J Inf Process Syst, Vol.18, No.2, pp.188~196, April 2022
https://doi.org/10.3745/JIPS.04.0242

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Path-Based Computation Encoder for Neural Architecture Search

Ying Yang, Xu Zhang*, and Hu Pan

Abstract

Recently, neural architecture search (NAS) has received increasing attention as it can replace human experts in designing the architecture of neural networks for different tasks and has achieved remarkable results in many challenging tasks. In this study, a path-based computation neural architecture encoder (PCE) was proposed. Our PCE first encodes the computation of information on each path in a neural network, and then aggregates the encodings on all paths together through an attention mechanism, simulating the process of information computation along paths in a neural network and encoding the computation on the neural network instead of the structure of the graph, which is more consistent with the computational properties of neural networks. We performed an extensive comparison with eight encoding methods on two commonly used NAS search spaces (NAS-Bench-101 and NAS-Bench-201), which included a comparison of the predictive capabilities of performance predictors and search capabilities based on two search strategies (reinforcement learning-based and Bayesian optimization-based) when equipped with different encoders. Experimental evaluation shows that PCE is an efficient encoding method that effectively ranks and predicts neural architecture performance, thereby improving the search efficiency of neural architectures.

Keywords

Neural Architecture Search, Path-based Computation, Performance Predictor

1. Introduction

Recently, there has been an increased interest in neural architecture search (NAS). It can replace human experts in designing the architecture of neural networks for different tasks, and has achieved remarkable results in numerous challenging tasks. During the search process, obtaining the performance of the model is extremely costly. A promising approach is to evaluate the neural architecture using specialized performance predictors. The encoder, regressor, and loss function constitute a normal performance predictor. The encoder affects the generalization ability of the entire predictor. Recent studies have mainly considered neural architectures as sequences or graphs. Adjacency matrix encoding is the most commonly used sequence-based encoding method, which concatenates the flattened adjacency matrix with the operation vector. The operation vector is represented by fixed encoding, such as an integer vector or a one-hot vector. As a result of this fixed encoding, the relationship between operations cannot be portrayed; for example, operations of the same type should be more similar than those of different types.

Manuscript received January 21, 2022; first revision March 7, 2022; accepted March 8, 2022.

***Corresponding Author:** Xu Zhang (zhangx@cqupt.edu.cn)

Dept. of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China (1608559026@qq.com, zhangx@cqupt.edu.cn, 1435896703@qq.com)

Sequences can also be encoded using path-based encoding [1]. Path-based encodings use the set of paths that traverse the architecture directed acyclic graph (DAG) from input into the output by providing a binary feature. In this case, the size of the encoding vector increases exponentially with the number of nodes. Graph convolutional networks (GCNs) have been applied to graph representations in graph-based encoding. The operation on the node is used as a feature of the node, and each node aggregates the features of neighboring nodes to capture the local features in the graph. However, as the neural architecture is a computation graph, we should focus on the output of the whole graph rather than some local information, and the representation of each node should be a computation rather than a property.

Our path-based computation encoding (PCE) simulates the computation of information on each path in the architecture, which is represented by aggregating the output of each path. Intuitively, PCE has some powerful advantages. These features are not as interdependent as adjacency matrix encoding, because each feature represents a unique path of the data tensor from the input node to the output node. Inspired by the success of [2] in learning good operation embeddings, we proposed the application of embedded operations in the encoder to rectify the limitation of not being capable of reflecting the relationship between operations in the operation vector. The predictive performance of PCE was evaluated on two popular NAS search spaces: NAS-Bench-101 [3] and NAS-Bench-201 [4]. We adopted reinforcement learning (RL) and Bayesian optimization (BO) to conduct the NAS experiments. Based on the experimental results, PCE achieved better prediction performance than the other predictors in the two search spaces and achieved better search results when using the same search strategy.

Our contributions are summarized as follows:

- We proposed a method to encode the neural architecture that simulates the computation of each path in a neural network. A bidirectional long short-term memory network (BiLSTM) was used to model the computation in two directions: forward propagation and backward propagation.
- We improved the generalization of the neural predictor by using a continuous representation of the applied operators and the transformer to aggregate the encoding of each path to produce encoding for the entire network.
- Experimental evidence illustrating the efficacy of the proposed method. When PCE is followed by a regressor for performance prediction, it is significantly better than other encoders in terms of prediction accuracy and generalization, effectively improving the performance of the predictor.

2. Related Work

2.1 Neural Architecture Search

NAS has been successful in designing neural architectures automatically and has been applied to a wide variety of tasks, such as object detection, semantic segmentation, and few-shot learning [5]. Based on different search strategies and assumptions, many NAS algorithms have been proposed to improve the search speed and performance of the neural architecture, including reinforcement learning-based, evolutionary algorithm-based, Bayesian, gradient-based, and accuracy prediction-based methods [6].

2.2 Neural Architecture Encoders

Early NAS methods were sequence-based encodings, such as the adjacency matrix [3], and the obtained

features can be highly dependent on each other [1]. Recently, several studies have used input adjacency matrices encoding different neural networks—MLP, long short-term memory network (LSTM), and transformer—to obtain better sequence-based features. However, these approaches can only model topological information implicitly, which reduces the representation capability of the encoder. Path-based encoding [7] is another commonly used sequence-based encoding method. When path-based encoding is used, the length of the encoding vector increases exponentially as the number of nodes increases. The use of graph-based encoding that takes advantage of topological information provides better performance. In these methods, the adjacency matrix of the graph and one-hot encoding of each node are embedded into a fixed-length vector representation using a GCN [6,8]. Our work is similar to that of GATES [9] and D-VAE [10] in considering the embedding of neural architectures as computations in the architectures.

3. Method

In this section, we describe our proposed PCE, which simulates the computation of different paths in a neural network and the aggregation in the output layer. Compared with the traditional encoding scheme based on a sequence or graph, PCE encodes the computation of the entire graph instead of encoding the structure, and each feature represents a unique data computation path. Therefore, PCE is more interpretable than the adjacency matrix, which breaks the strong dependency between the features.

3.1 Operation Embeddings

Inspired by the success of operation embeddings [2], we applied a variational graph auto-encoder (VGAE) to learn operation embeddings instead of using a fixed operation vector, and the learned operation embeddings were incorporated into the BiLSTM. We define a neural network with a DAG as $G = (V, E)$, where V is the set of nodes and E is the set of edges. Given DAG G, we assume that there is a single input node and a single output node. A graph representing the neural architecture is represented by an adjacency matrix A and an operation matrix X. We trained the DVAE-EMB [2] model to obtain the operational embeddings $O(X)$, and the parameter settings in the model were consistent. More detailed information can be found in [2].

3.2 Path-based Computation Encoder

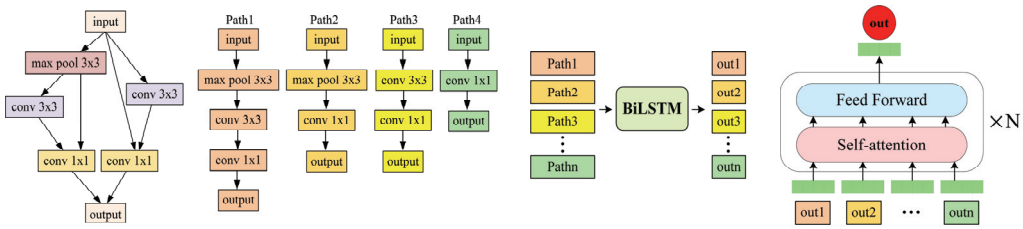

Fig. 1. (a) Example of data computation paths in the neural architecture. (b) Use BiLSTM encoding each computation path. (c) Illustration of the aggregating process of all computation path outputs.

As shown in Fig. 1(a), the input data were computed along each path from the input node to the output node. Therefore, we first determined all computed paths of the data flow and then encoded the computations on each path. The forward propagation process of the input data over the neural architecture is computed in topological order, the computational output of the previous operation affects the subsequent computations in the neural network, and the number of operation nodes on each computational path is not fixed. Based on the above observations, we adopted LSTM networks, which are suitable for processing ordered and variable-length sequences for modeling. Because the computation of the neural network is divided into two stages, forward and backward propagation, we incorporated the learned operation embeddings into the BiLSTM to encode the computation of each path.

The BiLSTM can be described by the following formulas

$$\left[h_1^1, h_2^1, \cdots, h_i^1\right] = \mathrm{LSTM}\left(C_0, \mathcal{O}(input), \mathcal{O}(\mathbf{X} \setminus \{input, output\})\right) \tag{1}$$

$$\left[h_1^2, h_2^2, \cdots, h_i^2\right] = \mathrm{LSTM}\left(C_0', \mathcal{O}(output), \mathcal{O}(\mathbf{X} \setminus \{input, output\})\right) \tag{2}$$

where $O(input)$ is the embedding of the input node and $O(output)$ is the embedding of the output node, and we use them as the initial hidden states of the forward and backward layers, respectively. C_0 and C_0' are the initial cell statuses that are randomly initialized.

$$h_i = concat\left\{h_i^1, h_i^2\right\} \tag{3}$$

$$out_n = concat\left\{h_1, h_2, \cdots, h_i\right\} \tag{4}$$

where out_n is the output state of BiLSTM, which represents the computational encoding of the n^{th} path in neural architecture, as shown in Fig. 1(b).

After all computation paths were encoded, we aggregated these outputs as the final encoding of the entire neural architecture, as shown in Fig. 1(c). We introduced a transformer, a powerful aggregation function, to obtain a representation of the entire architecture by aggregating the features of all paths in the neural architecture using a self-attention mechanism. This network includes a self-attention mechanism and fully connected networks with residual connections and layer normalization, which can avoid vanishing and exploding gradient problems. We aggregate the vector representations for all path outputs, and $out_n (i = 1,2, \ldots, n$, n is the total number of paths) as

$$Q = W^q O, K = W^k O, V = W^v O, O = \begin{bmatrix} out_1 \\ out_2 \\ \vdots \\ out_n \end{bmatrix} \tag{5}$$

where $W^q \in R^{d \times d}$ is a query-projection weight matrix, $W^k \in R^{d \times d}$ is a key-projection weight matrix, and $W^v \in R^{d \times d}$ is a value-projection weight matrix.

$$Z = LayerNorm\left(softmax\left(\frac{Q^{\mathrm{T}} K}{\sqrt{d}}\right)V + O\right) \tag{6}$$

where Z is the output after the input O passes through the self-attention layer, residual connection, and layer normalization.

The feedforward layer was relatively simple. It is a two-layer, fully connected layer. The activation function of the first layer was ReLU, and the second layer did not use the activation function. The corresponding formulae are as follows:

$$FFN(x) = \max\left(0, xW_1 + b_1\right)W_2 + b_2 \tag{7}$$

We input Z into the fully connected layer, and then had a residual connection and layer normalization to obtain the final output, which is expressed as follows

$$EMB = LayerNorm(Z + FFN(Z)) \tag{8}$$

We used the *EMB* as the encoding of the neural architecture.

The PCE framework is illustrated in Fig. 2. A cell-based search space is most commonly used to stack cells repeatedly to a final neural architecture; therefore, the NAS is transformed into a cell architecture search. When a cell architecture is sampled using the search algorithm, as shown in Fig. 2, its performance is predicted using the proposed method, and the predicted values are used to provide feedback and guidance to the search algorithm until the search is stopped.

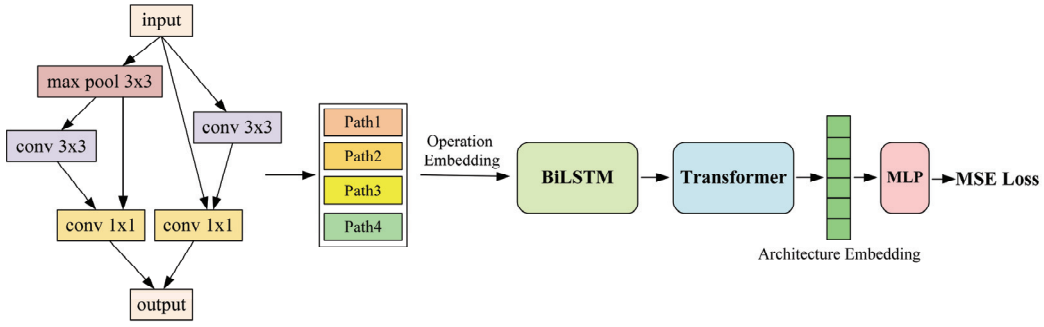

Fig. 2. PCE framework.

3.3 Optimization

To train the predictor, we used the mean square error (MSE) loss function

$$L = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - P(EMB(a_i))\right)^2 \tag{9}$$

qhere a_i is the given neural architecture, $EMB(a_i)$ is the encoding of a_i, y_i is the true performance of a_j, and $P(a_j)$ is the predicted value of a_j.

4. Experiments

4.1 Evaluation of Predictors on NAS-Bench-101

NAS-Bench-101 provides unified quantitative results. A cell is the smallest unit, containing an input node, an output node, and up to five intermediate nodes. In total, 423,624 neural architectures were evaluated, validated, and tested. In our experiment, to avoid the experimental result being the advantage

of the model over learning the dataset, we randomly generated 3,000 architectures in the search space each time, for which 1,000 were trained, 1000 were validated, and the remaining 1,000 were tested. We measured the performance of the predictor using two metrics: Kendall's tau and the mean absolute error (MAE). Both are widely used as neural predictors in NAS algorithms. There are four encoding methods based on sequence (path-based [7], transformer, MLP, and LSTM), two encoding methods based on the graph (two different GCN, i.e., neural predictors [6] and BONAS [8]), and two encoding methods based on computation (D-VAE and GATES). The results are shown in Table 1; the results in the table are the average results after 10 independents experiments.

Table 1. Ability of neural predictor with different encoders on NAS-Bench-101

Encoder	Valid MAE	Valid Kendall	Test MAE	Test Kendall	Encoding type
Path-based	2.402	0.646	2.524	0.657	Sequence-based
Transformer	2.316	0.549	2.221	0.610	Sequence-based
MLP	2.399	0.626	2.431	0.625	Sequence-based
LSTM	2.191	0.732	2.194	0.741	Sequence-based
Neural predictor	2.183	0.698	2.184	0.673	Graph-based
BONAS	2.179	0.682	2.143	0.671	Graph-based
D-VAE	2.205	0.622	2.198	0.635	Computation-based
GATES	2.401	**0.798**	2.421	0.787	Computation-based
PCE	**2.081**	0.782	**2.004**	**0.791**	Computation-based

The best values are shown in bold.

It can be observed that encoding the computations on the path improves both the prediction and ranking ability over using only path encoding, transformer, and LSTM, which indicates that encoding the calculations on the path and aggregation of the encoding extracts richer and more relevant features to the performance of the neural architecture. Our method is not much different from GATES' Kendall's tau on the validation and test sets, but it is nearly 2% lower than GATES in MAE. Our method has a lower MAE and higher Kendall's tau than all the sequence-based and graph-based encoding methods. This demonstrates that our encoding method has both ranking and absolute value prediction abilities.

4.2 Evaluation of Predictors on NAS-Bench-201

The search space defined by NAS-Bench-201 includes all possible 15,625 cell architectures produced by four nodes with five associated operation options. It provides training logs with the same settings for three datasets: CIFAR-10, CIFAR-100, and ImageNet-16-120, as well as the performance of each candidate architecture. For training, we used the first 90% (14,062) architectures of the CIFAR-10 dataset, and for testing the remaining 1,563 architectures. We only compared sequence-based and computation-based methods because the GCN encoding scheme cannot be directly implemented in the search space of NAS-Bench-201. The results are shown in Table 2.

The results show that our method can obtain a higher Kendall's tau and lower MAE than other encoders when there are only a small number of training samples. For example, in 141 training samples, our method can achieve a Kendall's tau of 0.7512 and an MAE of 2.649, which are better than those of other encoding methods. This shows the potential of the PCE encoder for generalization, which allows good performance predictors to be built with only a small amount of labeled data.

Table 2. MAE and Kendall's tau (KTau) of using different encoders on NAS-Bench-201

Encoder	Proportions of 14,062 training samples									
	1%		5%		10%		50%		100%	
	MAE	KTau	MAE	KTau	MAE	KTau	MAE	KTau	MAE	KTau
Path-based	2.862	0.1206	2.793	0.359	2.447	0.5398	1.5274	0.8567	0.595	0.8742
Transformer	2.879	0.3789	2.721	0.4578	2.403	0.6809	1.267	0.8754	0.604	0.8989
MLP	2.784	0.1274	2.682	0.4209	2.315	0.5609	1.124	0.8690	0.535	0.8901
LSTM	2.701	0.5664	2.601	0.6777	**2.012**	0.7498	1.013	0.8873	0.402	0.9092
GATES	2.905	0.7501	2.702	0.8509	2.321	**0.8721**	1.205	0.9015	0.647	**0.9237**
PCE	**2.649**	**0.7512**	**2.587**	**0.8621**	2.024	0.8697	**1.003**	**0.9101**	**0.397**	0.9209

The best values are shown in bold.

4.3 Search Experiment on NAS-Bench-101

Search processes can be improved when higher-performance predictors are used. Therefore, we used two search strategies and a variety of encoders to conduct search experiments on NAS-Bench-101. Table 3 shows the results of 500 independent runs, on average.

Table 3. Comparison of NAS performance using different encoders on NAS-Bench-101

NAS method		Number of queries	Test accuracy (%)
ADJ	RL	1,000	93.58
	BO	1,000	93.72
BONAS	RL	400	94.02
	BO	400	94.07
LSTM	RL	400	93.93
	BO	400	93.96
GATES	RL	400	94.03
	BO	400	94.04
PCE	RL	400	94.11
	BO	400	**94.14**

ADJ represents the adjacency matrix encoding and the bolded font shows the best value.

As shown in Table 3, the encoding method using the encoder can query approximately 400 times better than that using the adjacency matrix 1,000 times. Among these encoding methods, the architecture queried by PCE has the highest accuracy when the number of queries is the same, indicating that PCE is more competitive. This demonstrates that equipping a better neural architecture encoder in a predictor-based NAS process can improve search efficiency.

4.4 Search Experiment on NAS-Bench-201

NAS-bench-201 does not allow us to directly apply the GCN encoding scheme to the search space, and thus we can only compare with sequence-based and computation-based methods. Our method of implementation for CIFAR-10 is the same as that used in NAS-Bench-201, i.e., based on the validation

accuracy after 12 training epochs with converged learning rate scheduling. We have set the search budget at 1.2×10^4 seconds.. On CIFAR-100 and ImageNet-16-120, NAS is conducted using a budget based on the number of queries in CIFAR-10. The results are shown in Table 4, which are the average results of 500 independent runs.

Table 4. Mean and standard deviation of the validation and test accuracy of different algorithms under three datasets in NAS-Bench-201

Method		CIFAR-10		CIFAR-100		ImageNet-16-120	
		Validation	Test	Validation	Test	Validation	Test
ADJ	RL	91.03±0.33	93.82±0.31	72.35±0.63	72.13±0.79	45.58±0.62	45.30±0.86
	BO	90.82±0.53	93.61±0.52	72.59±0.82	72.37±0.90	45.44±0.70	45.26±0.83
LSTM	RL	90.91±0.32	93.72±0.39	72.94±0.46	72.72±0.46	46.05±0.54	45.89±0.56
	BO	90.68±0.49	93.64±0.32	72.95±0.59	72.85±0.67	46.01±0.61	45.99±0.46
GATES	RL	90.94±0.61	93.69±0.56	72.64±0.69	72.87±0.79	45.98±0.78	45.93±0.78
	BO	90.96±0.35	93.83±0.31	72.71±0.61	72.85±0.47	45.95±0.32	46.89±0.46
PCE	RL	91.36±0.35	94.16±0.58	**73.27±0.45**	73.16±0.35	46.20±0.31	46.10±0.53
	BO	**91.42±0.43**	**94.18±0.34**	73.01±0.69	**73.27±0.28**	**46.34±0.26**	**46.28±0.31**

ADJ represents the adjacency matrix encoding and the bolded font shows the best value.

As shown in Table 4, using PCE as an encoder can search for neural architectures with higher validation accuracy and test accuracy than using other encoders for the same search budget. The standard deviation of the search algorithm using the PCE encoder was also smaller than that of the other methods, reflecting the stability of the proposed method in the search task.

5. Conclusion

The PCE is a simple and effective neural architecture encoding method based on the computation path, which is encoded by simulating the real computation of a neural network. We treat the neural architecture as a computation graph and utilize BiLSTM to simulate the forward and backward propagation in the neural network, encoding each computational path in the neural architecture according to the topological order of the nodes, and then aggregating the information from each path using a transformer to make the whole encoding process closer to the real computational process of the neural network. We conducted many experiments using different encoders for prediction and search and proved the effectiveness of PCE on different downstream tasks.

Acknowledgement

This work was supported by the Natural Science Foundation of Chongqing, China under Grant cstc2019jscx-mbdxX0021, and in part by the Major Industrial Technology Research and Development Project of Chongqing High-tech Industry (No. D2018-82).

References

[1] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, Virtual Event, 2021, pp. 10293-10301.

[2] M. Chatzianastasis, G. Dasoulas, G. Siolas, and M. Vazirgiannis, "Operation embeddings for neural architecture search," 2021 [Online]. Available: https://arxiv.org/abs/2105.04885.

[3] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 7105-7114, 2019.

[4] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," 2020 [Online]. Available: https://arxiv.org/abs/2001.00326.

[5] X. Zhang, Y. Zhang, Z. Zhang, and J. Liu, "Discriminative learning of imaginary data for few-shot classification," *Neurocomputing*, vol. 467, pp. 406-417, 2022.

[6] W. Wen, H. Liu, Y. Chen, H. Li, G. Bender, and P. J. Kindermans, "Neural predictor for neural architecture search," in *Computer Vision – ECCV 2020.* Cham, Switzerland: Springer, 2020, pp. 660-676.

[7] C. White, W. Neiswanger, S. Nolen, and Y. Savani, "A study on encodings for neural architecture search," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20309-20319, 2020.

[8] H. Shi, P. Pi, H. Xu, Z. Li, J. Kwok, and T. Zhang, "Bridging the gap between sample-based and one-shot neural architecture search with BONAS," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1808-1819, 2020.

[9] X. Ning, Y. Zheng, T. Zhao, Y. Wang, and H. Yang, "A generic graph-based neural architecture encoding scheme for predictor-based NAS," in *Computer Vision - ECCV 2020*. Cham, Switzerland: Springer, 2020, pp. 189-204.

[10] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-VAE: a variational autoencoder for directed acyclic graphs," *Advances in Neural Information Processing Systems*, vol. 32, pp. 1586-1598, 2019.

Ying Yang **https://orcid.org/0000-0003-0166-1241**

She received her B.S. degree in Communication Engineering from Chongqing University of Posts and Telecommunications, Chongqing, China in 2019. She is currently pursuing an M.S. degree in the Department of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing China. Her research interests include neural architecture search.

Xu Zhang **https://orcid.org/0000-0002-7051-2736**

He received his Ph.D. in Computer and Information Engineering at Inha University, Incheon, Korea, in 2013. He is currently an associate professor at Chongqing University of Posts and Telecommunications. His research interests include urban computing, intelligent transportation system, and few-shot learning.

Hu Pan **https://orcid.org/0000-0002-4285-0965**

He received his B.S. degree in Communication Engineering from Chongqing University of Posts and Telecommunications, Chongqing, China in 2020. He is currently pursuing an M.S. degree in the Department of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing China. His research interests include neural architecture search.