Efficient Neural Architecture Design via Capturing Architecture-Performance Joint Distribution

Yue Liu Shanghai University
Ziyi Yu Shanghai University
Zitu Liu Shanghai University
Wenjie Tian Shanghai University

Abstract

The relationship between architecture and performance is critical for improving the efficiency of neural architecture design, yet few efforts have been devoted to understanding this relationship between architecture and performance, especially architecture-performance joint distribution. In this paper, we propose Semi-Supervised Generative Adversarial Networks Neural Architecture Design Method or SemiGAN-NAD to capture the architecture-performance joint distribution with few performance labels. It is composed of Bidirectional Transformer of Architecture and Performance (Bi-Arch2Perf) and Neural Architecture Conditional Generation (NACG). Bi-Arch2Perf is developed to learn the joint distribution of architecture and performance from bidirectional conditional distribution through the adversarial training of the discriminator, the architecture generator, and the performance predictor. Then, the incorporation of semi-supervised learning optimizes the construction of Bi-Arch2Perf by utilizing a large amount of architecture information without performance annotation in search space. Based on the learned bidirectional relationship, the performance of architecture is predicted by NACG in high-performance architecture space to efficiently discover well-promising neural architectures. The experimental results on NAS benchmarks demonstrate that SemiGAN-NAD achieves competitive performance with reduced evaluation time compared with the latest NAS methods. Moreover, the high-performance architecture signatures learned by Bi-Arch2Perf are also illustrated in our experiments.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

1 INTRODUCTION

Neural network has been witnessed impressive successes in computer vision tasks, such as image classification (He et al., 2016; Han et al., 2022) and object detection (Joseph et al., 2021). However, the well-designed neural architectures highly require expert experience with trial and error. It is extremely inefficient in real-world applications, which promotes the research on Neural Architecture Search (NAS). NAS attempts to design a high-performance neural architecture with an automatic manner (Zoph and Le, 2017; Zoph et al., 2018; Liu et al., 2019). It has been successfully applied in various tasks, and aroused widespread attention (Elsken et al., 2019; Xie et al., 2022).

The progress on architecture improvements (He et al., 2016; Szegedy et al., 2015; Li et al., 2021) has been observed that there exists a specific relationship between neural architecture and its performance, i.e., the performance of neural network heavily depends on its architecture design. This is particularly important for NAS, which still searches over all possible architectures in search space (Li et al., 2021; Ying et al., 2019). Two basic questions arise in this relationship: (1) How to efficiently determine the performance of a given neural network? (2) What are architecture signatures of high-performance neural networks? You et al. (2020) proposed a novel graph-based representation of neural network named relational graph to understand the relationship between architecture and performance in neural network. They found that the performance is approximately a smooth function with the average path length and clustering coefficient of the relational graph (Li et al., 2020; Ru et al., 2021). An inspiration is whether this relationship can be intuitively understood through capturing the joint distribution of architecture and performance.

In order to learn this joint distribution, existing methods built performance predictor to determine the per-

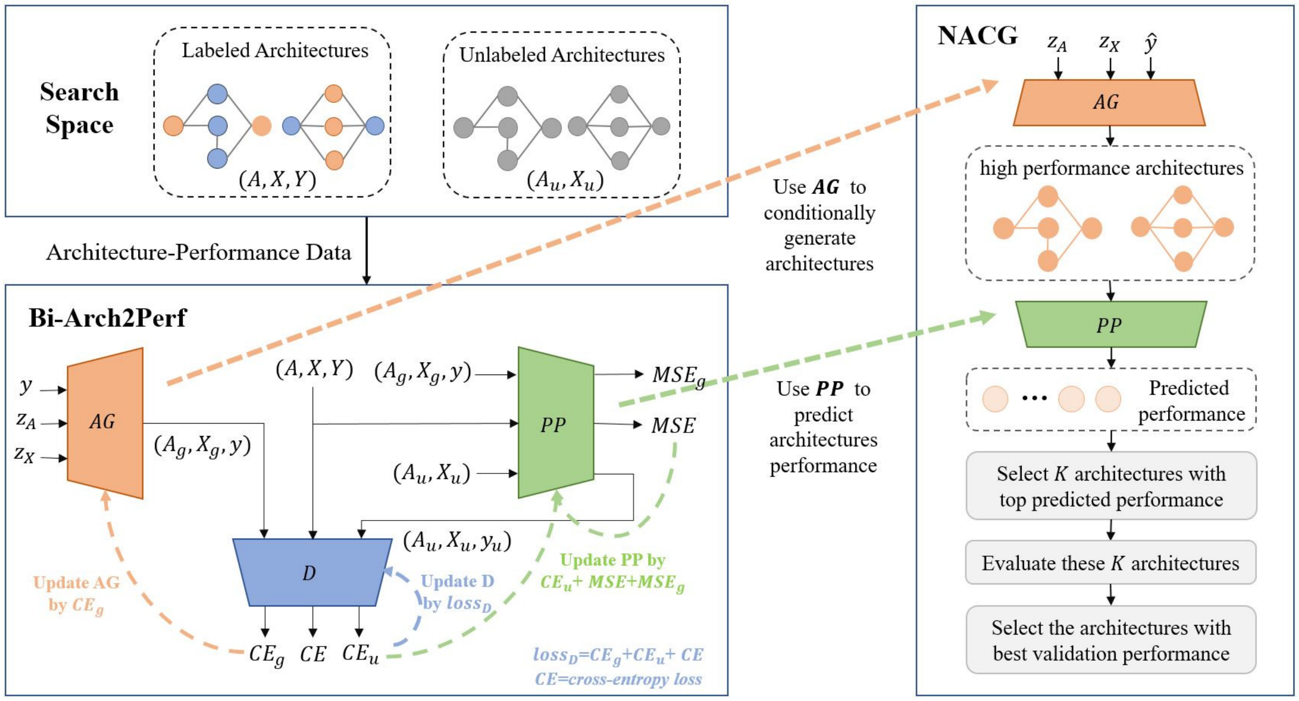
Search Space
Labeled Architectures
(A, X, Y)
Unlabeled Architectures
(A_u, X_u)
Architecture-Performance Data
Bi-Arch2Perf
y
z_A
z_X
AG
(A_g, X_g, y)
(A, X, Y)
(A_g, X_g, y)
PP
MSE_g
MSE
(A_u, X_u)
(A_u, X_u, y_u)
D
Update AG by CE_g
Update D by $loss_D$
Update PP by CE_u+ MSE+MSE_g
CE_g CE CE_u
$loss_D$=CE_g+CE_u+ CE
CE=cross-entropy loss
Use AG to conditionally generate architectures
Use PP to predict architectures performance
NACG
z_A z_X $\hat{y}$
high performance architectures
Predicted performance
Select K architectures with top predicted performance
Evaluate these K architectures
Select the architectures with best validation performance

Figure 1: The process of SemiGAN-NAD. It consists of Bi-Arch2Perf and NACG. Bi-Arch2Perf is built on the adversarial training of Architecture Generator (AG), Performance Predictor (PP) and Discriminator (D). After Bi-Arch2Perf has been built, NACG discovers high-performance architecture through architecture generation by AG and performance prediction by PP.

formance of a given architecture, and embedded the constructed predictor into NAS to improve the efficiency of architecture search. Luo et al. (2018) proposed a gradient-based optimization method for neural architecture, which performs the optimization within continuous space. Xu et al. (2019) proposed the feature tensor representation of neural network and further built LeNet-5 performance predictors for ranking different architectures. Li et al. (2020) and Wen et al. (2020) employed graph neural network to construct performance predictor. Moreover, Luo et al. (2020) and Tang et al. (2020) considered the insufficiency of architectures and their corresponding real performance, and proposed a semi-supervised performance predictor to utilize the architectures without performance annotation. The above works paid more attention to the conditional distribution from architecture to performance. However, the distribution from performance to architecture is also especially important in the relationship between architecture and performance. Therefore, it can learn comprehensive joint distribution to discover architecture signatures of high-performance neural networks, thereby accurately understanding the architecture-performance relationship in neural network.

Conditional GAN (CGAN) (Mirza and Osindero, 2014) generates data with a specific mode by introducing label information. One of its key features is generic nature in sampling from an unspecified distribution underlying the given data. This feature fits well when datasets are collected without knowing a prior distribution, such as in the case of the conditional distribution from performance to architecture in neural network. Therefore, it provides a promising approach to capture the architecture-performance relationship.

In this paper, we investigate the approach of incorporating GAN with performance predictor under semi-supervised learning to capture the architecture-performance joint distribution for understanding the relationship between architecture and its performance comprehensively, and propose a novel efficient Semi-Supervised Generative Adversarial Networks Neural Architecture Design Method (SemiGAN-NAD). It is composed of Bidirectional Transformer of Architecture and Performance (Bi-Arch2Perf) and Neural Architecture Conditional Generation (NACG). The Bi-Arch2Perf can learn the joint distribution of architecture and performance from bidirectional conditional distribution through the adversarial training of the discriminator, the architecture generator and the performance predictor. The semi-supervised architecture loss is introduced for optimizing the construction of Bi-

Arch2Perf by utilizing a large amount of architecture information without performance annotation in search space. Then, based on the bidirectional relationship learned from Bi-Arch2Perf, NACG employs the architecture generator to discover high-performance architecture space to qualitatively generate neural architectures with high performance and the performance predictor to quantitatively predict the performance of these architectures for designing promising neural architectures. Finally, the experimental evaluations conducted on NAS-Bench-101, NAS-Bench-201, and NAS-Bench-301 demonstrate that SemiGAN-NAD successfully discovers high-performance neural architectures that can compete with the latest methods in terms of accuracy, while requiring less evaluation time. Furthermore, the experimental results also indicate that the architecture signatures of the high-performance neural networks obtained through our method are close to that of top architectures on the NAS datasets.

2 RELATED WORKS

The goal of NAS is to quickly and accurately sample high-performance architectures from the search space. The reinforcement learning based NAS method (Zoph and Le, 2017) and the evolutionary algorithm based NAS method (Peng et al., 2023) can discover architectures that are superior to those designed manually on the datasets. However, the search process of these methods is based on the discrete architecture search space, resulting in low efficiency. Consequently, NAO (Luo et al., 2018) maps the discrete encoding of the architecture to a continuous space through LSTM and predicts the accuracy of the architecture representation through performance predictor. It performs architecture optimization in the continuous space through gradient ascent to find high-performance architectures. On this basis, SeimNAS (Luo et al., 2020) uses a semi supervised strategy to train a performance predictor. It employs some architectures with performance labels to train the initial performance predictor. Then, it conducts the performance prediction on unlabeled architectures and integrates the outcomes into the training set to retrain the performance predictor. Arch2Vec (Yan et al., 2020) utilizes unsupervised architecture representation learning to construct the continuous space, learning architecture representations without their labels. This helps to map architectures with similar performance to the same regions in the latent space, improving the efficiency of downstream architecture search. Recently, NAS methods based on generation strategy have become popular. GA-NAS (Rezaei et al., 2021) incorporates adversarial learning, where its generator is iteratively trained to continuously sample in more important regions of the search space and learn the distribution of winning architectures. Its discriminator distinguish the winning architectures from randomly generated ones in each iteration. Similarly, AG-Net (Lukasik et al., 2022) optimizes the architecture representation space via weighted retraining for efficient architecture search, using only generator and performance predictor.

Compared with the above methods of learning the conditional distribution from architecture to performance, we incorporate the performance predictor into the training process of GAN and learn the architecture-performance joint distribution in a semi supervised manner to explore the bidirectional relationship between architecture and performance. Then we use conditional generation to find high-performance architectures.

3 METHODS

This paper proposes SemiGAN-NAD, as shown in Figure 1, to efficiently design high-performance neural architecture based on bidirectional architecture-performance relationship. It is composed of Bidirectional Transformer of Architecture and Performance (Bi-Arch2Perf) and Neural Architecture Conditional Generation (NACG). Inspired by TripleGAN (Li et al., 2017), Bi-Arch2Perf is built on the adversarial training of Architecture Generator (AG), Performance Predictor (PP), and Discriminator (D) under semi-supervised learning to learn the joint distribution of architecture and performance. Based on the Bi-Arch2Perf, NACG is proposed for efficient architecture design through high-performance architecture generation by AG and performance prediction by PP.

3.1 Bi-Arch2Perf: Bidirectional Transformer of Architecture and Performance

3.1.1 Extended Adversarial Training for Learning Bidirectional Relationship

Suppose that a neural network can be viewed as Directed Acyclic Graph consisting of N nodes and E edges, where each node represents the operation of a layer in neural network (e.g., convolution, pooling), and each edge represents the connection between nodes. Let us denote a neural architecture to be adjacency matrix $A \in R^{N\times N}$ that represents edge connection, and one-hot operation matrix $X \in R^{N\times E}$ that represents the category of node operation. The joint distribution of architecture and performance can be decomposed into two conditional distributions as follows:

$$p(A, X, y) = p(y)p(A, X|y), \qquad (1)$$

$$p(A,X,y) = p(A,X)p(y|A,X), \quad (2)$$

where y represents the performance of neural network. $p(A,X|y)$ and $p(y|A,X)$ are captured by Architecture Generator AG and Performance Predictor PP respectively. When a neural architecture is drawn from $p(A,X)$, PP produces a pseudo performance following the conditional distribution $p(y|A,X)$. Therefore, a pseudo architecture-performance pair can be viewed as sampling from the joint distribution $p_{pp}(A,X,y) = p(A,X)p(y|A,X)$. Similarly, a pseudo architecture-performance pair also can be sampled from $p_{AG}(A,X,y) = p(y)p(A,X|y)$. It first draws y from $p(y)$, and transforms the architecture latent variables z_A, operation latent variable z_X and conditions y to neural architecture, i.e.,$(A,X) = AG(y,z_A,z_X)$ where $z_A \sim p_z(z_A)$ and $z_X \sim p_z(z_X)$ are sampled from prior distribution. In order to estimate these two conditional distributions jointly, discriminator D takes the pseudo architecture-performance pairs produced by PP and AG as negative samples, and the ones sampled from the real architecture-performance distribution as positive samples. Then, D distinguishes whether an input pair is a positive sample. The joint distribution learned by Bi-Arch2Perf is defined as:

$$p(A,X,y) = \alpha p_{PP}(A,X,y) + (1-\alpha)p_{AG}(A,X,y), \quad (3)$$

where $p_{pp}(A,X,y)$ and $p_{AG}(A,X,y)$ represent the joint distribution captured by PP and AG. $\alpha \in (0,1)$ is a constant that controls the relative importance of them. The adversarial training objective of Bi-Arch2Perf can be defined as follows:

$$\min_{PP,AG} \max_{D} V(PP,AG,D) = \lambda_1 + \alpha\lambda_2 + (1-\alpha)\lambda_3, \quad (4)$$

$$\begin{aligned}\lambda_1 &= \mathrm{E}_{(x,y)\sim p(x,y)}[\log D(x,y)],\\ \lambda_2 &= \mathrm{E}_{(x,y)\sim p_{PP}(x,y)}[\log(1-D(x,PP(x)))],\\ \lambda_3 &= \mathrm{E}_{(x,y)\sim p_{AG}(x,y)}[\log(1-D(AG(y,z),y))],\end{aligned} \quad (5)$$

where $x = (A,X)$ and $z = (z_A, z_X)$. λ_1 denotes that D maximizes its ability to identify positive samples, i.e., architectural-performance pairs in the dataset. λ_2 denotes that PP enables negative samples it produces to be discriminated as positive samples by D. PP provides performance labels for given architectures, which are then fed into D as negative samples. λ_3 denotes that AG enables it to generate negative samples that can be discriminated by D as positive samples. Specifically, AG produces architectures that satisfy the given performance constraints, which are then treated as negative samples by D.

3.1.2 Semi-supervised Architecture Loss for Conveying Unlabeled Architecture Information

Since the expensive cost of annotating a neural architecture with its real performance, the training set for Bi-Arch2Perf is insufficient. Several valid neural architectures without performance annotations can easy to be sampled from search space. In order to utilize the invaluable information of these architectures, semi-supervised learning is integrated to optimize the training process of Bi-Arch2Perf. Intuitively, a good performance predictor PP can provide labeled architecture-performance data through predicting performance for architectures without annotations beyond the training set, thereby promoting the training process of architecture generator AG and discriminator D. Thus, the objective function of D is redefined as:

$$\max_{D} \beta_1 + \alpha\beta_2 + (1-\alpha)\beta_3, \quad (6)$$

$$\begin{aligned}\beta_1 &= \mathrm{E}_{(x,y)\sim p(x,y)}[\log D(x,\hat{y})],\\ \beta_2 &= \mathrm{E}_{x_u\sim p_u(A,X)}[\log(1-D(x_u,\hat{y}_{pp}))],\\ \beta_3 &= \mathrm{E}_{y\sim p(y)}[\log(1-D(AG(\hat{y},z_A,z_X),\hat{y}))],\end{aligned} \quad (7)$$

where $x = (A,X)$ and $x_u = (A_u, X_u)$. A and X are the adjacency matrix and operation matrix of architecture with annotation respectively. A_u and X_u represent the ones of architecture without annotation. $\hat{y}$ represents the performance rank of architecture and $\hat{y}_{pp}$ represents the performance rank corresponding to the output of PP. β_1 trains D to identify real samples as positive samples, using real architecture-performance pairs of the dataset. β_2 trains D to identify negative samples provided by performance predictor PP as negative samples. In negative samples, architectures are x_u and performance labels are $\hat{y}_{pp}$ representing the performance ranks of x_u. β_3 trains D to identify negative samples provided by architecture generator AG as negative samples. β_2 and β_3 can incorporate the information from architectures without annotation as adversarial loss. Moreover, the goal of PP is to accurately predict the performance of architecture. Therefore, its objective function consists of two parts: adversarial loss and performance prediction loss, which are defined as follows:

$$\min_{PP} \alpha\gamma_1 + \gamma_2 + \alpha_p\gamma_3, \quad (8)$$

$$\begin{aligned}\gamma_1 &= \mathrm{E}_{x_u\sim p_u(A,X)}[\log(1-D(x_u,\hat{y}_{pp}))],\\ \gamma_2 &= \mathrm{E}_{(A,X,y)\sim p(A,X,y)}\|y-PP(A,X)\|_2^2,\\ \gamma_3 &= \mathrm{E}_{y\sim p(y)}\|y-PP(AG(z_A,z_X,\hat{y}))\|_2^2,\end{aligned} \quad (9)$$

where $\alpha_p \in (0,1)$ is a constant that balances the prediction accuracy of real architecture-performance pairs and pseudo ones. γ_1 is adversarial loss, which trains PP to provide negative samples that can be classified as positive samples by D. γ_2 is one part of performance prediction loss, which minimizes the difference between predicted and real performance of architectures. γ_3 is another part of performance prediction loss, in which PP predicts the performance for architectures generated by AG under performance conditions $\hat{y}$ corresponding to y. The purpose of γ_3 is to accurately predict the performance of architectures generated by AG. Here, Mean Square Error (MSE) is used to compute the loss. Moreover, the objective function of AG is defined as follows:

$$\min_{AG}(1-\alpha)\mathrm{E}_{y\sim p(y)}[\log(1-D(AG(\hat{y}, z_A, z_X), \hat{y}))]. \quad (10)$$

It represents AG generates architectures under performance conditions, and inputs pseudo architecture-performance pairs to D for obtaining the adversarial loss. It trains AG to make negative samples provided by AG convince D. Herein, the joint distribution can be learned through the adversarial training of Bi-Arch2Perf.

3.2 NACG: Neural Architecture Conditional Generation

In order to design high-performance neural architecture efficiently, Neural Architecture Conditional Generation (NACG) is proposed based on Bi-Arch2Perf. Specially, Bi-Arch2Perf is built based on N architecture with performance annotation and the remaining architecture without annotation in search space. Architecture generator AG is to qualitatively generate architecture under the performance condition within a certain range instead of a specific value. Therefore, the performance of N architectures is discretized to obtain the corresponding categories (0-4). Category 0 indicates the category with the highest performance. The discretization can be defined as follows:

$$\hat{y} = \begin{cases} \lfloor (t + 0.02 - y)/0.02 \rfloor, & y > t - 0.06 \\ 4, & y \leq t - 0.06 \end{cases}, \quad (11)$$

where y represents the real performance (validation or test accuracy for different NAS datasets), and $\hat{y}$ represents the discretized category that implies the performance rank of an architecture. t represents the customized bottom bound of category 0 depending on different datasets (e.g., set 0.93 for NAS-bench-101). Based on the discrete performance, AG performs qualitative architecture generation under performance conditions.

When Bi-Arch2Perf has been trained, NACG generates architectures under given high-performance conditions through AG, and predicts the performance of architectures through PP. Then, it selects top K neural architectures with the highest predicted performance for final validation. These K neural networks are trained on the training set and evaluated on validation or test set to get accuracy in datasets. Finally, NACG selects the best architecture with the highest accuracy. This strategy can improve the robustness of NACG, since it allows NACG to utilize a more reliable measurement to select the final neural architecture. Moreover, NACG produces architectures in high-performance space learned by AG. Therefore, the number of architectures for final evaluation can be small.

4 EXPERIMENTS

4.1 Experiments Datasets

The experiments are conducted on NAS benchmarks, including NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong and Yang, 2020) and NAS-Bench-301 (Zela et al., 2022), to evaluate the effectiveness of SemiGAN-NAD. Details of datasets are provided in Supplementary Materials Section A.

4.2 Experiments Setup

The experiments from two aspects are conducted: neural architecture design and architecture generation respectively. In the neural architecture design comparison experiments (Section 4.3.1), 0.025% (105) architectures with performance annotation are sampled from NAS-Bench-101. The number of final evaluation architectures K is set as 5. The constants α and α_p in equations of Section 3.1 are both set as 0.5. To execute experiments on NAS-Bench-301, following arch2vec (Yan et al., 2020), normal cells and reduction cells are the same, and we randomly sample 600,000 unique architectures in this search space. Following AG-NET (Lukasik et al., 2022), performance queries are generated by the XGBoost surrogate model of NAS-Bench-301. The architecture generation experiments are discussed in Section 4.3.2 and Section 4.3.3. More implementation details, including training process information, model information, and technical limitations, can be found in Supplementary Materials Section C.

4.3 Experimental Results and Analysis

4.3.1 Comparison of Neural Architecture Design Performance

The goal of SemiGAN-NAD is to efficiently find high-performance neural architecture through conditional architecture generation and performance prediction. In order to verify the efficiency and effectiveness of SemiGAN-NAD for finding high-performance architecture, we compare our method with traditional NAS methods including Random Search (RS), Reinforcement Learning (RL), Bayesian Optimization (BO), Regularized Evolution (RE) and Gradient Descent (NAO (Luo et al., 2018)), as well as the latest methods such as SeimNAS (Luo et al., 2020), Arch2Vec (Yan et al., 2020), GA-NAS (Rezaei et al., 2021), AG-Net (Lukasik et al., 2022) and PRE-NAS (Peng et al., 2023). Lukasik et al. (2022) and Yan et al. (2020) provide the experimental results for traditional methods, and we reimplement some latest methods. NAS-Bench-{101, 201, 301} is used to query the accuracy of architectures. The experiment results are shown in Table 1, Table 2 and Table 3. For all methods in the tables, we report their best results.

Table 1: The test accuracy comparison of CIFAR-10 on NAS-Bench-101. '*' Obtained through our replication. The bolded numbers mark the optimal query times and test accuracy.

Methods	#Queries	Test Acc (%)	Search Strategy
RS	1000	93.54	Random
RL	1000	93.58	Reinforce
BO	1000	93.72	Bayesian
RE	1000	93.72	Evolution
NAO	1000	93.74	Gradient Decent
SemiNAS*	**110**	93.24	-
Arch2vec-RL*	150	93.78	Reinforce
Arch2vec-BO*	150	93.76	Bayesian
GA-NAS	378	**94.23**	Generation
AG-Net	192	94.18	Generation
SemiGAN-NAD(ours)	**110**	94.06	Generation

The second column represents the number of queries in NAS-Bench-101, NAS-Bench-201 and NAS-Bench-301, which can be equivalent to the architecture evaluation process. Table 1 shows the comparison results on NAS-Bench-101. It can be seen that SemiGAN-NAD achieves 94.06% test accuracy, which increased by 0.52%, 0.48%, 0.34%, 0.34%, 0.32% than RS, RL, BO, RE, NAO methods respectively. SemiGAN-NAD has an absolute advantage in terms of query times, using only 110 queries, while these traditional methods use 1000 queries, resulting in a reduction of over 89% in queries. Compared with the latest methods, SemiGAN-NAD outperforms Arch2Vec and SemiNAS by 0.28% and 0.82% respectively. Although SemiNAS also incorporates the semi-supervised strategy, it is almost only a performance predictor. The advantage of SemiGAN-NAD lies in its ability to effectively generate high-performance architectures via capturing the architecture-performance joint distribution. Concurrently, SemiGAN-NAD achieves similar test accuracy to the state-of-the-art method (GA-NAS) with a gap of less than 0.17%, but our unique advantage stems from leveraging semi-supervised learning to mitigate annotation costs. Therefore, SemiGAN-NAD uses a minimum number of queries to obtain similar or higher accuracy than most of the comparison methods, demonstrating its competitiveness against the best method.

To understand whether SemiGAN-NAD can generalize to different datasets, it is compared with the test accuracy of NAS methods on NAS-Bench-201, which has architecture performance pairs for CIFAR-10, CIFAR-100 and ImageNet16-120 datasets. In Table 2, compared with traditional methods, the test accuracy of SemiGAN-NAD is higher than RS, RL and BO on CIFAR-10, and RS, RL on CIFAR-100. On ImageNet16-120, it can be seen that SemiGAN-NAD achieves an improvement of 0.49% test accuracy than RE with less computation. Compared with the latest methods, SemiGAN-NAD achieves higher test accuracy than SemiNAS on three datasets, with the same queries. Meanwhile, SemiGAN-NAD is similar to the state-of-the-art methods (Arch2vec-RL on CIFAR-10 and CIFAR-100, GA-NAS on ImageNet16-120) with test accuracy. Particularly, the difference between SemiGAN-NAD and the best method on ImageNet16-120 is only 0.33%, and 0.32% on CIFAR-10. Notably, SemiGAN-NAD requires minimal queries in all methods. Therefore, SemiGAN-NAD provides an effective and efficient neural architecture design strategy.

To further evaluate the performance, we evaluate the performance of SemiGAN-NAD on NAS-Bench-301, as shown in Table 3. NAS-Bench-301 has a search space of at least 10^9 architectures, which is multiple orders of magnitude larger than those of other benchmarks. Its search space is harder to learn, but closer to reality. The validation accuracy achieved by SemiGAN-NAD is better than almost all baselines, with the same or fewer queries. Although AG-Net maintains the highest validation accuracy, SemiGAN-NAD differs from it by only 0.06% and requires fewer queries.

Test/Val Acc is the crucial criterion for evaluating the performance of NAS methods, there is an upper limit to the accuracy of the NAS datasets (Lukasik et al., 2022; Peng et al., 2023). The difference between the best results of the latest NAS methods and the upper limit are mostly no more than 1%. This is a reason why the results of our method do not seem significant enough, but our exploration of generating high-performance architectures is significant. SemiGAN-NAD provides a new idea of learning architecture representation space by learning the joint distribution of architecture and performance, which is sufficient to compete with the best methods. It has a notable result in large search space (NAS-Bench-301). Although most methods on small search spaces (NAS-Bench-{101, 201}) are prone to learn the implicit laws of datasets and thus obtain favorable results, our method also has some advantages. We believe that this may provide a new insight for architecture design.

Table 2: The test accuracy comparison on NAS-Bench-201.

Methods	#Queries	Test Acc (%)			Search Strategy
		CIFAR-10	CIFAR-100	ImageNet16-120	
RS	400	94.12	72.82	46.12	Random
RL	400	94.13	72.92	46.16	Reinforce
BO	400	94.13	73.27	46.06	Bayesian
RE	400	94.27	73.41	46.27	Evolution
NAO	-	-	-	-	Gradient Decent
SemiNAS*	**160**	93.68	72.20	46.03	-
Arch2vec-RL	400	**94.54**	**73.93**	46.54	Reinforce
Arch2vec-BO	400	94.42	73.67	46.64	Bayesian
GA-NAS	444	94.39	73.45	**47.09**	Generation
AG-Net	400	94.37	73.51	46.42	Generation
PRE-NAS	500	94.38	73.24	46.37	Evolution
SemiGAN-NAD(ours)	**160**	94.22	73.22	46.76	Generation

Table 3: The validation accuracy comparison of CIFAR-10 on NAS-Bench-301.

Methods	#Queries	Val. Acc (%)	Search Strategy
RS	192	94.31	Random
RL	-	-	Reinforce
BO	192	94.71	Bayesian
RE	192	94.75	Evolution
NAO	-	-	Gradient Decent
SemiNAS*	**150**	94.37	-
Arch2vec-RL*	**150**	94.54	Reinforce
Arch2vec-BO*	**150**	94.54	Bayesian
GA-NAS	-	-	Generation
AG-Net	192	**94.79**	Generation
PRE-NAS	-	-	Evolution
SemiGAN-NAD(ours)	**150**	94.73	Generation

4.3.2 Analysis of Generated High-performance Architecture

The architecture generator *AG* in Bi-Arch2Perf can find high-performance architecture space for SemiGAN-NAD. In order to verify its results, 5000 architecture latent variables and operation latent variables are sampled from standard normal distribution to generate neural architectures with class 0 (high-performance) as the condition.

The valid architectures that conform to the constraints of NAS-Bench-101 search space account for 36.73% in generated architectures. Thus, the valid architectures are compared with architectures in NAS-Bench-101. Table 4 gives the average proportion of the valid architectures with validation accuracy greater than 93% and 93.5% respectively, i.e., 70.30% and 65.58%, which are much higher than NAS-Bench-101 (only 15.49% and 7.34%). Similarly, we also count the proportions of architectures with test accuracy greater than 93% and 93.5%. They are 56.53% and 9.92% higher than

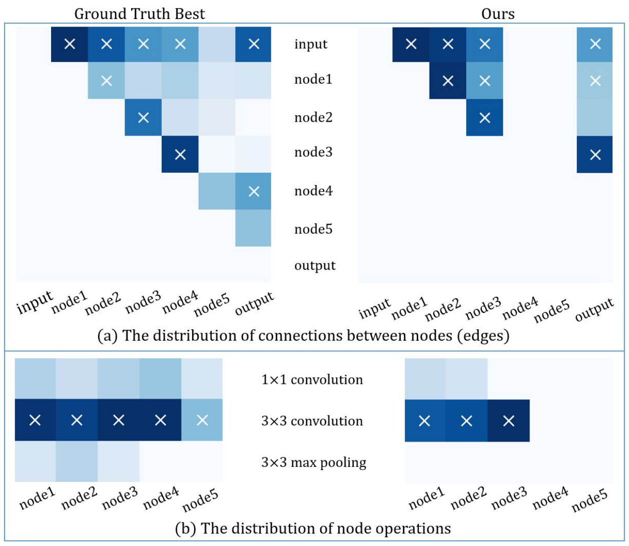

Figure 2: Comparison of top architectures generated by SemiGAN-NAD with ground truth on NAS-Bench-101 CIFAR-10 test accuracy. 'X' marks operations with the highest probability for nodes.

those in NAS-Bench-101. Therefore, it can be concluded that *AG* can obtain high-performance architecture space under the given architecture condition, which can improve the effectiveness of the architecture design of SemiGAN-NAD. Additional detailed analysis is provided in Supplementary Material Section B.

4.3.3 Distribution Analysis of Generated High-performance Architectures

Following the statistical approach of L^{2}NAS (Mills et al., 2021), we compare high-performance architectures generated by our method with ground-truth

Table 4: The comparison of generated architectures under the high-performance condition and the architectures in NAS-Bench-101.

Arch. space	Val.Acc>93%	Val.Acc>93.5%	Test Acc>93%	Test Acc>93.5%
NAS-Bench-101	15.49%	7.34%	6.46%	1.03%
Ours	70.30%	65.58%	62.99%	10.95%

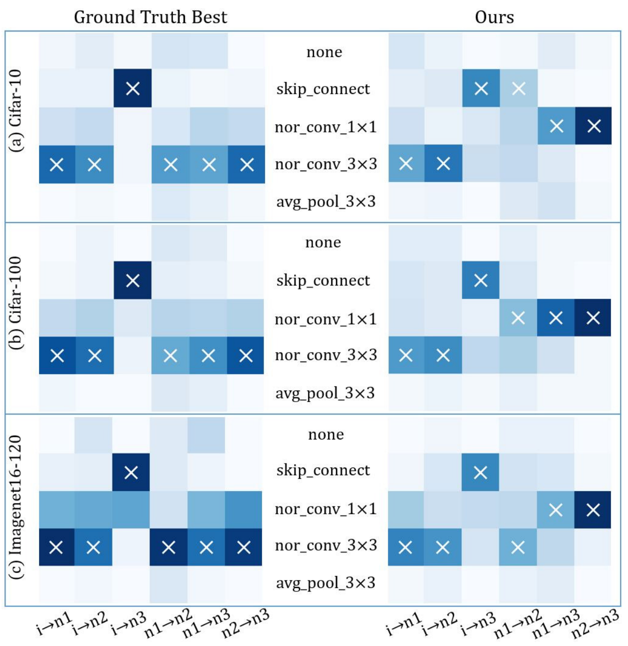

Figure 3: Comparison of top architectures generated by SemiGAN-NAD with ground truth on NAS-Bench-201 test accuracy. Each figure indicates the distribution of edge operations. Rows indicate operations, columns indicate edges. 'i' means input, and 'n' denotes an intermediate node.

best architectures. Figure 2 and Figure 3 illustrate the comparison results on NAS-Bench-101 and NAS-Bench-201 respectively. The number of top architectures is set to 64, and the values are normalized between 0 and 1 by dividing by the maximum value. Darker color indicates higher values. 'X' marks the high probability edges and nodes, associated with the best architecture.

In Figure 2(a), the top-9 edges with the highest values are marked since NAS-Bench-101 allows up to 9 edges in a cell. Although the edge distribution of SemiGAN-NAD hits 6 out of 9 marks 'X' in the ground truth, SemiGAN-NAD tends to find high-performance architectures in relatively compact structures. In Figure 2(b), the operations of the highest probability on three nodes remain consistent. Therefore, the distributions produced by top-64 architectures are close to the ground truth. It implies that SemiGAN-NAD can effectively generate high-performance and lightweight architectures. In Figure 3, we mark the most possible operation on each edge, and the distributions of edge operations have a small gap with the ground truth. In Figure 3(a) and (b), SemiGAN-NAD and ground truth agree on the first three of the six edges, and both suggest that the first three edges of the high-performance architecture should use nor_conv_3×3 or skip_connect. On the last two edges, SemiGAN-NAD prefers nor_conv_1×1. In Figure 3(c), SemiGAN-NAD hits 4 out of 6 marks in the ground truth. The joint distribution learned by SemiGAN-NAD is limited by the distribution of the sampled data, thus our generator generates architectures that are optimal for the sampled data within the learned distribution. This may be the reason why our generated architectures contain more nor_conv_1×1. Furthermore, our results also provide evidence that SemiGAN-NAD is capable of generating architectures with certain performance advantages while utilizing fewer parameters.

5 CONCLUSION AND FUTURE WORK

In order to efficiently design high-performance architecture, it is necessary to understand the relationship between architecture and performance. Here, we propose SemiGAN-NAD, which is composed of Bi-Arch2Perf and NACG. Bi-Arch2Perf is constructed with the adversarial training of performance predictor, architecture generator and discriminator to capture the joint distribution from conditional distribution of architecture and performance in both two directions. NACG can predict the performance of architecture in high-performance architecture space to efficiently discover well-promising neural architectures. The experiment results show that SemiGAN-NAD obtains comparable accuracies compared with the latest NAS methods when using a small number of queries in NAS-Bench-101, NAS-Bench-201 and NAS-Bench-301. Meanwhile, the high-performance architecture signatures we obtained are similar to those of the top architectures in the datasets. For future work, the causal approach can be introduced to explain the relationship between architecture and performance, and then improve the interpretability of NAS.

Acknowledgements

This work is supported by the Programs of National Natural Science Foundation of China (No. 52073169 and No. 92270124) and the National Key Research and Development Program of China (Grant No. 2021YFB3802101).

References

Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20:55:1–55:21.

Han, C., Pan, S., Que, W., Wang, Z., Zhai, Y., and Shi, L. (2022). Automated localization and severity period prediction of myocardial infarction with clinical interpretability based on deep learning and knowledge graph. *Expert Syst. Appl.*, 209:118398.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Joseph, K. J., Khan, S. H., Khan, F. S., and Balasubramanian, V. N. (2021). Towards open world object detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5830–5840. Computer Vision Foundation / IEEE.

Li, C., Xu, T., Zhu, J., and Zhang, B. (2017). Triple generative adversarial nets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4088–4098.

Li, J., Liu, Y., Liu, J., and Wang, W. (2020). Neural architecture optimization with graph VAE. *arXiv preprint arXiv:2006.10310*.

Li, Y., Wen, Z., Wang, Y., and Xu, C. (2021). One-shot graph neural architecture search with dynamic search space. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 8510–8517. AAAI Press.

Liu, H., Simonyan, K., and Yang, Y. (2019). DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lukasik, J., Jung, S., and Keuper, M. (2022). Learning where to look - generative NAS is surprisingly efficient. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXIII*, volume 13683 of *Lecture Notes in Computer Science*, pages 257–273. Springer.

Luo, R., Tan, X., Wang, R., Qin, T., Chen, E., and Liu, T. (2020). Semi-supervised neural architecture search. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Luo, R., Tian, F., Qin, T., Chen, E., and Liu, T. (2018). Neural architecture optimization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7827–7838.

Mills, K. G., Han, F. X., Salameh, M., Rezaei, S. S. C., Kong, L., Lu, W., Lian, S., Jui, S., and Niu, D. (2021). L2NAS: learning to optimize neural architectures via continuous-action reinforcement learning. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 1284–1293. ACM.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Peng, Y., Song, A., Ciesielski, V., Fayek, H. M., and Chang, X. (2023). PRE-NAS: evolutionary neural architecture search with predictor. *IEEE Trans. Evol. Comput.*, 27(1):26–36.

Rezaei, S. S. C., Han, F. X., Niu, D., Salameh, M., Mills, K. G., Lian, S., Lu, W., and Jui, S. (2021). Generative adversarial neural architecture search. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 2227–2234. ijcai.org.

Ru, B. X., Wan, X., Dong, X., and Osborne, M. A. (2021). Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society.

Tang, Y., Wang, Y., Xu, Y., Chen, H., Shi, B., Xu, C., Xu, C., Tian, Q., and Xu, C. (2020). A semi-supervised assessor of neural architectures. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1807–1816. Computer Vision Foundation / IEEE.

Wen, W., Liu, H., Chen, Y., Li, H. H., Bender, G., and Kindermans, P. (2020). Neural predictor for neural architecture search. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX*, volume 12374 of *Lecture Notes in Computer Science*, pages 660–676. Springer.

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Wang, L., Chen, Z., Xiao, A., Chang, J., Zhang, X., and Tian, Q. (2022). Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Comput. Surv.*, 54(9):183:1–183:37.

Xu, Y., Wang, Y., Han, K., Chen, H., Tang, Y., Jui, S., Xu, C., Tian, Q., and Xu, C. (2019). RNAS: architecture ranking for powerful networks. *arXiv preprint arXiv:1910.01523*.

Yan, S., Zheng, Y., Ao, W., Zeng, X., and Zhang, M. (2020). Does unsupervised architecture representation learning help neural architecture search? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. (2019). Nas-bench-101: Towards reproducible neural architecture search. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114. PMLR.

You, J., Leskovec, J., He, K., and Xie, S. (2020). Graph structure of neural networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10881–10891. PMLR.

Zela, A., Siems, J. N., Zimmer, L., Lukasik, J., Keuper, M., and Hutter, F. (2022). Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. Computer Vision Foundation / IEEE Computer Society.

Efficient Neural Architecture Design via Capturing Architecture-Performance Joint Distribution Supplementary Materials

A NAS Datasets

NAS-Bench-101 contains 423K neural architectures, which are both trained on CIFAR-10 for image classification from scratch to full convergence to get validation and test accuracy. NAS-Bench-101 is a cell-based search space. The cell can take on DAG structure, which consists of at most 7 nodes and 9 edges with the first node as input and the last node as output. The operation type of intermediate nodes can be selected from 3×3 convolution (kernel size is 3), 1×1 convolution and 3×3 max pooling. Moreover, in NAS-Bench-101, the architecture performance evaluation is repeated over 3 random initialization, and the average validation accuracy of three times is used to build Bi-Arch2Perf. The accuracy query of architecture from NAS-Bench-101 is equivalent to the architecture evaluation process.

NAS-Bench-201 contains all 15,625 unique neural architectures in the search space. The search space is an acyclic directed graph with 4 nodes and 6 edges. Each edge corresponds to an operation selected from the set of 5 possible options: conv 1×1, conv 3×3, avgpool 3×3, skip-connect and zeroize. This search space is applicable to almost all up-to-date NAS algorithms. Note although the search space of NAS-Bench-201 is more general, it is smaller than that of NAS-Bench-101. Each architecture is trained for 200 epochs and evaluated on 3 image datasets: CIFAR10, CIFAR100, ImageNet16-120. The evaluation is repeated over 3 random initialization seeds. We can access the training accuracy/loss, validation accuracy/loss after every training epoch, the final test accuracy/loss, the number of parameters as well as FLOPs from the dataset. The dataset and its API can be downloaded from https://github.com/D-X-Y/NAS-Bench-201.

NAS-Bench-301 provides a surrogate benchmark, which allows for fast evaluation of NAS methods on the DARTS (Liu et al., 2019) search space by querying the validation accuracy on CIFAR-10 image classification task. In NAS-Bench-301, each cell has 7 nodes (2 input nodes, 4 intermediate nodes and 1 output node) and 8 non-empty operation edges. The node represents feature maps, and the edge uses one of the following operations: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling and skip connection. The input nodes receive outputs from the previous cells, each intermediate node has two incoming edges and the output node receives the outputs of all intermediate nodes. About 10^{18} architectures exist in the search space.

B Analysis of Generated High-performance Architecture

In order to discover the architecture signatures of high-performance neural networks and interpret the neural network modeling process, the valid architectures generated by *AG* are statistically analyzed from two aspects: (1) the cell whether includes the operation of 3×3 max pooling, 1×1 convolution and 3×3 convolution. (2) skip connection: the shortest path length from the input node to the output node. Figure 4 shows the results of 3×3 max pooling, 1×1 convolution and 3×3 convolution in generated architectures respectively. It can be observed that most generated high-performance architectures contain 3×3 convolution (1373), which accounts for 74.78% in valid architectures. 1×1 convolution and 3×3 max pooling only account for 5.88% (108) and 3.21% (59). Therefore, this result indicates that compared with 3×3 max pooling and 1×1 convolution, 3×3 convolution appears more frequently in high-performance neural architectures, and it can also help the neural network to obtain better performance.

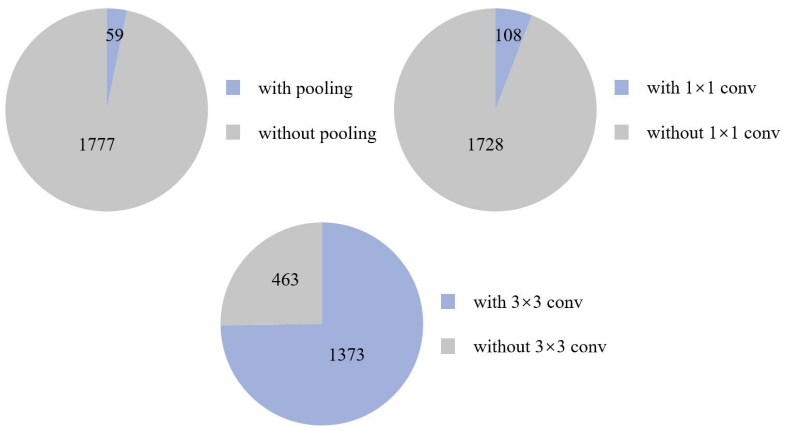

Figure 4: Generated architectures whether include 3×3 max pooling operation, 1×1 convolution operation and 3×3 convolution operation.

Next, the shortest path length of the input and output node in valid architectures is calculated by Breadth-First Search in graph. From Figure 5, it illustrates that the architectures with the shortest path length of 1 account for 70.81% (1300) in generated architecture, and 2 account for 24.73%. Moreover, the longer the shortest path length is, the smaller the ratio in generated architectures is, and AG does not generate architectures with the shortest path length of 5 and 6 under the high-performance condition. Therefore, it can be concluded that the generated high-performance architectures tend to choose the neural architecture with a shorter shortest path length between the input and output node, which indicates the importance of skip connection. It inspires us to add skip connection operation appropriately when designing neural architecture.

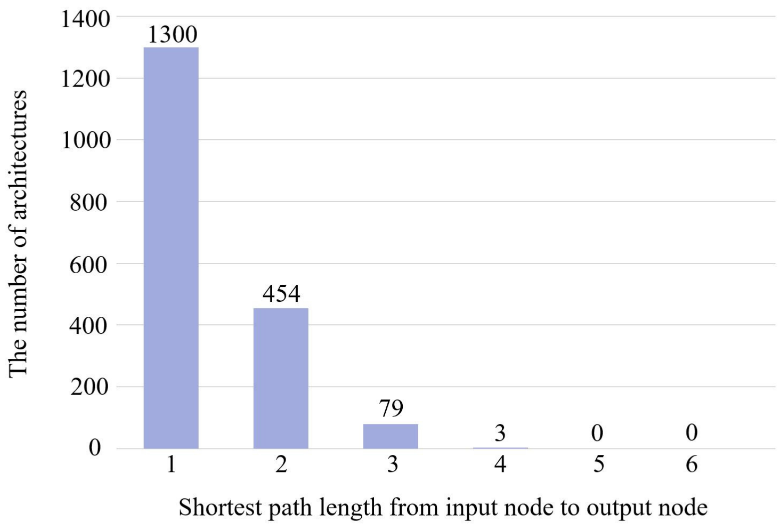

Figure 5: The comparison of the shortest path length of the input and output nodes in generated architectures by AG.

Moreover, three architectures with the highest test accuracy in NAS-Bench-101 are visualized in Figure 6 to verify the correctness of the architecture signatures found by AG. For the rank 1 neural architecture (94.32%), the shortest path length of the input and output node is 1, and the input node is connected with the other four nodes in this architecture. Moreover, three of the five variable nodes adopt the 3×3 convolution operation. For the rank 2 neural architecture, the shortest path length of the input and output node is 1, and there are connections from the input node to all other nodes in this architecture. Meanwhile, the 3×3 convolution operation is adopted in all variable nodes. For rank 3 neural architecture, the shortest path length between the input and output node is also 1, and the input node is connected to all other nodes in this architecture. Two of the three variable nodes chose the 3×3 convolution operation. Therefore, it can be concluded that the high-performance neural architectures are more inclined to add skip connection and 3×3 convolution compared with 1×1 convolution and 3×3 max pooling operation, which verifies the correctness of the high-performance architecture space found by AG, thereby promoting the neural architecture design of SemiGAN-NAD.

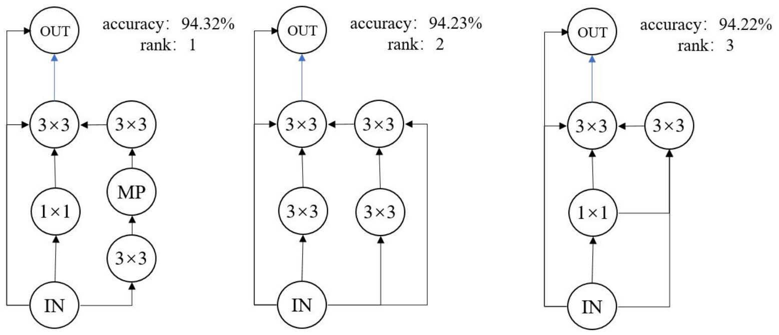

Figure 6: The top three neural architectures with the highest test accuracy in NAS-Bench-101.

C Implementation Details

For the training process information, we train the model components alternately when learning the architecture-performance joint distribution. The discriminator, the architecture generator and the performance predictor are trained once in turn in each batch of every epoch. Due to the rapid convergence of the loss, the entire model can be trained for a maximum of 60 epochs and a minimum of 40 epochs. In each batch, the architecture generator provides the same number of negative samples as the number of samples with performance labels. The adversarial loss is calculated using the cross-entropy loss. In NAS-Bench-101, the validation accuracy of the architecture is used for training and evaluating the output of the condition generation, and finally the test accuracy corresponding to the validation accuracy is used as the result of the architecture search. In NAS-Bench-201, test accuracy is employed in the whole process. In NAS-Bench-301, only the validation accuracy can be accessed. We use a NVIDIA GTX 1660 Ti in all experiments.

For the model information, the discriminator extracts the features of the architecture using a 3-layer Graph Isomorphism Network (GIN) and connects the features obtained from each layer to obtain the architecture feature. The architecture feature is concatenated with the embedding of the architecture performance rank to obtain the sample representation. The discriminator then utilizes the linear layers and the sigmod activation function to judge the probability that the sample is a positive sample. The settings of GIN follows arch2vec (Yan et al., 2020). The architecture generator generates an architecture by sampling architecture latent variables from the standard normal distribution, along with a given conditional performance rank. The dimensions of the latent variable and the embedding performance rank are both 16. The architecture generator first transforms the input into the architecture. To further ensure the validity of the generated architecture, a 5-layer GIN is employed for feature extraction. Subsequently, the extracted feature is transformed into an architecture. The transformation operation uses the linear layers. The performance predictor uses a 4-layer GIN and connects the features of each layer. It outputs the architecture performance prediction by the linear layers.

In terms of the technical limitations, our method involves conditional generation of architecture. The architecture latent variables are randomly sampled from distributions. Compared with other methods, this process may have lower scalability efficiency in new datasets or downstream tasks. However, the rich literature on the search strategy may inspire us to make improvements in the future.