Neural Architecture Search with Representation Mutual Information

Xiawu Zheng[1,2†], Xiang Fei[1†], Lei Zhang[1†], Chenglin Wu[3], Fei Chao[1],
Jianzhuang Liu[4], Wei Zeng[2,5], Yonghong Tian[2,5], Rongrong Ji[1,2,6,7*]
[1]Media Analytics and Computing Lab, Department of Artificial Intelligence, School of Informatics,
Xiamen University. [2]Peng Cheng Laboratory. [3]DeepWisdom Inc. [4]Huawei Noah's Ark Lab.
[5]National Engineering Research Center for Visual Technology, School of Computer Science,
Peking University. [6]Institute of Artificial Intelligence, Xiamen University. [7]Fujian Engineering
Research Center of Trusted Artificial Intelligence Analysis and Application, Xiamen University
{zhengxiawu, xiangf, leizhang}@stu.xmu.edu.cn, rrji@xmu.edu.cn
alexanderwu@fuzhi.ai, liu.jianzhuang@huawei.com, {weizeng, yhtian}@pku.edu.cn

Abstract

Performance evaluation strategy is one of the most important factors that determine the effectiveness and efficiency in Neural Architecture Search (NAS). Existing strategies, such as employing standard training or performance predictor, often suffer from high computational complexity and low generality. To address this issue, we propose to rank architectures by Representation Mutual Information (RMI). Specifically, given an arbitrary architecture that has decent accuracy, architectures that have high RMI with it always yield good accuracies. As an accurate performance indicator to facilitate NAS, RMI not only generalizes well to different search spaces, but is also efficient enough to evaluate architectures using only one batch of data. Building upon RMI, we further propose a new search algorithm termed RMI-NAS, facilitating with a theorem to guarantee the global optimal of the searched architecture. In particular, RMI-NAS first randomly samples architectures from the search space, which are then effectively classified as positive or negative samples by RMI. We then use these samples to train a random forest to explore new regions, while keeping track of the distribution of positive architectures. When the sample size is sufficient, the architecture with the largest probability from the aforementioned distribution is selected, which is theoretically proved to be the optimal solution. The architectures searched by our method achieve remarkable top-1 accuracies with the magnitude times faster search process. Besides, RMI-NAS also generalizes to different datasets and search spaces. Our code has been made available at https://git.openi.org.cn/PCL_AutoML/XNAS.

1. Introduction

Neural Architecture Search (NAS) is proposed to facilitate the design of deep neural networks, which is a challenging task and has demonstrated superior performance on various computer vision tasks, including but not limited to image classification [61, 67], object detection [10, 53] and segmentation [7, 33]. As a widely accepted standpoint, a conventional NAS algorithm is divided into three components [19]: search space, search algorithm and performance estimation strategy. The search space defines the scope of the search, the search strategy investigates how to explore the search space and the performance estimation refers to how to estimate the performance for the architectures.

Through extensive experiments, previous works [62, 64] have demonstrated that performance estimation is the most important component in NAS. In particular, an optimal estimation strategy is capable of improving both efficiency and effectiveness simultaneously with different search algorithms in NAS, which is investigated in [62, 64]. According to the previous works [19, 31], performance estimation strategies include multiple-fidelity training methods* [20, 42, 57, 62, 67], accuracy predictor based methods [2, 37, 38, 49], one-shot methods [5, 35, 48, 51, 54, 56, 58–61, 65] and training-free based methods [8, 31, 39].

The key challenge of the performance estimation is the trade-off among the accuracy, generalization and computation cost. Although multiple fidelity and accuracy predictor based methods are accurate and generalize to different search spaces, such methods require a lot of computation resources which hinder the usage of NAS in practical applications. For example, AmoebaNet [42] costs more than $3,150$

*Corresponding author.
†These authors contributed equally to this work.

*These methods use different training hyper-parameters for acceleration, including but not limited to fewer epochs, subset of data and downscaled models.

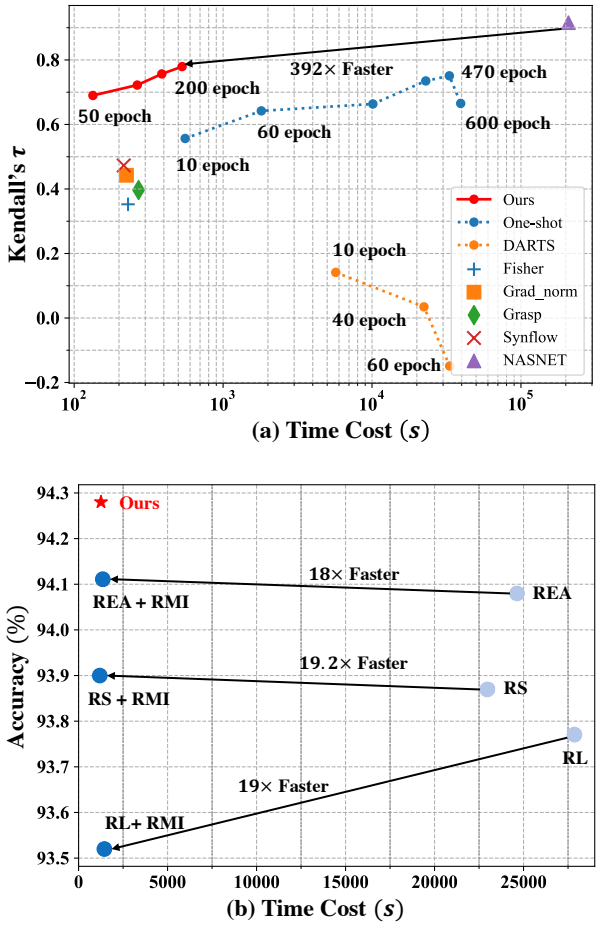

Figure 1. (a) The illustration of time cost for evaluating 100 architectures in the NAS-Bench-201 [16] benchmark. RMI is much faster with marginal correlation drop compared to NASNet [66], DARTS [35] and one-shot based methods [59]. (b) By incorporting RMI, we can largely accelerate NAS methods including reinforcement learnming (RL) [66], evolution algorithm (REA) [42] and random search (RS). Meanwhile, the proposed method further improves the accuracy by a clear margin with less search costs.

GPU-days to search for the optimal on CIFAR-10 [27]. To reduce the computation costs, one-shot and training-free based methods are proposed. These methods are based on parameter sharing and deep learning theory respectively. Albeit being more efficient than multiple fidelity and accuracy predictor based methods, one-shot and training-free based methods still have severe generalization issue. For example, the widely-used DARTS [35] is efficient only on the cell-based search space [66], where the more practical chain-structure search space [48] is rarely adopted in these works [34, 35, 54]. Meanwhile, another popular one-shot strategy [4] only employs the MobileNet [23, 44] search space for validation. Such a problem also exists in training-free based methods [8, 31, 39]. Overall, their search spaces are carefully selected or designed in their work.

In this paper, we propose a novel performance estimation strategy that ranks architectures by using the hidden Representation Mutual Information (RMI). In particular, through extensive experiments we find architectures that have good accuracies always yield high RMI scores. In other words, an arbitrary architecture that has decent accuracy can be considered as an accurate indicator by using RMI to facilitate NAS. For example, in the widely-used dataset CIFAR-10 [27], any architectures including human-designed or random sampled from pre-defined search spaces that have $> 85\%$ top classification accuracy is used as an accurate performance indicator. In practice, the calculation and optimization of RMI between two architectures only require a mini-batch of data. Compared to using the entire dataset during regular training, our approach enables a very significant speedup to estimate each architecture. Moreover, RMI also generalizes to different search spaces, which makes it potentially widely applicable to real-world problems.

We further propose an effective and efficient NAS algorithm that explores the search space by using the RMI score. In terms of efficiency, the proposed RMI performs as the guidance in NAS. It eliminates the need for laborious training on the whole dataset, thus significantly reducing computation complexity. Specifically, RMI-NAS first randomly samples architectures from the search space, which are then classified as positive or negative samples using RMI score. These samples are used to train a random forest to further accelerate the exploration of the unseen regions. Meanwhile, we keep tracking of the distribution of positive architectures. In terms of effectiveness, the architecture found is also theoretically guaranteed to be the optimal solution. After the whole search space is fully explored by random forest [3], we then select the architecture with the largest probability from the aforementioned distribution. To summarize, our main contributions are two-fold:

- Based upon extensive statistical verification, we empirically demonstrate that representation mutual information is a stable and accurate indicator to find the optimal architecture[†]. To the best of our knowledge, RMI is introduced for the first time to the NAS community and can be easily incorporated into most existing NAS algorithms to speed up the search process.

- We introduce a novel NAS optimization method termed RMI-NAS, which is efficient, fast and generalizes to different search spaces. We employ RMI and random forest to effectively explore the whole search space. Meanwhile, we also mathematically prove that

[†]Extensive experiments in Sec. 4 show that RMI shows a high correlation with the performance of the architectures in the search space.

the solution found by RMI-NAS is more likely to be the optimal solution in the search space.

Extensive experimental results demonstrate the efficiency and effectiveness of the proposed method on search spaces and datasets. Notably, under the setting of NAS-Bench-201 [16], our searched model achieves 94.36% test accuracy on CIFAR-10 dataset within 30 minutes, which is attributed to the proposed RMI and search algorithm.

2. Related work

As mentioned before, performance estimation strategy is critical to the efficiency and effectiveness of NAS. Therefore, we review NAS from the perspective of performance estimation. A comprehensive review of NAS is provided in the monograph [19]. In early works [42, 66, 67], multiple-fidelity training based methods are adopted to obtain the accuracy. These methods reduce the time of performance estimation by tuning training hyperparameters, including but not limited to employing downscaled proxy models [66], training for fewer epochs and using proxy datasets [42]. For instance, AmoebaNet [42] searchs on CIFAR-10 [27] and then transfers the searched architecture on ImageNet [15]. Such search process takes $3,150$ GPU-days and the searched architecture achieves 74.5% top-1 accuracy.

As we can see, multiple-fidelity training based methods need lots of training which hinders the usage of NAS practically. To solve this issue, one-shot methods are proposed recently, which employ the weight sharing strategy to avoid the exhausted and repeated training process. Such a method is widely adopted successfully in many efficient NAS applications, such as DARTS [35], PDARTS [34], PC-DARTS [54] and FBNet [50]. Although the mentioned efforts have reduced the searching cost, their generalizability and efficiency in performance estimation have been deeply challenged. For example, PC-DARTS [54] proposes to use partial channels in the search space for acceleration. Nevertheless, PC-DARTS does not generalize to the channel number search space, which is an important research area in network compression. Moreover, recent work [30] has found that the search algorithms in these works may not be better than random search.

Until very recently, training-free based methods [8, 31, 39] are proposed and actively explored in the literature. These methods attempt to employ some easily accessible architecture attributes to probe good architectures. As the name implies, these methods do not need any training process and thus lead to extremely effective performance estimation. However, these methods cannot generalize to different search spaces. Concretely, the recent work [39] is effective only in the cell-based search space [35], where the chain-structure is not investigated. Besides, different training-free based methods usually search on different datasets, e.g., ImageNet and the elaborate search space in Lin et al. [31] and CIFAR-10 in Mellor et al. [39].

In this paper, we propose a novel performance estimation method termed RMI, which focuses on estimation accuracy, generalization and computation cost simultaneously. Incorporated with our new search algorithm, RMI-NAS achieves comparable or better results on several benchmarks and search spaces with much less computation costs.

3. The Proposed Method

Notation. In this paper, we use upper case letters (e.g., X, Y) as random variables, and bold as vectors (e.g., $\boldsymbol{x}, \boldsymbol{y}$), matrices or tensors (e.g., $\boldsymbol{X}, \boldsymbol{Y}$). Calligraphic font denotes spaces or loss functions (e.g., $\mathcal{X}, \mathcal{Y}, \mathcal{L}$). To better describe the proposed method, we further define some common symbols in CNNs. $\boldsymbol{\alpha} \in \mathcal{A} \in \mathbb{R}^{N \times M}$ denotes an indicator vector, where N and M are the numbers of edges and operations, respectively. We use the superscript of $\boldsymbol{\alpha}^a$ to represent a specific architecture a and the subscript of $\boldsymbol{\alpha}_i$ as a certain edge i. $\boldsymbol{W} \in \mathcal{W}$ denotes the architecture weights.

Problem Formulation. In this paper, we solve the following optimization problem

$$\max_{\boldsymbol{W} \in \mathcal{W}, \boldsymbol{\alpha} \in \mathcal{A}} \mathcal{L}\left(\boldsymbol{W}, \boldsymbol{\alpha}\right), \text{ s.t. } \mathcal{T}(\boldsymbol{\alpha}) < \Omega. \tag{1}$$

$\mathcal{L} : \mathcal{W} \times \mathcal{A} \rightarrow \mathbb{R}$ is the loss function that is not differentiable with respect to $\boldsymbol{\alpha}$, and is differentiable with respect to $\boldsymbol{W}$. $T(\cdot)$ is a function that denotes the constraints for architectures, such as FLOPs, and Ω is given for different hardwares. We present an overview of RMI-NAS in Fig. 2, which aims to automatically discover the optimal neural architecture. The detailed motivations, descriptions and analysis are presented in the following sub-sections.

3.1. Representation Mutual Information

RMI Score. As mentioned before, performance estimation directly determines the efficiency of NAS. Fast performance estimation allows the search algorithm to explore the search space widely, while accurate performance estimation facilitates the search algorithm to better perceive the probability distribution of the search space. Rather than estimating architectures by using laborious training methods, we propose Representation Mutual Information (RMI) to achieve effective and efficient performance estimation. In particular, given an arbitrary network $\boldsymbol{\alpha}^+$ that has a decent accuracy, we use $X^{1+}, ..., X^{L+}$ to represent the random variables of feature maps in each layer. For any architecture $\boldsymbol{\alpha}$ that is sampled from the search space, we formally define the RMI score as

$$\phi(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}) = \sum_{i=1}^{L} I\left(X^{i+}, X^{i}\right). \tag{2}$$

In general, an architecture that has a high RMI score tends to be a good architecture, as the proposed RMI score

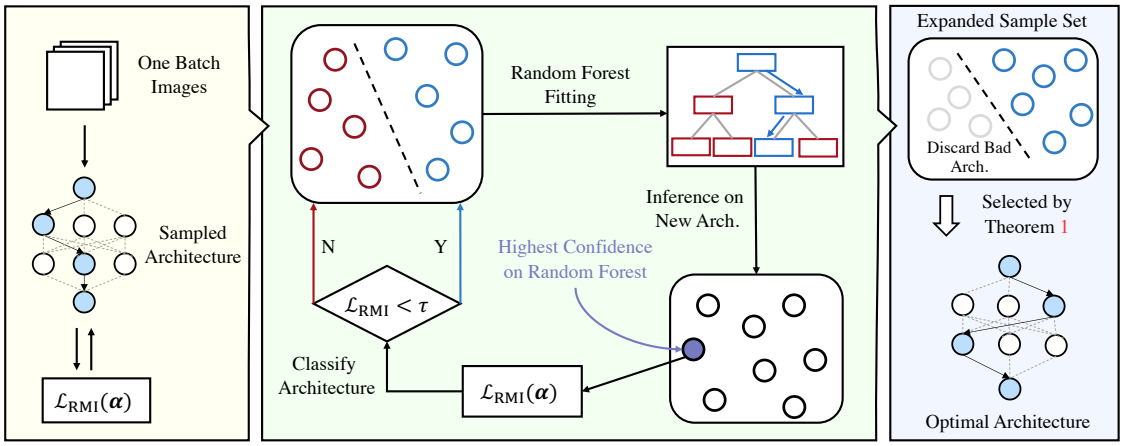

Figure 2. Overview of the proposed RMI-NAS. (a) In particular, we first sample n architectures and classify them as good or bad according to the RMI score, which is used to warm-up the random forest classifier. (b) We then employ the random forest to explore the search space effectively and iteratively. In each iteration, a large amount of architectures are sampled, where the architecture that has the highest confidence on the random forest is selected for training and calculating the RMI score. Depending on the score, the architecture is then classified and added to the sample set for updating the random forest in the next iteration. (c) After collecting a certain amount of good architectures, we use these architectures to construct an expanded sampling set and approximate the probability distribution of operations. The architecture that has the largest probability is considered as the optimal, according to Theorem 1.

is robust, effective and efficient, which is also demonstrated in Sec. 4. More specifically, the RMI score shows a high correlation with different $\boldsymbol{\alpha}^+$ and training conditions. In fact, we directly optimize the RMI score using only one training batch, where the result is accurate enough to explore the search space.

Acceleration of RMI Score Computation. The RMI score is hard to calculate since the distribution of hidden representations is intractable and time-consuming to estimate. Besides, the high dimension of X^i also suffers from the curse of dimensionality. Here we introduce the normalized Hilbert-Schmidt Independence Criterion (HSIC) [21, 63][‡] to solve this problem. Specifically, let $\mathcal{D} := \{(x_1, y_1), ..., (x_n, y_n)\}$ contain n i.i.d. samples drawn from the distribution P_{XY}. In this case, the normalized HSIC [21, 26, 63] is defined as

$$I(X,Y) \approx \text{nHSIC}_{\text{linear}}(X,Y) = \frac{||\boldsymbol{Y}^T\boldsymbol{X}||_F^2}{||\boldsymbol{X}^T\boldsymbol{X}||_F||\boldsymbol{Y}^T\boldsymbol{Y}||_F}, \quad (3)$$

where $||\cdot||_F$ is the Frobenius norm or the Hilbert-Schmidt norm. Calculating of Eq. 3 is effective, i.e., $\mathcal{O}(n^2)$, where n is the number of samples. It only takes a few seconds on a single GPU and CPU in practice.

Representation Learning and Performance Estimation with the RMI Loss. As the RMI score needs only one batch of data for calculation, the selection of data is of great importance. In particular, discriminative and representative features are more suitable for the RMI score. In deep learning, samples with a large confidence are close to the class center, where the corresponding features tend to be discriminative and representative. Therefore, we first set the expansion factor γ and randomly select data with γ times of the batch size as a data pool. The pretrained architecture $\boldsymbol{\alpha}^+$ is then used as a teacher network to select n_{batch} samples with the largest classification confidences from it. Subsequently, the samples are used to obtain the feature maps $\boldsymbol{X}^+$ and $\boldsymbol{X}$ of each stage/layer in $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}$, respectively. The RMI loss and classification error are optimized simultaneously, which is formally defined as

$$\begin{aligned}\mathcal{L}_{\text{loss}} &= \beta\mathcal{L}_{\text{RMI}} + (1-\beta)\mathcal{L}_{\text{cls}} \\ &= \beta\sum_{i=1}^{L}\frac{||\boldsymbol{X}^{i+T}\boldsymbol{X}^i||_F^2}{||\boldsymbol{X}^{iT}\boldsymbol{X}^i||_F||\boldsymbol{X}^{i+T}\boldsymbol{X}^{i+}||_F} + (1-\beta)\mathcal{L}_{\text{cls}}.\end{aligned} \quad (4)$$

In RMI-NAS, we use Eq. 4 to estimate an architecture. That is, after optimizing with certain epochs, a lower RMI loss indicates a better performance. Notably, optimizing Eq. 4 is extremely effective, as it only takes less than 10s for an architecture sampled from NAS-Bench-201 [16].

3.2. Search Algorithm

Here we describe the proposed RMI-NAS, which is also illustrated in Fig. 2. Since the RMI score is employed for performance estimation, only one batch of training data is required. Meanwhile, a random forest model with sequential updating is introduced to improve the sampling efficiency, in which the final architecture is obtained based on

[‡]Normalized HSIC is also known as CKA [26], RV coefficient [43] and Tucker's congruence coefficient [36], which is also used to learn diversified representations [24].

the samples. In general, RMI-NAS consists of three steps: random forest training, expanded sampling and architecture selection, which are elobrated in the following contents.

Random Forest Training. Even though the proposed RMI is effective in estimating neural architectures, it is still time-consuming to explore the whole search space. Therefore, we employ a random forest model to facilitate the exploration. In particular, we first randomly sample $n_{\text{warm}} = 100$ architectures, which are then optimized with the loss function in Eq. 4. After training these architectures for a certain number of iterations, their loss values are used for performance evaluation. In this case, top $k = 5\%$ architectures with the lowest loss value are considered as good and labeled to 1, while the rest are considered as bad and labeled to 0. Such classification also generates a loss value threshold τ for the following steps. This classification result is then used to construct a dataset for training the random forest model. Specifically, conventional NAS can be abstracted as a discrete optimization problem, i.e., candidate operation selection for each edge. We thus encode the selection in each dimension (edge) in $\boldsymbol{\alpha}$ as a one-hot vector. In this case, $\boldsymbol{\alpha}$ is encoded as a matrix that becomes the input to the random forest, where the classification results are used as labels to supervise the output. When the random forest is trained to convergence, it is then used to explore the search space effectively and iteratively.

Expanded Sampling. During this phase, we first randomly sample n_{fitting} network architectures in the search space for each iteration. Then, the random forest is employed to find the architecture with the largest classification confidence, which is then evaluated by the proposed RMI loss in Eq. 4. The actual loss value will be used to update the random forest in the next iteration. This architecture will be marked as a good sample if its loss value is below the threshold τ, and when n_{collect} good architectures are collected, the update step of the random forest is terminated.

Architecture Selection. In the previous step, a set with n_{collect} good architectures are collected. We have a new theorem to obtain the final architecture, which is described as:

Theorem 1: *Assume that $P(\boldsymbol{\alpha})$ obeys the uniform distribution on the domain of definition for an arbitrary black box function $f(\boldsymbol{\alpha})$. For a specific threshold τ, it holds that*

$$\arg\max_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) = \arg\max_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha} | f(\boldsymbol{\alpha}) + \sigma\epsilon > \tau), \quad (5)$$

where $\sigma > 0$, $\epsilon \sim \mathcal{N}(0, 1)$.

The proof is provided in our supplementary materials for a better understanding. According to Theorem 1, we can employ the distribution $P(\boldsymbol{\alpha} | f(\boldsymbol{\alpha}) + \sigma\epsilon > \tau)$ to find the optimal solution. However, this distribution is intractable in practise. Fortunately, we have effectively explored the search space and collected n_{collect} architectures whose loss values are smaller than threshold τ. Therefore, we can obtain the final architecture accordingly. In particular, for the collected architecture set $\mathcal{A}^{\text{collect}}$, we calculate the statistical mode in each edge, and set the operation with the largest frequency as the optimal architecture for this edge. Formally, for a specific edge i, the architecture is obtained through

$$\boldsymbol{\alpha}_i^* = \text{Mo}\left(\mathcal{A}_i^{\text{collect}}\right), \quad (6)$$

where Mo denotes the statistical mode function. The algorithm is also summarized in Alg. 1

Discussion. RMI-NAS is a fast, generalizable and efficient method. Firstly, our proposed performance estimation metric RMI score is effective and generalizes to different search spaces. In particular, the RMI loss does not need any training process on the entire dataset. Besides, the caulation of $I(X, Y)$ is effective, i.e., $\mathcal{O}(n^2)$, where n is the number of samples. Secondly, the proposed random forest further improves the effectiveness of NAS, where the optimal architecture is also guaranteed by Theorem 1. Concretely, it only takes less than 30 minutes on NAS-Bench-201 [16] to complete searching. Moreover, RMI-NAS can be incorporated with latency constraints by using reject sampling.

4. Experiments

In this section, we quantitatively present empirical evaluation of the proposed method on several widely-used datasets and search spaces. In particular, we first apply RMI-NAS to the NAS for image classification on the widely-used CIFAR-100, CIFAR-10 and ImageNet datasets with different search spaces and NAS benchmarks in Sec. 4.1. We then conduct ablation studies to investigate the efficiency of each component in Sec. 4.2. We have released all the source code including baselines, hyperparameter settings, training and searching code. At the same time, we carefully check all the option checklist [32] and confirm that the open-source code meets all the requirements.

4.1. Comparison with SOTA methods

Implementation Details. For the computation of RMI, teacher networks can be either complex, large-scale architectures, or tiny models with elaborate layers. In practice, we use ResNet [22] with different depths on CIFAR [27] and ImageNet [15] datasets. Meanwhile, a similar architecture is adopted to the search space on the ImageNet16-120 [11] dataset. Both methods have been proved to be effective through extensive experiments.

As we mentioned before, since only one batch of data is needed for the complete training in RMI-NAS, the quality of data is crucial to the final models' performance. Therefore, the aforementioned teacher network is subsequently used to judge the quality of the input images. In particular, during the data selection phase, we set the expansion

Algorithm 1: RMI-NAS

input : Sampling Number $n_{\text{warm}}, n_{\text{fitting}}, n_{\text{collect}}$; Quantile $k \in (0, 100)$; Architecture Search Space $\mathcal{A}$; Random Forest Model $\pi_\theta : \mathcal{A} \to [0, 1]$.

output: Searched Architecture $\boldsymbol{\alpha}^*$

Initial sampling set $\mathcal{D} = \emptyset$, $\mathcal{D}_{\text{warm}} = \emptyset$;
Pick preferred batch data as described in Sec. 4.1;
for $i \leftarrow 1$ **to** n_{warm} **do**
  $\boldsymbol{\alpha}^i \leftarrow$Sample an architecture from $\mathcal{A}$;
  $l^i \leftarrow$ Optimize $\boldsymbol{\omega}_{\boldsymbol{\alpha}^i}$ by Eq. 4;
  $\mathcal{D}_{\text{warm}} \leftarrow \mathcal{D}_{\text{warm}} \cup \{(\boldsymbol{\alpha}^i, l^i)\}$;
end
$\tau \leftarrow$ Get top $k\%$ quantile in $\mathcal{D}_{\text{warm}}$;
$z^i = \mathbb{I}\left[l^i \leq \tau\right]$ for $i = 1, \ldots, n_{\text{warm}}$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\alpha}^i, z^i)\}_{i=1}^{n_{\text{warm}}}$;
$\pi_{\theta^*} \leftarrow$ Fit π_θ by using $\mathcal{D}$;
$i \leftarrow n_{\text{warm}} + 1$;
while $i \leq n_{collect}$ **do**
  $\mathcal{A}^{\text{fitting}} \leftarrow$ Sample n_{fitting} architectures;
  $\boldsymbol{\alpha}^i \leftarrow \arg\max_{\boldsymbol{\alpha} \in \mathcal{A}^{\text{fitting}}} \pi_{\boldsymbol{\theta}_\star}(\mathcal{A}^{\text{fitting}})$;
  $l^i \leftarrow$ Optimize $\boldsymbol{\alpha}^i$ by Eq. 4;
  $z^i = \mathbb{I}\left[l^i \leq \tau\right]$;
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\alpha}^i, z^i)\}$;
  $\pi_{\theta^*} \leftarrow$ Fit π_θ by using $\mathcal{D}$;
  $i \leftarrow i + 1$;
end
Obtain the final architecture $\boldsymbol{\alpha}^*$ through Eq. 6;

factor γ and select data with γ times of the batch size as the data pool. A batch of data with the lowest classification loss in the data pool is then selected by the teacher network for subsequent experiments. In practice, we set the batch size to 32 and $\gamma = 16$ as a trade-off between search speed and accuracy. We also use MindSpore to validate the generalizability of our algorithm.

In the post-processing phase, the architecture must be properly encoded for the random forest in Alg. 1. We thus encode each operation choice as a one-hot vector. In this case, the architecture can be represented as a matrix. The overall process of the random forest has been described in Sec. 3.2. More specifically, we implement it with the sklearn [40] library, setting the number of trees to 30 and keeping all other hyperparameters as defaults. In the expanded sampling phase, one key step is to select the architecture with the highest confidence in the random forest from a larger set. Since this process is parallel, inference can be completed within seconds even if we set $n_{\text{fitting}} = 1000$. The selected architecture is retrained using RMI and classified according to the threshold set in the warm-up phase. Empirically, we set quantile $k = 5$ as the threshold for best performance. This iterative process is terminated when the random forest finds n_{succ} good architectures. As the number of choices for each operation is generally less than 10, it is sufficient to set $n_{\text{collect}} = 100$. Finally, in order to further stabilize the results, we further select half of the architectures with lower RMI loss in n_{collect} to calculate its preferable in the distribution, which eliminates quantile bias in the warm-up step.

We first conduct our experiments in NAS-Bench-201 [16] and DARTS [35]. We set different training epochs for different search spaces. In particular, we train 150 epochs for each architecture in the NAS-Bench-201 search space, while more epochs and time are required in larger search spaces. Additional experimental settings are provided in the supplementary materials.

Results on NAS-Bench-201. In this work, we follow the settings in the NAS-Bench-201 search space and implement our method on a single GTX 1080ti GPU to ensure a fair comparison with the baselines.

As shown in Table. 1, RMI-NAS is completed within 30 minutes, which is several times faster than the other baselines. In the meanwhile, the accuracy of the searched architecture has been significantly improved. We attribute these superior results to the efficient sampling strategies and lightweight metrics of RMI-NAS. Notably, the variance of our results is also reduced by a large margin, which indicates the stability and robustness of our method.

To discover the capability of RMI, we first make comparisons among several performance estimation strategies in Fig. 1(a). Together with Tab. 1, we obtain the following observations: Training-based evaluation strategies, which are widely used in traditional NAS algorithms like RS or RL, result in severe time consumption, but help them achieve stable performance and lower variance on all datasets. On the contrary, the efficient weight sharing and training-free strategies produce varied results under different settings. For instance, the Jacob_cov [39] method, though owning the best average performance, produces up to 2.07% variance on the ImageNet16-120 dataset. To summarize, the training-based methods focus more on stability and generalizability but result in huge time overhead, while the weight sharing and training-free strategies improve search speed but lead to performance decrease. Based on these observations, RMI-NAS shows the most stable and balanced results on all datasets, achieving state-of-the-art performance and verifying our claim in Sec. 3.1.

To demonstrate the generalizability of RMI, we further extend it to other commonly-used methods in Fig. 1(b). Specifically, we replace the training-based strategy with RMI on three different algorithms, which yields an average speedup of 19 times faster while maintaining almost the same accuracies. This result highlights that our RMI-NAS is able to learn the representation from the teacher network

Method	Search Cost (Seconds)	CIFAR-10		CIFAR-100		ImageNet16-120	
		Valid	Test	Valid	Test	Valid	Test
ResNet [22]	-	90.83	93.97	70.42	70.86	44.53	43.63
RS	22993.93	90.93 ± 0.36	93.80 ± 0.36	70.93 ± 1.09	71.04 ± 1.07	44.45 ± 1.10	44.57 ± 1.25
RL	27870.7	91.09 ± 0.37	93.85 ± 0.37	71.61 ± 1.12	71.71 ± 1.09	45.05 ± 1.02	45.24 ± 1.18
ENAS [41]	14058.8	37.51 ± 3.19	53.89 ± 0.58	13.37 ± 2.35	13.96 ± 2.33	15.06 ± 1.95	14.84 ± 2.10
DARTS-V2 [35]	35781.80	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
SETN [17]	34139.53	84.04 ± 0.28	87.64 ± 0.00	58.86 ± 0.06	59.05 ± 0.24	33.06 ± 0.02	32.52 ± 0.21
GDAS [18]	31609.80	89.89 ± 0.08	93.61 ± 0.09	71.34 ± 0.04	70.70 ± 0.30	41.59 ± 1.33	41.71 ± 0.98
FairNAS [13]	9845.00	90.97 ± 0.57	93.23 ± 0.18	70.94 ± 0.94	71.00 ± 1.46	41.09 ± 1.00	42.19 ± 0.31
MDENAS [61]	5300.00	-	89.07 ± 0.60	-	-	-	-
MIGONAS [59]	3200.00	-	93.15 ± 0.18	-	-	-	-
Jacob_cov [39]	-	89.69 ± 0.73	92.96 ± 0.80	69.87 ± 1.22	70.03 ± 1.16	43.99 ± 2.05	44.43 ± 2.07
Mag [47]	-	89.94 ± 0.34	93.35 ± 0.04	70.18 ± 0.66	70.47 ± 0.18	42.57 ± 2.14	43.17 ± 2.57
Ours	**1258.21**	$\mathbf{91.44 \pm 0.09}$	$\mathbf{94.28 \pm 0.10}$	$\mathbf{73.38 \pm 0.14}$	$\mathbf{73.36 \pm 0.19}$	$\mathbf{46.37 \pm 0.00}$	$\mathbf{46.34 \pm 0.00}$
Ours(best)	-	**91.55**	**94.36**	**73.49**	**73.51**	**46.37**	**46.34**
Optimal	-	91.61	94.37	73.49	73.51	46.77	47.31

Table 1. Classfication accuracies and average search cost for RMI-NAS, and other NAS algorithms on NAS-Bench-201 [16]. RS denotes random search and RL denotes reinforcement learning, each of them is tested by sampling 200 architectures following [55]. We use horizontal lines to divide our compared methods from top to bottom into training based, one-shot based and training-free based methods.

accurately, thus achieving better performance.

Results on the DARTS Search Space. In practice, we follow the settings in previous work, using the same operation set but shrink the search space by half for simplicity. In other words, the same architecture is used for both normal cell and reduction cell. With the help of random forest, we can still obtain comparable accuracy but deliver substantial improvements on time consumption, as shown in Table 2. Considering the randomness of neural network training, we also retrain baseline architectures under exactly the same setting and report the corresponding results using NAS-Bench-301 [46] for fair comparison. As we can see, the searched architectures show varied performance even with different random seeds. We thus conclude that RMI-NAS achieves comparable or better performance with the minimum search cost. The same results are observed in ImageNet, which is reported in supplementary materials.

4.2. Ablation Study

In this section we conduct ablation experiments to validate the efficiency of each component, which is based on the CIFAR-10 dataset with NAS-Bench-201 for simiplicity. Meanwhile, Kendall's $\tau \in [-1, +1]$ [45] is directly adopted as the metric following previous works [61,62], which measure the ordinal association of architecture performance between the ground truth and the estimation. In particular, a large Kendall's τ means the estimation method is highly correlated to the ground truth, and vice versa.

Influence of β. We first investigate the effectiveness of RMI score by adjusting β in Eq. 4. In particular, when β is set to 0, the RMI loss degenerates to the cross-entropy loss. As we can see in Fig. 3(a), the RMI loss has the highest kendall's τ when $\beta = 0.8$. Meanwhile, setting $\beta = 1$ only shows a marginal performance drop. On the contrary, using only the classification loss as the measure yields a much lower correlation coefficient of 0.25, which confirms the effectiveness of the proposed RMI score.

Influence of the $\boldsymbol{\alpha}^+$. Since the mutual information with the $\boldsymbol{\alpha}^+$ needs to be calculated in Eq. 4, the selection of the $\boldsymbol{\alpha}^+$ could impact the final performance. To this end, we first select networks with different accuracies as the $\boldsymbol{\alpha}^+$, and calculate the correlation between the RMI score and the accuracy by sampling architectures on NAS-Bench-201. In Fig. 3(b), we set horizontal axis as the accuracy of the different architectures that selected to be $\boldsymbol{\alpha}^+$. We can see that the accuracy is positively correlated to Kendall's τ. This reflects that choosing a predictive accurate architecture is beneficial to performance estimation. Another interesting observation in Fig. 3(b) is that a network architecture with an accuracy greater than 85% is suitable enough to make RMI score a good indicator. In other words, almost all decent human-designed architectures can be selected as $\boldsymbol{\alpha}^+$.

Influence of Optimization Iteration. We select ResNet-20 [22] as $\boldsymbol{\alpha}^+$ and test Kendall's τ of the RMI score using different optimization iterations, as reported in Fig. 3(c). As we can see in the figure, the Kendall's τ is also positively correlated to the optimization iteration. The optimization iteration is set to 150 for the consideration of the trade-off between efficiency and effectiveness.

Generalizability of RMI Score and Efficiency of the Search Algorithm. As mentioned before, the proposed RMI score is an indicator that can be flexibly combined with any search algorithm. Therefore, we conduct experiments combining different search strategies to validated the

Method	Search Cost (GPU-days)	CIFAR-10 Test Err. (%)			Search Method
		NAS-Bench-301	Paper	Retrain	
DenseNet-BC [25]	-	-	3.46	-	Manual
AmoebaNet-B [42]	3150	-	2.55 ± 0.05	-	Evolution
NASNet-A [67]	1800	-	2.65	-	RL
ENAS [41]	0.5	-	2.89	-	RL
DARTS (2nd) [35]	1	5.83	2.76 ± 0.09	2.60	Gradient
SNAS [52]	1.5	6.03	2.85 ± 0.02	2.68	Gradient
GDAS [18]	0.17	5.38	2.93	2.65	Gradient
ASNG-NAS [1]	0.11	-	2.83 ± 0.14	$2.85 \pm 0.12^*$	ASNG
P-DARTS (CIFAR-10) [9]	0.3	5.52	2.50	$2.70 \pm 0.15^*$	Gradient
PC-DARTS (CIFAR-10) [54]	0.1	5.51	2.57 ± 0.07	$2.71 \pm 0.11^*$	Gradient
PARSEC [6]	1	-	2.81 ± 0.03	-	Gradient
GAGE [29]	0.3	5.54	2.50	2.67^*	Gradient
MdeNAS [61]	0.16	5.80	2.55	$2.80 \pm 0.24^*$	MDL
FairDARTS-D [14]	0.4	6.10	2.54 ± 0.05	2.71	Gradient
DARTS- [12]	0.4	5.84	2.59 ± 0.08	2.62	Gradient
SGAS [28]	0.25	6.19	2.66 ± 0.24	2.71	Gradient
Ours	**0.08**	**5.61**	**-**	**2.64 ± 0.04**	**Random Forest**

Table 2. Classfication accuracies and average search cost for RMI-NAS and the other NAS algorithms on DARTS. To make the fair comparison, we retrain all the searched architectures with the released training code [59], where the performance in NAS-Bench-301, original paper are also reported for a better illustration. * denotes the corresponding results are referenced from the open-sourced code [59].

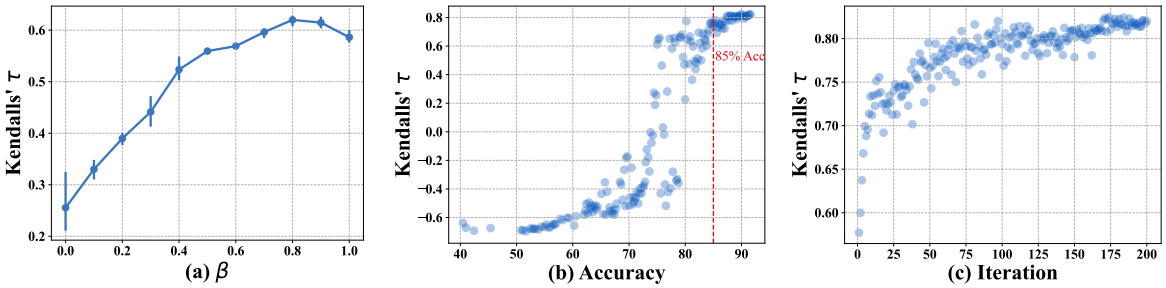

Figure 3. (a) Kendall's τ in different β. In Eq. 4, β controls the ratio of the RMI loss between classfication and RMI score. We repeat experiments over 3 random seeds, where the mean and standard deviation are illustrated in the figure. (b) Kendall's τ using architectures with different accuracy. (c) Kendall's τ employing different optimization iterations in Eq. 4.

effectiveness of RMI and the proposed algorithm. As shown in Fig. 1, integrating with different search algorithms definitely accelerate the search process with better or negligible performance drop. Another observation from Fig. 1 is that the search algorithm further improve the performance with a clearly gap compared to RS, RL and REA, which indicate the efficiency of the proposed search algoritm.

5. Conclusion

In this paper, we propose a new performance estimation strategy incorprated with a novel search algorithm. In particular, we find that representation mutual information is an effective and efficienct indicator to estimate architectures. Then, we propose a new search algorithm to further accelerate the search process. Meanwhile, the searched solution is also mathematically guaranteed by the proposed Theorem 1. Extensive experiments on various search spaces demonstrate the effectiveness in accelerating the search process and searching better architecture.

Acknwoledgement. This work was supported by the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No. U21B2037, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, and No. 62002305), Guangdong Basic and Applied Basic Research Foundation (No.2019B1515120049), the Natural Science Foundation of Fujian Province of China (No.2021J01002) and CAAI-Huawei MindSpore Open Fund. Thanks for the support provided by OpenI Community https//git.openi.org.cn.

References

[1] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *ICML*, 2019. 8

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv*, 2016. 1

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 2

[4] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *AAAI*, 2018. 2

[5] Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *arXiv*, 2019. 1

[6] Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. Probabilistic neural architecture search. *arXiv*, 2019. 8

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE TPAMI*, 40(4):834–848, 2017. 1

[8] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *ICLR*, 2020. 1, 2, 3

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *arXiv*, 2019. 8

[10] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *NeurIPS*, pages 6638–6648, 2019. 1

[11] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv*, 2017. 5

[12] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: robustly stepping out of performance collapse without indicators. *arXiv*, 2020. 8

[13] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *ICCV*, pages 12239–12248, 2021. 7

[14] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *ECCV*, pages 465–480. Springer, 2020. 8

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009. 3, 5

[16] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2019. 2, 3, 4, 5, 6, 7

[17] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *ICCV*, pages 3681–3690, 2019. 7

[18] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 2019. 7, 8

[19] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *JMLR*, 2019. 1, 3

[20] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*, pages 1437–1446. PMLR, 2018. 1

[21] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *ICALT*, pages 63–77. Springer, 2005. 4

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 7

[23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv*, 2017. 2

[24] Jie Hu, Rongrong Ji, ShengChuan Zhang, Xiaoshuai Sun, Qixiang Ye, Chia-Wen Lin, and Qi Tian. Information competing process for learning diversified representations. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[25] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 8

[26] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, pages 3519–3529. PMLR, 2019. 4

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 3, 5

[28] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *CVPR*, pages 1620–1630, 2020. 8

[29] Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search. *arXiv*, 2020. 8

[30] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv*, 2019. 3

[31] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance image recognition. In *ICCV*, pages 347–356, 2021. 1, 2, 3

[32] Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *arXiv*, 2019. 5

[33] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019. 1

[34] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 2, 3

[35] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 1, 2, 3, 6, 7, 8

[36] Urbano Lorenzo-Seva and Jos MF Ten Berge. Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2(2):57–64, 2006. 4

[37] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. Semi-supervised neural architecture search. In *NeurIPS*, 2020. 1

[38] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, 2018. 1

[39] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *ICML*, pages 7588–7598. PMLR, 2021. 1, 2, 3, 6, 7

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011. 6

[41] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv*, 2018. 7, 8

[42] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv*, 2018. 1, 2, 3, 8

[43] Paul Robert and Yves Escoufier. A unifying tool for linear multivariate statistical methods: The rv-coefficient. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 25(3):257–265, 1976. 4

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 2

[45] Pranab Kumar Sen. Estimates of the regression coefficient based on kendall's tau. *Journal of the American Statistical Association*, 63(324):1379–1389, 1968. 7

[46] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv*, 2020. 7

[47] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *NeurIPS*, 2020. 7

[48] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *CVPR*, pages 12965–12974, 2020. 1, 2

[49] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *ECCV*, pages 660–676. Springer, 2020. 1

[50] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, pages 10734–10742, 2019. 3

[51] Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, 2017. 1

[52] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search. *arXiv*, 2018. 8

[53] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *CVPR*, pages 6649–6658, 2019. 1

[54] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *arXiv*, 2019. 1, 2, 3, 8

[55] Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. Does unsupervised architecture representation learning help neural architecture search? In *NeurIPS*, 2020. 7

[56] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. Cars: Continuous evolution for efficient neural architecture search. In *CVPR*, pages 1829–1838, 2020. 1

[57] Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *arXiv*, 2018. 1

[58] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *CVPR*, pages 7809–7818, 2020. 1

[59] Xiawu Zheng, Rongrong Ji, Yuhang Chen, Qiang Wang, Baochang Zhang, Jie Chen, Qixiang Ye, Feiyue Huang, and Yonghong Tian. Migo-nas: Towards fast and generalizable neural architecture search. *IEEE TPAMI*, 43(9):2936–2952, 2021. 1, 2, 7, 8

[60] Xiawu Zheng, Rongrong Ji, Lang Tang, Yan Wan, Baochang Zhang, Yongjian Wu, Yunsheng Wu, and Ling Shao. Dynamic distribution pruning for efficient network architecture search. *arXiv*, 2019. 1

[61] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *ICCV*, 2019. 1, 7, 8

[62] Xiawu Zheng, Rongrong Ji, Qiang Wang, Qixiang Ye, Zhenguo Li, Yonghong Tian, and Qi Tian. Rethinking performance estimation in neural architecture search. In *CVPR*, 2020. 1, 7

[63] Xiawu Zheng, Yuexiao Ma, Teng Xi, Gang Zhang, Errui Ding, Yuchao Li, Jie Chen, Yonghong Tian, and Rongrong Ji. An information theory-inspired strategy for automatic network pruning. *arXiv*, 2021. 4

[64] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *CVPR*, pages 11396–11404, 2020. 1

[65] Qinqin Zhou, Xiawu Zheng, Liujuan Cao, Bineng Zhong, Teng Xi, Gang Zhang, Errui Ding, Mingliang Xu, and Rongrong Ji. Ec-darts: Inducing equalized and consistent optimization into darts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11986–11995, 2021. 1

[66] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv*, 2016. 2, 3

[67] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1, 3, 8