Surrogate-Assisted Evolutionary Neural Architecture Search with Isomorphic Training and Prediction

Pengcheng Jiang[1], Yu Xue[1(✉)], Ferrante Neri[1,2], and Mohamed Wahib[3]

[1] School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
xueyu@nuist.edu.cn

[2] NICE Group, School of Computer Science and Electronic Engineering, University of Surrey, Guildford GU2 7XH, UK

[3] High Performance Artificial Intelligence Systems Research Team, RIKEN Center for Computational Science, 2-1 Hirosawa, Wako 351-0198, Saitama, Japan

Abstract. Neural architecture search (NAS) is an increasingly popular method for the automatic design of neural networks. Although promising, NAS is often associated with a significant computational cost. Surrogate models, which predict the performance of candidate networks without training them, are thus used to speed up NAS calculations. Since surrogate models must be trained, their performance depends on the dataset of labelled candidate architectures. The generation of these samples can be time-consuming as it requires the training of an architecture. The present paper proposes an inexpensive way of generating training data for the surrogate model. Specifically, the proposed algorithm makes use of isomorphism to obtain more training data for the graph-based encoding. We propose an isomorphic training which combines the use of the Mean Squared Error (MSE) with a novel isomorphic loss function. Then, we propose an isomorphic score to predict the performance of candidate architectures. The proposed isomorphic-based surrogate is integrated within an evolutionary framework for NAS. Numerical experiments are performed on NAS-Bench101 and NAS-Bench201 search spaces. The experimental results demonstrate that the proposed Isomorphic Training and Prediction Evolutionary Neural Architecture Search (ITP-ENAS) algorithm can identify architectures with better performance than other state-of-the-art algorithms, despite training only 424 architectures.

Keywords: Neural architecture search · Evolutionary algorithm · Surrogate model · Isomorphic graphs · Convolutional neural networks

1 Introduction

Convolutional neural networks (CNNs) are classical deep learning models, with emerging architectures like ResNet and MobileNet. Despite their effectiveness in tasks such as image classification, object detection, and semantic segmentation, designing CNN structures remains challenging, leading to the adoption of neural architecture search (NAS)

D.-S. Huang et al. (Eds.): ICIC 2024, LNCS 14863, pp. 191–203, 2024.
https://doi.org/10.1007/978-981-97-5581-3_16

methods. Among NAS methods, Evolutionary NAS (ENAS) field has attracted significant interest. Real et al. achieved superior results using a large-scale mutation method [2]. However, evolutionary algorithms introduce significant computational overhead, with each individual requiring evaluation through training to obtain fitness values, see e.g., [4].

Several strategies aim to mitigate the time consumption of ENAS. Among these, surrogate models have proven to be a viable option within evolutionary frameworks. Surrogate models expedite evaluations by swiftly obtaining fitness values through prediction. A well-constructed surrogate model can substantially decrease ENAS time requirements. For instance, Sun et al. employed an end-to-end prediction approach for AECNN [5]. Lu et al. used a supernet with an adaptive switchable surrogate model to further diminish search time [6]. However, existing surrogate models heavily depend on the quantity and quality of the training dataset, especially regarding information about the trained architecture. Many surrogate models exhibit unreliable prediction accuracy when trained with limited data [7, 8]. Current ENAS research faces a trade-off. Surrogate models require more data for effective training and predictions, but generating this data is computationally expensive, potentially undermining the surrogate approach's advantages.

To overcome the challenge of insufficient architecture information for training surrogate models, researchers have developed data augmentation methods to generate more labelled architectures without additional network training. One approach involves exploiting graph structures within search spaces. Some studies have used genetic algorithms to explore network architectures represented by adjacency matrices of graphs [9]. These encoding strategies enable the generation of numerous network representations through graph isomorphism. Liu et al. leverage the NASBench-101 encoding space by swapping node order to enrich the surrogate model's dataset [11], improving prediction accuracy without additional computational costs. However, this approach still underutilises evaluated architecture information.

In this paper, we introduce the Isomorphic Training and Prediction-assisted Evolutionary-based NAS (ITP-ENAS) approach. It leverages isomorphic architectures in the graph encoding space to expand the dataset, enhancing training and prediction performance to maximize information utilization. Specifically, ITP-ENAS swaps node orders within evaluated architectures to generate new training data without additional overhead, maintaining the same classification accuracy information. Additionally, we introduce an isomorphic loss to extract information between isomorphic networks, improving model prediction. During the search process, fitness values are assigned based on isomorphic predictions. Applied to NASBench-101 and NASBench-201, this approach significantly enhances prediction ability and performs well with limited data. Moreover, it can be combined with any neural network-based neural predictor, enhancing surrogate prediction performance through isomorphic loss and prediction embedding.

The remainder of this paper is organized as follows: the implementation details of the proposed ITP-ENAS are introduced in Sect. 2. Sect. 3 presents experimental results to demonstrate the performance of ITP-ENAS. Finally, Sect. 4 comprises the conclusion and outlines future works.

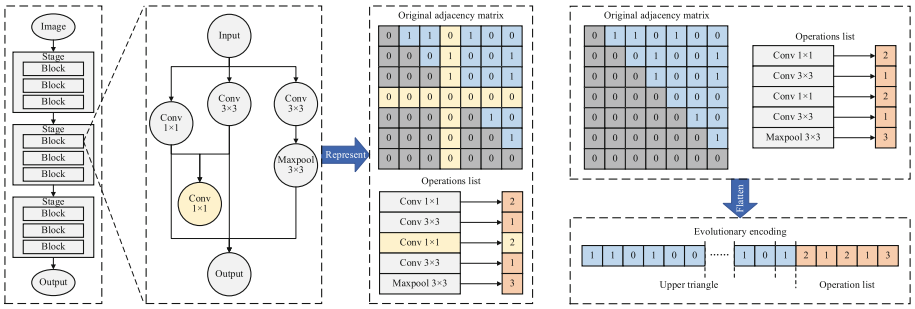
2 Isomorphic Training and Prediction-Assisted Evolutionary-Based NAS

2.1 Encoding

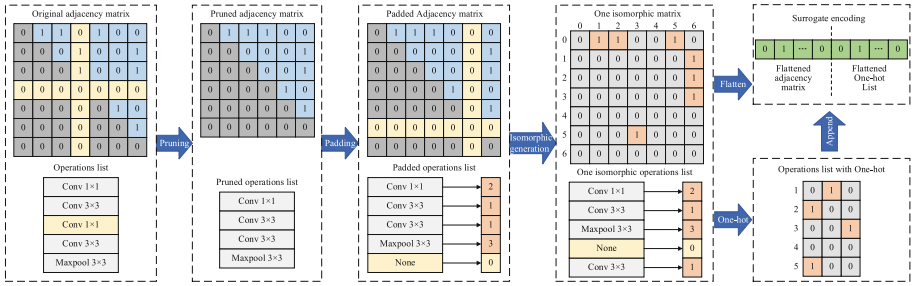
In ITP-ENAS, considering the distinct features of the evolutionary algorithm and the surrogate model, we propose the combined use of two encoding strategies, namely evolutionary encoding and surrogate encoding, respectively, see Fig. 1 as an example on NASBench-101.

For the evolutionary encoding part, this study utilized the original encodings from EvoXBench for the two search spaces, NASBench-101 and NASBench-201 [12]. Specifically, the network architectures in NASBench-101 were represented as graphs using adjacency matrices. Each element m_{ij} within the matrix indicates whether the $i-th$ node and the $j-th$ node are connected (1) or not (0). The matrix size corresponds to the number of operation nodes in the graph, considering inputs and outputs as two distinct nodes for this search space, resulting in a size of 7. The node numbering is based on the order in which the modules are used, with inputs in the first rows and columns and outputs in the last rows and columns. Numerical values are assigned to represent the type of operation, resulting in a fully digitized encoding denoted as O. The upper triangular part of the adjacency matrix is flattened into a one-dimensional vector and spliced in front of the type vector, as shown in Fig. 1. In the case of NASBench-201, only 6 operation locations were chosen, and each architecture is encoded with 6 bits, denoted as {0, 1, 2, 3, 4} to represent the type of operation.

For surrogate encoding, we introduce a novel representation suitable for architectures in both NASBench-101 and NASBench-201 benchmarks. Using a matrix denoted as M, we establish the adjacency matrix for NASBench-101, while a transformation from the Homogeneous Architecture Augmentation for Neural Predictor (HAAP) package [11] ensures compatibility with NASBench-201 architectures. After normalising the graph representation to NASBench-101 logic, we create the adjacency matrix M, generating isomorphic graphs illustrated in Fig. 1. Each matrix element m_{ij} indicates the connection between the $i-th$ and the $j-th$ nodes. Unlike the evolutionary representation, '1's are allowed in the lower triangular part of M. Operation encoding is transformed using the One-hot method to produce the operation type matrix O. Finally, the flattened and spliced adjacency and operation matrices form a one-dimensional vector I. It's important to note that invalid operations, such as those without inputs or outputs in NASBench-101 individuals (see Conv 1×1 highlighted in yellow in Fig. 1), are retained to maintain the number of nodes during evolution. However, prior to surrogate encoding, the positions of invalid operations in the matrix are set to 0, and their order is adjusted to precede the outputs, ensuring consistency. The missing operation types are encoded using the null vector in the One-hot encoding scheme.



a) Representation with adjacency matrix and b) Evolutionary encoding



c) Surrogate encoding

Fig. 1. Example of the evolutionary encoding and surrogate encoding for a graph-base network architecture in NASBench-101.

2.2 Isomorphic Predictor

In the surrogate model phase, we propose a novel training and prediction method based on isomorphic graphs, as illustrated in Fig. 2. Initially, using the filled adjacency matrix from Fig. 1, we consider all possible column rearrangements for the nodes in the middle (excluding inputs and outputs) to generate isomorphic graphs or surrogate encodings. Consequently, a graph containing S internal nodes (excluding inputs and outputs) can yield $G_{iso} = \{G_1, G_2, \ldots, G_{\mathcal{A}}\}$, where $\mathcal{A} = S!$ represents the total number of isomorphic graphs. To address potential redundancy in the original data where some isomorphic graphs may produce the same isomorphic groups, we retain only one group from G_{iso} generated from individual graphs in the history records, ensuring that each graph group has an equal amount of isomorphic data to prevent data imbalance.

Once the set of $S!$ isomorphic networks have been generated, their training process ensues. The use of a classical loss function such as the Mean Squared Error (MSE) is inadequate in this case as MSE could only use one network at the time as an input. Conversely, we make use of the isomorphic nature of the generated architecture and design around this fact a loss function, namely isomorphic loss, that can handle multiple architectures at once, whose details are described in the following paragraph.

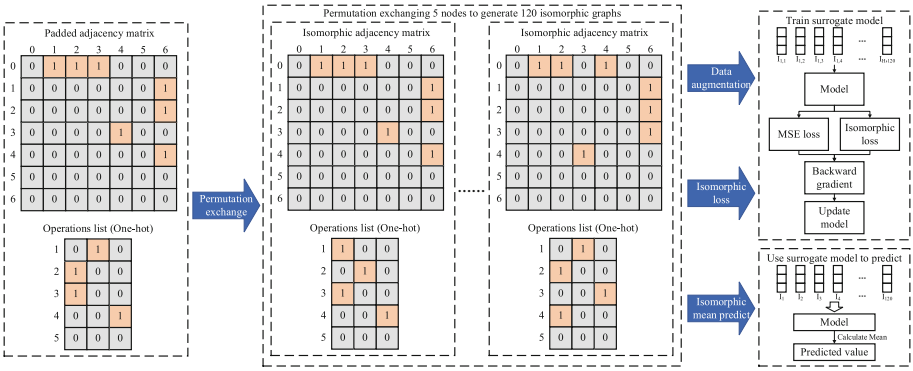


Fig. 2. The process method of isomorphic graphs generation, surrogate encoding, isomorphic training and prediction.

By applying a flattening operation, we obtain the corresponding surrogate encoding vectors $I_{iso} = \{I_1, I_2, \ldots, I_{\mathcal{A}}\}$ for graphs in one G_{iso}. All graphs in the history data are then used for the training. To better distinguish the isomorphic graph groups generated by different graphs, we add an index to I, using I_{ij} to represent the $j-th$ isomorphic graph in the $i-th$ I_{iso} temporarily. The training data is represented by the formula:

$$D_{MSE} = \{I_{1,1}, \ldots, I_{1,\mathcal{A}}, \ldots, I_{H\prime,1}, \ldots, I_{H\prime,\mathcal{A}}\} \tag{1}$$

$$D_{iso} = \left\{I_{iso_1}, \ldots, I_{iso_{\mathcal{A}}}\right\} = \{(I_{1,1}, \ldots, I_{1,\mathcal{A}}), \ldots, (I_{H\prime,1}, \ldots, I_{H\prime,\mathcal{A}})\} \tag{2}$$

in which $H\prime$ is the current length of the architecture history, D_{MSE} is used for training with MSE loss function and D_{iso} is used for training with isomorphic loss function. The size of batch data is set as B, so in each batch we select B samples from D_{MSE} and $\lambda = \max(1, \lfloor \frac{B}{\mathcal{A}} \rfloor)$ samples from D_{iso}. The predicted value of D_{MSE} is $y_i\prime$ while the true label is y_i, so we can get the MSE loss value as the formula:

$$L_1 = \frac{1}{B} \sum_{i=1}^{B} (y_i - y_i')^2 \tag{3}$$

To focus more on the inner relationships between pairs of isomorphic graphs, the proposed isomorphic loss is represented by the formula:

$$L_2 = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \frac{2}{\mathcal{A}(\mathcal{A}-1)} \sum_{j-2}^{\mathcal{A}} \sum_{k=1}^{j} (y'_{i,j} - y'_{i,k})^2 \tag{4}$$

in which $y_{i,j}\prime$ is the predict value of the isomorphic graph D_{iso_i}. Therefore, the final loss function for this batch data is as $L = L_1 + L_2$. After the forward progress of the loss value, $\beta \| W \|_2^2$ is added into the loss value as regularization where W is the weights of surrogate model and β is a hyper-parameter to control the strength of the regularization. The final loss value is used for back-propagation and model update.

In the phase of using the surrogate model for prediction, for an individual G to be predicted, the same steps are first adopted to generate an encoding for $\mathcal{A}$ isomorphic

representations $I_1, I_2, \ldots, I_{\mathcal{A}}$. These are then used as the inputs of the surrogate model to obtain $y_1\prime, \ldots y_{\mathcal{A}}\prime$. In the end, the final prediction value for this individual employs the mean of the $\mathcal{A}$ prediction values, which is noted as isomorphic mean in Fig. 2. The prediction process for one individual G is represented by the formula:

$$Pred(G) = \frac{1}{\mathcal{A}} \sum_{i=1}^{\mathcal{A}} Model(I_{iso}) \tag{5}$$

where I_{iso} is the corresponding encodings of isomorphic graph group for G.

2.3 Evolutionary Framework

The comprehensive workflow of the entire search methodology is delineated in Fig. 3. As illustrated, the proposed ITP-ENAS is partitioned into two distinct blocks enveloped by dotted rectangles in Fig. 3. The lower block outlines the procedures for constructing a surrogate predictor, as elucidated in Sect. 2.2, while the upper block delineates the Neural Architecture Search (NAS) operations orchestrated by the evolutionary framework. Moreover, for a more intricate understanding of the algorithmic operations, we have presented the pseudocode of the proposed ITP-ENAS in Algorithm 1.

Initially, P individuals, denoted as candidate architectures, are randomly sampled. From this pool of P candidates, N_{init} individuals undergo evaluation, meaning each architecture I_i is trained to acquire the corresponding fitness value y_i. The values of each trained architecture are recorded in an archive, referred to as *history*, which is utilised for training the surrogate model, as detailed in Sect. 2.2. Subsequently, the trained surrogate model predicts the fitness value of all P individuals, including the N_{init} that were trained, ensuring comparability across the entire population. At each generation, Q offspring architectures are generated through uniform crossover and random mutation, with probabilities p_c and p_m respectively. Parents are selected via tournament selection with a tournament size of 2. Every T generations, provided the *history* archive has not reached its maximum capacity H, the top K individuals from the offspring population undergo training, and the *history* archive is updated accordingly. The surrogate model is then re-trained, and the entire population is re-estimated using the updated surrogate model. Parents and offspring are merged to select P elite individuals (candidate architectures with the highest predicted performance) for the subsequent generation. Finally, individuals with the best fitness values from the final generation are chosen for real evaluation on the test set, yielding the final architecture and test accuracy.

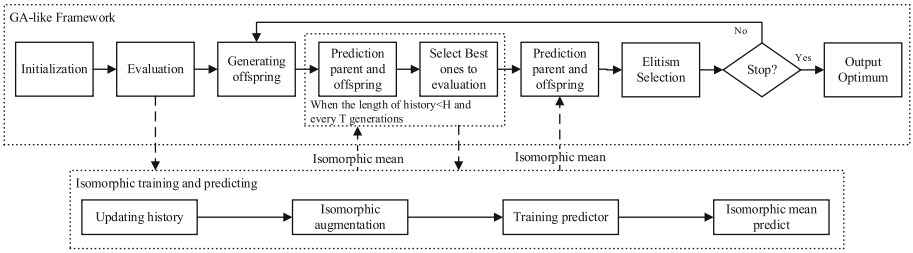


Fig. 3. The framework of ITP-ENAS, consisting with a GA-like search strategy and the isomorphic training and prediction progress.

Algorithm 1: Proposed ITP-ENAS Framework

Input: Size of population P, size of initial surrogate dataset N_{init}, number of offspring Q, number of generations $maxGen$, Number of individuals selected to expand the surrogate dataset K, interval generations between two real evaluations T, maximum length of history H.

Initialise P individuals (candidate architectures).
Truly evaluate (train) N_{init} individuals and get accuracy for them $R = \{(G_i, y_i)\}_{N_{init}}$.
Add records R to $history$, and train the surrogate model with loss function L calculated by means of Eqs. 3 and 4.
Predict P individuals with Eq. 5 and get fitness value $[y'_1, \ldots, y'_i]_P$.
While $Gen < maxGen$
Generate Q new individuals as offspring with binomial crossover and choice mutation.
If length$(history) < H$ and $Gen\%T = 0$
Predict offspring's fitness as $[y'_1, \ldots, y'_i]_Q$ with Eq. 5.
Selecting the K best predicted individuals in the parent and offspring with the best surrogate fitness for calculating their actual accuracy $R = \{(G_i, y_i)\}_K$.
Add records R to $history$, and train the surrogate model with Loss L.
Predict parent and offspring again and get fitness value $[y'_1, \ldots, y'_i]_{P+Q}$.
Else
Predict P individuals with Eq. 5 and get fitness value $[y'_1, \ldots, y'_i]_P$.
End
Elite retention of P individuals based on fitness values of parents and offspring.
$Gen \mathrel{+}= 1$.
End
Get last retained P individuals and get best individual based on fitness value.
Truly evaluate the best individual X_{best} and get the final test accuracy.
Return X_{best}

3 Experimental Results

To evaluate the performance of the proposed ITP-ENAS and compare it against state-of-the-art algorithms, we selected NASBench-101 and NASBench-201. Each experiment was conducted five times, and the mean and standard deviation of each experiment were reported.

3.1 Performance-Based Comparison

We employed a population size of $P = 250$ individuals and selected $N_{init} = 100$ for the initial surrogate training. The number of offspring architectures is set to $Q = 500$,

Table 1. Performance of the searched networks on NASBench-101

Methods	#Queries	Accuracy (mean ± std)	Accuracy (best)	Rank (%)
NAR (statistics) [7]	4236	94.07 ± 0.09	94.19	0.0054
NAR (random) [7]	4236	94.06 ± 0.04	94.10	0.0061
Peephole* [13]	1000	93.41 ± 0.34	–	1.6387
E2EPP [5]	1000	93.77 ± 0.13	–	0.1445
SSANA [14]	1000	94.01 ± 0.12	–	0.0111
HAAP [11]	1000	94.09 ± 0.11	–	0.0038
NAO* [15]	1000	–	93.74	0.1900^{+}
RFGIAug [16]	424	–	94.23	0.0005^{+}
RFGIAug [16]	1000	–	94.20	0.0009^{+}
MbML-NAS(GB) [17]	860	93.26 ± 0.01	–	3.0466
HybridNAS [18]	250	–	94.10	0.0061^{+}
HybridNAS [18]	400	–	94.17	0.0017^{+}
GenNAS-N [19]	500	93.92 ± 0.004	–	0.0321
BANANAS* [20]	500	–	94.08	0.0047^{+}
ReNAS [8]	423	93.95 ± 0.11	–	0.0222
CTNAS [21]	423	93.92 ± 0.18	94.22	0.0321
Random Search* [21]	423	89.31 ± 3.92	93.46	71.2759
RegressionNAS* [21]	423	89.51 ± 4.94	93.65	68.6082
ITP-ENAS (Ours)	424	**94.18 ± 0.001**	**94.23**	**0.0012** (**0.0005^{+}**)
Oracle	N/A	N/A	94.32	N/A

and every $T = 3$ generations, $K = 50$ individuals are chosen for real evaluation. The maximum capacity of the *history* archive is set to $H = 424$. The crossover and mutation probabilities are configured with $p_c = 0.5$ and $p_m = 0.1$.

Table 1 presents ITP-ENAS performance results on NASBench-101, comparing them with 29 other algorithms, including NAS methods with and without surrogate assistance. Methods denoted with '*' were not originally run on NASBench-101 but have since been applied to this search space in subsequent research. '#Queries' indicates the number of trained candidate architectures, serving as a measure of computational cost. 'Rank (%)' denotes the percentage rank based on mean accuracy within the search space, with '+' indicating the best result used for rank calculation in the absence of mean accuracy. Our method sets a new state-of-the-art for this space, achieving the highest average task accuracy with minimal standard deviation across 5 experiments, indicating its stable search capability. Although it didn't discover the best architecture in terms of test accuracy, ITP-ENAS still found an architecture with 94.23% accuracy, surpassing the best validation accuracy among all considered NAS methods. The average accuracy over 5 trials is 94.18%, outperforming other state-of-the-art methods.

Table 2. Performance of the searched networks on NASBench-201 with 3 dataset (CIFAR-10, CIFAR-100, ImageNet-16-120)

Methods	#Queries	CIFAR-10		CIFAR-100		ImageNet-16-120	
		Validation	Test	Validation	Test	Validation	Test
NAR [7]	1000	91.44 ± 0.10	94.33 ± 0.05	72.54 ± 0.44	72.89 ± 0.37	46.16 ± 0.37	46.66 ± 0.23
GenNAS-N [19]	1000	–	94.18 ± 0.10	–	72.56 ± 0.74	–	45.59 ± 0.54
NASWOT [25]	1000	89.69 ± 0.73	92.96 ± 0.81	69.86 ± 1.21	69.98 ± 1.22	43.95 ± 2.05	44.44 ± 2.10
MbML-NAS (RF) [17]	860	–	93.36 ± 0.20	–	70.33 ± 0.85	–	42.99 ± 4.21
MbML-NAS (GB) [17]	860	–	93.03 ± 0.52	–	70.02 ± 1.17	–	44.28 ± 1.42
ITP-ENAS (Ours)	424	**91.60 ± 0.12**	**94.36 ± 0.02**	**73.28 ± 0.32**	**73.44 ± 0.14**	**46.87 ± 0.35**	**46.81 ± 0.33**
NASWOT [25]	100	89.55 ± 0.89	92.81 ± 0.99	69.35 ± 1.70	69.48 ± 1.70	42.81 ± 3.05	43.10 ± 3.16
Random Search* [26]	100	91.07 ± 0.27	93.82 ± 0.24	71.44 ± 0.83	71.40 ± 0.83	45.37 ± 0.63	45.36 ± 0.67
REA* [27]	100	91.37 ± 0.25	94.06 ± 0.29	72.79 ± 0.69	72.72 ± 0.72	46.15 ± 0.45	45.99 ± 0.51
BANANAS*[20]	100	91.50 ± 0.15	94.23 ± 0.30	73.27 ± 0.57	73.25 ± 0.63	46.45 ± 0.25	46.31 ± 0.31
GP bayesopt*[20]	100	91.45 ± 0.23	94.16 ± 0.31	73.11 ± 0.65	73.05 ± 0.75	46.51 ± 0.25	46.25 ± 0.34
DNGO* [26]	100	91.41 ± 0.17	94.08 ± 0.26	72.71 ± 0.66	72.66 ± 0.67	46.11 ± 0.44	46.00 ± 0.48
Bohamiann* [20]	100	91.41 ± 0.18	94.09 ± 0.26	72.73 ± 0.64	72.65 ± 0.66	46.15 ± 0.45	46.03 ± 0.48
GCN Predictor* [20]	100	91.02 ± 0.32	93.74 ± 0.33	71.43 ± 0.72	71.43 ± 0.77	45.47 ± 0.72	45.36 ± 0.78
BONAS* [26]	100	91.56 ± 0.10	94.32 ± 0.15	73.32 ± 0.40	73.30 ± 0.46	46.56 ± 0.19	46.31 ± 0.30
NPENAS-SSRL* [28]	100	91.56 ± 0.14	94.32 ± 0.19	73.47 ± 0.22	73.47 ± 0.30	46.53 ± 0.33	45.83 ± 0.60
NPENAS-CCL* [28]	100	91.57 ± 0.13	94.32 ± 0.19	**73.48 ± 0.15**	**73.49 ± 0.23**	**46.62 ± 0.34**	45.61 ± 0.41
SAENAS-NE [26]	100	91.58 ± 0.09	94.34 ± 0.12	73.46 ± 0.18	73.46 ± 0.20	46.59 ± 0.14	46.36 ± 0.26
ReNAS [8]	90	90.90 ± 0.31	93.99 ± 0.25	71.96 ± 0.99	72.12 ± 0.79	45.85 ± 0.47	45.97 ± 0.49
ITP-ENAS (Ours)	100	**91.59 ± 0.13**	**94.35 ± 0.03**	73.34 ± 0.37	73.36 ± 0.19	46.10 ± 0.44	**46.40 ± 0.34**
Oracle	N/A	91.61	94.37	73.49	73.51	46.73	47.31

Table 2 presents results for the NASBench-201 search space. In this case, we set$H = 100$, $N_{init} = 30$, $K = 30$, and $T = 5$, with all other parameters consistent with those for NASBench-101. We compare our method with 20 others. ITP-ENAS achieves superior performance when training over 1% of the search space, significantly outperforming other state-of-the-art methods. When using less than 1% of the search space, we still achieve the best results on CIFAR-10 and ImageNet-16-120 but observe slightly inferior performance compared to some methods on CIFAR-100.

3.2 Comparison of Surrogate Model

To demonstrate the effectiveness of the surrogate model in ITP-ENAS, we compared its predictive ability with 10 competitor algorithms on NASBench-101, as shown in Table 3. We randomly sampled #Querys architectures and trained the surrogate model using validation accuracy as the label. We then predicted the accuracy of 5000 new architectures and calculated the Kendall's Tau between the predicted and actual values. Our method achieved the highest prediction accuracy using only 424 #Querys, approximately 0.1% of the search space. The only method outperforming ITP-ENAS was HAAP, but it required 1,000 #Querys, over twice the computational budget of ITP-ENAS.

Table 3. Kendall's Tau correlation index between the test ground-truth and surrogate ranking, and number of queries to train the surrogate models on the NASBench-101 dataset.

Methods	Kendall's Tau	#Queries
RegressionNAS* [21]	0.430	423
NAO* [15]	0.6550	423
Peephole [13]	0.4556	424
Peephole [13]	0.4373 ± 0.0112	1000
E2EPP [5]	0.5038	424
E2EPP [5]	0.5705 ± 0.0082	1000
HAAP [11]	0.7010 ± 0.0022	424
HAAP [11]	0.7126 ± 0.0024	1000
NPNAS [29]	0.6945	424
SSANA [14]	0.6541 ± 0.0078	1000
ReNAS [8]	0.6574	424
ITP-ENAS (Ours)	0.7084 ± 0.0077	424

4 Conclusion

The present paper proposes a novel surrogate model for NAS, with training augmented by the use of isomorphic graphs devised from a candidate architecture. A domain-specific isomorphic loss is also formulated to perform training combined with a standard MSE.

This method significantly expands the training dataset of the surrogate model, enhancing its prediction performance. Its benefit derives from the utilisation of evaluated individuals to generate new training data by changing the order of internal nodes without any additional overhead. Experimental results carried out on NASBench-101 and NASBench-201 display outstanding results for NAS that uses only 424 actual fitness evaluations and establishes a new state-of-the-art for both search spaces. This method can be applied to any neural-network-based neural predictor and appears promising in conjunction with the use of random forest. Further investigations include the study of the suitability of isomorphic data for various machine learning models.

Acknowledgments. This work was partially supported by the National Natural Science Foundation of China (62376127, 61876089, 61876185, 61902281), the Natural Science Foundation of Jiangsu Province (BK20141005) and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (14KJB520025), Jiangsu Distinguished Professor Programmer.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G.G., Tan, K.C.: A survey on evolutionary neural architecture search. IEEE Trans. Neural Netw. Learn. Syst. **34**, 550–570 (2023)
2. Real, E., et al.: Large-scale evolution of image classifiers. In: International Conference on Machine Learning, pp. 2902–2911 (2017)
3. Sun, Y., Xue, B., Zhang, M., Yen, G.G.: Completely automated CNN architecture design based on blocks. IEEE Trans. Neural Netw. Learn. Syst. **51**, 1242–1254 (2020)
4. Lu, Z., Sreekumar, G., Goodman, E., Banzhaf, W., Deb, K., Boddeti, V.N.: Neural architecture transfer. IEEE Trans. Pattern Anal. Mach. Intell. **43**, 3037–3054 (2021)
5. Sun, Y., Wang, H., Xue, B., Jin, Y., Yen, G.G., Zhang, M.: Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. IEEE Trans. Evol. Comput. **24**, 350–364 (2020)
6. Lu, Z., Deb, K., Goodman, E., Banzhaf, W., Boddeti, V. N.: NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I, pp. 35–51. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_3
7. Guo, B., et al.: Generalized global ranking-aware neural architecture ranker for efficient image classifier search. In: Proceedings of the 30th ACM International Conference on Multimedia, pp. 3730–3741. ACM, Lisboa Portugal (2022)
8. Xu, Y., et al.: ReNAS: Relativistic Evaluation of Neural Architecture Search. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4409–4418. IEEE, Nashville, TN, USA (2021)
9. Xie, L., Yuille, A.: Genetic CNN. In: 2017 IEEE International Conference on Computer Vision, pp. 1379–1388 (2017)
10. Irwin-Harris, W., Sun, Y., Xue, B., Zhang, M.: A graph-based encoding for evolutionary convolutional neural network architecture design. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 546–553 (2019)

11. Liu, Y., Tang, Y., Sun, Y.: Homogeneous architecture augmentation for neural predictor. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 12229–12238. IEEE, Montreal, QC, Canada (2021)
12. Lu, Z., Cheng, R., Jin, Y., Tan, K.C., Deb, K.: Neural architecture search as multiobjective optimization benchmarks: problem formulation and performance assessment. IEEE Trans. Evol. Comput. **139**, 323–337 (2024)
13. Deng, B., Yan, J., Lin, D.: Peephole: predicting network performance before training. http://arxiv.org/abs/1712.03351 (2017)
14. Tang, Y., et al.: A semi-supervised assessor of neural architectures. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1807–1816 (2020)
15. Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.-Y.: Neural architecture optimization. In: Advances in Neural Information Processing Systems, pp. 7827–7838. Curran Associates, Inc. (2018)
16. Xie, X., Sun, Y., Liu, Y., Zhang, M., Tan, K.C.: Architecture augmentation for performance predictor via graph isomorphism. IEEE Trans. Cybern. **54**, 1828–1840 (2024)
17. Sun, Y., et al.: Neural architecture search with interpretable meta-features and fast predictors. Inf. Sci. **649**, 119642 (2023)
18. Xun, Z., Songbai, L., Ka-Chun, W., Qiuzhen, L., Kaychen, T.: A hybrid search method for accelerating convolutional neural architecture search. In: Proceedings of the 2023 15th International Conference on Machine Learning and Computing, pp. 177–182. ACM, Zhuhai China (2023)
19. Li, Y., Hao, C., Li, P., Xiong, J., Chen, D.: Generic neural architecture search via regression. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., and Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, pp. 20476–20490. Curran Associates, Inc. (2021)
20. White, C., Neiswanger, W., Savani, Y.: BANANAS: Bayesian optimization with neural architectures for neural architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 10293–10301 (2021)
21. Chen, Y., et al.: Contrastive neural architecture search with neural architecture comparators. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9497–9506 (2021)
22. Guo, Z., et al.: Single path one-shot neural architecture search with uniform sampling. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI, pp. 544–560. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58517-4_32
23. Chu, X., Zhang, B., Xu, R.: FairNAS: rethinking evaluation fairness of weight sharing neural architecture search. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 12219–12228. IEEE, Montreal, QC, Canada (2021)
24. Liu, H., Simonyan, K., Yang, Y.: DARTS: differentiable architecture search. In: International Conference on Learning Representations (2018)
25. Mellor, J., Turner, J., Storkey, A., Crowley, E.J.: Neural architecture search without training. In: Proceedings of the 38th International Conference on Machine Learning, pp. 7588–7598. PMLR (2021)
26. Fan, L., Wang, H.: Surrogate-assisted evolutionary neural architecture search with network embedding. Complex Intell. Syst. **9**, 3313–3331 (2023)
27. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4780–4789 (2019)

28. Wang, H., et al.: Self-supervised representation learning for evolutionary neural architecture search. IEEE Comput. Intell. Mag. **16**, 33–49 (2021)
29. Wen, W., Liu, H., Chen, Y., Li, H., Bender, G., Kindermans, P.-J.: Neural predictor for neural architecture search. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) European Conference on Computer Vision, pp. 660–676. Springer, Cham, Cham (2020)