A Generic Graph-Based Neural Architecture Encoding Scheme for Predictor-Based NAS

Xuefei Ning[1], Yin Zheng[2], Tianchen Zhao[3], Yu Wang[1(✉)], and Huazhong Yang[1]

[1] Department of Electronic Engineering, Tsinghua University, Beijing, China
yu-wang@tsinghua.edu.cn
[2] Weixin Group, Tencent, Shenzhen, China
foxdoraame@gmail.com
[3] Department of Electronic Engineering, Beihang University, Beijing, China

Abstract. This work proposes a novel Graph-based neural ArchiTecture Encoding Scheme, a.k.a. GATES, to improve the predictor-based neural architecture search. Specifically, different from existing graph-based schemes, GATES models the operations as the transformation of the propagating information, which mimics the actual data processing of neural architecture. GATES is a more reasonable modeling of the neural architectures, and can encode architectures from both the "operation on node" and "operation on edge" cell search spaces consistently. Experimental results on various search spaces confirm GATES's effectiveness in improving the performance predictor. Furthermore, equipped with the improved performance predictor, the sample efficiency of the predictor-based neural architecture search (NAS) flow is boosted.

Keywords: Neural architecture search (NAS) · Predictor-based NAS

1 Introduction

Recently, Neural Architecture Search (NAS) has received extensive attention due to its capability to discover neural network architectures in an automated manner. Substantial studies have shown that the automatically discovered architectures by NAS are able to achieve highly competitive performance.

Generally speaking, there are two key components in a NAS framework, the *architecture searching module* and the *architecture evaluation module*. Specifically, the architecture evaluation module provides the signals of the architecture performance, e.g., accuracy, latency, etc., which are then used by the architecture searching module to explore architectures in the search space. In the seminal work of [25], the architecture evaluation is conducted by training every candidate architecture

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58601-0_12) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12358, pp. 189–204, 2020.
https://doi.org/10.1007/978-3-030-58601-0_12

until convergence, and thousands of architectures need to be evaluated during the architecture search process. As a result, the computational burden of the whole NAS process is extremely large. There are two directions to address this issue, which focus on improving the searching and evaluation module, respectively. 1) Evaluation: accelerating the evaluation of each individual architecture, and in the meanwhile, keep the evaluation meaningful in the sense of ranking correlation; 2) Searching: increasing the sample efficiency so that fewer architectures are needed to be evaluated for discovering a good architecture.

To improve the sample efficiency of the architecture searching module, a promising idea is to learn an approximated performance predictor, and then utilize the predictor to sample architectures that are more worth evaluating. We refer to these NAS methods [8,13,21] as the predictor-based NAS methods, and their general flow will be introduced in Sect. 3.1. The generalization ability of the predictor is crucial to the sample efficiency of predictor-based NAS flows. Our work follows the line of research of predictor-based NAS, and focus on improving the performance predictor of neural architectures.

A performance predictor predicts the performance of architectures based on the encoding of them. Existing neural architecture encoding schemes include the sequence-based scheme and the graph-based scheme. The sequence-based schemes [8,13,21] rely on specific serialization of the architecture. They model the topological information only implicitly, which deteriorates the representational power and interpretability of the predictor. Existing graph-based schemes [4,19] usually apply graph convolutional networks (GCN) [6] to encode the neural architectures. For the "operation on node" (OON) search spaces, in which the operations (e.g., `Conv3x3`) are on the nodes of the directed acyclic graph (DAG), GCN can be directly applied to encode architectures. Nevertheless, since a neural architecture is a "data processing" graph, where the operations behave as the data processing functions (e.g., `Conv3x3`, `MaxPool`), existing methods' modeling of operations as the node attributes in OON search spaces is not suitable. Instead of modeling the operations as node attributes, a more natural solution is to treat them as the transforms of the node attributes (i.e., mimic the processing of the information). On the other hand, for the "operation on edge" (OOE) search spaces,[1] the handling of edge information in the existing graph-based scheme [4] is even more unsatisfying regarding its poor generalizability and flawed handling of architecture isomorphism.

In this work, we propose a general encoding scheme: *Graph-based neural ArchiTecture Encoding Scheme (GATES)*, which is suitable for the representation learning of data processing graphs such as neural architectures. Specifically, to encode a neural architecture, GATES models the information flow of the actual data processing of the architecture. First, GATES models the input information as the attributes of the input nodes. And the input information will be propagated along the architecture DAG. The data processing of the operations (e.g., `Conv3x3`, `MaxPool`) are modeled by GATES as different transforms of the information. Finally, the output information is used as the embedding of

[1] Figure 2 illustrates the OON and OOE search spaces.

the cell architecture. Since the encoding process of GATES mimics the actual computation flow of the architectures, GATES intrinsically maps isomorphic architectures to the same representation. Moreover, GATES can encode architectures from both the OON and OOE cell search spaces in a consistent way. Due to the superior representational ability of GATES, the generalization ability of the architecture performance predictor using GATES is significantly better than other encoders. Experimental results confirm that GATES is effective in improving the architecture performance predictors. Furthermore, by utilizing the improved performance predictor, the sample efficiency of the NAS process is improved.

2 Related Work

2.1 Architecture Evaluation Module

One commonly used technique to accelerate architecture evaluation is parameter sharing [15], where a super-net is constructed such that all architectures in the search space share a superset of weights and the training costs of architectures are amortized to an "one-shot" super-net training. Parameter sharing dramatically reduces the computational burden and is widely used by recent methods. However, recent studies [12,17] find that the ranking of architecture candidates with parameter sharing does not reflect their true rankings well, which dramatically affects the effectiveness of the NAS algorithm. Moreover, the parameter sharing technique is not generally applicable, since it is difficult to construct the super-net for some search spaces, for example, in NAS-Bench-101 [24], one operation can have different output dimensions in different candidate architectures. Due to these limitations, this work does not use the parameter sharing technique, and focus on improving the sample efficiency of the architecture searching module.

2.2 Architecture Searching Module

To improve the sample efficiency of the architecture search module, a variety of search strategies have been used, e.g., RL-based methods [3,15,25], Evolutionary methods [9,16], gradient-based method [7,10], Monte Carlo Tree Search (MCTS) method [14], etc.

A promising direction to improve the sample efficiency of NAS is to utilize a performance predictor to sample new architectures, a.k.a. *predictor-based NAS*. An early study [8] trains a surrogate model (predictor) to identify promising architectures with increasing complexity. NASBot [5] design a distance metric in the architecture space and exploits gaussian process to get the posterior of the architecture performances. Then, it samples new architectures based on the acquisition function calculated using the posterior. NAO [13] trains an LSTM-based autoencoder together with a performance predictor based on the latent representation. After updating the latent representation following the predictor's gradients, NAO decodes the latent representation to sample new architectures.

2.3 Neural Architecture Encoders

Existing neural architecture encoding schemes include the sequence-based and the graph-based schemes. In the sequence based scheme, the neural architecture is *flattened* into a string encoding the architecture decisions, then encoded using either an LSTM [8,13,21] or a Multi-Layer Perceptron (MLP) [8,21]. In these methods, the topological information could only be modeled implicitly, which deteriorates the encoder's representational ability. Also, the search efficiency would deteriorate since these encoders could not guarantee to map isomorphic architectures [20,24] to the same representation, and data augmentation and regularization tricks are utilized to alleviate this issue [13].

Recently, the graph-based encoding scheme that utilizes the topological information explicitly has been used to get better performance. In these graph-based schemes, graph convolutional networks (GCN) [6] are usually used to embed the graphs to fixed-length vector representations. For the "operation on node" search spaces, in which the operations (e.g., `Conv3x3`) are on the nodes of the DAG, GCN can be directly applied [19] to encode architectures, i.e., using adjacency matrix and operation embedding of each node as the input. However, for the "operation on edge" search spaces, in which the operations are on the edges, GCN cannot be applied directly. A recent study [4] proposes an ad-hoc solution for the ENAS search space. They represent each node by the concatenation of the operation embeddings on the input edges. This solution is contrived and cannot generalized to search spaces where nodes could have different input degrees. Moreover, since the concatenation is not commutative, this encoding scheme could not handle isomorphic architectures correctly. In brief, existing graph-based encoding schemes are specific to different search spaces, and a generic approach for encoding the neural architectures is desirable in the literature.

3 Method

3.1 Predictor-Based Neural Architecture Search

The principle of predictor-based NAS is to increase the sample efficiency of the NAS process, by utilizing an approximated performance predictor to sample architectures that are more worth evaluating. Generally speaking, the flow of predictor-based NAS could be summarized as in Algorithm 1 and Fig. 1.

In line 6 of Alg. 1, the architecture candidates are sampled based on the approximated evaluation of the predictor. Utilizing a more accurate predictor, we could choose better architectures for further evaluation. The better the generalization ability of the predictor is, the fewer architectures are needed to be exactly evaluated to get a highly accurate predictor. Therefore, the generalization ability of the predictor is crucial for the efficiency and effectiveness of the NAS method.

The model design (i.e., how to encode the neural architectures) of the predictor is crucial to its generalization ability. We'll introduce our main effort to improve the predictor from the "model design" aspect in the following section.

Algorithm 1. The flow of predictor-based neural architecture search

$\mathcal{A}$: Architecture search space
P : $\mathcal{A} \to \mathbb{R}$: Performance predictor that outputs the predicted performance given the architecture
$N^{(k)}$: Number of architectures to sample in the k-th iteration

k = 1
while $k \leq$ MAX_ITER **do**
    Sample a subset of architectures $S^{(k)} = \{a_j^{(k)}\}_{j=1,\cdots,N^{(k)}}$ from $\mathcal{A}$, utilizing P
    Evaluate architectures in $S^{(k)}$, get $\tilde{S}^{(k)} = \{(a_j^{(k)}, y_j^{(k)})\}_{j=1,\cdots,N^{(k)}}$ (y is the performance)
    Optimizing P using the ground-truth architecture evaluation data $\tilde{S} = \cup_{i=1}^{k} \tilde{S}^{(i)}$
end while
Output $a_{j^*} \in \cup_{i=1}^{k} S^{(i)}$ with best corresponding y_{j^*}; Or, $a^* = \text{argmax}_{a \in \mathcal{A}} \text{P}(a)$

3.2 GATES: A Generic Neural Architecture Encoder

A performance predictor P is a model that takes a neural architecture a as input, and outputs a predicted score $\hat{s}$. Usually, the performance predictor is constructed by an encoder followed by an MLP, as shown in Eq. 1. The encoder Enc maps a neural architecture into a continuous embedding space, and its design is vital to the generalization ability of the performance predictor. Existing encoders include the sequence-based ones (e.g., MLP, LSTM) and the graph-based ones (e.g., GCN). We design a new graph-based neural architecture encoder GATES that is more suitable for modeling neural architectures.

又改对Encoder部分，但 MLP不用管？

$$\hat{s} = \text{P}(a) = \text{MLP}(\text{Enc}(a)) \tag{1}$$

To encode a cell architecture into an embedding vector, GATES follows the ideology of modeling the information flow in the architecture, and uses the output information as the embedding of the architecture. The notations are summarized in Table 1.

Specifically, we models the input information as the embedding of the input nodes $E \in \mathbb{R}^{n_i \times h_i}$, where n_i is the number of input nodes, and h_i is the embed-

Table 1. Notations of GATES. E, EMB, W_o and W_x are all trainable parameters

n_i	Number of input nodes: 1, 1, 2 for NAS-Bench-101, NAS-Bench-201 and ENAS, respectively
N_o	Number of operation primitives
h_o	Embedding size of operation
h_i	Embedding size of information
$E \in \mathbb{R}^{n_i \times h_i}$	The embedding of the information at the input nodes
$\text{EMB} \in \mathbb{R}^{N_o \times h_o}$	The operation embeddings
$W_o \in \mathbb{R}^{h_o \times h_i}$	The transformation matrix on the operation embedding
$W_x \in \mathbb{R}^{h_i \times h_i}$	The transformation matrix on the information

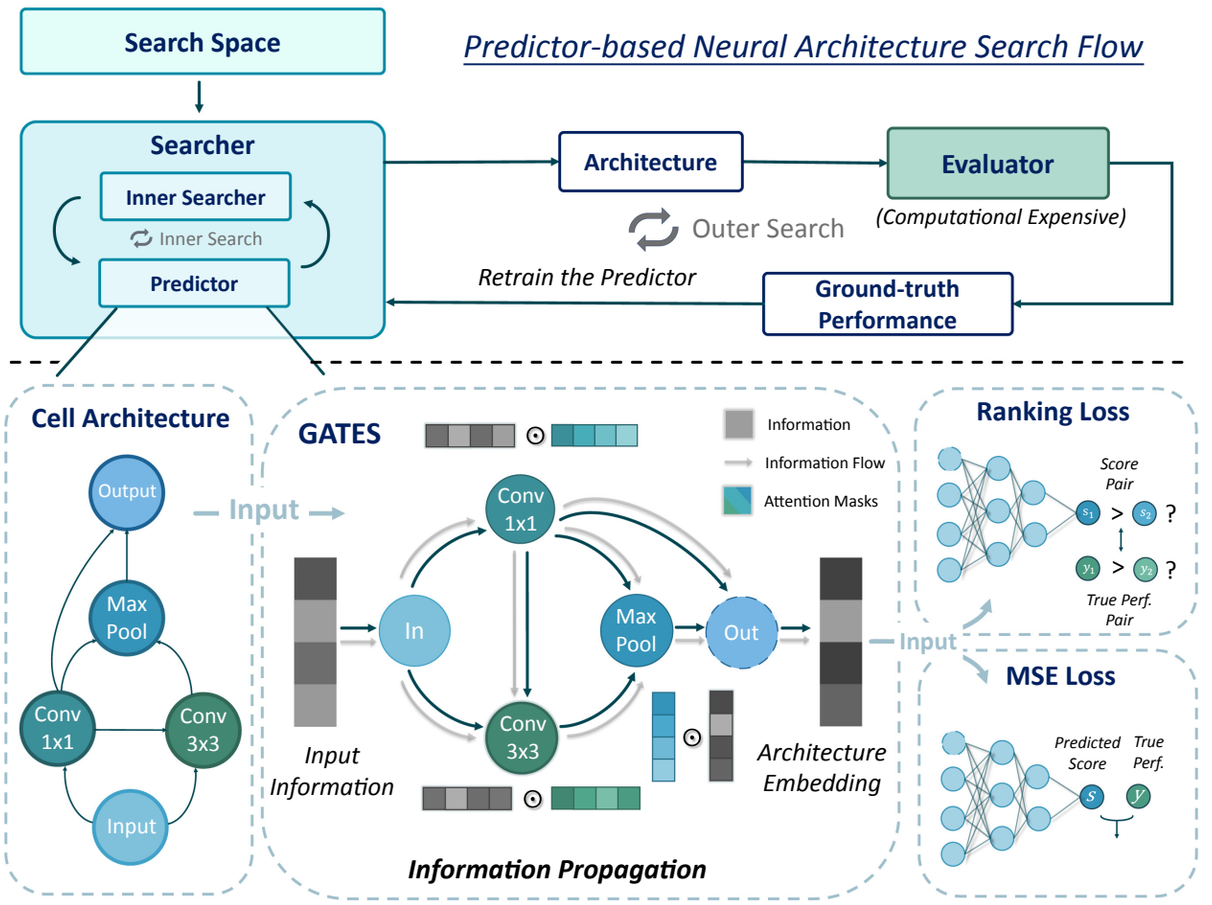


Fig. 1. The overview of the proposed algorithm. Upper: The general flow of the predictor-based NAS. Lower: Illustration of the encoding processes of GATES of an OON cell architecture

ding size of the information. The information (embedding of the input nodes) is then "processed" by the operations and "propagates" along the DAG.

The encoding process of GATES goes as follows: Upon each unary operation o (e.g., `Conv3x3`, `MaxPool`, etc.), the input information x_{in} of this operation is processed by a linear transform W_x and then elementwise multiplied with a soft attention mask $m = \sigma(\text{EMB}(o)W_o) \in \mathbb{R}^{1\times h_i}$.

$$x_{\text{out}} = m \odot x_{\text{in}} W_x \tag{2}$$

where $\odot$ denotes the elementwise multiplication. And the mask m is calculated from the operation embedding $\text{EMB}(o) = \text{onehot}(o)^T\text{EMB} \in \mathbb{R}^{1\times h_o}$.

Multiple pieces of information are aggregated at each node using summation. Finally, after obtaining the virtual information at all the nodes, the information at the output node is used as the embedding of the entire cell architecture. For search spaces with multiple cells (e.g., normal and reduce cells in ENAS), GATES encodes each cell independently, and concatenate the embeddings of cells as the embedding of the architecture.

Figure 2 illustrates two examples of the encoding process in the OON and OOE search spaces. As can be seen, the encoding process of GATES mimics the actual feature map computation. For example, in the example of the OON search

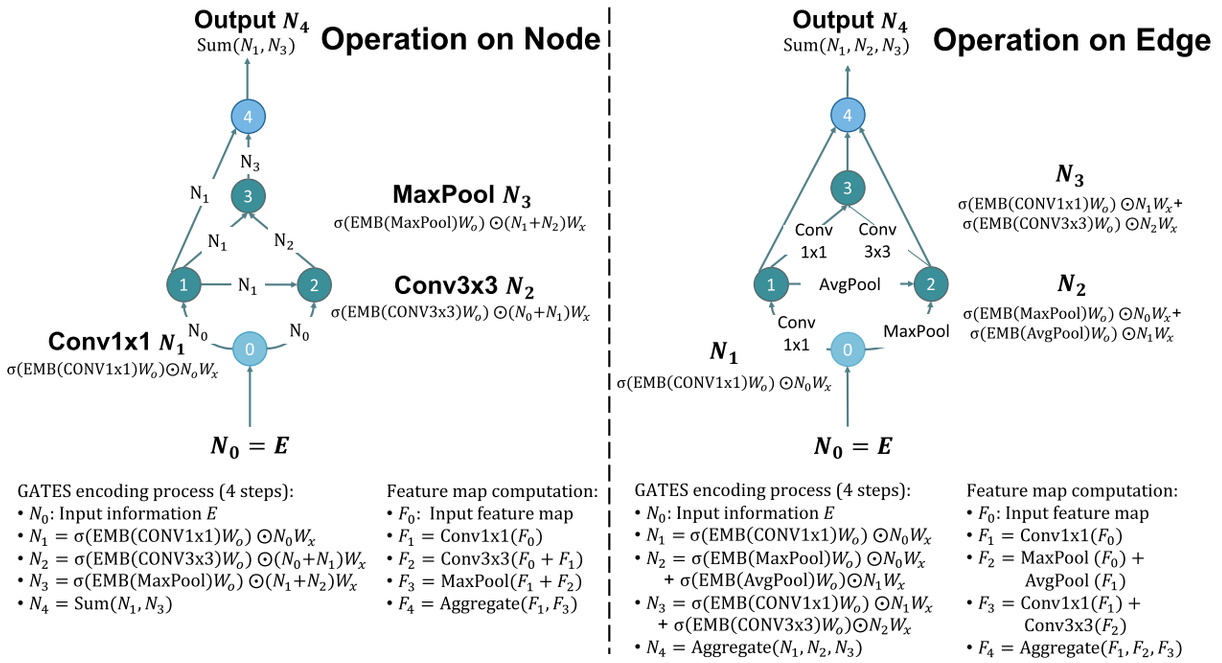


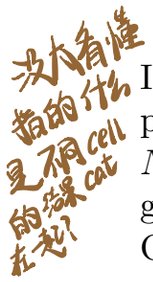
Fig. 2. Feature map (F_i) computation and GATES encoding process (N_i). Left: The "operation on node" cell search space, where operations (e.g., `Conv3x3`) are on the nodes of the DAG (e.g., NAS-Bench-101 [24], randomly wired search space [22]). Right: The "operation on edge" cell search space, where operations are on the edges of the DAG. (e.g., NAS-Bench-201 [2], ENAS [15])

space, the actual feature map computation at node 2 is $F_2 = \text{Conv3x3}(F_0 + F_1)$, where F_i is the feature map at node i. To model the information processing of this feature map computation, GATES calculates the information (node embedding) at node 2 by $N_2 = \sigma(\text{EMB}(\text{Conv3x3})W_o) \odot (N_0 + N_1)W_x$, where $\sigma(\cdot)$ is the sigmoid function, and $W_o \in \mathbb{R}^{h_o \times h_i}$ is a transformation matrix that transforms the h_o-dim operation embedding into a h_i-dim feature. That is to say, the summation of feature maps $F_0 + F_1$ corresponds to the summation of the virtual information $N_0 + N_1$, and the data processing function $o(\cdot)$ (`Conv3x3`) corresponds to a transform $f(\cdot)$ that processes the information $x = N_0 + N_1$ by $f_o(x) = \sigma(\text{EMB}(o)W_o) \odot xW_x$.

Intuitively, to model a cell architecture, GATES models the operations in the architecture as the "soft gates" that control the flow of the virtual information, and the output information is used as the embedding of the cell architecture. The key difference between GATES and GCN is: In GATES, the operations (e.g., `Conv3x3`) are modeled as the processing of the node attributes (i.e., virtual information), whereas GCN models them as the node attributes themselves.

The representational power of GATES for neural architectures comes from two aspects: 1) The more reasonable modeling of the operations in data-processing DAGs. 2) The intrinsic proper handling of DAG isomorphism. The discussion and experiments on how GATES handles the isomorphism are in the "Discussion on Isomorphism" section in the appendix.

In practice, to calculate the information propagation following the topological order of different graphs in a batched manner, we use a stack of GATES layers.



In the forward process of each layer, one step of information propagation is taken place at every node. That is to say, if a graph is input to a GATES encoder with N layers, the information is propagated and aggregated for N steps along the graph. The batched formulas and specific implementations of a GATES layer for OON and OOE search spaces are elaborated in the "Implementation of GATES" section in the appendix.

The Optimization of GATES. The most common practice [8,13] to train the architecture performance predictors is to minimize the Mean Squared Error (MSE) between the predictor outputs and the true performances.

$$L(\{a_j, y_j\}_{j=1,\cdots,N}) = \sum_{j=1}^{N} (\mathrm{P}(a_j) - y_j)^2 \tag{3}$$

where a_j denotes one architecture, and y_j denotes the true performance of a_j.

In NAS applications, what is really required to guide the search of architectures is the relative ranking order of architectures rather than the absolute performance values. In this paper, we adopt Kendall's Tau ranking correlation [18] as the measure as the direct criterion for evaluating architecture predictors. And since ranking losses are better surrogate losses [1,11,23] for the ranking correlation than the regression loss, in addition to the MSE loss, we use a hinge pair-wise ranking loss with margin $m = 0.1$ to train the predictors.[2]

$$L(\{a_j, y_j\}_{j=1,\cdots,N}) = \sum_{j=1}^{N} \sum_{i, y_i > y_j} \max[0, m - (\mathrm{P}(a_i) - \mathrm{P}(a_j))] \tag{4}$$

3.3 Neural Architecture Search Utilizing the Predictor

We follow the flow in Algorithm 1 to conduct the architecture search. There are multiple ways of utilizing the predictor P to sample architectures (line 6 in Alg. 1), i.e., the choice of the inner search method. In this work, we use two inner search methods for sampling architecture for further evaluation:[3]

- Random sample n architectures from the search space, then choose the best k among them according to the evaluation of the predictor.
- Search with Evolutionary Algorithm (EA) for n steps, and then choose the best k with the highest predicted scores among the seen architectures.

Compared with the evaluation (line 7 in Alg. 1) in the outer search process, the evaluation of each architecture in the inner search process is very efficient with only a forward pass of the predictor. The sample ratio $r = \frac{n}{k}$ indicates the

[2] A more comprehensive comparison of the MSE regression loss and multiple ranking losses is shown in the appendix.

[3] Note that this inner search component could be easily substituted with other search strategies.

equivalent number of the architectures need to be evaluated by the predictor to make one sample decision. And it is not the case that bigger r leads to better sample efficiency of the overall NAS process. If n is too large (the limiting case is to exhaustive test the whole search space with $n = |\mathcal{A}|$), the sampling process would overfit onto exploiting the current performance predictor and fails to explore. Therefore, there is a trade-off between exploration and exploitation controlled by n, which we verify in Sect. 4.3.

4 Experiments

The experiments in Sect. 4.1 and Sect. 4.2 verify the effectiveness of the GATES encoder on both the OON and OOE search spaces. Then, in Sect. 4.3, we demonstrate that by utilizing GATES, the sample efficiency of the NAS process surpasses other searching strategies, including the predictor-based methods with other baseline encoders. Finally, in Sect. 4.4, we apply the proposed algorithm to the ENAS search space.

4.1 Predictor Evaluation on NAS-Bench-101

Setup. NAS-Bench-101 [24] provides the performances of the 423k unique architectures in a search space. The NAS-Bench-101 search space is an OON search space, in which sequence based encoding schemes [21], and graph based encoding schemes [19] are proposed for encoding architectures. We use the Kendall's Tau ranking correlation [18] as the measure for evaluating the architecture performance predictors. The first 90% (381262) architectures are used as the training data, and the other 42362 architectures are used for testing.[4]

We conduct a more comprehensive comparison of the MSE loss and multiple ranking losses on NAS-Bench-101, and the results are shown in the appendix. We find that compared to the MSE loss, ranking losses bring consistent improvements, and hinge pair wise loss is a good choice. Therefore, in our experiments, unless otherwise stated, the hinge pairwise loss with margin 0.1 is used to train all the predictors.

Results. Table 2 shows the comparison of the GATES encoder and various baseline encoders trained using different proportions of the training data. As can be seen, GATES could achieve higher Kendall's Taus on the testing architectures than the baseline encoders consistently with different training proportions. The advantages are especially significant when there are few training architectures. For example, when only 190 (0.05%) architectures are seen by the performance predictor, utilizing the same training settings, GATES achieves a test Kendall's Tau of *0.7634*, whereas the Kendall's Tau results achieved by MLP, LSTM, and the best GCN variant are 0.3971, 0.5509 and 0.5343, respectively. This demonstrates the surpassing generalization ability of the GATES encoder, which

[4] See "Setup and Additional Results" section in the appendix for more details.

Table 2. The Kendall's Tau of using different encoders on the NAS-Bench-101 dataset. The first 90% (381262) architectures in the dataset are used as the training data, and the other 42362 architectures are used as the testing data

Encoder	Proportions of 381262 training samples							
	0.05%	0.1%	0.5%	1%	5%	10%	50%	100%
MLP [21]	0.3971	0.5272	0.6463	0.7312	0.8592	0.8718	0.8893	0.8955
LSTM [21]	0.5509	0.5993	0.7112	0.7747	0.8440	0.8576	0.8859	0.8931
GCN (w.o. global node)	0.3992	0.4628	0.6963	0.8243	0.8626	0.8721	0.8910	0.8952
GCN (global node) [19]	0.5343	0.5790	0.7915	0.8277	0.8641	0.8747	0.8918	0.8950
GATES	**0.7634**	**0.7789**	**0.8434**	**0.8594**	**0.8841**	**0.8922**	**0.9001**	**0.9030**

Table 3. N@K on NAS-Bench-101. All predictors are trained with 0.1% of the training data (i.e., 381 architectures)

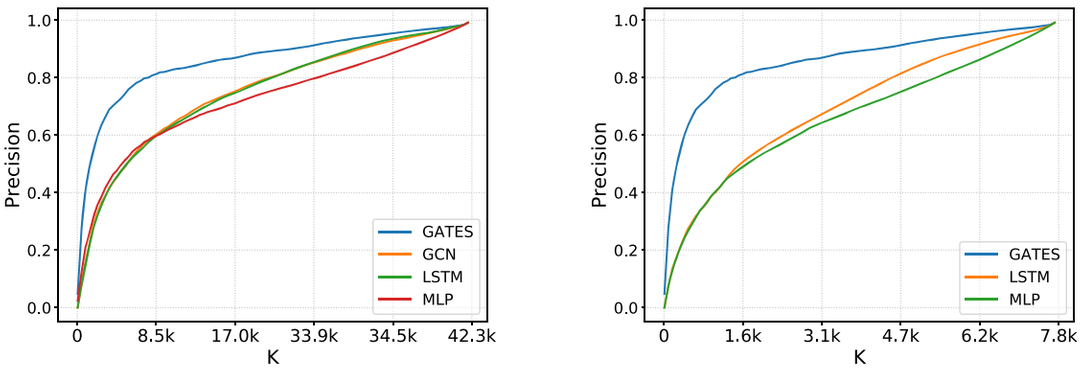
Encoder	Ranking loss		Regression loss	
	N@5	N@10	N@5	N@10
MLP [21]	57 (0.13%)	58 (0.13%)	1397 (3.30%)	552 (1.30%)
LSTM [21]	1715 (4.05%)	1715 (4.05%)	1080 (2.54%)	312 (0.73%)
GCN [19]	2025 (4.77%)	1362 (3.21%)	405 (0.95%)	405 (0.95%)
GATES	**22 (0.05%)**	**22 (0.05%)**	**27 (0.05%)**	**27 (0.05%)**

enables one to learn a good performance predictor for unseen architectures after evaluating only a small set of architectures.

In the Kendall's Tau measure, all discordant pairs are treated equally. However, in NAS applications, the relative rankings among the poorly performing architectures are not of concern. Therefore, we compare different predictors in the form of other measures that have a more direct correspondence with the NAS flow: 1) N@K: The best true ranking among the top-K architectures selected according to the predicted scores. 2) Precision@K: The proportion of true top-K architectures among the top-K predicted architectures. Table 3 and Fig. 3(a) show these two measures of the predictors with different encoders on the testing set of NAS-Bench-101. As can be seen, GATES achieves consistently better performances than other encoders across different Ks.

4.2 Predictor Evaluation on NAS-Bench-201

Setup. NAS-Bench-201 [2] is another NAS benchmark that provides the performances of 15625 architectures in an OOE search space. In our experiments, we use the first 50% (7813) as the training data, and the remaining 7812



(a) NAS-Bench-101 (b) NAS-Bench-201

Fig. 3. Precision@K

Table 4. The Kendall's Tau of using different encoders on the NAS-Bench-201 dataset. The first 50% (7813) architectures in the dataset are used as the training data, and the other 7812 architectures are used as the testing data

Encoder	Proportions of 7813 training samples				
	1%	5%	10%	50%	100%
MLP [21]	0.0974	0.3959	0.5388	0.8229	0.8703
LSTM [21]	0.5550	0.6407	0.7268	0.8791	0.9002
GATES	**0.7401**	**0.8628**	**0.8802**	**0.9192**	**0.9259**

architectures as the testing data. Since GCN encoders could not be directly applied to the OOE search spaces, we compare GATES with the sequence-based encoders: MLP and LSTM (Table 4).[5]

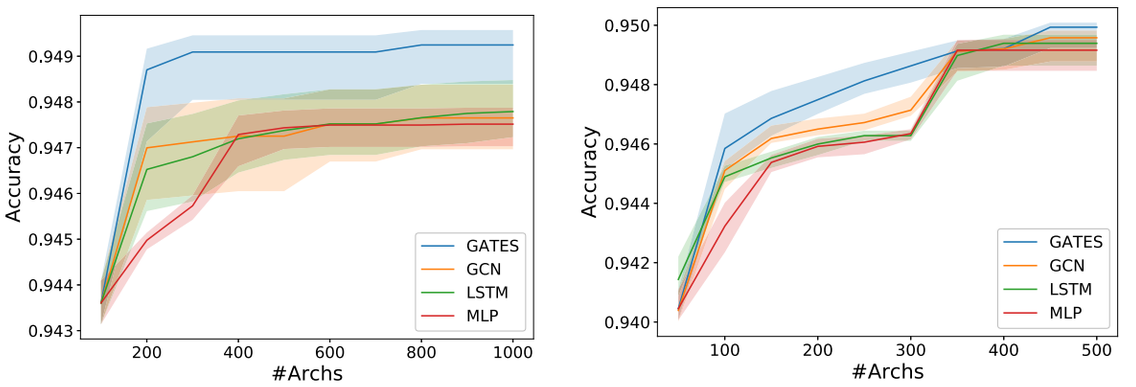
Results. Table 2 shows the evaluation results of GATES. GATES could achieve significantly higher ranking correlations than the baseline encoders, especially when there are only a few training samples. For example, with 78 training samples, "GATES + Pairwise loss" could achieve a Kendall's Tau of 0.7401, while the best baseline result is 0.5550 ("LSTM + Pairwise loss").

The N@K and Precision@K measures on NAS-Bench-201 are shown in Table 5 and Fig. 3(b), respectively. We can see that GATES can achieve an N@5 of 1 on the 7812 testing architectures, with either ranking loss or regression loss. And, not surprisingly, GATES outperforms the baselines consistently on the Precision@K measure too.

[5] We also implement an ad-hoc solution of applying GCN on OOE architectures referred to as the Line Graph GCN solution, in which the graph is first converted to a line graph. See "Setup and Additional Results" section in the appendix for more details.

Table 5. N@K on NAS-Bench-201. All the predictors are trained using 10% of the training data (i.e., 781 architectures)

Encoder	Ranking loss		Regression loss	
	N@5	N@10	N@5	N@10
MLP [21]	7 (0.09%)	7 (0.09%)	1538 (19.7%)	224 (3.87%)
LSTM [21]	8 (1.02%)	2 (0.01%)	250 (6.65%)	234 (2.99%)
GATES	**1 (0.00%)**	**1 (0.00%)**	**1 (0.00%)**	**1 (0.00%)**



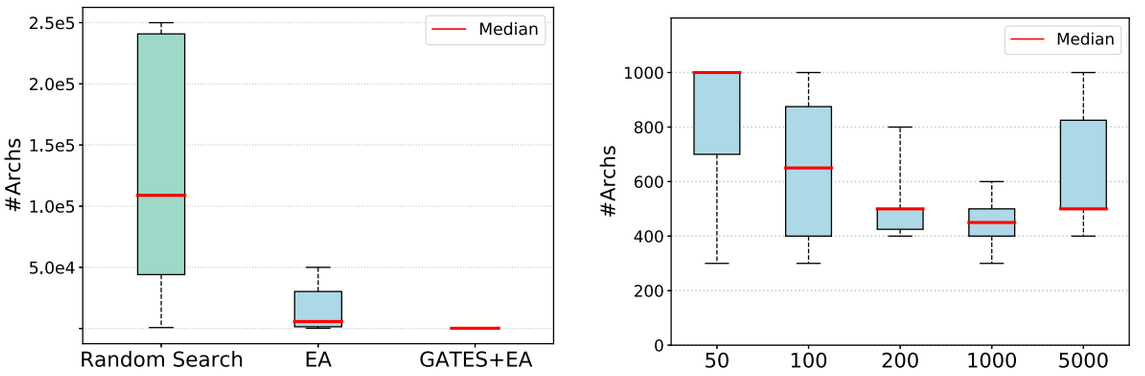
a) RS inner search method ($r = 500$) (b) EA inner search method ($r = 100$)

Fig. 4. Comparison of predictor-based NAS with different encoders: The best validation accuracy during the search process over 10/15 runs for the RS and EA inner serach method, respectively. r is the sample ratio (see Sect. 3.3)

4.3 Neural Architecture Search on NAS-Bench-101

Equipped with a better performance predictor, the sample efficiency of the predictor-based NAS process can be significantly improved. To verify that, we conduct the architecture search on NAS-Bench-101 using various searching strategies. As the baseline of our method, we run a random search, regularized evolution [16], and predictor-based NAS methods equipped with the baseline encoders (i.e., LSTM, MLP, GCN).

Comparison of Sample Efficiency. The results of running predictor-based NAS methods with different encoders are shown in Fig. 4. We conduct experiments with two inner search methods: random search, and evolutionary algorithm. In each stage, 100 random samples are used to train the predictor (50 for evolutionary algorithm), and the predictor is trained for 50 epochs with hinge ranking loss. When using random search, $n = 2500$ architectures are randomly sampled, and the top $k = 5$ architectures with high predicted scores are chosen to be further evaluated by the ground truth evaluator. When using the evolutionary algorithm for the inner search, n is set to 100, and k is set to 1. And the population and tournament size is 20 and 5, respectively. We can see that the sample efficiency using GATES surpasses the baselines with different inner



(a) Comparison of search methods (b) Ablation study of the sample ratio r

Fig. 5. Left: Number of architectures evaluated to acquire the best validation accuracy on NAS-Bench-101 over 100 runs. We use the mean validation accuracy as the search reward. GATES-powered predictor-based NAS is 511.0× and 59.25× more sample efficient than random search and regularized evolution. Right: Number of architectures evaluated to acquire the best validation accuracy over 10 runs with different r

search methods. This verifies the analysis that utilizing a better neural architecture encoder in the predictor-based NAS flow leads to better sample efficiency.

The comparison of the sample efficiency of two baseline searching strategies and the predictor-based method with GATES is shown in Fig. 5(a). The median counts of evaluated architectures of RS, Regularized EA and GATES-powered NAS over 100 runs are 220400, 23700 and 400 (50 as the granularity), respectively. GATES-powered NAS is 551.0× and 59.25× more sample efficient than the random search and evolution algorithm.

Ablation Study of the Sample Ratio r. The ablation study of the sample ratio r (Sect. 3.3) is shown in Fig. 5(b). We run GATES-powered predictor-based search with evolutionary algorithm, and shows the architectures needed to evaluate before finding the architecture with the best validation accuracy. We can see that the sample ratio r should be neither too big nor too small, since a too small n leads to bad exploitation and a too large n leads to bad exploration.

4.4 Neural Architecture Search in the ENAS Search Space

In this section, we apply our method on the ENAS search space. This search space is an OOE search space that is much larger than the benchmark search spaces. We first randomly sample 600 architectures and train them for 80 epochs. Then we train a GATES predictor using the performance of the 600 architectures and use it to sample 200 architectures, by randomly sampling 10k architectures and taking the top 200 with the highest predicted scores (sample ratio $r = 50$). After training these 200 architectures for 80 epochs, we pick the architecture with the best validation accuracy. Finally, after the channel and layer augmentation, the architecture is trained from scratch for 600 epochs.

Table 6. Comparison of NAS-discovered architectures on CIFAR-10

Method	Test Error (%)	#Params (M)	#Archs Evaluated
NASNet-A + cutout [25]	2.65	3.3	20000
AmoebaNet-B + cutout [16]	2.55	2.8	27000
NAONet [13]	2.98	28.6	1000
PNAS [8]	3.41	3.2	1160
NAONet-WS† [13]	3.53	2.5	–
DARTS+cutout† [10]	2.76	3.3	–
ENAS + cutout† [15]	2.89	4.6	–
Ours + cutout	2.58	4.1	800

The comparison of the test errors of different architectures is shown in Table 6, and the discovered architecture is shown in the appendix. As can be seen, our discovered architecture can achieve a test error rate of 2.58%, which is better than those architectures discovered with parameter sharing evaluation. Compared to the other methods, much fewer samples are truly evaluated to discover an architecture with better or comparable performance. When transferred to ImageNet, the discovered architecture achieves a competitive top-1 error of 24.1% with 5.6M parameters.

5 Conclusion

In this paper, we propose GATES, a graph-based neural architecture encoder with better representation ability for neural architectures. Due to its reasonable modeling of the neural architectures and intrinsic ability to handle DAG isomorphism, GATES significantly improves the architecture performance predictor for different cell-based search spaces. Utilizing GATES in the predictor-based NAS flow leads to consistent improvements in sample efficiency. Extensive experiments demonstrate the effectiveness and rationality of GATES. Employing GATES to encode architectures in larger or hierarchical topological search spaces is an interesting future direction.

Acknowledgments. This work was supported by National Natural Science Foundation of China (No. 61832007, 61622403, 61621091, U19B2019), Beijing National Research Center for Information Science and Technology (BNRist). The authors thank Novauto for the support.

References

1. Chen, W., Liu, T.-Y., Lan, Y., Ma, Z., Li, H.: Ranking measures and loss functions in learning to rank. In: Bengio, Y., Schuurmans, D., Lafferty, J.D., Williams, C.K.I., Culotta, A. (eds.) Advances in Neural Information Processing Systems 22, pp. 315–323. Curran Associates, Inc., New York (2009)

2. Dong, X., Yang, Y.: Nas-bench-201: extending the scope of reproducible neural architecture search. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=HJxyZkBKDr
3. Guo, Y., et al.: Breaking the curse of space explosion: towards effcient nas with curriculum search. In: International Conference on Machine Learning (2010)
4. Guo, Y., et al.: Nat: neural architecture transformer for accurate and compact architectures. In: Advances in Neural Information Processing Systems, pp. 735–747 (2019)
5. Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., Xing, E.P.: Neural architecture search with Bayesian optimisation and optimal transport. In: Advances in Neural Information Processing Systems, pp. 2016–2025 (2018)
6. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
7. Lian, D., et al.: Towards fast adaptation of neural architectures with meta learning. In: International Conference on Learning Representations (2020)
8. Liu, C., et al.: Progressive neural architecture search. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11205, pp. 19–35. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01246-5_2
9. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436 (2017)
10. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
11. Liu, T.Y., et al.: Learning to rank for information retrieval. Found. Trends® Inf. Retrieval **3**(3), 225–331 (2009)
12. Luo, R., Qin, T., Chen, E.: Understanding and improving one-shot neural architecture optimization. arXiv preprint arXiv:1909.10815 (2019)
13. Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.Y.: Neural architecture optimization. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 7816–7827. Curran Associates, Inc. (2018). http://papers.nips.cc/paper/8007-neural-architecture-optimization.pdf
14. Negrinho, R., Gordon, G.: Deeparchitect: automatically designing and training deep architectures. arXiv preprint arXiv:1704.08792 (2017)
15. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268 (2018)
16. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. Proc. AAAI Conf. Artificial Intell. **33**, 4780–4789 (2019)
17. Sciuto, C., Yu, K., Jaggi, M., Musat, C., Salzmann, M.: Evaluating the search phase of neural architecture search. arXiv preprint arXiv:1902.08142 (2019)
18. Sen, P.K.: Estimates of the regression coefficient based on Kendall's tau. J. Am. Stat. Assoc. **63**(324), 1379–1389 (1968)
19. Shi, H., Pi, R., Xu, H., Li, Z., Kwok, J.T., Zhang, T.: Multi-objective neural architecture search via predictive network performance optimization. arXiv preprint arXiv:1911.09336 (2019)
20. Stagge, P., Igel, C.: Neural network structures and isomorphisms: random walk characteristics of the search space. In: 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (Cat. No. 00), pp. 82–90. IEEE (2000)

21. Wang, L., Zhao, Y., Jinnai, Y., Fonseca, R.: Alphax: exploring neural architectures with deep neural networks and monte carlo tree search. arXiv preprint arXiv:1805.07440 (2018)
22. Xie, S., Kirillov, A., Girshick, R., He, K.: Exploring randomly wired neural networks for image recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1284–1293 (2019)
23. Xu, Y., et al.: Renas:relativistic evaluation of neural architecture search (2019)
24. Ying, C., Klein, A., Real, E., Christiansen, E., Murphy, K., Hutter, F.: Nas-bench-101: towards reproducible neural architecture search. arXiv preprint arXiv:1902.09635 (2019)
25. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR (2017). https://arxiv.org/abs/1611.01578