CARL: Causality-guided Architecture Representation Learning for an Interpretable Performance Predictor

Supplementary Material

1. Search Space Description

In this section, we will present detailed information about the five NAS search spaces used in the experiments.

- **NAS-Bench-101** [25] is an operation-on-node (OON) search space where node denotes the operation and edge denotes the connection. It is a cell-based search space that is composed of over 423k architectures. Each architecture cell includes at most seven nodes and nine edges. The search space provides the accuracy of architectures on the CIFAR-10 dataset.
- **NAS-Bench-201** [5] is an operation-on-edge (OOE) and cell-based search space which contains 15625 architectures. Unlike OON search spaces, the operation type is represented by edges on NAS-Bench-201. There exist four nodes and six edges in each architecture. NAS-Bench-201 provides the accuracy of architectures on CIFAR-10, CIFAR-100 [10], and ImageNet-16-120 [4] datasets.
- **TransNAS-Bench-101** [6] includes two types of search spaces. The micro one is cell-based and contains 4096 architectures, and the macro one is skeleton-based and contains 3256 architectures. The cell in the micro space is the same as NAS-Bench-201. We encode the cell in the macro space with the same encoding in Arch-Graph [7]. This search space provides architecture performance across seven tasks, including segmentation, regression, pixel-level prediction, and self-supervised tasks.
- **NAS-Bench-NLP** [9] is a cell-based and OON search space which contains 14322 architectures. The maximum number of edges and nodes in each cell is set to 26 and 24, respectively. This search space focuses on architectures for NLP tasks, providing architecture performance on Penn Tree Bank [17] and WikiText2 [16] datasets.
- **DARTS** [11] is a cell-based and OOE search space which contains around 10^{18} architectures without available performance. Each architecture is represented by a normal cell and a reduction cell, which has seven nodes and eight edges. In DARTS, we only search for the normal cell and keep the reduction cell identical.

To unify the encoding of architectures from different search spaces, we convert the architectures in the OOE search spaces into an OON format.

2. Search Algorithms

We integrate CARL into two popular predictor-based NAS frameworks: predictor-guided random search [1] and predictor-guided evolutionary search [20] on search spaces except for DARTS. The pipelines of two frameworks are presented in Algorithm 1 and Algorithm 2. The total query budget of the algorithms is set to N architectures. The search algorithms aim to explore the search space S in pursuit of the optimal architecture A^*.

Algorithm 1: Predictor-guided Random Search

```
Initialize: Sample a few architectures randomly
  from search space $S$ to construct an initialization
  space $S_0$, train them for ground-truth performance
  and add them to $history$.
while $|history| + |S_0| < N$ do
    Train Predictor $F$ with architectures in $history$ ;
    Sample $M$ candidate architectures at random
      from $S - history$ ;
    Utilize $F$ to predict the performance of each
      candidate architecture ;
    Update $history$ by adding architectures with
      top-$K$ predicted values ;
return Architecture $A'$ with best ground-truth
  performance in $history$
```

Algorithm 2: Predictor-guided Evolutionary Search

```
Initialize: Sample a few architectures randomly
  from search space $S$ to construct an initialization
  space $S_0$, train them for ground-truth performance
  and add them to $history$ and population $P$.
while $|history| + |S_0| < N$ do
    Train predictor $F$ with architectures in $history$ ;
    Select $T$ well-performing architectures in $P$
      through Tournament ;
    Mutate the selected architectures and generate
      $M$ candidate architectures ;
    Utilize $F$ to predict the performance of each
      candidate architecture ;
    Update $history$ and $P$ by adding architectures
      with top-$K$ predicted values ;
    Remove the oldest architecture from $P$;
return Architecture $A'$ with best ground-truth
  performance in $history$
```

The size of initialization space S_0 is set to 20, and the number of candidate architectures M is set to 200. The number of updated architectures in each generation K is set to {10,10,5,10} for NAS-Bench-101, NAS-Bench-201, TransNAS-Bench-101, and NAS-Bench-NLP, respectively.

3. A Motivating Example for Critical Features

As revealed in recent studies [21, 22], architecture features can be split into two parts: critical features that dominantly determine the performance and redundant features that have negligible effects. To explicitly display the impact of critical and redundant features on architecture performance, we give a motivating example on NAS-Bench-201 in Figure 1.

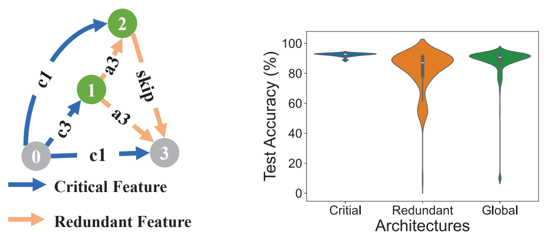

Figure 1. A motivating example for critical and redundant features on CIFAR-10 within NAS-Bench-201. (*Left*) Architecture features can be split into critical and redundant features. (*Right*) The performance gap between architectures containing these critical and redundant features.

We compare the test accuracy of 125 architectures containing the highlighted critical features (blue ones) with that of another 125 architectures containing the highlighted redundant features (orange ones) on the CIFAR-10 dataset. Figure 1 shows a significant gap between the accuracy of architectures containing critical and redundant features. The former is densely distributed in the high-accuracy region, while the latter has a much wider range. Besides, the distribution of architectures containing redundant features is very similar to the global distribution (15625 architectures), indicating that redundant features can not determine the architecture performance. These results suggest that critical features have a dominant impact on performance and should be identified correctly for better prediction results.

4. Explanations for Causal Analysis

In SCM, there exist probabilistic dependencies between causal features C and redundant features R: $C \dashleftarrow\dashrightarrow R$. There can be two types of relationships. First, C is the direct cause of R, i.e., $C \rightarrow R$. Alternatively, there is an unobserved common cause M, i.e., $C \leftarrow M \rightarrow R$. On both conditions, there exists a backdoor path between redundant features R and performance Y ($R \leftarrow C \rightarrow Y$ or $R \leftarrow M \rightarrow C \rightarrow Y$). This can induce spurious correlations between R and Y, causing poor generalization of performance predictors.

5. Implementation Details

We opt for a four-layer GCN [8] as the encoder and a three-layer MLP as the regressor to compose the predictor. It is trained with a learning rate of 1e-4, a dropout rate of 0.15, and a weight decay of 1e-3. In the settings of ranking experiments, the batch size is set to 32 and 8 on NAS-Bench-101 and NAS-Bench-201, respectively. λ_1 and λ_2 are set to 0.5 and 0.5. The margin a in hinge ranking loss is set to 0.5 on all experiments. As for the searching experiments, the batch size is set to 10 on all search spaces. The experiments on DARTS are conducted on a single RTX 3090 GPU, while others are conducted on a single RTX 2080 GPU.

6. Additional Experiments and Results

6.1. More Ablation Studies

In this section, we provide more results of the ablation study for the generation of interventional samples, optimization objective of redundant features, and hyperparameters in the final loss function on NAS-Bench-201.

Generation of Interventional Samples. During the causal intervention, we generate interventional samples by combining various redundant representations with critical ones in mini-batch architectures using random pairing. To investigate the effect of the random pairing, we replace it with the orderly pairing that performs the pairing operation in order. The comparison is presented in Table 1.

Table 1. Ablation results on generation of interventional samples.

Training Portion	0.5%	1%	3%
Orderly Pairing	0.614	0.652	0.714
Random Pairing	**0.656**	**0.697**	**0.755**

We can observe that random pairing outperforms orderly pairing by a significant margin. This is because the random pairing adds to the diversity of the interventional samples.

Optimization Objective of Redundant Features. To reduce the impact of redundant features on predictions, we use a uniform value, i.e., the mean accuracy of training samples, as the optimization objective. To inspect its effectiveness, we use another uniform value (zero) and randomly shuffle the ground-truth accuracy in a mini-batch to train CARL.

As shown in Table 2, using a uniform value for learning redundant representations can better facilitate the ranking ability of CARL. Although Random shuffling helps CARL learn representations uncorrelated with ground-truth, it also introduces additional noise during training. Meanwhile, the choice of uniform value is not that significant as long as it does not differ from the ground-truth accuracy by several

Table 2. Ablation results on optimization objective of redundant features.

Training Portion	0.5%	1%	3%
Random Shuffle	0.618	0.679	0.751
Uniform Zero	0.652	**0.697**	0.754
Uniform Mean	**0.656**	**0.697**	**0.755**

Table 3. Ablation results of λ_1 and λ_2 on NAS-Bench-201 with 1% training samples.

λ_1&λ_2	**0**	**0.2**	**0.5**	**0.7**	**1**
0	0.638	0.654	0.654	0.659	0.645
0.2	0.643	0.665	0.678	0.669	0.668
0.5	0.645	0.692	**0.697**	0.695	0.668
0.7	0.624	0.653	0.681	0.663	0.660
1	0.643	0.654	0.655	0.656	0.641

Table 4. Ablation results of λ_1 and λ_2 on NAS-Bench-201 with 3% training samples.

λ_1&λ_2	**0**	**0.2**	**0.5**	**0.7**	**1**
0	0.715	0.736	0.742	0.743	0.738
0.2	0.734	0.738	0.746	0.741	0.741
0.5	0.739	0.753	**0.755**	0.752	0.748
0.7	0.738	0.750	0.751	0.748	0.742
1	0.738	0.745	0.744	0.741	0.738

orders of magnitude. We can see that using zero and mean accuracy achieves similar results.

Impact of Disentanglement and Intervention Intensity. We provide the ablation results for λ_1 and λ_2 with more training data on NAS-Bench-201 in Table 3 and Table 4. We observe a similar trend that λ_1 and λ_2 can not be too small or too large. Hence, we use 0.5 and 0.5 for λ_1 and λ_2. We also validate the generalization of two hyperparameters on NAS-Bench-Graph [19] with the Cora dataset. As shown in Table 5, CARL achieves promising performance when λ_1 and λ_2 range from 0.2 to 0.7, outperforming competitive PINAT [15] (0.467). This demonstrates the strong generalizability of hyperparameters λ_1 and λ_2 to new search spaces, eliminating the need for fine-grained grid search and preserving CARL's data-efficiency advantages.

Table 5. Ablation results of λ_1 and λ_2 on NAS-Bench-Graph Cora with 1% training samples.

λ_1&λ_2	**0.2**	**0.5**	**0.7**
0.2	0.491	0.501	0.498
0.5	**0.509**	0.508	0.508
0.7	0.501	0.505	0.493

6.2. More Ranking Results

Results on More Training Samples. We validate the ranking ability of CARL using more training samples on NAS-Bench-101 and NAS-Bench-201. As shown in Table 6, CARL outperforms PINAT on NAS-Bench-201 with 5% and 10% training samples. On NAS-Bench-101, PINAT slightly takes the lead. We hypothesize that the sub-optimal performance of CARL stems from the narrowed distribution shift between training and test samples as the training portion grows. With sufficient and unbiased training samples, predictors are more likely to capture critical features of architectures for correct predictions.

Table 6. Ranking results of CARL on NAS-Bench-101 and NAS-Bench-201 with more training samples.

Search Space	NAS-Bench-101		NAS-Bench-201	
Training Portion	0.1%	1%	5%	10%
PINAT [15]	0.772	**0.846**	0.761	0.784
CARL (ours)	**0.774**	0.844	**0.794**	**0.805**

Results on Graph Tasks. We analyze the performance of CARL on NAS-Bench-Graph [19], a new challenging search space on graph node classification tasks. As shown in Table 7, CARL beats the competitive PINAT [15] on three datasets with 1% training data. This suggests the excellent generalization ability of CARL beyond conventional computer vision and NLP tasks.

Table 7. Ranking results on NAS-Bench-Graph.

Dataset	Cora	Arxiv	Citeseer
PINAT [15]	0.467	0.419	0.413
CARL (ours)	**0.508**	**0.452**	**0.456**

6.3. Results on Regression Errors

We report comparisons of CARL and PINAT in RMSE on NAS-Bench-201 in Table 8. CARL achieves lower RMSE than competitive PINAT.

Table 8. Regression results (RMSE) on NAS-Bench-201.

Training portion	0.5%	1%	3%
PINAT [15]	0.1105	0.1017	0.0879
CARL (ours)	**0.1072**	**0.0957**	**0.0730**

6.4. Plug-and-Play Solution

We also apply CARL to a simple MLP-based predictor to validate its effects. We use the implementation of OFA [2]. As shown in Table 9, the MLP-based predictor gains significant improvements on both search spaces, indicating the effectiveness of CARL as a plug-and-play solution.

Table 9. Ranking results of the MLP-based predictor with/without CARL on NAS-Bench-101 and NAS-Bench-201.

Search Space	NAS-Bench-101			NAS-Bench-201		
Portion	0.02%	0.04%	0.1%	0.5%	1%	3%
MLP	0.469	0.472	0.509	0.458	0.534	0.629
MLP w/ CARL	**0.515**	**0.596**	**0.644**	**0.529**	**0.613**	**0.722**

6.5. Results for Generalizable Encoding

To inspect the effectiveness of CARL for generalizable architecture encoding [13, 14, 18] where architecture features from multiple search spaces are used, we conduct the experiment on a cross-domain NAS task as CDP [13], a GCN-based predictor. Table 10 presents the improvement in Ktau with the proposed CARL, reflecting that it can work well with generalizable architecture encoding.

Table 10. Ranking results of CARL on the cross-domain task (NAS-Bench-101 and NAS-bench-201 to ShallowDARTS).

Method	CDP [13]	CDP w/ CARL
Ktau	0.5306	**0.5685**

6.6. Computational Efficiency

To investigate the computational efficiency of CARL, we compute the training and inference speed and compare it with two state-of-the-art performance predictors: PINAT [15] and NAR-Former [24]. We also compare the computational efficiency of GCN with and without CARL. For a fair comparison, we train all the predictors for 300 epochs with a training portion of 0.1% and a batch size of 10 on NAS-Bench-101 using a single RTX 2080 GPU.

Table 11. Training and inference times on NAS-Bench-101 with a training portion of 0.1%.

Method	Training time (sec)	Inference time (sec)	# Params (M)
PINAT [15]	750.61	253.22	0.55
NAR-Former [24]	2802.27	**66.23**	4.82
CARL (ours)	**242.49**	133.45	**0.21**

Table 12. Training and inference times of GCN with and without our CARL on NAS-Bench-101 with a training portion of 0.1%.

Method	Training time (sec)	Inference time (sec)	# Params (M)
GCN w/o CARL	185.79	132.59	0.15
GCN w/ CARL (ours)	242.49	133.45	0.21

According to Table 11, CARL consumes the least training time. This superior efficiency of CARL originates from its lightweight model, which has $2.6\times$ and $23.0\times$ fewer parameters than PINAT and NAR-Former, respectively. As for the inference time, CARL ranks second to NAR-Former. In practical performance evaluation, it takes significant time to obtain the ground-truth performance of an architecture. For instance, about 32 minutes is required to train a single architecture on the TPU v2 accelerator [25]. Therefore, the difference in inference time of predictors is negligible compared with the expensive cost of training architectures. The excellent computational efficiency of CARL allows it to achieve accurate and fast architecture performance prediction in challenging NAS tasks where the computation resource is limited. Table 12 reflects that CARL introduces minor computational overhead. This is because most of CARL's complexity lies in its GCN encoder rather than in other components.

6.7. Results on Real-World Networks

We search large-scale EfficientNets (depth=242) using NNLQP search space [12], which belongs to hardware-aware tasks. As shown in Table 13, CARL achieves the best MAPE and Acc(10%), demonstrating CARL's effectiveness on challenging real-world tasks.

Table 13. Searching results on NNLQP search space.

Method	CARL (ours)	NAR-Former [24]	nn-Meter [26]
MAPE↓	**18.54%**	28.05%	18.93%
Acc(10%)↑	**32.30%**	24.08%	23.40%

6.8. Searched Architecture in DARTS

We first search for architectures on CIFAR-10 and transfer the searched architecture to ImageNet. Figure 2 demonstrates the architecture in DARTS discovered by CARL, which achieves a test accuracy of 97.67% on CIFAR-10 and 76.1% on ImageNet, respectively. The normal cell and the reduction cell are the same in the searched architecture.

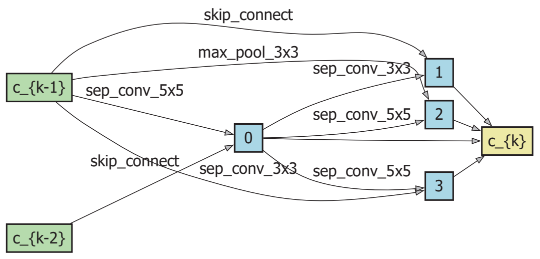

Figure 2. Cell architecture found by CARL on CIFAR-10 and ImageNet datasets.

6.9. More Visualization Results

Results on DARTS. We provide more visualization results on DARTS. The relative significance of architecture features in normal cells of SOTA methods [3, 15, 23] is visualized in Figure 3. We discover that skip connects and separable convolutions between node 0 and the input1/2 cause good performance.

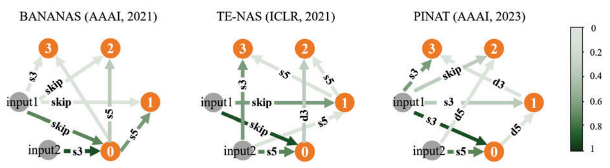

Figure 3. Visualization results for the significance of architecture features on DARTS.

Robustness of Feature Separation. We validate the robustness of feature separation by masking one feature at a time on NAS-Bench-201 CIFAR-10. As shown in Figure 4, masking high-significance features (OP1, OP4, and OP6) leads to large performance drops, demonstrating the effectiveness and robustness of CARL's feature separation.

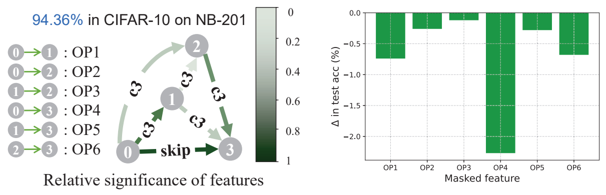

Figure 4. Visualization results for CARL's feature separation on NAS-Bench-201. (*Left*) Feature significance of the architecture feature. (*Right*) The performance drops after masking the corresponding feature.

7. Limitations

CARL is primarily designed for single-objective NAS and may require customized strategies for multi-objective scenarios, as different objectives may depend on different critical features. We will explore this in future work.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2), 2012. 1

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 3

[3] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021. 4

[4] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 1

[5] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2019. 1

[6] Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. TransNAS-Bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5251–5260, 2021. 1

[7] Minbin Huang, Zhijian Huang, Changlin Li, Xin Chen, Hang Xu, Zhenguo Li, and Xiaodan Liang. Arch-Graph: Acyclic architecture relation predictor for task-transferable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11881–11891, 2022. 1

[8] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016. 2

[9] Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, Alexander Filippov, and Evgeny Burnaev. Nas-bench-nlp: neural architecture search benchmark for natural language processing. *IEEE Access*, 10:45736–45747, 2022. 1

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 1

[11] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2018. 1

[12] Liang Liu, Mingzhu Shen, Ruihao Gong, Fengwei Yu, and Hailong Yang. Nnlqp: A multi-platform neural network latency query and prediction system with an evolving database. In *Proceedings of the 51st International Conference on Parallel Processing*, pages 1–14, 2022. 4

[13] Yuqiao Liu, Yehui Tang, Zeqiong Lv, Yunhe Wang, and Yanan Sun. Bridge the gap between architecture spaces via a cross-domain predictor. *Advances in Neural Information Processing Systems*, 35:13355–13366, 2022. 3, 4

[14] Hrushikesh Loya, Łukasz Dudziak, Abhinav Mehrotra, Royson Lee, Javier Fernandez-Marques, Nicholas D Lane, and Hongkai Wen. How much is hidden in the nas benchmarks? few-shot adaptation of a nas predictor. *arXiv preprint arXiv:2311.18451*, 2023. 3

[15] Shun Lu, Yu Hu, Peihao Wang, Yan Han, Jianchao Tan, Jixiang Li, Sen Yang, and Ji Liu. Pinat: A permutation invariance augmented transformer for nas predictor. In *Proc. of AAAI*, 2023. 3, 4

[16] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. 1

[17] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, pages 1045–1048. Makuhari, 2010. 1

[18] Keith G Mills, Fred X Han, Jialin Zhang, Fabian Chudak, Ali Safari Mamaghani, Mohammad Salameh, Wei Lu, Shangling Jui, and Di Niu. Gennape: Towards generalized neural architecture performance estimators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9190–9199, 2023. 3

[19] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. Nas-bench-graph: Benchmarking graph neural architecture search. *Advances in neural information processing systems*, 35:54–69, 2022. 3

[20] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4780–4789, 2019. 1

[21] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *International Conference on Learning Representations*, 2020. 2

[22] Xingchen Wan, Binxin Ru, Pedro M Esperança, and Zhenguo Li. On redundancy and diversity in cell-based neural architecture search. In *International Conference on Learning Representations*, 2021. 2

[23] Colin White, Willie Neiswanger, and Yash Savani. BANANAS: Bayesian optimization with neural architectures for

neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10293–10301, 2021. 4

[24] Yun Yi, Haokui Zhang, Wenze Hu, Nannan Wang, and Xiaoyu Wang. NAR-Former: Neural architecture representation learning towards holistic attributes prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7715–7724, 2023. 4

[25] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-Bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019. 1, 4

[26] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. Nn-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–93, 2021. 4