r, sampled pool $\mathcal{R}$, top k selections.
Output: Auto-GAS proxy with highest Kendall's Tau τ coefficient.

1: $\mathcal{P}0$:= Initialize candidate set(P_i);
2: Sampled pool $\mathcal{R} := \emptyset$;
3: **for** $i = 1, 2, \ldots, \mathcal{T}$ **do**
4: Select ${G_i}^s$:= SelectTopk($\mathcal{R}$, k);
5: // greedy evolutionary strategy
6: Empty sampled pool $\mathcal{R} := \emptyset$;
7: Mutate := MUTATE(G_i^s);
8: Crossover G_i^c := CROSSOVER(G_i^s);
9: Incorporate G_i^m into R;
10: Incorporate G_i^c into R;
11: **if** $\mathcal{R} < \mathcal{P}$ **then**
12: Introduce random samples
13: **else**
14: Proceed to line 4;
15: **end if**
16: **end for**

the existing TransNAS-Bench-101 [15] benchmarks. Specifically, this benchmark involves Autoencoding task with 7k different generators in macro-level search space and their evaluation metric (*i.e.*, SSIM). Each generator network follows an encoder-decoder structure in Pix2Pix, where the encoders are the searched backbones, and the decoders contain 14 layers of convolution and deconvolution. Thanks to this existing benchmark, we do not need to train performance ground-truths again, and **it only takes 1.5 hours in our proxy search process.** In light of excellent generality (proven in our extensive experiments), our searched proxy $\rho_{Auto-GAS}$ can be well transferred to similar search spaces and can be directly used in most scenarios without additional search budget.

3.3 Discovered Proxy Analysis

Searched Zero-cost Proxy. We present formulas of the searched proxies on the generation benchmark as follows:

$$\rho^*_{Auto-GAS} = (\text{slogdet}([\mathbf{F}', {\mathbf{F}'}^T]))^2, \mathbf{F}' = \text{above-mean}(\text{swish}(\mathbf{F})) \tag{4}$$

where $\mathbf{F}$ represents the original features, and transformed features $\mathbf{F}'$ are obtained by applying the swish operation followed by the above-mean operation to the original features $\mathbf{F}$. The $[\mathbf{F}', {\mathbf{F}'}^T]$ indicates the $N \times N$ gram-matrix of the transformed features $\mathbf{F}'$. The proxy value ρ^* is calculated by taking the square of the slogdet of the concatenated gram-matrix $[\mathbf{F}', {\mathbf{F}'}^T]$.
Analysis of searched proxy. The discovered formulations indicate that (1) Mathematical operations, such as swish activation and slogdet, significantly con-

Table 2: Ranking results on TransNAS-Bench-101 [15].

Method	Kendall	Spearman	Pearson	Method	Kendall	Spearman	Pearson
SNIP [1]	-2.04%	-3.55%	-0.26%	ZiCo [29]	18.63%	15.53%	21.13%
EZNAS [2]	23.30%	25.91%	32.03%	Synflow [57]	-50.91%	-65.01%	-49.17%
NWOT [45]	51.28%	63.94%	48.88%	Params.	-12.97%	-15.24%	-5.80%
SNIP [28]	30.81%	23.31%	29.34%	FLOPs	48.91%	64.01%	52.36%
Zen [39]	-2.27%	2.58%	5.39%	**Auto-GAS**	**61.14%**	**79.32%**	**81.25%**

tribute to the predictability of the proxy. These operations enhance the proxy's ability to capture important features and characteristics of the generator. (2) Gram-Matrix and Mutual Information: The expression utilizes a sample-wise gram-matrix, which accurately estimates the latent mutual information of the generator. The gram-matrix captures the relationships and dependencies among the transformed features, providing valuable insights into the generator's behaviour. The discovered proxy of Auto-GAS, as illustrated in Figure 1, showcases a substantial improvement in ranking. This improvement suggests that the searched proxy effectively captures essential characteristics of the generator, leading to enhanced performance in the benchmark.

Understanding why our Auto-GAS surpasses other NAS methods. (1) Many train-based NAS methods employ weight-sharing and other proxy settings, which often lead to inaccurate estimations of architecture performance. In contrast, our Auto-GAS method searches for proxies based on actual performance, resulting in a proxy that demonstrates strong ranking capability. (2) Other train-free NAS methods are primarily designed for classification rather than generative tasks. Consequently, these methods search for architectures using weak correlation proxies. In contrast, our proxy exhibits a stronger correlation, ensuring the discovery of more efficient models.

3.4 Training-free Generative Models Search

Auto-GAS has made efforts to search generative Models in a non-training way to ease the instability of GAN training and improve search efficiency. In this stage, we directly search the generator in $\rho_{Auto-GAS}$. In each iteration, we generate P candidate generators to make up the *population* $\mathcal{A}$, where all these candidates use the randomly initialized weights W. Each candidate is characterized by both architecture and weight variables. In each iteration, we evaluate P candidates without training. We employ the information-aware proxy to assign scores to all these candidates. The formulation of the search stage can be succinctly stated as follows:

$$\alpha^*_{Auto-GAS} = \arg\max_{\alpha \in \mathcal{A}} \rho^*_{Auto-GAS}(\alpha, \mathcal{W}). \tag{5}$$

Note that in scenarios involving discriminator searches, we also use $\rho_{Auto-GAS}$ for searching the discriminator architectures with real sample inputs.

Table 3: Comparison results on the CIFAR-10 and STL-10 datasets. We use FID (the lower the better) and the mIoU metric (the higher the better) for evaluations.

Method	Algorithm	GPU days	CIFAR-10 (32×32)		STL-10 (48×48)	
			IS↑	FID↓	IS↑	FID↓
WGAN-GP [20]	Manual	–	7.86	29.3	–	–
SN-GAN [46]			8.22	21.70	9.16	40.1
ProbGAN [21]			7.75	24.60	8.87	46.74
Improv MMD GAN [66]			8.29	16.21	9.34	37.63
BigGAN [5]			**9.22**	14.73	-	-
AGAN [65]	RL	1200	8.29	30.50	9.23	52.70
AutoGAN [18]		2.00	8.55	12.42	9.16	31.01
AlphaGAN [59]		1.2	8.70	15.56	-	-
AdversarialNAS [17]		1.00	7.86	24.04	8.52	38.85
EGAN [64]	EA	1.25	6.90	-	-	-
EAS-GAN [40]		1.00	7.45	33.20	-	38.84
EAGAN [70]		0.8	8.41	12.83	9.69	23.82
Fisher [1]	Training-free	0.12	8.03	17.66	9.37	29.82
Zen [39]		0.11	8.02	15.99	9.45	28.66
SNIP [28]		0.12	7.96	16.67	9.58	29.25
NWOT [45]		0.09	8.05	16.32	9.23	27.46
Auto-GAS		0.09	8.44	**12.21**	**9.72**	**23.19**

3.5 Generative Models Training

After the architecture searches, we train generative models with randomly selected data samples. Simultaneously, a batch of noise samples is generated from a noise distribution. The generator tries to generate data from random noise z as $G(z)$, which should be as close as possible to the real data distribution. The discriminator tries to distinguish the generated data from the real data. The value function $V(D, G)$ for a minimax game is defined as follows:

$$\begin{aligned} \min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] \\ +\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))] \end{aligned} \tag{6}$$

where $\mathbf{x} \sim p_{data}(\mathbf{x})$ denotes the distribution of real data, $\mathbf{z} \sim p_z(\mathbf{z})$ denotes the input noise to the generator.

4 Experiments

In this section, we first perform proxy search experiments on the generation benchmark, and then we conduct architecture search experiments on image generation and image-to-image translation tasks by translocating the searched proxies. Finally, we provide a detailed analysis of our approach.

4.1 Experiments on Generation Benchmark

Dataset and implementation. We adopt the search space and performance ground truth provided by the Autoencoding task in TransNAS-Bench-101 [15] as

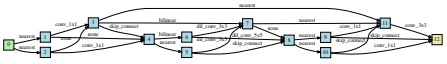

Figure 3: Searched model on CIFAR-10.

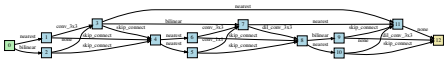

Figure 4: Searched model on STL-10.

the benchmark for searching and evaluating GAS proxies. We split the 4:3 ratio of the benchmark into the validation and test sets. During the proxy search phase, we set ($\mathcal{P}$, $\mathcal{T}$, r, k) in Alg. 1 as $(20, 100, 0.9, 5)$ and evaluate with 50 architectures in the validation set. After the search, we fairly compare the Auto-GAS proxy with the traditional ones by randomly sampling 200 architectures in the test set. Note that our validation and test sets do not overlap to ensure fair proxy comparisons. Detailed implementations are in the Appendix.

Comparison results. Table 5 demonstrates the ranking capabilities of different methods on Generation Benchmark. Auto-GAS (ours) achieves the best rankings across all three measures: Kendall's Tau (61.14%), Spearman (79.32%), and Pearson (81.25%). These results indicate that Auto-GAS performed well in preserving the rankings compared to other methods. These findings highlight the effectiveness of Auto-GAS for generating meaningful and accurate rankings.

4.2 Experiments on Image Generation

Datasets and implementation. We perform evaluations using CIFAR-10 [27] and STL-10 [8] datasets, aligning with other GAS methodologies [17, 18, 70]. CIFAR-10 comprises 50k training and 10k test 32×32 images, while STL-10 contains 100,500 images at a higher 48×48 resolution. We directly apply searched proxies to identify optimal architectures within EAGAN's [70] search space. The exploration process spans 150 epochs with a $P = 32$ individual population. Generator and discriminator batch sizes are 80 and 40, respectively. We utilize 16 randomly generated Gaussian noise images for proxy score computation to expedite evaluation. Post-search, we conduct full training of the best-scoring generator for 500 epochs. For evaluation, following previous GAS works [17, 18, 70], we generate 50,000 images to calculate IS and FID metrics. Figures 3 and 4 illustrate the discovered architectures for CIFAR-10 and STL-10.

Training-based NAS method comparison. We benchmark Auto-GAS against various training-based NAS approaches: BigGAN [5], E2GAN [44], AdversarialNAS [17], and EAGAN [70]. As shown in Table 3, Auto-GAS demonstrates superior efficiency. For CIFAR-10, Auto-GAS achieves lower FID than BigGAN and comparable overall results to E2GAN, while outperforming AdversarialNAS in both FID and IS. On STL-10, Auto-GAS significantly outperforms E2GAN and AdversarialNAS in both metrics, rivaling EAGAN in IS and surpassing it in FID. In summary, Auto-GAS emerges as an efficient and competitive approach, either surpassing or matching several state-of-the-art training-based NAS methods across diverse evaluation metrics and datasets.

Zero-shot proxy comparison. Table 3 also juxtaposes the Auto-GAS proxy with four alternative zero-shot proxies. Fisher and SNIP, relying on gradient-

Table 4: Image translation evaluation on horse→zebra and Cityscapes datasets.

Model	Dataset	Method	#Parameters		GPU days		FID (↓)		mIoU (↑)	
			Value	Ratio	Value	Ratio	Value	Increase	Value	Drop
CycleGAN	horse→zebra (512×512)	Original	11.4M	–	–	–	61.53	–	–	
		COEV [53]	–	–	–	–	96.15	34.6	–	
		GAN Comp. [38]	0.34M	33.3×	11.60	1.00×	71.81	10.3	–	
		Fast GAN Comp. [38]	0.36M	32.1×	3.12	3.72×	65.19	3.66	–	
		Auto-GAS	0.39M	29.3×	**0.11**	**105.45**×	**63.66**	**2.13**	–	
GauGAN	Cityscapes (512×512)	Original	93.0M	–	–	–	57.60	–	62.18	–
		GAN Comp. [38]	20.4M	4.6×	17.70	1.00×	**55.19**	**2.41**	61.22	0.96
		Fast GAN Comp. [38]	20.2M	4.6×	10.40	1.70×	56.25	1.35	61.17	1.01
		Auto-GAS	21.0M	4.5×	**0.16**	**110.63**×	56.04	1.56	**61.21**	**0.97**

based computations, yield similar scores. Zen-Score and NWOT exhibit positive correlations with model accuracy, outperforming gradient-based methods. These results validate the Auto-GAS proxy's efficacy, positioning it as a formidable contender among zero-shot proxy methods.

4.3 Experiments on Image-to-image Translation.

Datasets and implementation. We comprehensively evaluate our approach on image translation tasks using high-resolution datasets, including [9] and horse→zebra [71] at 512×512 resolutions. We apply the proxy for architecture exploration, adopting GAN Compression's [38] search space for fair comparisons, focusing on channel and layer number configurations. We evolve to compress Pix2Pix [23] on horse→zebra and GauGAN [48] on cityscapes. The search process involves 500 iterations, followed by model retraining with a 0.0002 learning rate and 16 batch size, adhering to the original paper [23].

GAN compression method comparison. Table 4 demonstrates that Auto-GAS achieves the highest FID score of 63.66 for CycleGAN with a superior compression ratio (29.3×) compared to alternatives. This indicates Auto-GAS effectively reduces model size while preserving image quality. For GauGAN, Auto-GAS exhibits a marginally higher FID score and a competitive mIoU score of 61.22.

Qualitative Analysis. Figure 5 illustrates Auto-GAS's proficiency in translation tasks. In contrast, FastGAN Compression not only yields a lower FID score but also fails to eliminate artifacts like brown discoloration in zebra models. Conversely, Auto-GAS-generated zebra images accurately capture authentic black-and-white zebra patterns.

4.4 Ablation Study

Search efficiency benefits. For image generation (see Table 3), our Auto-GAS method exhibits significantly superior search efficiency to training-based NAS GANs and has a definite advantage over other zero-cost proxies. The Fisher and

Figure 5: Visualization results on image translation tasks using high-resolution datasets, *e.g.*, horse→zebra and Cityscapes with 512×512 resolution.

Table 5: Ablations of our proxy search.

Exp.	Method	Kendall	Spearman	FID↓@CIFAR-10	FID↓@STL-10
I	Vanilla Gram-matrix	24.28%	35.75%	17.88	29.62
	Auto-GAS	61.14%	79.32%	12.21	23.19
II	Auto-GAS without Transform	55.78%	73.60%	14.22	25.58
	Auto-GAS without Encoding	51.64%	68.62%	15.26	26.49
	Auto-GAS without Reduction	52.78%	71.95%	14.63	25.74
	Auto-GAS without Augment	58.40%	76.06%	13.86	25.12
III	Auto-GAS (Gram-matrix)	61.14%	79.32%	12.21	23.19
	Auto-GAS (Pearson)	60.07%	77.53%	12.85	24.62
	Auto-GAS (Hamming)	60.58%	78.57%	12.35	23.88
IV	Auto-GAS (Proxy Random)	29.49%	38.25%	16.65	28.12
	Auto-GAS (Proxy Evolution)	61.14%	79.32%	12.21	23.19

SNIP require gradient backpropagation, leading to increased time consumption during the search process. In contrast, our Auto-GAS only requires a single forward pass, making it faster than methods that necessitate multiple forwards or gradient backpropagation. For image translation, as presented in Table 4, Auto-GAS achieves a 110× enhancement compared to GAN Compression [38].

Ablation of proxy search. (1) Our proxy search space includes transform, encoding, reduction, and augmentation operations. Ablations in Table 5 (II) indicate that all these operators contribute, with encoding operators having a greater influence. (3) Ablations for different encoding operators in Table 5 (III) show that the gram-matrix is important, but other operators (*e.g.*, Pearson correlation matrix) also achieve similar performance. (2) Our contribution stems from search space design and proxy search processes. Ablations in Table 5 demonstrate that the various components of our search space contribute. In addition, our proxy evolution search plays an important role in exploring extensive search space and it significantly outperforms random search in Table 5 (IV).

Operations for mutual information estimation. (1) Gram-matrix calculates matrix correlation for mutual information, which measures the salient features and relevant information captured by generative models. The comparison in Table 5 (I) shows that the vanilla Gram-matrix can be used as a proxy for NAS but is weaker than our Auto-GAS. (2) We also compare other operations for fea-

Table 6: Results of different correlation methods on the CIFAR-10 dataset.

Correlation Methods	CIFAR-10	
	IS↑	FID↓
Hessian Trace	8.36	15.47
L_2 norm	7.56	24.62
Entropy	7.70	25.43
Variance	7.67	23.75
Gram-matrix	8.12	17.88
Auto-GAS	8.44	12.21

Table 7: Results of hyperparameter analysis on the CIFAR-10 dataset.

Hyperparameter	CIFAR-10	
	IS↑	FID↓
Batch Size (20)	8.52	12.28
Batch Size (40)	8.02	16.52
Number of Individuals (16)	8.44	12.21
Number of Individuals (64)	8.11	16.32
EA iteration interval (5)	7.78	18.25
EA iteration interval (10)	8.38	13.69

ture information estimation in Table. 6. The results of L_2 norm|Entropy|Variance are worse than Auto-GAS, because they overlooks higher-order feature statistics. Hessian Trace [43] yields relatively promising performance but its computational complexity is excessive.

Hyperparameter analysis. The results in Table 7 indicate that a larger batch size may speed up convergence but also risk overfitting. An appropriate population size ensures our more efficient architecture exploration. Larger iteration intervals allow for a more comprehensive exploration in our architecture space.

5 Conclusion

In this paper, we present Auto-GAS, a novel training-free generative architecture search framework by evolving customized proxy. We derive from the information bottleneck theory using the latent layer feature as input and building candidate proxies with various transform, encoding, reduction, and augment options. We search for the best proxy using evolutionary algorithms. With customized proxies, our Auto-GAS allows an efficient search for promising architectures. Comprehensive experiment results on image generation and image translation illustrate the promising performance of Auto-GAS. For future work, we will augment our Auto-GAS with advanced methods [31–35, 37, 52, 69]. We hope our novel investigations will provide some insight for the generation and NAS research community.

Iimitation. We mainly evaluate the same benchmarks as other NAS methods for fair comparison. We will extend Auto-GAS to more generative models in future work. **Social impact.** Our Auto-GAS focuses on technology improvement without social and ethical implications.

Acknowledgements

The research was supported by Theme-based Research Scheme (T45-205/21-N) from Hong Kong RGC, and Generative AI Research and Development Centre from InnoHK.

References

1. Abdelfattah, M.S., Mehrotra, A., Dudziak, Ł., Lane, N.D.: Zero-cost proxies for lightweight nas. arXiv preprint arXiv:2101.08134 (2021)
2. Akhauri, Y., Munoz, J.P., Jain, N., Iyer, R.: EZNAS: Evolving zero-cost proxies for neural architecture scoring. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) NeurIPS (2022), https://openreview.net/forum?id=lSqaDG4dvdt
3. Alemi, A.A., Fischer, I., Dillon, J.V., Murphy, K.: Deep variational information bottleneck. ICLR (2017)
4. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. In: ICLR (2017)
5. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. In: ICLR (2019)
6. Cavagnero, N., Robbiano, L., Pistilli, F., Caputo, B., Averta, G.: Entropic score metric: Decoupling topology and size in training-free nas. In: ICCV workshop (2023)
7. Chen, K., Yang, L., Chen, Y., Chen, K., Xu, Y., Li, L.: Gp-nas-ensemble: a model for the nas performance prediction. In: CVPRW (2022)
8. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Fourteenth International Conference on Artificial Intelligence and Statistics (2011)
9. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. CVPR (2016)
10. Dong, P., Li, L., Wei, Z.: Diswot: Student architecture search for distillation without training. In: CVPR (2023)
11. Dong, P., Li, L., Wei, Z., Niu, X., Tian, Z., Pan, H.: Emq: Evolving training-free proxies for automated mixed precision quantization. In: ICCV (2023)
12. Dong, P., Niu, X., Li, L., Tian, Z., Wang, X., Wei, Z., Pan, H., Li, D.: Rd-nas: Enhancing one-shot supernet ranking ability via ranking distillation from zero-cost proxies. arXiv preprint arXiv:2301.09850 (2023)
13. Dong, P., Niu, X., Li, L., Xie, L., Zou, W., Ye, T., Wei, Z., Pan, H.: Prior-guided one-shot neural architecture search. arXiv preprint arXiv:2206.13329 (2022)
14. Dong, P., Niu, X., Tian, Z., Li, L., Wang, X., Wei, Z., Pan, H., Li, D.: Progressive meta-pooling learning for lightweight image classification model. In: ICASSP (2023)
15. Duan, Y., Chen, X., Xu, H., Chen, Z., Liang, X., Zhang, T., Li, Z.: Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In: CVPR. pp. 5251–5260 (2021)
16. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. arXiv preprint arXiv:1808.05377 (2018)
17. Gao, C., Chen, Y., Liu, S., Tan, Z., Yan, S.: Adversarialnas: Adversarial neural architecture search for gans. In: CVPR (2020)
18. Gong, X., Chang, S., Jiang, Y., Wang, Z.: Autogan: Neural architecture search for generative adversarial networks. ICCV (2019)
19. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS (2014)
20. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. NeurIPS (2017)

21. He, H., Wang, H., Lee, G.H., Tian, Y.: Probgan: Towards probabilistic gan with theoretical guarantees. In: ICLR (2018)
22. Hu, Y., Wang, X., Li, L., Gu, Q.: Improving one-shot nas with shrinking-and-expanding supernet. Pattern Recognition (2021)
23. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017). https://doi.org/10.1109/CVPR.2017.632
24. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
25. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (June 2019)
26. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.E.: Similarity of neural network representations revisited. In: ICML (2019)
27. Krizhevsky, A., Nair, V., Hinton, G.: The cifar-10 dataset. online: http://www. cs. toronto. edu/kriz/cifar. html (2014)
28. Lee, N., Ajanthan, T., Torr, P.H.: Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340 (2018)
29. Li, G., Yang, Y., Bhardwaj, K., Marculescu, R.: Zico: Zero-shot NAS via inverse coefficient of variation on gradients. In: The Eleventh ICLR (2023), https://openreview.net/forum?id=rwo-ls5GqGn
30. Li, H., Fu, T., Dai, J., Li, H., Huang, G., Zhu, X.: Autoloss-zero: Searching loss functions from scratch for generic tasks. In: CVPR (2022)
31. Li, L.: Self-regulated feature learning via teacher-free feature distillation. In: ECCV (2022)
32. Li, L., Bao, Y., Dong, P., Yang, C., Li, A., Luo, W., Liu, Q., Xue, W., Guo, Y.: Detkds: Knowledge distillation search for object detectors. In: ICML (2024)
33. Li, L., Dong, P., Li, A., Wei, Z., Yang, Y.: Kd-zero: Evolving knowledge distiller for any teacher-student pairs. NeuIPS (2024)
34. Li, L., Dong, P., Wei, Z., Yang, Y.: Automated knowledge distillation via monte carlo tree search. In: ICCV (2023)
35. Li, L., Jin, Z.: Shadow knowledge distillation: Bridging offline and online knowledge transfer. In: NeuIPS (2022)
36. Li, L., Sun, H., Dong, P., Wei, Z., Shao, S.: Auto-das: Automated proxy discovery for training-free distillation-aware architecture search. In: ECCV (2024)
37. Li, L., Wang, Y., Yao, A., Qian, Y., Zhou, X., He, K.: Explicit connection distillation. In: ICLR (2020)
38. Li, M., Lin, J., Ding, Y., Liu, Z., Zhu, J.Y., Han, S.: Gan compression: Efficient architectures for interactive conditional gans. CVPR (2020)
39. Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., Li, H., Jin, R.: Zen-nas: A zero-shot nas for high-performance image recognition. ICCV (2021)
40. Lin, Q., Fang, Z., Chen, Y., Tan, K.C., Li, Y.: Evolutionary architectural search for generative adversarial networks. IEEE Transactions on Emerging Topics in Computational Intelligence (2022)
41. Liu, H., Simonyan, K., Yang, Y.: DARTS: differentiable architecture search. In: 7th ICLR, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. vol. abs/1806.09055 (2019)
42. Liu, L., Zhang, Y., Deng, J., Soatto, S.: Dynamically grown generative adversarial networks. AAAI (2021)
43. Liu, Y., Yu, S., Lin, T.: Hessian regularization of deep neural networks: A novel approach based on stochastic estimators of hessian trace. Neurocomputing **536**, 13–20 (2023)

44. Luo, Y., Zhang, Y., Cai, X., Yuan, X.: E2gan: End-to-end generative adversarial network for multivariate time series imputation. In: 28th international joint conference on artificial intelligence. pp. 3094–3100. AAAI Press Palo Alto, CA, USA (2019)
45. Mellor, J., Turner, J., Storkey, A., Crowley, E.J.: Neural architecture search without training. In: ICML (2021)
46. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. ArXiv **abs/1802.05957** (2018)
47. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. In: ICML. PMLR (2017)
48. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: CVPR (2019)
49. Qin, J., Wu, J., Xiao, X., Li, L., Wang, X.: Activation modulation and recalibration scheme for weakly supervised semantic segmentation. In: AAAI (2022)
50. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR **abs/1511.06434** (2016)
51. Real, E., Liang, C., So, D.R., Le, Q.V.: Automl-zero: Evolving machine learning algorithms from scratch (2020)
52. Shao, S., Dai, X., Yin, S., Li, L., Chen, H., Hu, Y.: Catch-up distillation: You only need to train once for accelerating sampling. arXiv preprint arXiv:2305.10769 (2023)
53. Shu, H., Wang, Y., Jia, X., Han, K., Chen, H., Xu, C., Tian, Q., Xu, C.: Co-evolutionary compression for unpaired image translation. In: ICCV (2019)
54. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv:1703.00810 (2017)
55. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv preprint, arXiv:1703.00810 (2017)
56. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML (2019)
57. Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. NeurIPS (2020)
58. Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. NeurIPS **33**, 6377–6389 (2020)
59. Tian, Y., Shen, L., Su, G., Li, Z., Liu, W.: Alphagan: Fully differentiable architecture search for generative adversarial networks. arXiv preprint arXiv:2006.09134 (2020)
60. Tian, Y., Wang, Q., Huang, Z., Li, W., Dai, D., Yang, M., Wang, J., Fink, O.: Off-policy reinforcement learning for efficient and effective gan architecture search. In: ECCV (2020)
61. Tishby, N., Zaslavsky, N.: Deep learning and the information bottleneck principle. In: ITW. pp. 1–5. IEEE (2015)
62. Wang, C., Zhang, G., Grosse, R.: Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376 (2020)
63. Wang, C., Zhang, G., Grosse, R.: Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:2002.07376 (2020)
64. Wang, C., Xu, C., Yao, X., Tao, D.: Evolutionary generative adversarial networks. IEEE Transactions on Evolutionary Computation **23**(6), 921–934 (2019)
65. Wang, H., Huan, J.: Agan: Towards automated design of generative adversarial networks. arXiv preprint arXiv:1906.11080 (2019)
66. Wang, W., Sun, Y., Halgamuge, S.: Improving MMD-GAN training with repulsive loss function. In: ICLR (2019)

67. Wei, Z., Pan, H., Li, L.L., Lu, M., Niu, X., Dong, P., Li, D.: Convformer: Closing the gap between cnn and vision transformers. arXiv preprint arXiv:2209.07738 (2022)
68. Wolchover, N., Reading, L.: New theory cracks open the black box of deep learning. Quanta Magazine (2017)
69. Xiaolong, L., Lujun, L., Chao, L., Yao, A.: Norm: Knowledge distillation via n-to-one representation matching. In: ICLR (2023)
70. Ying, G., He, X., Gao, B., Han, B., Chu, X.: Eagan: Efficient two-stage evolutionary architecture search for gans. In: ECCV. Springer (2022)
71. Yu, A., Grauman, K.: Fine-grained visual comparisons with local learning. CVPR (2014)
72. Zhang, M., Li, H., Pan, S., Chang, X., Su, S.: Overcoming multi-model forgetting in one-shot nas with diversity maximization. In: CVPR. pp. 7809–7818 (2020)
73. Zhong, Z., Yang, Z., Deng, B., Yan, J., Wu, W., Shao, J., Liu, C.L.: Blockqnn: Efficient block-wise neural network architecture generation. IEEE transactions on pattern analysis and machine intelligence (2020)
74. Zhu, C., Li, L., Wu, Y., Sun, Z.: Saswot: Real-time semantic segmentation architecture search without training. In: AAAI (2024)
75. Zimian Wei, Z., Li, L.L., Dong, P., Hui, Z., Li, A., Lu, M., Pan, H., Li, D.: Autoprox: Training-free vision transformer architecture search via automatic proxy discovery. In: AAAI (2024)
76. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR (2018)