HEP-NAS: Towards Efficient Few-shot Neural Architecture Search via Hierarchical Edge Partitioning

Jianfeng Li[1], Jiawen Zhang[1], Feng Wang[1*], Lianbo Ma[2*]

[1]School of Computer Science, Wuhan University
[2]College of Software, Northeastern University
fengwang@whu.edu.cn, malb@swc.neu.edu.cn

Abstract

One-shot methods have significantly advanced the field of neural architecture search (NAS) by adopting weight-sharing strategy to reduce search costs. However, the accuracy of performance estimation can be compromised by co-adaptation. Few-shot methods divide the entire supernet into individual sub-supernets by splitting edge by edge to alleviate this issue, yet neglect relationships among edges and result in performance degradation on huge search space. In this paper, we introduce HEP-NAS, a hierarchy-wise partition algorithm designed to further enhance accuracy. To begin with, HEP-NAS treats edges sharing the same end node as a hierarchy, permuting and splitting edges within the same hierarchy to directly search for the optimal operation combination for each intermediate node. This approach aligns more closely with the ultimate goal of NAS. Furthermore, HEP-NAS selects the most promising sub-supernet after each segmentation, progressively narrowing the search space in which the optimal architecture may exist. To improve performance evaluation of sub-supernets, HEP-NAS employs search space mutual distillation, stabilizing the training process and accelerating the convergence of each individual sub-supernet. Within a given budget, HEP-NAS enables the splitting of all edges and gradually searches for architectures with higher accuracy. Experimental results across various datasets and search spaces demonstrate the superiority of HEP-NAS compared to state-of-the-art methods.

Code — https://github.com/Jianf-l/hepnas

Introduction

In recent years, NAS has received widespread attention from both academia and industry (Jiang, Wang, and Bie 2024; Wang et al. 2023; Wu et al. 2024) for its ability to automatically search for optimal network architecture for specific tasks. Compared to traditional manually designed network (He et al. 2016), NAS simplifies the human iterative design process and uncovers more efficient and innovative network architectures.

NAS was first introduced by (Zoph and Le 2016) using reinforcement learning, which requires substantial computational resources, thereby impeding its practical application. To improve search efficiency, (Pham et al. 2018) proposed one-shot NAS, which integrates all potential networks within a supernet and trains them simultaneously using a weight-sharing strategy, then uses this supernet as an estimator to evaluate the performance of candidate architectures. This innovation significantly reduces the time cost from thousands of GPU-Days to merely a few. However, despite notable improvements, one-shot NAS is criticized (Bender et al. 2018; Wang et al. 2021) for its diminished effectiveness as a proxy for evaluating candidate architecture performance and difficulties in identifying superior architectures within the search space, due to the inherent properties of joint training, namely co-adaptation. To elaborate, consider two sub-supernets $\mathcal{N}_a(w_a, w_s)$ and $\mathcal{N}_b(w_b, w_s)$ that share the same weight w_s. If they produce mismatched gradients for w_s when optimized independently, then optimizing the whole supernet $\mathcal{N}_s(w_a, w_b; w_s)$ would lead to a compromise weight aimed at a global optimum, which may not be ideal for either $\mathcal{N}_a$ or $\mathcal{N}_b$ individually. As a result, evaluation of $\mathcal{N}_a$ and $\mathcal{N}_b$ might not reflect their true performance. To improve its accuracy, few-shot methods (Zhao et al. 2021; Hu et al. 2022; Ly-Manson et al. 2024) divide operations into distinct sub-supernets via edge-wise partition strategy, facilitating weight-sharing within each sub-supernet while maintaining separation across them, thereby mitigating disagreement among sub-supernets on how to update the weights of shared module, which proved to be effective.

Although few-shot methods outperform one-shot counterparts in performance, their partition strategy is not ideal. Firstly, it overlooks the interrelationships among edges, providing limited alleviation to co-adaption within a hierarchy. To begin with, selecting the optimal combination of operations for each intermediate node, works more relevant to the objectives of NAS than choosing the best operation for each edge independently. However, with edge-wise partition strategy, after segmenting a specific edge, each resulting sub-supernet still retains all operations on other edges within the same hierarchy, leading to co-adaptation within this hierarchy persisting during the subsequent joint training process until all edges in this hierarchy are split. As a result, when assessing the performance of candidate sub-supernets and determining the optimal connections for an intermediate node using weights inherited from the supernet, the performance estimation remains inaccurate since the combina-

*Corresponding authors.

tion of operations connected to this node is not fully trained in isolation, which can hinder the identification of architectures with truly high accuracy. Secondly, it maintains unnecessary search spaces, which splits all sub-supernets and constructs a full multifork partition tree, leading to significantly increased time overhead. Therefore, only a limited number of edges are split in most cases to balance search cost, constraining its improvement over one-shot methods. However, not all branches warrant further exploration, as high-performance architectures often favor similar operations on the same edge, rendering many other operations redundant (Wan et al. 2022). Therefore, exploring these search spaces may waste search budgets with little performance benefit.

In this paper, we propose HEP-NAS, a hierarchical edge partitioning algorithm to address above issues. The key concept involves two alternate process. The first one is splitting all edges via hierarchy-wise partition to ensure operation combinations can be trained in isolation. The other one is gradually narrowing the search space to save search cost. Furthermore, as greedily selecting the optimal search space may entail the risk of falling into a local optimum and require sufficient retraining process, we introduce SMD (search space mutual distillation) to ensure the efficacy of search space reduction. Specifically, sub-supernets partitioned within the same hierarchy engage in collaborative learning and mutual teaching to boost the convergence of individual sub-supernets, while the optimal sub-supernet identified in the preceding hierarchy guide all sub-supernets on the current hierarchy to mitigate performance degeneration caused by discretization, thereby offering more accurate selection of the optimal search space.

To evaluate the efficacy of HEP-NAS, we conducted extensive experiments across various datasets and search spaces. The experimental outcomes demonstrate that HEP-NAS outperforms alternative state-of-the-art methods. The discovered architectures not only achieve 97.56% accuracy on CIFAR-10, but also exhibit a 76.4% top-1 accuracy when directly transferred to ImageNet, underscoring the exceptional generalization capability of HEP-NAS.

Related Work

One-shot methods builds one single supernet that includes all candidate architectures in the search space as sub-paths, where architectures can share weights with each other as long as they contain the same operation on the same edge. In the training process, it employs a weight-sharing strategy to optimize the weight of supernet $\mathcal{W}_{\mathcal{A}}$ only once. Subsequently, in the evaluation process, the sampled architectures treat the weights inherited from the supernet w_α as if they were trained alone and are ranked by their accuracy on the validation set to derive the optimal one. Later, differentiable methods (Liu, Simonyan, and Yang 2018; Chu et al. 2020a,b) eschew discrete architecture candidate searches in favor of continuous relaxation of the search space. This enables optimization through efficient gradient descent.

Despite the search efficiency, one-shot methods often yields suboptimal results due to inaccurate performance estimation of child networks caused by co-adaptation. Various methods have been introduced to improve its accuracy. For example, SPOS (Guo et al. 2020) and its successors (Chu et al. 2023; Chen et al. 2023; Zhang et al. 2024) construct a supernet with choice blocks, each containing different operations with only one activated during training to mitigate co-adaptation within the blocks. However, this approach gives rise to multi-model forgetting (Zhang et al. 2020), introducing additional complexity.

Few-shot NAS divides the supernet into sub-supernets through edge-wise partitioning, restricting weight-sharing within each sub-supernet while maintain separation across them. GM-NAS (Hu et al. 2022), an evolution of few-shot NAS, groups individual operations on each edge using gradient matching, which reduces the number of sub-supernets significantly, yet still splits limited edges due to maintaining unnecessary search spaces. Recent research (Ly-Manson et al. 2024) extends gradient matching and introduces diverse metrics to group operations within an edge but overlooks correlations among edges. Therefore, it is still required to effectively develop a more precise performance estimator to search for architectures with higher accuracy.

Methodology

Preliminaries

Gradient Matching Gradient matching approach proposed by (Hu et al. 2022) significantly reduces the number of generated sub-supernets after segmentation through grouping together operations whose standalone gradient is similar. The gradient matching metric introduced to quantify the similarity of operations can be formulated as,

$$GM(o_i^k, o_j^k) = \mathcal{S}_{cos}(\nabla\mathcal{L}(\mathcal{M}_{o_i}^k, \omega), \nabla\mathcal{L}(\mathcal{M}_{o_j}^k, \omega)) \quad (1)$$

where $\mathcal{S}_{cos}$ represents the cosine similarity function, o_i^k is the *i*-th operation on the *k*-th edge e_k, and $\mathcal{M}_{o_i}^k$ stands for a sub-supernet with only operation o_i left enabled on edge e_k. After evaluating the gradient matching score for each pair of operations on e_k, groups can be formed via min-cut optimization, defined as,

$$\mathcal{G}_k = \arg\min_{\mathcal{G}_k \subseteq \mathcal{O}} \sum_{o_i^k \in \mathcal{G}_k, o_j^k \in \mathcal{O}\setminus\mathcal{G}_k} GM(o_i^k, o_j^k) \quad (2)$$

where $\{\mathcal{G}_k, \mathcal{O}\setminus\mathcal{G}_k\}$ are the obtained operation groups on e_k.

We utilize the gradient matching technique to group operations, and progressively search for architectures with higher accuracy via alternate hierarchical edge partitioning and search space reducing process (see Fig. 1). Initially, HEP-NAS trains the entire supernet in a one-shot fashion. It then proceeds, following a hierarchical sequence, to rearrange operations within the current hierarchy, combining and assigning them to distinct sub-supernets. These sub-supernets undergo training using SMD, with the most accurate one on the validation dataset chosen as optimal for further splitting. This iterative process continues for subsequent hierarchies until all hierarchies have been partitioned. Details are present in the following subsections.

Hierarchical Edge Partitioning

Consider an intermediate node N_k within a given search space $\mathcal{S}$. N_k is connected to its predecessor nodes, where it

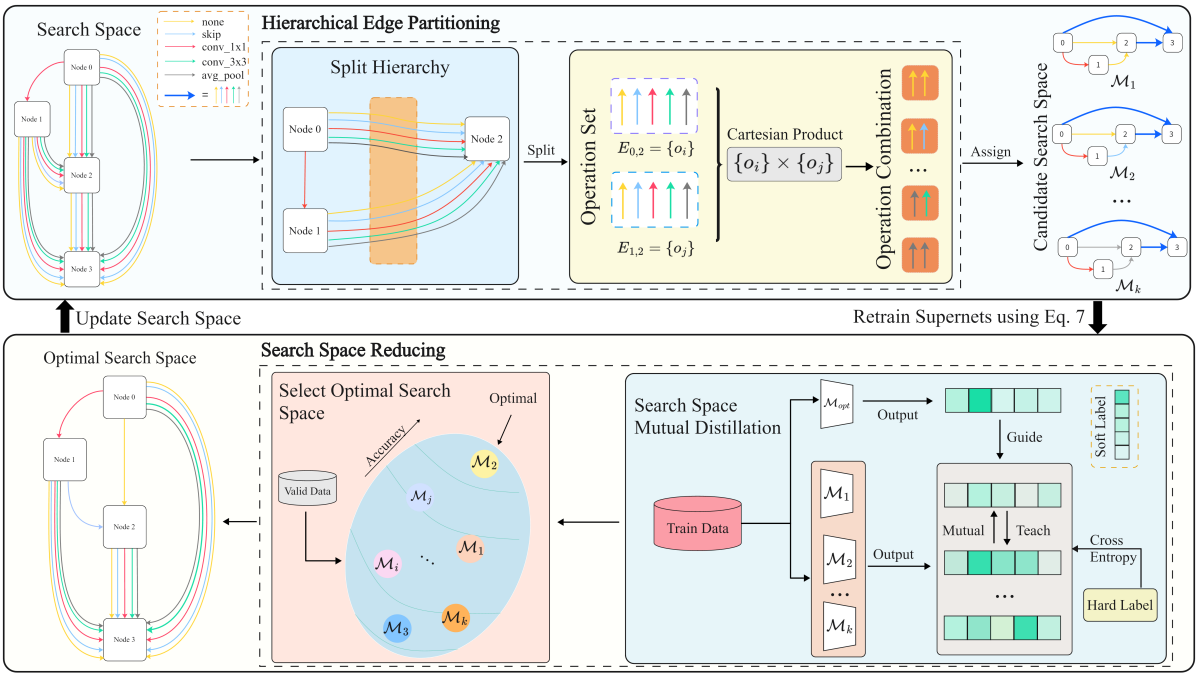

Figure 1: Overall illustration of HEP-NAS. A hierarchy-wise partition strategy is utilized to create sub-supernets. Subsequently, SMD is employed to expedite and stabilize the training process of these sub-supernets. The next step involves selecting the sub-supernet with the highest accuracy on the validation dataset, replacing the previous optimal one, and proceeding to split the next hierarchy until all hierarchies have been partitioned.

aggregates, collects, and processes the feature maps passed by them, without involving the connections of the rest of the nodes. This connectivity is expressed as:

$$x^{(k)} = \sum_{i<k} \sum_{o \in \mathcal{O}} o^{(i,k)}(x^{(i)}) \tag{3}$$

Here, $x^{(k)}$ denotes the output feature map of node N_k, and $o^{(i,k)}$ represents an individual operation on edge $E_{i,k}$ connecting N_i and N_k. Since the ultimate objective is to simultaneously select the optimal operation on each edge $E_{i,k}$ for $i = 0, 1, \cdots, k-1$, we treat these edges connected to the same end node N_k as a hierarchy h_k, defined as $h_k = \{E_{i,k}\}_{0 \le i < k}$, and directly search for the best combination of operations within h_k for N_k. To elaborate, let ψ^k represent the operation combinations in h_k. ψ^k is calculated by:

$$\forall k \in \{2, 3, \cdots, N-1\}, \psi^k = \prod_{i<k} \{\mathcal{G}^{(i,k)}, \mathcal{O} \setminus \mathcal{G}^{(i,k)}\} \tag{4}$$

where $\prod$ denotes the Cartesian product and $\mathcal{G}^{(i,k)}$, $\mathcal{O} \setminus \mathcal{G}^{(i,k)}$ are operation groups on $E_{i,k}$ calculated by Eq. 2. Instead of splitting one edge and assigning each individual group of operation into separate sub-supernets, we rearrange operation groups within the same hierarchy in combination and allocate these combinations to distinct sub-supernets $\{\mathcal{M}_k\}$ derived from the segmentation of h_k, with their edges replaced accordingly and their weights inherited from parent sub-supernet. Therefore, there will be only one operation group reserved on each edge connecting N_k to its predecessor node after segmentation, ensuring each combination can be fully trained in isolation as much as possible to obtain its truly performance.

The hierarchical segmentation strategy establishes a broader and shallower partition tree, with each layer of the tree training the sub-supernets created by the varied connectivity of the corresponding predecessor nodes directly. This approach more effectively addresses co-adaptation at the hierarchical level and searches out architectures with higher accuracy.

Additionally, unlike few-shot methods where edges can be split in any order, hierarchical edge partitioning splits each hierarchy in node numbering sequence. This is because each successive intermediate node is connected to more predecessor nodes than the previous one, and a premature perturbation of a significant number of edges can destabilize the architectural training, resulting in biased outcomes. Therefore, we gradually increase the strength of splitting, ensuring that the segmentation of h_{k+1} generates more sub-supernets than the segmentation of h_k.

Search Space Reducing

Selection of Optimal Search Space Prior to advancing the segmentation process on h_{k+1}, we retrain $\{\mathcal{M}_k\}$ for a few epochs and then identify the optimal sub-supernet $\mathcal{M}_{opt}$ for continued splitting, while discarding others to save search costs. Since the hierarchy-wise splitting strategy generates more sub-supernets in one stage and allows for fuller

Algorithm 1: Main process of HEP-NAS

Input: Search space $\mathcal{S}$, warmup epochs $warm\_epo$ to train sub-supernets before selecting the optimal one, the set of each epoch $split\_epos$ to split the supernet.
Parameters: The optimal sub-supernet $\mathcal{M}_{opt}$ found on previous hierarchy, the set of candidate operations $\{o^{(i,k)}\}_{i<k}$ on edges connecting the *i*-th node and *k*-th node.
Output: The most promising architecture

$start\_epo \leftarrow 0$;
$k \leftarrow 2$;
for end_epo in $split\_epos$ **do**
  Train $\mathcal{S}$ for $end\_epo - start\_epo$ epochs;
  $\mathcal{M}_{opt} \leftarrow \mathcal{S}$;
  Generate combinations ψ^k of $\{o^{(i,k)}\}_{i<k}$ using Eq. 4;
  Split $\mathcal{S}$ on the *k*-th hierarchy and assign each element in ψ^k into sub-supernets $\{\mathcal{M}_k\}$;
  for $i = 1, 2, \cdots, warm\_epo$ **do**
    Sequentially train each sub-supernet *m* in $\{\mathcal{M}_k\}$ for one epoch with the guidance of $\mathcal{M}_{opt}$ via minimizing Eq. 7;
  end for
  Select sub-supernet $\mathcal{M}^*$ with the highest accuracy on validation dataset using Eq. 5;
  $\mathcal{S} \leftarrow \mathcal{M}^*$;
  $start\_epo \leftarrow end\_epo$;
  $k \leftarrow k + 1$;
end for
Derive the final architecture from $\mathcal{S}$ using corresponding base model selection methods.

comparisons, it does not easily fall into a local optimum compared to the edge-wise strategy with pruning. This strategy greatly improves search efficiency by constantly trimming the number of sub-supernets when splitting all edges, while ensuring that the global optimal architecture can be searched as much as possible. In the meantime, by affecting only a few edges in one partitioning stage, gradually increasing the perturbation strength and shrinking search space, HEP-NAS is able to reduce the discretization error (He et al. 2024) caused by direct one-step selection from the supernet to final architecture, which can significantly change the computation of the feature map and leads to suboptimal results. This means it suffers less from accuracy degradation after discarding unimportant operations in the final model selection stage.

A straightforward criterion for selection is the accuracy of sub-supernets on the validation dataset,

$$\mathcal{M}_{opt} = \arg\max_{\mathcal{M}_k} \ Acc_{val}(\mathcal{M}_k; w_{\mathcal{M}_k}) \tag{5}$$

where Acc_{val} stands for the top-1 accuracy on validation dataset. However, segregating supernets inevitably disrupts the network structure and alters the computation of feature maps, leading to suboptimal weights for operations in the contracted architecture, thereby impacting sub-supernet performance and evaluation outcomes. Moreover, determining the appropriate number of training epochs for $\{\mathcal{M}_k\}$ poses a challenge, as too few epochs result in inaccurate performance evaluations due to delayed convergence of operations with more parameters, while too many epochs incur significant time overhead. These factors can lead to the selection of a suboptimal search space.

Search Space Mutual Distillation To stabilize the training process and boost the convergence of sub-supernets for better performance evaluation, we introduce SMD. Different from previous works commonly introducing a large pretrained model or a third-party model to distill knowledge, which limits flexibility when there are no existing pretrained models, SMD can employ knowledge transfer among segmented sub-supernets in the same hierarchy without requiring an external teacher model, as these sub-supernets can learn collaboratively and teach each other throughout the training process, thus facilitating the convergence and enhancing generalization capabilities of individual architectures. This is realized by appending an objective function $\mathcal{L}_{dis}$, the cross entropy with the soft target labels, and this cross entropy distills knowledge among candidate sub-supernets, defined as:

$$\mathcal{L}_{dis}(\mathcal{M}_i, \mathcal{M}_j) = \sum_{c=1}^{C} p_c(x; w_{\mathcal{M}_i}) \log \frac{p_c(x; w_{\mathcal{M}_i})}{p_c(x; w_{\mathcal{M}_j})} \tag{6}$$

where C refers to the number of classes, $p(x; w_{\mathcal{M}_i})$ is soft targets generated by a softmax function that converts feature logits to a probability distribution, and x stands for a batch of given input data. Meanwhile, to alleviate the performance degradation of sub-supernets caused by splitting operations, SMD also uses the optimal sub-supernet $\mathcal{M}_{opt}$ searched in the previous hierarchy to guide all the sub-supernets to be trained in the current hierarchy, thereby reducing the difference before and after discretization and stabilizing the training process. To elaborate, let $\mathcal{M}_m^k$ be the *m*-th segmented sub-supernet in h_k, $\mathcal{M}_{opt}^{k-1}$ be the optimal architecture found in the previous hierarchy h_{k-1}, and $\mathcal{L}_{cls}$ is the classification loss with the correct labels y, the complete training loss function of $\mathcal{M}_m^k$ can be formulated as:

$$\begin{aligned}\mathcal{L}(\mathcal{M}_m^k) =& \mathcal{L}_{cls}(p(x; w_{\mathcal{M}_m^k}), y) + \mathcal{L}_{dis}(\mathcal{M}_m^k, \mathcal{M}_{opt}^{k-1}) \\ &+ \frac{1}{\mathcal{D}^k - 1} \sum_{i=0, i\neq m}^{\mathcal{D}^k - 1} \mathcal{L}_{dis}(\mathcal{M}_m^k, \mathcal{M}_i^k)\end{aligned} \tag{7}$$

where $\mathcal{D}^k$ stands for the number of sub-supernets in h_k ($\mathcal{D}^k = |\{\mathcal{M}_k\}|$), and the initial weight $w_{\mathcal{M}_m^k}$ is inherited from $w_{\mathcal{M}_{opt}^{k-1}}$ via weight-sharing strategy.

Algorithm 1 summarizes the main process of HEP-NAS. The number of elements in $split\_epos$ is equal to the number of intermediate nodes since we split all edges with the same end node each time, and the initial value of *k* is the serial number of the first intermediate node. After the segmentation of the last hierarchy, we could derive the most promising architecture via correponding model selection methods (e.g., DARTS) from smaller search space $\mathcal{S}$.

Experiments and Analysis

We evaluate the performance of HEP-NAS on the DARTS search space with CIFAR-10, CIFAR-100, and ImageNet for image classifcation, as well as the NAS-Bench-201 search space with CIFAR-10, CIFAR-100, and ImageNet16-120. All experiments are conducted on single NVIDIA RTX 4090 GPU and the results are obtained in 4 independent runs. Experimental results demonstrate that HEP-NAS outperforms the state-of-the-art algorithm in most cases.

Architecture	Params (M)	Top-1 (%)	Cost (G·d)
ResNet (He et al. 2016)	1.7	4.61	-
DenseNet (Huang et al. 2017)	25.6	3.46	-
ProxylessNAS (Cai, Zhu, and Han 2018)	7.1	**2.08**	4.0
PC-DARTS (Xu et al. 2019)	3.6	2.57	**0.1**
P-DARTS (Chen et al. 2019)	3.4	2.50	0.3
DARTS- (Chu et al. 2020a)	3.5	2.50	0.4
DARTS-PT (Wang et al. 2021)	3.0	2.61	0.8
DDPNAS (Zheng et al. 2023)	3.16	2.59	0.075
ADARTS (Xue and Qin 2023)	2.9	3.70	0.2
IS-DARTS (He et al. 2024)	4.25	2.56	0.42
MOEA (Xue, Chen, and Słowik 2023)	3.0	2.77	2.6
MixPath-c (Chu et al. 2023)	5.4	2.6	0.25
EOFGA (Yuan, Xue, and Zhang 2023)	2.11	2.59	0.49
EAEPSO (Yuan et al. 2023)	2.94	2.74	2.2
few-shot DARTS (Zhao et al. 2021)	3.6	2.60	1.1
GM DARTS (Hu et al. 2022)	3.7	2.46	1.1
HEP-NAS	3.6	**2.44**	1.5

Table 1: Comparison results of HEP-NAS with state-of-the-art methods on CIFAR-10.

Architecture	Params (M)	Top-1 (%)	Cost (G·d)
ResNet (He et al. 2016)	1.7	22.10	-
DenseNet (Huang et al. 2017)	25.6	17.18	-
ShuffleNet (Zhang et al. 2018)	1.06	22.86	-
GDAS (Dong and Yang 2019)	3.4	18.38	0.2
P-DARTS (Chen et al. 2019)	3.6	17.20	0.3
PC-DARTS (Xu et al. 2019)	4.0	17.01	**0.1**
DARTS- (Chu et al. 2020a)	3.3	17.51	0.4
ADRATS (Xue and Qin 2023)	2.9	17.06	0.2
OLES (Jiang et al. 2024)	3.4	17.30	0.4
MOEA (Xue, Chen, and Słowik 2023)	5.8	18.97	5.2
EOFGA (Yuan, Xue, and Zhang 2023)	2.18	17.23	0.94
EAEPSO (Yuan et al. 2023)	2.94	16.94	2.2
few-shot DARTS (Zhao et al. 2021)	3.4	18.59	1.3
few-shot SNAS (Zhao et al. 2021)	3.3	18.39	1.3
HEP-NAS	3.6	**16.83**	1.6

Table 2: Comparison results of HEP-NAS with state-of-the-art methods on CIFAR-100.

Architecture	Params (M)	Top-1 (%)	Cost (G·d)
ShuffleNetV2 (Ma et al. 2018)	5	25.1	-
MobileNetV3 (Howard et al. 2019)	7.4	23.4	-
SNAS (Xie et al. 2018)	4.3	27.3	1.5
ProxylessNAS†(Cai, Zhu, and Han 2018)	-	24.9	8.3
GDAS (Dong and Yang 2019)	5.3	26.0	**0.3**
P-DARTS (Chen et al. 2019)	-	24.4	**0.3**
PC-DARTS† (Xu et al. 2019)	5.3	24.2	3.8
DARTS-† (Chu et al. 2020a)	4.9	23.8	4.5
DARTS-PT (Wang et al. 2021)	4.6	25.5	0.8
OLES (Jiang et al. 2024)	4.7	24.5	0.4
EAEPSO (Yuan et al. 2023)	4.9	26.9	4.0
MOEA (Xue, Chen, and Słowik 2023)	4.7	26.4	2.6
EOFGA† (Yuan, Xue, and Zhang 2023)	5.7	24.4	8.0
MixPath-A† (Chu et al. 2023)	5.0	**23.1**	10.3
few-shot Proxyless†(Zhao et al. 2021)	4.9	24.1	11.7
GM DARTS(2nd) (Hu et al. 2022)	5.1	24.5	2.7
GM Proxyless† (Hu et al. 2022)	4.9	23.4	24.9
HEP-NAS	5.1	**23.6**	1.5

Table 3: Comparison results of HEP-NAS with state-of-the-art methods on ImageNet. † means searching directly on ImageNet, otherwise on CIFAR-10.

Results on DARTS Search Space

Settings DARTS search space is widely used for gradient-based NAS algorithm, which contains 8 candidate operations. All the corresponding hyper-parameter settings are kept the same as DARTS. We train the supernet for 45 epochs with batch size 128 and split the supernet at 15, 25, 35, and 45 epochs respectively. we set $warmup\_epo$ to 5 initially and decrease it sequentially when splitting the next hierarchy, as the sub-supernets have already been sufficiently trained. We report the params, top-1 error rate and search cost (GPU-Days, referred to G·d) for each architecture.

Results Results on CIFAR-10 are presented in Table 1. HEP-NAS outperforms few-shot counterparts and other prevailing algorithms with slight additional time overhead. This is achieved by splitting all edges via hierarchy-wise partition strategy to minimize co-adaptation, thereby enhancing evaluation accuracy. Although ProxylessNAS (Cai, Zhu, and Han 2018) can obtain less error rate, it suffers from a larger parameter size and significantly longer search time. Note that HEP-NAS selects the most promising sub-supernet after each segmentation, therefore additional epochs are required to accurately evaluate performance during search, resulting in a higher cost than original few-shot methods. However, HEP-NAS only keeps one sub-supernet after reduction, allowing direct selection and retraining of the best architecture, while few-shot DARTS and GM DARTS retain all generated sub-supernets and use Successive Halving to progressively discard poor architectures, which requires much more time (almost 40 GPU hours, while HEP-NAS only needs 16) to derive the final well-trained network. Results on CIFAR-100 are presented in Table 2. HEP-NAS

Architecture	CIFAR-10		CIFAR-100		ImageNet16-120	
	Valid	Test	Valid	Test	Valid	Test
REA (Real et al. 2019)	91.25±0.31	94.02±0.31	72.28±0.95	72.23±0.84	45.71±0.77	45.77±0.80
ENAS (Pham et al. 2018)	39.77±0.00	54.30±0.00	10.23±0.12	10.62±0.27	16.43±0.00	16.32±0.00
DARTS (Liu, Simonyan, and Yang 2018)	39.77±0.00	54.30±0.00	38.57±0.00	38.97±0.00	18.87±0.00	18.41±0.00
PC-DARTS (Xu et al. 2019)	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
SPOS (Guo et al. 2020)	88.40±1.07	92.24±1.16	67.8±2.00	68.0±2.25	39.28±3.00	40.28±3.00
RSPS (Li and Talwalkar 2020)	84.16±1.69	87.66±2.69	45.78±6.33	46.60±6.57	31.09±5.65	30.78±6.12
CyDAS (Yu et al. 2022)	**91.12±0.44**	94.02±0.31	72.12±1.23	71.92±1.30	40.09±0.61	45.51±0.72
RD-NAS (Dong et al. 2023)	90.44±0.27	93.36±0.04	70.96±2.12	71.83±1.33	43.81±0.09	44.46±1.58
DDPNAS (Zheng et al. 2023)	90.12±0.05	93.56±0.02	70.78±0.12	70.91±0.07	44.89±0.29	46.13±0.46
EG-NAS (Cai et al. 2024)	90.12±0.05	93.56±0.02	70.78±0.12	70.91±0.07	44.89±0.29	46.13±0.46
OLES (Jiang et al. 2024)	90.88±0.10	93.70±0.15	70.56±0.28	70.40±0.22	44.17±0.49	43.97±0.38
few-shot DARTS (Zhao et al. 2021)	84.70±0.44	88.55±0.02	70.17±2.66	70.16±2.87	31.16±3.93	30.09±4.43
GM DARTS (Hu et al. 2022)	91.03±0.24	93.72±0.12	71.61±0.62	71.83±0.97	42.19±0.00	42.60±0.00
HEP-NAS	91.07±0.39	**93.86±0.21**	**73.47±0.66**	**73.48±0.55**	**46.28±0.33**	**46.51±0.35**
Optimal	91.61	94.37	73.49	73.51	46.73	47.31

Table 4: Accuracy rates with standard deviation for HEP-NAS on the NAS-Bench-201 search space.

still achieves remarkable performance and surpasses few-shot counterparts in a large margin. Results on ImageNet are presented in Table 3. We directly transfer the architecture searched on CIFAR-10 to the ImageNet dataset to verify its performance. HEP-NAS surpasses most prevailing NAS algorithms, achieving a top-1 test error rate of 23.6%, which is state-of-the-art when searching architecture directly on CIFAR-10 to save time expenditure, confirming its effectiveness and excellent generalization capability in more complex datasets (see Fig. 2).

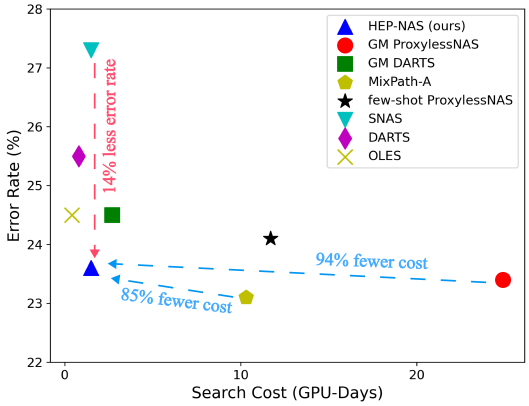

Figure 2: Performance comparation of HEP-NAS with various NAS methods on ImageNet.

Fig. 3 illustrates the distribution of accuracy rates of sub-supernets and remaining search space size after each segmentation stage. The figure clearly shows a consistent improvement in the accuracy of sub-supernets as the partitioning advances, since we progressively narrow down the search space and employ SMD to enhance the training process, effectively reducing performance degradation. Consequently, this approach provides a more precise evaluation of performance.

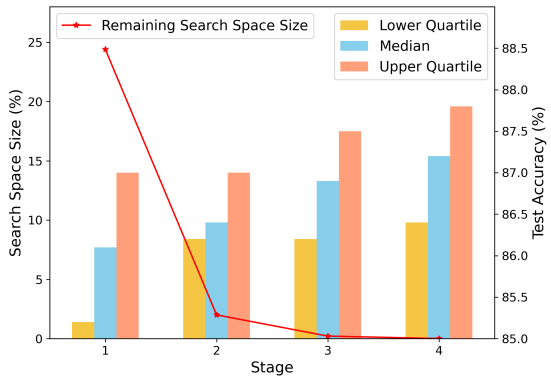

Figure 3: Remaining search space size and accuracy distribution of sub-supernets after each segmentation stage.

Results on NAS-Bench-201 Search Space

Settings NAS-Bench-201 (Dong and Yang 2020) is another widely used NAS benchmark to analyze the performance of different NAS algorithms, which provides the performance of 15,625 architectures. It is a cell-like search space, including 4 nodes and 5 candidate operations. We only search architectures on CIFAR-10 and CIFAR-100 , then transfer to ImageNet16-120. The $warmup\_epo$ is set to 10. We report the mean accuracy rate with standard deviation for each architecture.

Results The performance of HEP-NAS on NAS-Bench-201 is summarized in Table 4. Compared to few-shot counterparts and other prevailing algorithms, HEP-NAS achieves remarkable performance in most cases by further mitigating co-adaptation to enhance performance evaluation accuracy. Particularly, it achieves an average test accuracy of 46.51%, ranking highest in the challenging ImageNet16-120 dataset, demonstrating the superiority and generalization capability of HEP-NAS. We also search directly on ImageNet16-120 and obtain 46.34% accuracy, which is still optimal among counterparts. In addition, HEP-NAS also achieves the highest test accuracy on CIFAR-10 and CIFAR-100, resulting in a 5.31% improvement over few-shot DARTS on CIFAR-10, and 3.32% on CIFAR-100 respectively.

We also evaluate the ranking correlation with Spearman correlation, a particularly important measure to quantify the degree of co-adaption, among architectures in the final reduced search space. We obtain 0.665 Spearman correlation, proving that HEP-NAS can estimate the performance of architectures accurately.

Ablation Study

Effectiveness of Hierarchy-wise Partition Narrowing the search space offers both increased efficiency and risk of converging to a local optimum. By selectively choosing the optimal sub-supernet for continued exploration after splitting each edge, the number of resulting sub-supernets can be reduced further. However, as previously mentioned, the primary goal of the search process is more relevant to identify the best combination of operations and determine the most suitable predecessor nodes for each intermediate node. Therefore, considering the connections between nodes is essential. The hierarchy-wise partition strategy accounts for this combinatorial relationship, reducing the likelihood of being trapped in a local optimum and ensuring the exploration of the best overall architecture as far as possible. The experimental results presented in Table 5 confirm the efficacy of hierarchical segmentation.

Method	Error Rate (%)		Search Cost
	CIFAR-10	CIFAR-100	(GPU-Days)
edge-wise	2.86	17.7	0.36
hierarchy-wise	2.51	17.08	0.8

Table 5: Ablation study for hierarchy-wise and edge-wise splitting strategy used in conjunction with shrinking search space, using batch size 256.

Order of Hierarchical Segmentation HEP-NAS employs a hierarchical partitioning strategy, where each successive hierarchy discretizes a larger number of operations than the previous one. This progressive discretization approach is designed to enhance the stability of the training process in the initial phases and accelerate the discovery of the optimal search space in the later stages of training. Therefore, the sequence of hierarchical segmentation is crucial, as premature perturbation of a significant number of edges can destabilize the architectural training, resulting in biased outcomes. To assess the impact of the splitting order, we conducted experiments using reverse order and random order with a batch size of 256. The results are detailed in Table 6.

Split Order	Error Rate (%)		Search Cost
	CIFAR-10	CIFAR-100	(GPU-Days)
random order	2.63	17.08	1.125
reverse order	2.68	17.17	1.4

Table 6: Ablation Study for splitting order. Results of random order are obtained in 6 independent runs.

Effectiveness of Search Space Mutual Distillation To further confirm the impact of SMD on the training process of sub-supernets, we evaluated four distillation methods: (A) Training sub-supernets without guidance (baseline); (B) Training current sub-supernets using only the previous optimal architecture's guidance; (C) Training current sub-supernets using only guidance from other sub-supernets in the same hierarchy; (D) Training current sub-supernets using guidance from both the previous optimal architecture and other sub-supernets in the same hierarchy, which represents our HEP-NAS algorithm. Experiment results are summarized in Table 7. Method-D boosts the convergence and stabilizes the training process, thereby achieved the lowest test error rate among these methods, showcasing the efficacy of SMD.

Guidance		Error Rate (%)	
previous	other sub-supernets	CIFAR-10	CIFAR-100
-	-	2.51	17.10
✓	-	2.46	17.10
-	✓	2.44	17.07
✓	✓	2.44	16.83

Table 7: Ablation study for different distillation method. 'previous' refers to the optimal architecture found in previous hierarchy, and 'other sub-supernets' refers to candidate sub-supernets in current splitting hierarchy.

Conclusion

In this paper, we introduce HEP-NAS to search out architectures with higher accuracy. The core concept involves utilizing a hierarchy-wise strategy to partition all edges and progressively narrowing the search space to identify the optimal sub-supernet. To enhance performance evaluation, we concurrently transfer knowledge from the optimal sub-supernet identified in the previous hierarchy and other candidate sub-supernets in the current hierarchy to stabilize the training process and improve convergence of each individual sub-supernet. Extensive experiments across diverse datasets and search spaces demonstrate that HEP-NAS outperforms previous state-of-the-art NAS algorithms in most scenarios.

Acknowledgments

This work is supported by the National Nature Science Foundation of China [Grant Nos. 62173258, T2495254].

References

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *International conference on machine learning*, 550–559. PMLR.

Cai, H.; Zhu, L.; and Han, S. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

Cai, Z.; Chen, L.; Liu, P.; Ling, T.; and Lai, Y. 2024. EG-NAS: Neural Architecture Search with Fast Evolutionary Exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 11159–11167.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1294–1303.

Chen, Z.; Qiu, G.; Li, P.; Zhu, L.; Yang, X.; and Sheng, B. 2023. Mngnas: distilling adaptive combination of multiple searched networks for one-shot neural architecture search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chu, X.; Lu, S.; Li, X.; and Zhang, B. 2023. Mixpath: A unified approach for one-shot neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5972–5981.

Chu, X.; Wang, X.; Zhang, B.; Lu, S.; Wei, X.; and Yan, J. 2020a. Darts-: robustly stepping out of performance collapse without indicators. *arXiv preprint arXiv:2009.01027*.

Chu, X.; Zhou, T.; Zhang, B.; and Li, J. 2020b. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European conference on computer vision*, 465–480. Springer.

Dong, P.; Niu, X.; Li, L.; Tian, Z.; Wang, X.; Wei, Z.; Pan, H.; and Li, D. 2023. RD-NAS: Enhancing one-shot supernet ranking ability via ranking distillation from zero-cost proxies. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.

Dong, X.; and Yang, Y. 2019. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1761–1770.

Dong, X.; and Yang, Y. 2020. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single path one-shot neural architecture search with uniform sampling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, 544–560. Springer.

He, H.; Liu, L.; Zhang, H.; and Zheng, N. 2024. IS-DARTS: Stabilizing DARTS through Precise Measurement on Candidate Importance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 12367–12375.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1314–1324.

Hu, S.; Wang, R.; Hong, L.; Li, Z.; Hsieh, C.-J.; and Feng, J. 2022. Generalizing few-shot nas with gradient matching. *arXiv preprint arXiv:2203.15207*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Jiang, S.; Ji, Z.; Zhu, G.; Yuan, C.; and Huang, Y. 2024. Operation-level early stopping for robustifying differentiable NAS. *Advances in Neural Information Processing Systems*, 36.

Jiang, T.; Wang, H.; and Bie, R. 2024. MeCo: zero-shot NAS with one data and single forward pass via minimum eigenvalue of correlation. *Advances in Neural Information Processing Systems*, 36.

Li, L.; and Talwalkar, A. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, 367–377. PMLR.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Ly-Manson, T.; Leonardon, M.; El Bey, A. A.; Hacene, G. B.; and Mauch, L. 2024. Analyzing Few-Shot Neural Architecture Search in a Metric-Driven Framework.

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, 116–131.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, 4095–4104. PMLR.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, 4780–4789.

Wan, X.; Ru, B.; Esperança, P. M.; and Li, Z. 2022. On redundancy and diversity in cell-based neural architecture search. *arXiv preprint arXiv:2203.08887*.

Wang, R.; Cheng, M.; Chen, X.; Tang, X.; and Hsieh, C.-J. 2021. Rethinking architecture selection in differentiable NAS. *arXiv preprint arXiv:2108.04392*.

Wang, W.; Zhang, X.; Cui, H.; Yin, H.; and Zhang, Y. 2023. FP-DARTS: Fast parallel differentiable neural architecture

search for image classification. *Pattern Recognition*, 136: 109193.

Wu, F.; Gao, J.; Hong, L.; Wang, X.; Zhou, C.; and Ye, N. 2024. G-NAS: Generalizable Neural Architecture Search for Single Domain Generalization Object Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 5958–5966.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*.

Xue, Y.; Chen, C.; and Słowik, A. 2023. Neural Architecture Search Based on a Multi-Objective Evolutionary Algorithm With Probability Stack. *IEEE Transactions on Evolutionary Computation*, 27(4): 778–786.

Xue, Y.; and Qin, J. 2023. Partial Connection Based on Channel Attention for Differentiable Neural Architecture Search. *IEEE Transactions on Industrial Informatics*, 19(5): 6804–6813.

Yu, H.; Peng, H.; Huang, Y.; Fu, J.; Du, H.; Wang, L.; and Ling, H. 2022. Cyclic differentiable architecture search. *IEEE transactions on pattern analysis and machine intelligence*, 45(1): 211–228.

Yuan, G.; Wang, B.; Xue, B.; and Zhang, M. 2023. Particle swarm optimization for efficiently evolving deep convolutional neural networks using an autoencoder-based encoding strategy. *IEEE Transactions on Evolutionary Computation*.

Yuan, G.; Xue, B.; and Zhang, M. 2023. An effective one-shot neural architecture search method with supernet fine-tuning for image classification. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 615–623.

Zhang, B.; Wang, X.; Qin, X.; and Yan, J. 2024. Boosting Order-Preserving and Transferability for Neural Architecture Search: a Joint Architecture Refined Search and Fine-tuning Approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5662–5671.

Zhang, M.; Li, H.; Pan, S.; Chang, X.; and Su, S. 2020. Overcoming multi-model forgetting in one-shot NAS with diversity maximization. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 7809–7818.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6848–6856.

Zhao, Y.; Wang, L.; Tian, Y.; Fonseca, R.; and Guo, T. 2021. Few-shot neural architecture search. In *International Conference on Machine Learning*, 12707–12718. PMLR.

Zheng, X.; Yang, C.; Zhang, S.; Wang, Y.; Zhang, B.; Wu, Y.; Wu, Y.; Shao, L.; and Ji, R. 2023. Ddpnas: Efficient neural architecture search via dynamic distribution pruning. *International Journal of Computer Vision*, 131(5): 1234–1249.

Zoph, B.; and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.