NAS-LID: Efficient Neural Architecture Search with Local Intrinsic Dimension

Xin He[1,4], **Jiangchao Yao**[2,3], **Yuxin Wang**[1], **Zhenheng Tang**[1], **Ka Chun Cheung**[1,4], **Simon See**[2,4,6,7], **Bo Han**[1], **Xiaowen Chu** [5,1*]

[1] Hong Kong Baptist University
[2] Shanghai Jiao Tong University
[3] Shanghai AI Laboratory
[4] NVIDIA AI Tech Center
[5] The Hong Kong University of Science and Technology (Guangzhou)
[6] Mahindra University
[7] Coventry University

Abstract

One-shot neural architecture search (NAS) substantially improves the search efficiency by training one supernet to estimate the performance of every possible child architecture (i.e., subnet). However, the inconsistency of characteristics among subnets incurs serious interference in the optimization, resulting in poor performance ranking correlation of subnets. Subsequent explorations decompose supernet weights via a particular criterion, e.g., gradient matching, to reduce the interference; yet they suffer from huge computational cost and low space separability. In this work, we propose a lightweight and effective local intrinsic dimension (LID)-based method *NAS-LID*. NAS-LID evaluates the geometrical properties of architectures by calculating the *low-cost* LID features layer-by-layer, and the similarity characterized by LID enjoys *better separability* compared with gradients, which thus effectively reduces the interference among subnets. Extensive experiments on NASBench-201 indicate that NAS-LID achieves superior performance with better efficiency. Specifically, compared to the gradient-driven method, NAS-LID can save up to 86% of GPU memory overhead when searching on NASBench-201. We also demonstrate the effectiveness of NAS-LID on ProxylessNAS and OFA spaces. Source code: https://github.com/marsggbo/NAS-LID.

Introduction

Neural architecture search (NAS) (Elsken, Metzen, and Hutter 2019; He, Zhao, and Chu 2021) has been widely used to discover models automatically in various tasks (He et al. 2021; Liu et al. 2019; Ying et al. 2022; He et al. 2022). Vanilla NAS (Zoph and Le 2017; Real et al. 2019) trains and evaluates each architecture separately, which obtains the true performance of all searched architectures at the cost of substantial computations. One-shot NAS (Pham et al. 2018; Liu, Simonyan, and Yang 2019) drastically reduces the cost by training only one supernet as an estimator of the performance of all subnets in the search space. However, the subnets with inconsistent characteristics interfere with each other during training in the shared supernet, resulting in inaccurate estimation (Bender et al. 2018).

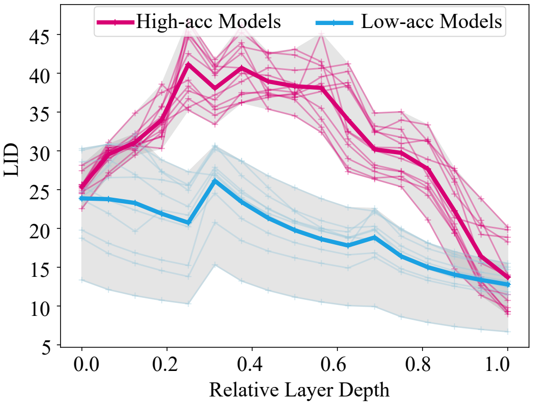

Figure 1: Layer-wise LID of sampled architectures in NASBench-201. Each curve indicates a model. The LID profiles of high-accuracy (>92%) models are arch-shaped (i.e., first increasing and then decreasing), while those of low-accuracy (<86%) models show an approximately monotonically decreasing trend.

Recent works to reduce the interference among subnets can be categorized into two groups. One is to treat all subnets as student networks and use a well-designed teacher network to force the similarity of their layer-wise outputs (Peng et al. 2021; Xu et al. 2022b; Bashivan, Tensen, and DiCarlo 2019). However, Liu et al. (2020) empirically showed that the optimal student would be different under different teachers even trained on the same task and dataset. In other words, choosing a different teacher will lead to quite different results for the performance ranking of subnets. Instead, few-shot NAS (Zhao et al. 2021; Hu et al. 2022) aggregates the subnets with aligned inputs and outputs into the same sub-supernet by splitting the supernet into multiple sub-supernets. For example, GM-NAS (Hu et al. 2022) splits the supernet via the gradient similarity between sub-supernets, which achieves state-of-the-art (SOTA) performance. However, the gradients are high-dimensional and sparse data that inevitably encounter the curse of dimensionality on space separability. Our experiments in Sec. show that the gradient similarities among sub-supernets are very close, leading

[*] Corresponding author: xwchu@ust.hk

to low separability. Besides, GM-NAS requires the forward and backward processes for each sub-supernet before similarity calculation, incurring huge GPU memory occupation and high computational complexity.

To address the issue, we explore a different criterion, namely local intrinsic dimension (LID) (Houle 2017), which measures the minimal number of parameters to describe the data representation learned from the model. We randomly sample multiple models from NASBench-201 (Dong and Yang 2020) and train them from scratch to obtain accurate LID estimations of all layers. Fig. 1 presents LIDs against relative depth, where the LID is orders of magnitude smaller than the number of parameters per layer. Notably, the models with higher accuracy have arch-shaped LID profiles, i.e., LID first increases and then decreases, which has been similarly observed in (Ansuini et al. 2019). On the other hand, we show that the LID profiles of the models with lower accuracy are monotonically decreasing, except for a slight increase at relative depths of 0.3 and 0.7. If the LIDs of two data are close, it means that they are close in the low-dimensional manifold, as well as in high-dimensional space. Models with similar LID profiles tend to have similar layer-wise outputs. The observation implies that LID profiles can capture the geometrical properties of architectures and determine the intrinsic training dynamics.

Thus, we propose a new method *NAS-LID*, which leverages LID to characterize the similarity among architectures for NAS. Each time we compare all unpartitioned layers to select the one with the highest partition score based on LID (see Sec.). Then, we partition the candidate operations on the selected layer into two sub-supernets. Iterating this partition allows us to get smaller sub-supernets. Thanks to the advantages of LID, we both effectively avoid the curse of dimensionality and reduce the computational cost in this procedure. In a nutshell, the contribution of this work can be summarized as follows.

1. We are the first to provide a proof-of-concept for the potential application of LID in splitting one-shot supernet, and discover that the LID-based characterization exhibits better space separability and higher performance ranking scores than the gradient counterpart.
2. Compared to the gradient-based split scheme, we propose a novel method NAS-LID, which can significantly reduce the GPU memory overhead (saving up to 86% of GPU memory overhead on NASBench-201) and guarantee the superior performance.
3. We demonstrate the effectiveness of NAS-LID by conducting extensive experiments on multiple search spaces (NASBench-201, ProxylessNAS, and OFA) and datasets (CIFAR10, CIFAR100, and ImageNet).

Related Work

Neural Architecture Search

Vanilla NAS (Zoph and Le 2017; Zoph et al. 2018; Real et al. 2019) requires vast computational resources to find the best architecture by training all sampled architectures and comparing their performance. One-shot NAS (Pham et al. 2018; Liu, Simonyan, and Yang 2019; Cai, Zhu, and Han 2019) uses a supernet to represent the search space, where each possible architecture is a subnet, reducing the need to train all subnets. However, subnets' weights are coupled, causing interference during training and resulting in inaccurate performance predictions.

To reduce interference, one approach aligns the subnets' outputs with those of a teacher network (Bashivan, Tensen, and DiCarlo 2019; Peng et al. 2021; Xu et al. 2022b) using knowledge distillation (Hinton et al. 2015). For example, Xu et al. (2022b) selected a top-performing subnet as the teacher network to align other subnets, but this approach may introduce innate biases, leading to unfair assessments of model performance. Another approach is to split the supernet into sub-supernets based on similar architecture characteristics (Zhao et al. 2021; Hu et al. 2022; Su et al. 2021). However, exhaustively dividing the supernet into hundreds of sub-supernets can lead to high resource consumption. GM-NAS (Hu et al. 2022) used gradient information at the shared part of sub-supernets to partition the supernet, but it suffers from the curse of dimensionality due to the high-dimensional and sparse gradients.

Intrinsic Dimension

Intrinsic dimension (ID) represents the necessary degrees of freedom of a high-dimensional data space with minimal information loss. Researchers have designed ID estimators (Campadelli et al. 2015; Houle 2017) for different tasks to analyze complex and sparse high-dimensional data in deep neural networks. For example, Xu et al. (2022a) use ID for adversarial detection to tackle heterogeneous label noise, while Li et al. (2018) leverage ID to estimate objective landscapes and approximate the model's training in a random subspace. Recently, Aghajanyan, Gupta, and Zettlemoyer (2021) reduce parameter sizes for fine-tuning large language models using ID. In this work, we use LID (Houle 2017) to estimate ID, as it avoids expensive matrix operations by requiring only an ordered list of neighbor distances. We are the first to introduce LID for NAS as a promising criterion to achieve efficient NAS.

Methods

In this work, we propose a lightweight and effective LID-based method, *NAS-LID*, to show that LID can facilitate the separability of search space and thus improve the performance ranking correlation of subnets. In this section, we first introduce the LID and its estimation method. Then, we describe our proposed NAS-LID in two parts: 1) why and how we use LID for architecture characterization, and 2) how we split the supernet by LID similarity.

Preliminary of Local Intrinsic Dimension

LID (Houle 2017) measures the ID of data representations in deep networks, without accessing the underlying global data distribution, which can be defined as follows.

Definition 1 (Local Intrinsic Dimension (Houle 2017)). *Given a random variable* $\mathbf{R}$*, denoting the distance from a*

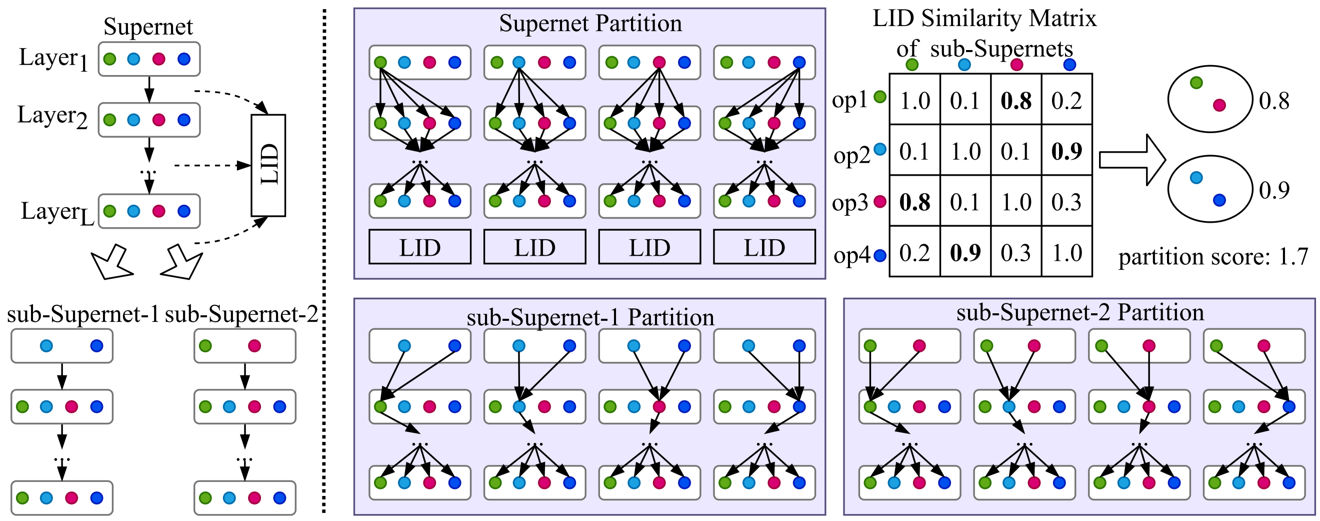

Figure 2: An illustration of the supernet partition method via local intrinsic dimension (LID). We compare all unpartitioned layers and only select the layer with the highest partition score to split the supernet. Taking the first layer as an example, we can split the operations into four different sub-supernets, each characterized by layer-wise LID. We calculate the similarity between LIDs of these sub-supernets and divide the operations into two groups by maximizing the sum of intra-similarity of two groups (i.e., partition score). Finally, we merge the operations of the same group to form two sub-supernets. We can obtain more sub-supernets by iterating the above steps on generated sub-supernets.

reference sample to other samples. For any distance threshold r such that the cumulative distance function $F_{\mathbf{R}}(r)$ is positive and continuously differentiable at r, then the LID of the reference sample at distance r is given by

$$\begin{aligned}\mathrm{LID}_{\mathbf{R}}(r) &\triangleq \lim_{\epsilon\to 0^+} \frac{\ln(F_{\mathbf{R}}((1+\epsilon)\cdot r)/F_{\mathbf{R}}(r))}{\ln((1+\epsilon)\cdot r/r)} \\ &= \frac{r\cdot F'_{\mathbf{R}}(r)}{F_{\mathbf{R}}(r)}\end{aligned} \tag{1}$$

The last line of Eq. (1) gives a low-dimensional computation of the sub-manifold in the limit. A popular empirical estimation from (Amsaleg et al. 2015) is maximum likelihood estimator (MLE). Specifically, given a reference data sample $x \sim \mathbf{X}$, where $\mathbf{X}$ represents the data distribution, the MLE estimator of the LID at x is defined as follows:

$$\widehat{\mathrm{LID}}(x) = -\left(\frac{1}{k}\sum_{i=1}^{k} \ln \frac{r_i(x)}{r_k(x)}\right)^{-1} \tag{2}$$

where $r_i(x)$ indicates the distance between x and its i-th nearest neighbor within samples drawn from $\mathbf{X}$, i.e., $r_k(x)$ is the maximum distance from x among its k nearest neighbors. In our work, we use Euclidean distance and set $k = 20$.

The LID-Based Characterization

Why Do We Use the LID-Based Characterization? In terms of space separability, model gradients used in previous SOTA GM-NAS are in a high-dimensional and sparse space. This is difficult to distinguish the difference between different gradients based on the Euclidean or cosine distance measure, making the partition inaccurate. In contrast, LID measures the dimensionality of the lower dimensional sub-manifold in which the high-dimensional data resides. As Fig. 1 shows, the high-dimensional outputs from each layer of the models in NASBench-201 fall in sub-manifolds with dimensions less than 50. Our experiments in Sec. show that our characterization can achieve much higher space separability than the gradient counterpart.

In terms of information bottleneck (Tishby, Pereira, and Bialek 2000; Shwartz-Ziv and Tishby 2017), layer-wise LID profiles capture models' geometrical properties (Lei et al. 2020) and describe how information changes layer by layer. For arch-shaped LID profiles (red lines in Fig. 1), the initial increase of LID indicates that models are learning data by continuously mapping the input data to higher-dimensional manifolds, and the decreasing part is to prune features irrelevant for prediction. On the other hand, the monotonically decreasing LID profiles (blue lines in Fig. 1) indicate that these models fail to extract informative features at the initial layers, resulting in worse performance.

Architecture LID Characterization Consider an architecture with L layers, and let $X = [x_1, x_2, ..., x_L]$ be the set of the feature representations of all layers, where $x_i \in \mathbb{R}^{b\times m_i}$, b is the number of data (i.e., batch size), and m_i is the size of the output representations in i-th layer. The LID of i-th layer representations (i.e., x_i) is calculated as follows

$$\mathrm{LID}(x_i) = \frac{1}{b}\sum_{j=0}^{b-1} \widehat{\mathrm{LID}}(x_i[j,:]) \tag{3}$$

where $x_i[j,:]$ is the j-th data sample of x_i. The architecture LID characterization is computed by stacking LIDs of all layers, namely $[\mathrm{LID}_1, ..., \mathrm{LID}_L]^T$, where $\mathrm{LID}_i = \mathrm{LID}(x_i)$.

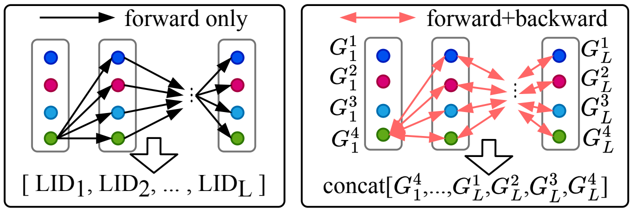

Figure 3: Overview of the L-layer sub-supernet characterization in our NAS-LID (left) and GM-NAS (right). NAS-LID is more efficient as it requires only the forward process and L parameters for characterization, while GM-NAS involves forward and backward processes and characterizes the sub-supernet by gradients of all selected operations.

Sub-supernet LID Characterization Although we can directly separate the supernet by merging architectures (i.e., subnets) with high LID similarity, a standard NAS search space usually contains millions or even more subnets; thus, it is too expensive to compute LIDs of all subnets. For example, suppose a supernet $\mathcal{A}$ has L searchable layers, each with n candidate operations; it contains 1,048,576 subnets even when $n = 4, L = 10$. To improve efficiency, we split the supernet into a collection of sub-supernets. As Fig. 2 shows, we can split the supernet by dividing the operations of one layer into different sub-supernets. In this way, the number of sub-supernets can be much less than the number of architectures, reducing the computational overhead. The LID characterization of the sub-supernet is similar to that of a single architecture. Without loss of generality, let's consider the LID of the i-th layer and denote by $\{x_i^{O_1}, x_i^{O_2}, ...x_i^{O_n}\}$ the output features of n candidate operations in i-th layer, by $I_i = \{I_i^{O_1}, I_i^{O_2}, ..., I_i^{O_n}\}, I_i^{O_j} \in \{0, 1\}$ the binary vector of candidate operations. Similar to Inception (Szegedy et al. 2015), the output of i-th layer in the sub-supernet is the sum of all selected operations' outputs, i.e., $z_i = \sum_{j=1}^{n} x_i^{O_j} \cdot I_i^{O_j}$; thus, the LIDs $\mathcal{L}$ of the sub-supernet $\mathcal{A}$ is defined as

$$\mathcal{L}_{\mathcal{A}} = [\mathrm{LID}(z_1), \mathrm{LID}(z_2), ..., \mathrm{LID}(z_L)]^T \tag{4}$$

Complexity: Gradient vs. LID Fig. 3 presents the difference in sub-supernet characterization between LID-based and gradient-based methods. Suppose we split a L-layer supernet into n sub-supernets. NAS-LID only requires $n \times L$ parameters to characterize these sub-supernets, while GM-NAS requires approximately $n \times M$ parameters, where M is the size of the supernet and $M \gg L$. In terms of GPU memory cost, NAS-LID involves only the forward process to compute layer-wise LID, while GM-NAS requires both forward and backward processes, and the overhead of backward is empirically twice that of the forward. Table 1 compares the GPU memory overhead consumed by GM-NAS and NAS-LID on a single V100 GPU (32GB). We can see that NAS-LID is more efficient as it can save up to 86% GPU memory on NASBench-201 with input size of $32 \times 3 \times 224 \times 224$. Thus, NAS-LID enables larger batch size for special needs of different computing tasks and scenarios (Yao et al. 2022).

Input size	Method	NASBench-201	ProxylessNAS
128×3	GM-NAS	3,303	2,247
$\times 32 \times 32$	NAS-LID	**1,539** ($\downarrow$ 53%)	**1,571** ($\downarrow$ 30%)
$32 \times 3 \times$	GM-NAS	24,073	6,707
224×224	NAS-LID	**3,413** ($\downarrow$ 86%)	**2,015** ($\downarrow$ 70%)

Table 1: Comparison of the GPU memory (MB) costs between GM-NAS and our proposed NAS-LID across different search spaces and input sizes.

LID-Based Supernet Partition

In this sub-section, we introduce how to split the supernet via LID. For simplicity, we use an example of a one-round supernet partition in Fig. 2 to describe this procedure. Actually, NAS-LID consists of $T \geq 1$ rounds of partition, where the first round splits only the supernet, and the subsequent rounds split the sub-supernets generated in the previous rounds. After T rounds of partition, we will get 2^T sub-supernets, each covering a different and non-overlapping region of the original search space. Alg. 1 details our LID-based supernet partition scheme. Without loss of generality, we introduce the partition on a sub-supernet in the t-th round. We denote by $\mathcal{A} = \{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_L\}$ a sub-supernet, by $\mathcal{L}_{\mathcal{A}}$ the LID characterizations, where $\mathcal{O}_l = \{O_1, ..., O_n\}$ is the set of n candidate operations of the l-th layer and associated with a binary vector I_l recording which operations have been deactivated (encoded by 0).

Sub-supernet Similarity As Fig. 2 shows, the supernet can be divided into n sub-supernets $\{\mathcal{A}_{O_1}, ..., \mathcal{A}_{O_n}\}$ by splitting the operations ($\mathcal{O}_l$) on the l-th layer; thus, the supernet partition is equivalent to operations partition. We use Eq. 4 to get LIDs of these sub-supernets, i.e., $\{\mathcal{L}_{\mathcal{A}_{O_1}}, ..., \mathcal{L}_{\mathcal{A}_{O_n}}\}$, based on which we can obtain the $n \times n$ symmetric similarity matrix. via Eq. 5. The similarity between two sub-supernets is defined as the reciprocal of the Euclidean distance of their LIDs.

$$S(\mathcal{A}_{O_i}, \mathcal{A}_{O_j}) = \frac{1}{\|\mathcal{L}_{\mathcal{A}_{O_i}}, \mathcal{L}_{\mathcal{A}_{O_j}}\|_2 + \epsilon} \tag{5}$$

where $\epsilon = 10^{-6}$ avoids the denominator being 0. We have also explored Pearson distance, but our results in Sec. show that Pearson distance is not suited for LID because the scale of LIDs is vital for characterization.

Sub-Supernet Merging Based on the similarity matrix, we can merge sub-supernets with high similarity by merging operations via the graph min-cut algorithm (Boykov and Jolly 2001), which aims to maximize the partition score γ, i.e., the sum of intra-similarity of each group, as below:

$$\gamma = \max_{\Gamma \subseteq \mathcal{O}} [\sum_{O, O' \in \Gamma} S(\mathcal{A}_O, \mathcal{A}_{O'}) + \sum_{O, O' \in \mathcal{O} \setminus \Gamma} S(\mathcal{A}_O, \mathcal{A}_{O'})]$$
$$\text{s.t. } \lfloor n/2 \rfloor \leq |\Gamma| \leq \lceil n/2 \rceil \tag{6}$$

Algorithm 1: NAS-LID: LID-based Supernet Partition

```
Require: $\mathcal{A}^0$: supernet, $T$: rounds of partition
Ensure: $\Omega$: the set of partitioned sub-supernets
procedure MAIN($\mathcal{A}^0, T$)
    $\Omega = \{\mathcal{A}^0\}$
    for $t \in [1, T]$ do
        $\Omega^t = \{\}$
        for $\mathcal{A} \in \Omega$ do
            Warmup: training $\mathcal{A}$ as a one-shot supernet
            $\Omega^t$.insert(SPLIT-SUPERNET($\mathcal{A}$))
        $\Omega = \Omega^t$
    $\Omega \leftarrow$ Finetuning all sub-supernets in $\Omega$
    return $\Omega$
procedure SPLIT-SUPERNET($\mathcal{A}$)
    $\mathcal{I} = \{I_1, ..., I_L\}$   ▷ binary vector of each layer in $\mathcal{A}$
    for unsplitted $l$-th layer do   ▷ $sum(I_l) == |I_l|$
        $\mathcal{O}_l = \{O_1, ..., O_n\}$   ▷ candidate operations
        $\{\mathcal{A}_{O_1}, ..., \mathcal{A}_{O_n}\} \leftarrow$ split $\mathcal{A}$ on the $l$-th layer
        $\{\mathcal{L}_{\mathcal{A}_{O_1}}, ..., \mathcal{L}_{\mathcal{A}_{O_n}}\} \leftarrow$ get LIDs via Eq. 4
        $\gamma \leftarrow$ partition score via Eq. 5∼6
    split $\mathcal{A}$ on the layer with the best $\gamma^*$
    $\Gamma^* \leftarrow$ best partitioned operation group
    return $\{\mathcal{A}_{\Gamma^*}, \mathcal{A}_{\mathcal{O}\backslash\Gamma^*}\}$
```

As a result, the candidate operations are divided into two groups $\{\Gamma, \mathcal{O}\backslash\Gamma\}$. In other words, the supernet $\mathcal{A}$ is split into two sub-supernets, i.e., $\mathcal{A} = \{\mathcal{A}_\Gamma, \mathcal{A}_{\mathcal{O}\backslash\Gamma}\}$. After merging, we will update the binary vector I of the partitioned layer for each generated sub-supernet. Next time, we will only split the unpartitioned layers (i.e., I is an all-one vector) with the highest partition score. In this way, we can prevent the sub-supernets from being too fragmented and limit the number of sub-supernets, thus reducing the computational overhead.

Experiments

In this section, we empirically verify the effectiveness of our proposed NAS-LID. We first conduct extensive experiments on NASBench-201 to compare the LID-driven and gradients-driven split schemes. We then evaluate the performance of NAS-LID on other open domain search spaces.

NASBench-201

NASBench-201 is a public tabular architecture dataset, which builds a DARTS-like (Liu, Simonyan, and Yang 2019) search space and provides the performance of 15,625 neural architectures on the CIFAR-10 and CIFAR-100 datasets (Krizhevsky and Hinton 2009). Each architecture is stacked with multiple cells, each sharing the same structure. As shown in Fig. 4, a cell is represented as a directed acyclic graph (DAG) containing four nodes, and each edge has 5 predefined operations, i.e., None, Skip-connection, Conv-1×1, Conv-3×3, and Avgpool-3×3. Thus, we can obtain five sub-supernets by splitting the operations on one edge. For example, if we split the edge (0-3), the five operations on this edge will be divided into five different sub-supernets that still retain five candidate operations on the remaining edges. We compare with GM-NAS (Hu et al. 2022) in three aspects: separability, ranking correlation, and performance of derived architectures.

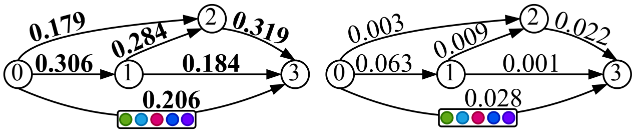

Figure 4: Comparison of separability score (the higher, the better) of each edge in NASBench-201 space between our NAS-LID (left) and GM-NAS (right).

Separability GM-NAS and our NAS-LID respectively split the supernet based on gradient and LID similarity. Here, we study how different these two criteria are to distinguish the sub-supernets, i.e., *separability*. For the quantitative comparison, we first train the supernet for 50 epochs via Random Sampling with Parameter Sharing (RSPS) (Li and Talwalkar 2020) and use the pretrained weights to calculate the separability score for each edge, formalized as follows.

$$D = \sqrt{\frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n}(s_{i,j} - \bar{s})^2}$$

$$\text{s.t., } \bar{s} = \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n} s_{i,j} \text{ and } i \neq j \tag{7}$$

where n is the number of candidate operations (i.e., 5 in NASBench-201), $s_{i,j}$ denotes the LID or gradient similarity between the two sub-supernets that only include operation i and j, respectively. $\bar{s}$ is the average similarity of all pairs of sub-supernets. Intuitively, the lower the separability score, the higher the inter-similarity between different sub-supernets, and the harder it is to split them. Fig. 4 compares the separability score of each edge of NAS-LID and GM-NAS. We can see that the separability scores of all edges obtained by NAS-LID are two orders of magnitude higher than those obtained by GM-NAS, showing NAS-LID is more confident and certain in the partition.

Ranking Correlation Reliable ranking performance is crucial to NAS algorithms. We compare the ranking performance with RSPS (Li and Talwalkar 2020), GM-NAS (Hu et al. 2022) among the top 50/100/150 architectures in NASBench-201 space. RSPS does not involve splitting but trains only one supernet for 300 epochs. Following GM-NAS, we split four edges based on LID similarity, leading to $4^2 = 16$ sub-supernets. We finetune each sub-supernet for several epochs after partition. We run each method three times and report the mean and standard deviation of correlation results in Table 2. GM-NAS and NAS-LID outperform RSPS, showing that splitting the supernet does help increase the ranking correlation among top models. We can also see that NAS-LID achieves 0.466 Kendall and 0.637 Spearman correlation, much higher than GM-NAS (0.335 Kendall and 0.484 Spearman). The lower standard deviation also indicates the stability of LID-driven characterization.

Method	Top50		Top100		Top150	
	Kendall	Spearman	Kendall	Spearman	Kendall	Spearman
SPOS (Guo et al. 2020)	0.14±0.02	0.16±0.01	0.19±0.03	0.27±0.04	0.12±0.03	0.17±0.05
GM-NAS (Hu et al. 2022)	0.34±0.09	0.48±0.11	0.21±0.04	0.29±0.07	0.23±0.03	0.31±0.06
NAS-LID (Ours)	**0.47**±0.01	**0.64**±0.03	**0.35**±0.03	**0.49**±0.04	**0.34**±0.02	**0.48**±0.03

Table 2: Ranking correlation (the higher, the better) among top50/100/150 architectures of NASBench-201. Our LID-driven splitting scheme achieves a much higher ranking correlation with lower variance.

Method	Type	CIFAR-10		CIFAR-100	
		validation	test	validation	test
ResNet (He et al. 2016)	Manual	90.83	93.97	70.42	70.86
NASNet (Zoph et al. 2018)	Vanilla	91.09 ± 0.37	93.85 ± 0.37	70.05 ± 1.67	70.17 ± 1.61
ENAS (Pham et al. 2018)	One-shot	37.51 ± 3.19	53.89 ± 0.58	13.37 ± 2.35	13.96 ± 2.33
DARTS (Liu, Simonyan, and Yang 2019)		39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00
GDAS (Dong and Yang 2019b)		90.00 ± 0.21	93.51 ± 0.13	71.14 ± 0.27	70.61 ± 0.26
DSNAS (Hu et al. 2020)		89.66 ± 0.29	93.08 ± 0.13	30.87 ± 16.40	31.01 ± 16.38
SETN (Dong and Yang 2019a)		82.25 ± 5.17	86.19 ± 4.63	58.86 ± 7.59	56.87 ± 7.77
PC-DARTS (Xu et al. 2020)		89.96 ± 0.15	93.41 ± 0.30	67.12 ± 0.39	67.48 ± 0.89
RSPS (Li and Talwalkar 2020)		84.16 ± 1.69	87.66 ± 1.69	59.00 ± 4.60	58.33 ± 4.34
FSNAS+RSPS (Zhao et al. 2021)	Few-shot	85.40 ± 1.28	89.11 ± 1.37	58.59 ± 3.45	58.69 ± 3.75
GM-NAS+RSPS (Li and Talwalkar 2020)		89.09 ± 0.40	92.70 ± 0.53	68.36 ± 0.91	68.81 ± 1.28
NAS-LID+RSPS (ours)		**89.74 ± 0.37**	**92.90 ± 0.27**	**69.38 ± 0.36**	**69.39 ± 0.77**
optimal		91.61	94.37	73.49	73.51

Table 3: Comparison with state-of-the-art NAS methods on NASBench-201.

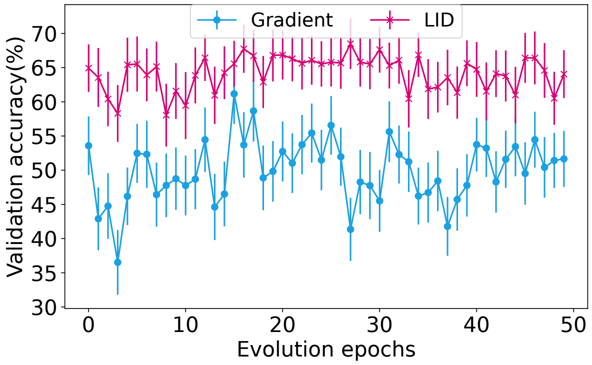

Figure 5: Validation accuracy of GM-NAS and our NAS-LID on NASBench-201 during the evolutionary search.

Performance of Derived Architectures For GM-NAS and our NAS-LID, we fine-tune the 16 sub-supernets for 50 epochs and then apply the evolutionary algorithm to search for superior architectures based on these sub-supernets. Prior to validation, each architecture inherits the weights from its corresponding sub-supernet. We conduct the evolutionary search for 50 epochs. We produce 50 promising architectures for each search epoch via the crossover, mutation, and selection steps. Fig. 5 compares the mean and standard deviation of validation accuracy of the searched architectures in each evolution epoch. Compared to the dramatic fluctuations in the validation accuracy of GM-NAS, we achieve higher and more stable validation performance, which further justifies that NAS-LID can better reduce the interference among architectures and thus help the training of sub-supernets. Table 3 compares the performance of the best architectures found by different algorithms on the CIFAR-10 and CIFAR-100 datasets. Note that the validation and test accuracy are queried from NASBench-201. The three Few-shot NAS methods adopt the same training strategy (i.e., RSPS) to train the sub-supernets and outperform RSPS by a large margin. Besides, NAS-LID can still achieve better performance and smaller variance than GM-NAS on both datasets, which confirms the advantages of LID.

Generalizing to Other Spaces

To evaluate the generalizability of NAS-LID, we further evaluate NAS-LID on other open domain search spaces, including OFA (Cai et al. 2020) and ProxylessNAS (Cai, Zhu, and Han 2019). We select two layers and split the supernet into four sub-supernets. We fine-tune each sub-supernet for 50 epochs and apply the evolutionary algorithm to search for promising architectures. We conduct the supernet partition and evolutionary search on the CIFAR-10 dataset for ProxylessNAS space. We then transfer the searched architectures to the ImageNet (Deng et al. 2009) dataset and present the results in Table 4. We took 2.3 GPU days to split the supernet and search promising architectures on ProxylessNAS. We achieve a top-1 test error rate of 23.0%, surpassing the one-shot and previous two few-shot methods. Our searched model also achieves competitive results in the OFA space, validating the effectiveness of NAS-LID.

Architecture	Test Error(%)		Params (M)	Flops (M)	Search Cost (GPU Days)	Search Method
	top-1	top-5				
Inception-v1 (Szegedy et al. 2015)	30.1	10.1	6.6	1448	-	manual
MobileNet (Howard et al. 2017)	29.4	10.5	4.2	569	-	
NASNet-A (Zoph et al. 2018)	26.0	8.4	5.3	564	2000	Vanilla NAS
PNAS (Liu et al. 2018)	25.8	8.1	5.1	588	225	
AmoebaNet-C (Real et al. 2019)	24.3	7.6	6.4	570	3150	
EfficientNet-B1 (Tan and Le 2019)	20.9	5.6	7.8	700	-	
MnasNet-92 (Tan et al. 2019)	25.2	8.0	4.4	388	-	
DARTS (Liu, Simonyan, and Yang 2019)	26.7	-	4.9	595	4	One-shot NAS
GDAS (Dong and Yang 2019b)	26.0	8.5	5.3	581	0.3	
BayesNAS (Zhou et al. 2019)	26.5	8.9	3.9	-	0.2	
P-DARTS (Chen et al. 2021b)	24.4	7.4	4.9	557	0.3	
DSNAS (Hu et al. 2020)	25.7	8.1	-	324	-	
ISTA-NAS (Yang et al. 2020)	24.0	7.1	5.7	638	-	
PC-DARTS (Xu et al. 2020)	24.2	7.3	5.3	597	3.8	
BigNAS-L (Yu et al. 2020)	20.5	-	6.4	586	-	
DrNAS (Chen et al. 2021a)	23.7	7.1	5.7	604	4.6	
SPOS (Guo et al. 2020)	25.3	-	3.4	328	8.3	
CLOSE (Zhou et al. 2022)	24.7	-	4.8	-	-	
ProxylessNAS (Cai, Zhu, and Han 2019)	24.9	7.5	7.1	465	8.3	
OFA (Cai et al. 2020)	20.0	5.1	9.1	595	1.7‡	
K-Shot-NAS-A (Su et al. 2021)	22.4	6.4	6.5	422	1	Few-shot NAS
FSNAS (ProxylessNAS) (Zhao et al. 2021)	24.1	-	4.9	521	20.8	
GM-NAS (ProxylessNAS) (Hu et al. 2022)	23.4	7.0	4.9	530	24.9	
NAS-LID (ProxylessNAS)	**22.9**	**6.3**	6.9	678	2.3†	
FSNAS (OFA) (Zhao et al. 2021)	20.2	5.2	9.2	600	1.7‡	
GM-NAS (OFA) (Hu et al. 2022)	19.7	5.0	9.3	587	1.7‡	
NAS-LID (OFA)	**19.5**	**5.0**	9.9	776	1.7‡	

Table 4: Comparison with state-of-the-art NAS methods on ImageNet. †The supernet partition and evolutionary search are conducted on the CIFAR-10 dataset. ‡ Only the search cost of evolutionary search on the ImageNet dataset is reported.

LID Similarity Measures

Measure	Kendall (τ)		Spearman (γ)	
	max	mean(std)	max	mean(std)
Pearson	0.30	0.26 (0.03)	0.45	0.39 (0.04)
Euclidean	**0.37**	**0.34 (0.02)**	**0.53**	**0.48 (0.03)**

Table 5: Ranking correlation of top 150 architectures on NASBench-201 with different similarity measures.

The above experiments used Euclidean distance to measure LID similarity. If we ignore the scaling effect and only consider the correlations among LIDs (e.g., treating $[2, 6, 4]$ and $[20, 60, 40]$ as the same), is it also effective? We explore this using the Pearson correlation coefficient. As shown in Table 5, Pearson similarity is not as effective as the Euclidean measure, which implies that the scale of the LID values is essential to characterize the geometric properties of different architectures in a high-dimensional space.

Conclusion & Future Work

Our empirical findings in this study demonstrate that LID is a highly promising criterion for characterizing architectures, with similar LID profiles indicating similar training dynamics. We propose NAS-LID as a proof-of-concept for using LID to split the supernet, grouping subnets into different clusters (i.e., sub-supernets) and reducing interference between architectures during training. Compared to the gradient-based method, NAS-LID effectively addresses the curse of dimensionality in high-dimensional architecture search spaces and achieves better space separability and higher ranking correlation of subnets. The scale of LID values plays a crucial role in architecture characterization. Our preliminary understanding is that the LID profile provides insights into the underlying mechanisms of neural network learning by revealing how each model layer transforms high-dimensional data into low-dimensional sub-manifolds.

While our focus in this work is on image classification models, we believe that exploring LID profiles of other models, including large language models, could provide deeper insights into their relationship with model generalization ability and broader potential. This creates opportunities for further research and applications of LID in various areas.

Acknowledgements

We thank the NVIDIA Academic Hardware Grant Program for its support of our work.

References

Aghajanyan, A.; Gupta, S.; and Zettlemoyer, L. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 7319–7328. Online: Association for Computational Linguistics.

Amsaleg, L.; Chelly, O.; Furon, T.; Girard, S.; Houle, M. E.; Kawarabayashi, K.-i.; and Nett, M. 2015. Estimating local intrinsic dimensionality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 29–38.

Ansuini, A.; Laio, A.; Macke, J. H.; and Zoccolan, D. 2019. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32.

Bashivan, P.; Tensen, M.; and DiCarlo, J. J. 2019. Teacher guided architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5320–5329.

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.

Boykov, Y. Y.; and Jolly, M.-P. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, 105–112. IEEE.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2020. Once-for-All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations*.

Cai, H.; Zhu, L.; and Han, S. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations*.

Campadelli, P.; Casiraghi, E.; Ceruti, C.; and Rozza, A. 2015. Intrinsic dimension estimation: Relevant techniques and a benchmark framework. *Mathematical Problems in Engineering*, 2015.

Chen, X.; Wang, R.; Cheng, M.; Tang, X.; and Hsieh, C.-J. 2021a. DrNAS: Dirichlet Neural Architecture Search. In *International Conference on Learning Representations*.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2021b. Progressive darts: Bridging the optimization gap for nas in the wild. *International Journal of Computer Vision*, 129(3): 638–655.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dong, X.; and Yang, Y. 2019a. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3681–3690.

Dong, X.; and Yang, Y. 2019b. 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. In *International Conference on Learning Representations*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.

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 *European conference on computer vision*, 544–560. Springer.

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.

He, X.; Wang, S.; Chu, X.; Shi, S.; Tang, J.; Liu, X.; Yan, C. C.; Zhang, J.; and Ding, G. 2021. Automated Model Design and Benchmarking of 3D Deep Learning Models for COVID-19 Detection with Chest CT Scans. In *AAAI*.

He, X.; Ying, G.; Zhang, J.; and Chu, X. 2022. Evolutionary Multi-objective Architecture Search Framework: Application to COVID-19 3D CT Classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 560–570. Springer.

He, X.; Zhao, K.; and Chu, X. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212: 106622.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Houle, M. E. 2017. Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications. In *International Conference on Similarity Search and Applications*, 64–79. Springer.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, S.; Wang, R.; HONG, L.; Li, Z.; Hsieh, C.-J.; and Feng, J. 2022. Generalizing Few-Shot NAS with Gradient Matching. In *International Conference on Learning Representations*.

Hu, S.; Xie, S.; Zheng, H.; Liu, C.; Shi, J.; Liu, X.; and Lin, D. 2020. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12084–12092.

Krizhevsky, A.; and Hinton, G. 2009. Learning Multiple Layers of Features from Tiny Images. *Tech Report*.

Lei, N.; An, D.; Guo, Y.; Su, K.; Liu, S.; Luo, Z.; Yau, S.-T.; and Gu, X. 2020. A geometric understanding of deep learning. *Engineering*, 6(3): 361–374.

Li, C.; Farkhoor, H.; Liu, R.; and Yosinski, J. 2018. Measuring the Intrinsic Dimension of Objective Landscapes. In *International Conference on Learning Representations*.

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

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 82–92.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, 19–34.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Liu, Y.; Jia, X.; Tan, M.; Vemulapalli, R.; Zhu, Y.; Green, B.; and Wang, X. 2020. Search to distill: Pearls are everywhere but not the eyes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7539–7548.

Peng, J.; Zhang, J.; Li, C.; Wang, G.; Liang, X.; and Lin, L. 2021. Pi-NAS: Improving neural architecture search by reducing supernet training consistency shift. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12354–12364.

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.

Shwartz-Ziv, R.; and Tishby, N. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Su, X.; You, S.; Zheng, M.; Wang, F.; Qian, C.; Zhang, C.; and Xu, C. 2021. K-shot NAS: Learnable Weight-Sharing for NAS with K-shot Supernets. In *International Conference on Machine Learning*, 9880–9890. PMLR.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2820–2828.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, 6105–6114. PMLR.

Tishby, N.; Pereira, F. C.; and Bialek, W. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Xu, J.; Chen, Z.; Quek, T. Q.; and Chong, K. F. E. 2022a. FedCorr: Multi-Stage Federated Learning for Label Noise Correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10184–10193.

Xu, J.; Tan, X.; Song, K.; Luo, R.; Leng, Y.; Qin, T.; Liu, T.-Y.; and Li, J. 2022b. Analyzing and mitigating interference in neural architecture search. In *International Conference on Machine Learning*, 24646–24662. PMLR.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2020. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In *International Conference on Learning Representations*.

Yang, Y.; Li, H.; You, S.; Wang, F.; Qian, C.; and Lin, Z. 2020. Ista-nas: Efficient and consistent neural architecture search by sparse coding. *Advances in Neural Information Processing Systems*, 33: 10503–10513.

Yao, J.; Zhang, S.; Yao, Y.; Wang, F.; Ma, J.; Zhang, J.; Chu, Y.; Ji, L.; Jia, K.; Shen, T.; et al. 2022. Edge-Cloud Polarization and Collaboration: A Comprehensive Survey for AI. *IEEE Transactions on Knowledge and Data Engineering*.

Ying, G.; He, X.; Gao, B.; Han, B.; and Chu, X. 2022. EAGAN: Efficient Two-Stage Evolutionary Architecture Search for GANs. In *European Conference on Computer Vision*, 37–53. Springer.

Yu, J.; Jin, P.; Liu, H.; Bender, G.; Kindermans, P.-J.; Tan, M.; Huang, T.; Song, X.; Pang, R.; and Le, Q. 2020. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, 702–717. Springer.

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.

Zhou, H.; Yang, M.; Wang, J.; and Pan, W. 2019. Bayesnas: A bayesian approach for neural architecture search. In *International conference on machine learning*, 7603–7613. PMLR.

Zhou, Z.; Ning, X.; Cai, Y.; Han, J.; Deng, Y.; Dong, Y.; Yang, H.; and Wang, Y. 2022. Close: Curriculum learning on the sharing extent towards better one-shot nas. In *European Conference on Computer Vision*, 578–594. Springer.

Zoph, B.; and Le, Q. 2017. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.