7. Appendix

7.1. Spearman Correlation on NAS-Bench-201

We also plot the Spearman ranking correlation between the ranking of architectures at current epoch and that of fully trained ones for every training epoch. As showing in Figure 4, the ranking correlation reaches 0.6 only after a few epochs of training and increases steadily after that on all three datasets. The trajectory of ranking correlation serves as an extra piece of evidence that shows we can terminate the training of architectures at early stages to save the resources without a big sacrifice of the search performance. Moreover, due to the multi-level nature of the NOSH algorithm, each level obtains a more accurate ranking between architectures than previous levels. At the top level, architectures will be fully trained, leading to the true ranking among them in terms of the final validation accuracy.

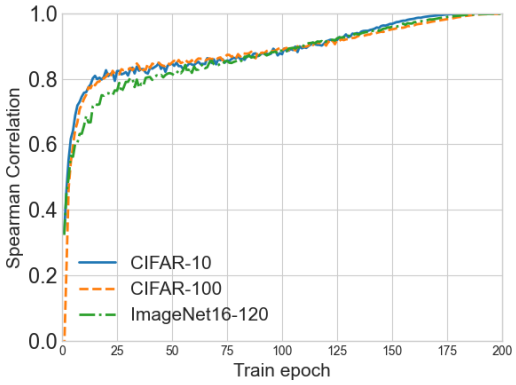

Figure 4: Spearman ranking correlation between the validation accuracy of partially and fully trained architectures on NAS-Bench-201.

7.2. Search Spaces

NAS-Bench-101 NAS-Bench-101 [46] is a generic cell-based search space where the searchable operations are defined on the nodes in the cell, and the edges denote the data flow. Each searchable cell includes seven nodes, with the first being the input node and the last being the output node. There are four operations for each searchable node in this search space: `conv_1x1`, `conv_3x3`, `conv_5x5` and `max_pool_3x3`, with `conv_5x5` approximated by two `conv_3x3`s. NAS-Bench-101 contains architectures with any arbitrary DAG structure between the input and output node with at most nine edges.

NAS-Bench-201 The search cell in NAS-Bench-201 consists of five nodes (two input/output nodes and three intermediate nodes) and six edges. Unlike NAS-Bench-101, the operations are defined on edges in this space, and nodes represent data flow. Each edge is associated with one of the five operations: `none`, `skip`, `conv_1x1`, `conv_3x3`, and `avg_pool_3x3`. Since we use the same GIN encoder as arch2vec [45], we follow their method to transform this operation-on-the-edge representation into a graph where nodes present searchable operations and edges represent the data flow to match NAS-Bench-101. We refer the reader to the original arch2vec paper [45] for further details.

The maximum training epoch on all three datasets is set to 200. However, for CIFAR-10, NAS-Bench-201 also provides a 12-epoch version, where architectures are trained for only 12 epochs with the learning rate scheduled accordingly. They use this version to bench its baselines on CIFAR-10. Therefore, for fair comparisons, we also use the 12-epoch version of CIFAR-10 for the main results in our experiment on NAS-Bench-201.

The DARTS Space Similar to NAS-Bench-201, the DARTS space [23] is an operation-on-the-edge space. Each cell contains six nodes, including two Input/Output nodes and four intermediate nodes. There are 14 possible edges in this search space, and eight of them will be selected to form an architecture. Every edge is associated with one of the following eight possible operations: `none`, `skip_connect`, `avg_pool_3x3`, `max_pool_3x3`, `sep_conv_3x3`, `sep_conv_5x5`, `dil_conv_3x3`, and `dil_conv_5x5`. For the DARTS space, we also transform its DAG into the unified graph representation where nodes present searchable operations and edges represent the data flow as done in NAS-Bench-201 [45].

7.3. Extended Discussion on Related Works

In this work, we focus on task-agnostic methods for improving the efficiency of predictor-based NAS. However, it is also possible to leverage task-specific prior knowledge to speed up architecture search. For example, FCOS [37] attempts to improve the search efficiency of RL-based NAS method on Object Detection tasks using fixed backbones and proxy tasks on VOC. Since the techniques proposed in these works are task-dependent, we could not compare them with our method or previous SOTA NAS algorithms in the main experiments.

The concept of pausing and resuming the training of a candidate has been explored in Bayesian Optimization [34]: FTBO [34] tries to decide when to pause or resume the training of a configuration via learning curve prediction. In comparison, NOSH adopts a fixed schedule for simplicity. In this sense, FTBO can be viewed as an orthogonal work to ours, and it might be an interesting future direction to study if FTBO could be applied to our framework to decide the NOSH schedules adaptively. Moreover, FTBO

focuses on the Hyperparameter Optimization task, whereas we mainly study Neural Architecture Search.

Neural Predictor [39] (NeuralPred) is an early and arguably the most straightforward predictor-based NAS algorithm. It trains a neural network predictor on a pool of N fully trained architectures at once, and use it to propose K new architectures. The total budget reported is > 100 architectures, which is higher than the latest SOTA predictor-based methods we compared with, such as arch2vec-BO [45] and BANANAS [40]. Moreover, NeuralPred focuses on the ProxylessNAS search space rather than the widely used DARTS Space. For these reasons, we exclude the comparison with this method in the main experiments. We encourage the readers to check out their paper for further details.

7.4. Extra Details on the Experimental Settings

Ranker Network As visualized in Figure 3, the ranker network consists of two MLPs on top of a pair of Siamese five-layer GIN encoders with shared weights. The GIN encoder produces a 16-dimensional embedding for each architecture. And the feature embeddings from two architectures are concatenated into a 32-dim feature. The first MLP transforms this feature into a 64-dim hidden vector, which will then be mapped to a 2-dim output by the second MLP. The GIN encoders are pretrained using reconstruction loss following arch2vec [45]; We refer the readers to their paper for further details of the pretraining step.

We train the ranker network with a batch size of 10 for 100 epochs using Binary Cross-Entropy loss and Adam optimizer. The learning rate is set as 0.01 and annealed to 0.00001 with a cosine schedule.

Train-free prior scores Abdelfattah *et al.* [1] demonstrates that several metrics previously used for network pruning [35] can serve as rough measures of architecture performance without training. In this work, we use the magnitude of model weights at initialization as our prior score due to its simplicity, although more complex and advanced metrics can be deployed to further improve the performance. Concretely, after the network is initialized, we sum up the magnitude of its weights and use it as the score. The implementation is taken directly from the official Synaptic-Flow [35] repo: https://github.com/ganguli-lab/Synaptic-Flow.

Search algorithm When proposing new architectures, we also deploy explicit exploration as done in BRP-NAS [12]. Concretely, the K proposals are constructed by selecting the top $\frac{K}{2}$ architectures using global ranking and randomly sampling the rest half from the top $2K$ architectures (excluding top $\frac{K}{2}$ to avoid duplicates). This strategy allows RANK-NOSH to explore more diverse architectures in the search space.

7.5. Complementary Results to Ablation Study

Table 8: Validation Accuracy of final architectures from RANK-NOSH on CIFAR-10, CIFAR-100 and ImageNet16-120 under various schedules and move ratios. Our method is relatively stable across various E and r on all three datasets.

Dataset	E	Search Budget	Valid Accuracy (%)
CIFAR-10	(10,50,200)	6,750	91.60 ± 0.03
	(10,50,100,200)	5,550	91.60 ± 0.02
	(5,25,50,200)	4,075	91.59 ± 0.03
	(5,10,25,200)	3,400	91.57 ± 0.06
CIFAR-100	(10,50,200)	6,750	73.49 ± 0.00
	(10,50,100,200)	5,550	73.49 ± 0.00
	(5,25,50,200)	4,075	73.42 ± 0.22
	(5,10,25,200)	3,400	73.42 ± 0.15
ImageNet16-120	(10,50,200)	6,750	46.42 ± 0.08
	(10,50,100,200)	5,550	46.37 ± 0.00
	(5,25,50,200)	4,075	46.47 ± 0.16
	(5,10,25,200)	3,400	46.33 ± 0.27

(a) Under different E

Dataset	r	Search Budget	Valid Accuracy (%)
CIFAR-10	0.7	9,750	91.58 ± 0.06
	0.6	7,400	91.59 ± 0.06
	0.5	5,550	91.60 ± 0.02
	0.4	4,100	91.58 ± 0.08
	0.3	2,950	91.40 ± 0.16
CIFAR-100	0.7	9,750	73.49 ± 0.00
	0.6	7,400	73.46 ± 0.11
	0.5	5,550	73.49 ± 0.00
	0.4	4,100	73.46 ± 0.11
	0.3	2,950	72.80 ± 0.5
ImageNet16-120	0.7	9,750	46.43 ± 0.13
	0.6	7,400	46.50 ± 0.25
	0.5	5,550	46.37 ± 0.00
	0.4	4,100	46.40 ± 0.08
	0.3	2,950	46.17 ± 0.5

(b) Under different r

We include extra ablation study results in this section. All experiments are conducted by running the search algorithm for 10 random seeds, as done in Section 5.

Results on other datasets We provide ablation study results for NOSH schedules on all three datasets on NAS-Bench-201. As shown in Table 8, RANK-NOSH is stable under a wide range of schedules and move ratios across datasets.

Effectiveness of the ranker model in RANK-NOSH To leverage the non-uniform signals produced by the NOSH algorithm, we adopt a ranker model as the performance predictor, optimized with discrete pairwise ranking loss. Compared with regression models, one potential downside of discrete pairwise loss is that it discards the fine-grain numerical values of validation accuracy. However, ranking-based methods also increase sample efficiency by creating $O(N^2)$ (pairs of) data points out of N original samples, which may cancel out the loss of fine-grain information.

Empirically, we observe a net gain of the adopted ranker model over the regression model used in arch2vec. To show this, we compare RANK with arch2vec at full budget (i.e., without early stopping or NOSH). As shown in Table 9, the ranker model alone leads to a near-oracle validation accuracy of $91.6\%/73.49\%/46.71\%$, outperforming arch2vec-BO.

Table 9: Validation accuracy (%) of the final architectures obtained by RANK and arch2vec-BO at full budget on NAS-Bench-201.

Dataset	Search Budget	arch2vec-BO	RANK-NOSH
CIFAR-10	20,000 (100%)	91.48 ± 0.16	91.60 ± 0.02
CIFAR-100	20,000 (100%)	73.29 ± 0.41	73.49 ± 0.00
ImageNet16-120	20,000 (100%)	46.27 ± 0.39	46.71 ± 0.12

Effectiveness of the NOSH algorithm in RANK-NOSH To further validate the necessity of NOSH algorithm over naive early stopping (ES), we compare RANK-NOSH with RANK-ES, i.e., replace the NOSH algorithm in the proposed method with early stopping. As shown in Table 10, RANK-NOSH consistently outperforms RANK-ES, demonstrating that the NOSH algorithm is critical to our framework. Note that from Table 6 and Table 10, we can see that the performance of ES is quite unstable over multiple runs; We conjecture that it is because the noisy signals produced by early stopping might mislead both predictor and final selection, resulting in much larger variances.

Table 10: Validation accuracy (%) of the final architectures obtained by RANK-NOSH v.s. RANK-ES on NAS-Bench-201.

Dataset	Search Budget	RANK-ES	RANK-NOSH
CIFAR-10	2,969	91.16 ± 0.32	91.56 ± 0.07
CIFAR-100	2,969	72.46 ± 0.30	73.44 ± 0.09
ImageNet16-120	2,969	45.42 ± 1.01	46.43 ± 0.21

7.6. Discovered Architectures

The best architecture discovered by RANK-NOSH on the DARTS space is visualized in Figure 5. As mentioned in the main text, we follow arch2vec [45] and use the same cell for both reduction and normal cells.

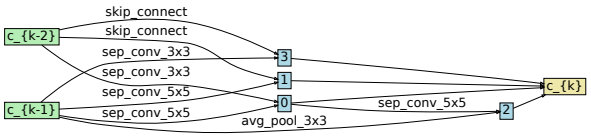

Figure 5: Cell Discovered by RANK-NOSH on the DARTS space.