Neural Architecture Search with Representation Mutual Information

Xiawu Zheng\textsuperscript{1,2,}\textdagger\textasteriskcentered, Xiang Fei\textsuperscript{1,}\textdagger\textasteriskcentered, Lei Zhang\textsuperscript{1,}\textdagger, Chenglin Wu\textsuperscript{3}, Fei Chao\textsuperscript{1}, Jianzhuang Liu\textsuperscript{4}, Wei Zeng\textsuperscript{2,5}, Yonghong Tian\textsuperscript{2,5}, Rongrong Ji\textsuperscript{1,2,6,7,}\textasteriskcentered

\textsuperscript{1}Media Analytics and Computing Lab, Department of Artificial Intelligence, School of Informatics, Xiamen University. \textsuperscript{2}Peng Cheng Laboratory. \textsuperscript{3}DeepWisdom Inc. \textsuperscript{4}Huawei Noah’s Ark Lab. \textsuperscript{5}National Engineering Research Center for Visual Technology, School of Computer Science, Peking University. \textsuperscript{6}Institute of Artificial Intelligence, Xiamen University. \textsuperscript{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 and accuracy predictor based methods\textsuperscript{*} [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

\textsuperscript{*}These methods use different training hyper-parameters for acceleration, including but not limited to fewer epochs, subset of data and down-scaled 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 incorporating RMI, we can largely accelerate NAS methods including reinforcement learning (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

\[ \text{Extensive experiments in Sec. 4 show that RMI shows a high correlation with the performance of the architectures in the search space.} \]
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] searches 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., \( \mathbf{x}, \mathbf{y} \)), matrices or tensors (e.g., \( \mathbf{X}, \mathbf{Y} \)). 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. \( \alpha \in \mathcal{A} \in \mathbb{R}^{N \times M} \) denotes an indicator vector, where \( N \) and \( M \) are the numbers of edges and operations, respectively. We use the superscript of \( \alpha^a \) to represent a specific architecture \( a \) and the subscript of \( \alpha_i \) as a certain edge \( i \). \( \mathbf{W} \in \mathcal{W} \) denotes the architecture weights.

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

\[
\max_{\mathbf{W} \in \mathcal{W}, \alpha \in \mathcal{A}} \mathcal{L} (\mathbf{W}, \alpha), \quad \text{s.t.} \quad T(\alpha) < \Omega. \tag{1}
\]

\( \mathcal{L} : \mathcal{W} \times \mathcal{A} \rightarrow \mathbb{R} \) is the loss function that is not differentiable with respect to \( \alpha \), and is differentiable with respect to \( \mathbf{W} \). \( T(\cdot) \) is a function that denotes the constraints for architectures, such as FLOPs, and \( \Omega \) 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 \( \alpha^+ \) that has a decent accuracy, we use \( X^{1+}, \ldots, X^{L+} \) to represent the random variables of feature maps in each layer. For any architecture \( \alpha \) that is sampled from the search space, we formally define the RMI score as

\[
\phi(\alpha^+, \alpha) = \sum_{i=1}^{L} I (X^{i+}, X^i). \tag{2}
\]

In general, an architecture that has a high RMI score tends to be a good architecture, as the proposed RMI score...
is robust, effective and efficient, which is also demonstrated in Sec. 4. More specifically, the RMI score shows a high correlation with different $\alpha^+$ 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 $D := \{(x_1, y_1), \ldots, (x_n, y_n)\}$ contain $n$ i.i.d. samples drawn from the distribution $P_{XY}$. In this case, the normalized HSIC \[21, 26, 63\] is defined as

$$ I(X, Y) \approx n \text{HSIC}_{\text{norm}}(X, Y) = \frac{||Y^T X||_F^2}{||X^T X||_F ||Y^T Y||_F}, $$

(3)

where $|| \cdot ||_F$ is the Frobenius norm or the Hilbert-Schmidt norm. Calculating 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 $\gamma$ and randomly select data with $\gamma$ times of the batch size as a data pool. The pretrained architecture $\alpha^+$ is then used as a teacher network to select $\gamma_{\text{batch}}$ samples with the largest classification confidences from it. Subsequently, the samples are used to obtain the feature maps $X^+$ and $X$ of each stage/layer in $\alpha^+$ and $\alpha$, respectively. The RMI loss and classification error are optimized simultaneously, which is formally defined as

$$ L_{\text{loss}} = \beta L_{\text{RMI}} + (1 - \beta)L_{\text{cls}} $$

$$ = \beta \sum_{i=1}^{T} \frac{||X^{i+T} X^+||_F^2}{||X^{i+T} X^+||_F} + (1 - \beta)L_{\text{cls}}. $$

(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
the samples. In general, RMI-NAS consists of three steps: random forest training, expanded sampling and architecture selection, which are elaborated 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 $\tau$ 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 $\alpha$ as a one-hot vector. In this case, $\alpha$ 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 $n_{\text{filing}}$ 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 $\tau$, and when $n_{\text{collect}}$ good architectures are collected, the update step of the random forest is terminated.

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

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

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

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

The proof is provided in our supplementary materials for a better understanding. According to Theorem 1, we can employ the distribution $P(\alpha|f(\alpha) + \sigma \epsilon > \tau)$ to find the optimal solution. However, this distribution is intractable in practice. Fortunately, we have effectively explored the search space and collected $n_{\text{collect}}$ architectures whose loss values are smaller than threshold $\tau$. Therefore, we can obtain the final architecture accordingly. In particular, for the collected architecture set $A_{\text{collect}}$, 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

$$\alpha^*_i = \text{Mo}(A_{\text{collect}}),$$

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 calculation of $I(X, Y)$ is effective, i.e., $O(n^2)$, 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
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 in the data pool as described in Sec. 4.1;

\begin{algorithm}
\textbf{input}: Sampling Number \(n_{\text{warm}}, n_{\text{fitting}}, n_{\text{collect}}\);
\text{Quantile} \(k \in (0, 100)\); Architecture Search Space \(\mathcal{A}\); Random Forest Model \(\pi_\theta: \mathcal{A} \rightarrow \{0, 1\}\).
\textbf{output}: Searched Architecture \(\alpha^*\)
Initial sampling set \(D = \emptyset\); \(D_{\text{warm}} = \emptyset\);
Pick preferred batch data as described in Sec. 4.1;
for \(i \leftarrow 1\) to \(n_{\text{warm}}\) do
\(\alpha' \leftarrow \text{Sample an architecture from } \mathcal{A}\);
\(l' \leftarrow \text{Optimize } \omega_{\alpha'}\text{ by Eq. 4};\)
\(D_{\text{warm}} \leftarrow D_{\text{warm}} \cup \{(\alpha', l')\};\)
end
\(\tau \leftarrow \text{Get top } k\%\text{ quantile in } D_{\text{warm}};\)
\(z^i = \mathbb{I}[l' \leq \tau]\) for \(i = 1, \ldots, n_{\text{collect}};\)
\(D \leftarrow D \cup \{(\alpha^i, z^i)\}_{i = 1}^{n_{\text{collect}}};\)
\(\pi_\theta \leftarrow \text{Fit } \pi_\theta\text{ by using } D;\)
i \(\leftarrow n_{\text{warm}} + 1;\)
while \(i \leq n_{\text{collect}}\) do
\(\mathcal{A}_{\text{fitting}} \leftarrow \text{Sample } n_{\text{fitting}}\text{ architectures};\)
\(\alpha' \leftarrow \text{arg max}_{\alpha' \in \mathcal{A}_{\text{fitting}}} \pi_\theta(\mathcal{A}_{\text{fitting}});\)
\(l' \leftarrow \text{Optimize } \alpha'\text{ by Eq. 4};\)
\(z^i = \mathbb{I}[l' \leq \tau];\)
\(D \leftarrow D \cup \{(\alpha', z^i)\};\)
\(\pi_\theta \leftarrow \text{Fit } \pi_\theta\text{ by using } D;\)
i \(\leftarrow i + 1;\)
end
Obtain the final architecture \(\alpha^*\) through Eq. 6;
\end{algorithm}

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.
Table 1. Classification accuracies and average search cost for RMI-NAS, and other NAS algorithms on NAS-Bench-201 [16]. RS 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Search Cost (Seconds)</th>
<th>CIFAR-10 Valid</th>
<th>CIFAR-10 Test</th>
<th>CIFAR-100 Valid</th>
<th>CIFAR-100 Test</th>
<th>ImageNet16-120 Valid</th>
<th>ImageNet16-120 Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet [22]</td>
<td>-</td>
<td>90.83</td>
<td>93.97</td>
<td>70.42</td>
<td>70.86</td>
<td>44.53</td>
<td>43.63</td>
</tr>
<tr>
<td>RS</td>
<td>22993.93</td>
<td>90.93 ± 0.36</td>
<td>93.80 ± 0.36</td>
<td>70.93 ± 1.09</td>
<td>71.04 ± 1.07</td>
<td>44.45 ± 1.10</td>
<td>44.57 ± 1.25</td>
</tr>
<tr>
<td>RL</td>
<td>27870.91</td>
<td>91.09 ± 0.37</td>
<td>93.85 ± 0.37</td>
<td>71.61 ± 1.12</td>
<td>71.71 ± 1.09</td>
<td>45.05 ± 1.02</td>
<td>45.24 ± 1.18</td>
</tr>
<tr>
<td>ENAS [41]</td>
<td>14058.8</td>
<td>37.51 ± 3.19</td>
<td>53.89 ± 0.58</td>
<td>13.37 ± 2.35</td>
<td>13.96 ± 2.33</td>
<td>15.06 ± 1.95</td>
<td>14.84 ± 2.10</td>
</tr>
<tr>
<td>DARTS-V2 [35]</td>
<td>35781.80</td>
<td>39.77 ± 0.00</td>
<td>54.30 ± 0.00</td>
<td>15.03 ± 0.00</td>
<td>15.61 ± 0.00</td>
<td>16.43 ± 0.00</td>
<td>16.32 ± 0.00</td>
</tr>
<tr>
<td>SETN [17]</td>
<td>34139.53</td>
<td>84.04 ± 0.28</td>
<td>87.64 ± 0.00</td>
<td>58.86 ± 0.06</td>
<td>59.05 ± 0.24</td>
<td>33.06 ± 0.02</td>
<td>32.52 ± 0.21</td>
</tr>
<tr>
<td>GDAS [18]</td>
<td>31609.89</td>
<td>89.89 ± 0.08</td>
<td>93.61 ± 0.09</td>
<td>71.34 ± 0.04</td>
<td>70.70 ± 0.30</td>
<td>41.59 ± 1.33</td>
<td>41.71 ± 0.98</td>
</tr>
<tr>
<td>FairNAS [13]</td>
<td>9845.00</td>
<td>90.97 ± 0.57</td>
<td>93.23 ± 0.18</td>
<td>70.94 ± 0.94</td>
<td>71.00 ± 1.46</td>
<td>41.09 ± 1.00</td>
<td>42.19 ± 0.31</td>
</tr>
<tr>
<td>MDENAS [61]</td>
<td>5300.00</td>
<td>-</td>
<td>89.07 ± 0.60</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MIGNONAS [59]</td>
<td>3200.00</td>
<td>-</td>
<td>93.15 ± 0.18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Jacob cov [39]</td>
<td>-</td>
<td>89.69 ± 0.73</td>
<td>92.96 ± 0.80</td>
<td>69.87 ± 1.22</td>
<td>70.03 ± 1.16</td>
<td>43.99 ± 2.05</td>
<td>44.43 ± 2.07</td>
</tr>
<tr>
<td>Mag [47]</td>
<td>-</td>
<td>89.94 ± 0.34</td>
<td>93.35 ± 0.04</td>
<td>70.18 ± 0.66</td>
<td>70.47 ± 0.18</td>
<td>42.57 ± 2.14</td>
<td>43.17 ± 2.57</td>
</tr>
</tbody>
</table>

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 simplicity. Meanwhile, Kendall’s τ ∈ [−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 β = 0.8. Meanwhile, setting β = 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 α+. Since the mutual information with the α+ needs to be calculated in Eq. 4, the selection of the α+ could impact the final performance. To this end, we first select networks with different accuracies as the α+, 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 α+. 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 α+

Influence of Optimization Iteration. We select ResNet-20 [22] as α+ 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
Table 2. Classification 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].

![Kendall's τ](image)

**Figure 3.** (a) Kendall’s τ in different β. In Eq. 4, β controls the ratio of the RMI loss between classification 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 algorithm.

5. Conclusion

In this paper, we propose a new performance estimation strategy incorporated with a novel search algorithm. In particular, we find that representation mutual information is an effective and efficient 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.

Acknowledgement. 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](https://git.openi.org.cn).
References


[31] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, and Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv, 2017. 2


[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
