PDNAS: On the Predictability of Supernet for Shrinking Architecture Search Space

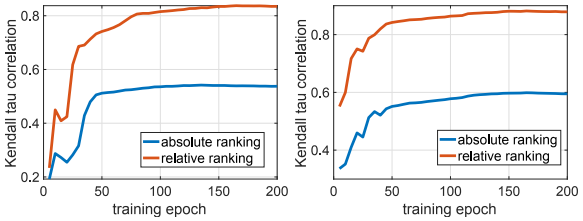
Lujun Li[1,2], Shan You[3,4]*, Fei Wang[5], Chen Qian[3], Changshui Zhang[4], Xiaogang Wang[3,6], Qingyi Gu[1]
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]SenseTime Research [4]Department of Automation, Tsinghua University
Institute for Artificial Intelligence, Tsinghua University (THUAI)
Beijing National Research Center for Information Science and Technology (BNRist)
[5]University of Science and Technology of China [6]Chinese University of Hong Kong
{lilujun2019,qingyi.gu}@ia.ac.cn, {youshan,qianchen}@sensetime.com
wangfei91@mail.ustc.edu.cn, zcs@mail.tsinghua.edu.cn, xgwang@ee.cuhk.edu.hk

Abstract

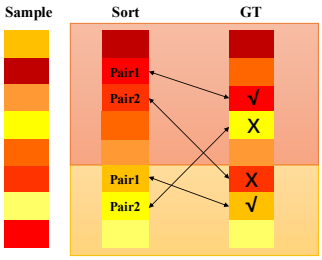
*Huge search space challenges the evaluation ability of supernet in one-shot neural architecture search (NAS) methods, and limits the searching performance accordingly. In this paper, we re-examine the role of supernet, and highlight its relative ranking ability (*i.e.*, predictability) during training. Based on this, we can leverage the supernet to distinguish good and weak paths, and shrink the search space by filtering those weak paths. In this way, we introduce a path discriminator (PD) during training, which enables to inherit the predictability of supernet to identify the paths. Thus the PD acts as a binary classifier, and can be jointly trained with the supernet. In this way, the search space can be continuously shrunk, and the target shrunk space will be indicated by the last PD. We implement our method PDNAS on the ImageNet dataset, and results show that PDNAS can achieve better Top-1 accuracy* 74.9% *on ShuffleNetV2 search space and* 77.5% *on SE-expanded search space. Besides, we also conduct experiments on NAS-Bench-201 dataset to investigate the effectiveness of our introduced PD in shrinking search space and boosting the searching performance. Code is available at: PDNAS-Series.*

1. Introduction

Early design of deep neural networks (DNNs) is mostly subject to human experience and requires much manual effort. Neural architecture search (NAS) thus aims at liberating human labor and automatically probing more promising structures in a data-driven manner. Pioneer work NASNet [39] used LSTM to represent network structures, and searched via reinforcement learning (RL) for 22,400 GPU-hours. For boosting searching efficiency, ENAS [20] and the following differentiable NAS methods (*e.g.*, DARTS [17], ISTA-NAS [34] and EnTranNAS [35]) adopt a weight sharing paradigm to reduce resource consumption. Recent one-shot methods (*e.g.*, SPOS [11], GreedyNAS [37], MCT-NAS [22], K-shot-NAS [26] and ViTAS [25]) leverage a supernet (group) for directly sharing weights and have



(a) CIFAR-10 (b) CIFAR-100



(c) Ranking and labeling

Figure 1: Relative correlation in supernet training. We uniformly sample 100 models from NAS-Bench-201 [7] and calculate for the absolute ranking and relative ranking. See Appendix B in supplementary materials for more details.

*Corresponding author.

achieved remarkable performance. Besides, hard constraint on the architectures can be easily integrated during searching, such as FLOPs and latency.

In one-shot NAS methods, the role of a supernet mainly comes at two folds. First, it indicates the search space since all valid architectures (paths) are derived as its subnetworks. Second, it is regarded as a performance evaluator, which enables to give accurate ranking over all paths. However, the volume of paths in supernet is rather huge. Thus it brings tough challenge for various searching algorithms (*e.g.*, random or evolutionary search), and sub-optimal results are expected accordingly. On the other hand, the huge search space is usually mingled with many weak paths. Under the context of weight sharing, training these weak paths does no good to the searching of optimal architectures, and affects training efficiency as well [37, 23, 13].

To improve the searching performance, we begin with re-examining the role of supernet. The supernet is supposed to provide accurate ranking over all paths after it is trained well. Nevertheless, besides this absolute ranking ability, we highlight its relative ranking ability, *i.e.*, *predictability*, which can be held through the whole training process. Concretely, during its training, though the accurate ranking can not be ensured over all paths, the supernet can still be predictive to identify whether a path is a good one or not. This predictability intuition is validated by the observations on the NAS-Bench-201 dataset [7]. As Figure 1 shows, though the correlation of absolute ranking is subtle during training, the relative ranking tends to be significant. Since the supernet is capable of identifying good paths to a great extent, it thus inspires us that we can leverage this predictability to block out the weak paths during training. The search space thus gets shrunk, and evolves towards a smaller space composed of actually good paths.

However, it is computationally unbearable if we directly use the supernet to examine every single path during training. Therefore, we introduce a path discriminator (PD) to encode and inherit the predictability of supernet. The PD receives architecture representations as input, and outputs a binary label as prediction about whether a path is good (+1) or weak (-1). The ground-truth labels of paths are supplied by the supernet, which are determined by the relative ranking results as Figure 1. Note that this labeling cost will hamper the searching efficiency if the quantity of labeled paths is too large. Thus we implement path evaluation only on a surrogate subset of validation dataset and exploit path augmentation as unlabeled data for PD. In this way, the PD can be trained as a binary classifier; besides, to better grasp the predictability of supernet, we incorporate a ranking loss for PD's training.

Briefly, our method PDNAS (PreDictive NAS with Path Discriminator) leverages the introduced PD to shrink the search space along with the training of supernet. In addition, to further narrow the search space and filter the weak paths, we encourage a continuous space shrinkage. After a PD is trained, the supernet will only get trained on the good paths predicted by this PD till the next PD is to be trained. In this way, both the supernet and PD get evolved towards a continuously reduced search space. And the optimal architecture will be searched on the smallest search space indicated by the last PD, by dint of random [32] or evolutionary search [27, 37, 24].

We conducted extensive experiments on the large-scale ImageNet dataset to verify the superiority of our proposed method. The results show that our PDNAS can achieve better classification accuracy under similar FLOPs level in the same ShuffleNetV2 search space. Besides, by switching to a larger space, our PDNAS can obtain new state-of-the-art architectures. For example, in the same MoblieNetV2 search space, we have a significant improvement of 1.3% Top-1 accuracy over SPOS [11] under the same training settings. And in the expanded search space, we can achieve 77.2% Top-1 accuracy with only 366M FLOPs. We also conducted comprehensive ablation studies on the recent NAS-Bench-201 dataset to investigate how our method can make use of predictability of supernet to boost the searching performance.

2. Revisiting One-shot NAS Methods

A NAS search space $\mathcal{A}$ can be represented by a directed acyclic graph (DAG) consisting of an order sequence of nodes, each node is denoted as a feature map and each directed edge between nodes indicates an operation transforms the feature. The purpose of NAS is to find a optimal operation o among candidate operations $\mathcal{O}$ such as convolution, pooling, identity or different types of building blocks.

In one-shot NAS, the whole search space can be modelled into an over-parameterized network $\mathcal{N}$ (i.e., supernet) with weights $W_{\mathcal{A}}$, and each network architecture $a \in \mathcal{A}$ is a subgraph of supernet, denoted as $\mathcal{N}(a, W_{\mathcal{A}}(a))$. The search process is divided into two stages: supernet training and architecture search. It first trains the supernet by sampling and training one single path each iteration, *i.e.*,

$$\mathcal{W}_{\mathcal{A}}^{*} = \arg\min_{\mathcal{W}_{\mathcal{A}}} \mathop{\mathbb{E}}_{\boldsymbol{a}\sim U(\mathcal{A})} [\![\mathcal{L}_{train}(\mathcal{N}(a, W_{\mathcal{A}}(a)))]\!]. \quad (1)$$

During search stage, the aim is to find the optimal architecture a^* by evaluating paths using the trained supernet $\mathcal{N}(W_{\mathcal{A}}^{*})$:

$$a^{*} = \operatorname*{argmax}_{a\in\mathcal{A}} \mathrm{ACC}_{\mathrm{val}}\left(\mathcal{N}\left(a, W_{\mathcal{A}}^{*}(a)\right)\right) \quad (2)$$

However, the weights in different architectures are highly coupled, which makes the supernet hard to estimate each architecture accurately. In this work, we aim to shrink the search space into those potentially-good ones by a path discriminator.

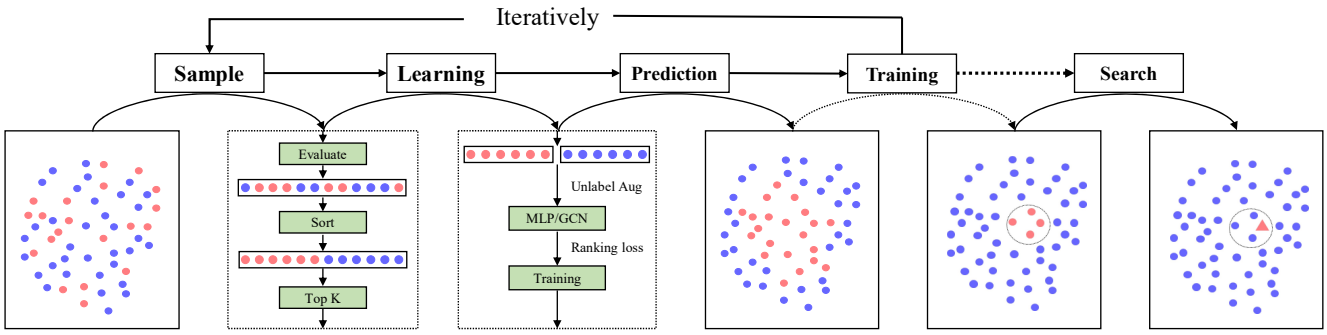


Figure 2: Framework of PDNAS by training the supernet via PDs. In each space shrinkage stage, some paths are first sampled, evaluated and sorted based on the previous PD. Then, we train the PD using ranking loss and unlabeled path augmentation. Finally, the PD is used to discriminate and filter the weak paths to continuously shrink the search space.

3. PDNAS: Predictive NAS with Path Discriminator

Basically, the one-shot supernet indicates the search space, where all architectures (paths) are just derived and evaluated by inheriting the weights. However, the volume of paths inside can be considerably huge, *e.g.*, 7^{21}. This huge search space induces two main issues. First, the ability of evaluating different paths is weakened as the correlation of ranking of paths between their ground-truth performance and that on the supernet is usually poor [1]. Second, various searching algorithms (*e.g.*, random or evolutionary search) are limited in such a huge space, and usually only search sub-optimal results [15]. In this way, the intuition is to explicitly shrink the search space. Besides, since the aim of NAS is to probe promising architectures; we expect the shrunk space is composed of actually competitive paths with those weak paths already filtered.

3.1. Estimating Space Shrinkage with PD

Formally, we partition the original search space $\mathcal{A}$ into N complete sub-spaces as

$$\begin{aligned} \mathcal{A} &= \mathcal{A}_1 \cup \mathcal{A}_2 \cup \cdots \cup \mathcal{A}_{N-1} \cup \mathcal{A}_N, \\ \text{s.t.}\ \forall\, i < j,\ &\mathcal{A}_i \cap \mathcal{A}_j = \emptyset,\ \mathcal{A}_i \preccurlyeq \mathcal{A}_j \end{aligned} \tag{3}$$

where $\preccurlyeq$ is a partial order defined on $\mathcal{A}$, which means that $\boldsymbol{b} \in \mathcal{A}$ has better performance (*e.g.*, ground-truth accuracy) than $\boldsymbol{a} \in \mathcal{A}$ if $\boldsymbol{a} \preccurlyeq \boldsymbol{b}$ holds. Then for a shrunk space, a good option would refer to $\mathcal{A}_N$ since its paths have better performance than any other sub-spaces. However, this oracle partition is unknown during searching. Note that the search space is exactly indicated by the supernet, and the supernet is supposed to make ranking over paths. As Figure 1 we have the observations that during the training of supernet, though the correlation of absolute ranking can not be ensured, the relative ranking by supernet usually has strong correlation with that of ground-truth. In other words, the supernet can be leveraged to predict whether a path is good or weak. For example, we can use the supernet to predict whether a path $\boldsymbol{a}$ is from $\mathcal{A}_1$ or not since the correlation of relative ranking is high. [1]

Concretely, given the supernet $\mathcal{N}$ with weights W, if all paths are ranked according to the performance on the supernet, then we can leverage a threshold $r \in (0, 1)$ to label the top-est r ratio of paths as $+1$ denoted by $\mathcal{P}^+$, and the rest of them as -1 also denoted by $\mathcal{P}^-$. Then we have high confidence that

$$\mathcal{A} = \mathcal{P}^+ \cup \mathcal{P}^-, \quad \mathcal{P}^+ \approx \bigcup_{i=2}^{N} \mathcal{A}_i, \quad \mathcal{P}^- \approx \mathcal{A}_1. \tag{4}$$

Ideally, we can simply block the training of $\mathcal{P}^-$, and continue the supernet training on the $\mathcal{P}^+$. The search space is thus shrunk from $\mathcal{A}$ into $\bigcup_{i=2}^{N} \mathcal{A}_i$, then the ability of absolute ranking of paths is also supposed to be improved for supernet since the space has been reduced. In fact, with the new search space $\bigcup_{i=2}^{N} \mathcal{A}_i$, we can repeat the shrinkage process above, thus the space can be continuously shrunk in a chain till we reach our target space $\mathcal{A}_N$.

However, the space shrinkage above is up to the partition $\mathcal{P}^+ \cup \mathcal{P}^-$, which needs to traverse all paths in supernet and is unbearable due to the huge scale of $\mathcal{A}$. Inspired by the data-free knowledge distillation [18] and literatures on the network predictors [29][10], the architecture itself can have informative knowledge benefiting the identifying whether a path is from $\mathcal{P}^+$ or $\mathcal{P}^-$ given a supernet. In this way, we introduce a **path discriminator** (PD), $\mathcal{D} : f(\boldsymbol{a}; W_{\mathcal{A}}) \rightarrow \{+1, -1\}$, to directly predict the labels of paths, with $+1$ for $\mathcal{P}^+$ and -1 for $\mathcal{P}^-$, respectively. Besides, due to the generalization ability of path features [38], learning of such a PD will not necessarily traverse all paths to query the label, but only needs a training dataset of paths

[1] Admittedly, we can also leverage the predictability of supernet to predict whether a path $\boldsymbol{a}$ is from the good $\mathcal{A}_N$ or not. Nevertheless, since we aim to shrink the space for NAS, it is more sensible to filter those weak paths.

and is learned under the typical empirical risk minimization (ERM) framework.

With a learned PD on the space $\mathcal{A} = (\bigcup_{i=2}^{N} \mathcal{A}_i) \cup \mathcal{A}_1 = \mathcal{P}^+ \cup \mathcal{P}^-$, we can use it to estimate the $\mathcal{P}^+$ and $\mathcal{P}^-$, *i.e.*,

$$\mathcal{P}^+ \approx \mathcal{A}_{\mathcal{D}}^+ := \{\boldsymbol{a} \in \mathcal{A} | f(\boldsymbol{a}) = +1\}, \tag{5}$$

$$\mathcal{P}^- \approx \mathcal{A}_{\mathcal{D}}^- := \{\boldsymbol{a} \in \mathcal{A} | f(\boldsymbol{a}) = -1\}. \tag{6}$$

Then combining Eq.(4) and Eq.(5), the PD is also regarded as a good indicator for the space $\mathcal{A}_1$. As a result, we can use PD to shrink the space since it inherits the predictability of the supernet. Then the supernet will simply get trained on the space $\bigcup_{i=2}^{N} \mathcal{A}_i \approx \mathcal{P}^+ \approx \mathcal{A}_{\mathcal{D}}^+$, *i.e.*,

$$W_{\mathcal{A}}^*(\mathcal{D}) = \arg\min_{W_{\mathcal{A}}} \mathop{\mathbb{E}}_{\boldsymbol{a} \sim U(\mathcal{A}_{\mathcal{D}}^+)} [\![\mathcal{L}_{train}(\boldsymbol{w}_{\boldsymbol{a}})]\!]. \tag{7}$$

Besides, the search space can be further shrunk by repeating the shrinkage process as previously, and the PD will get evolved by learning on the reduced space $\bigcup_{i=2}^{N} \mathcal{A}_i \approx \mathcal{P}^+ \approx \mathcal{A}_{\mathcal{D}}^+$, *i.e.*,

$$W_{\mathcal{D}}^*(\mathcal{A}) = \arg\min_{W_{\mathcal{D}}} \mathop{\mathbb{E}}_{\boldsymbol{a} \sim U(\mathcal{A}_{\mathcal{D}}^+)} [\![\mathcal{L}_{\mathcal{D}}(\boldsymbol{a}, y_{\boldsymbol{a}}(W_{\mathcal{A}}))]\!], \tag{8}$$

where $W_{\mathcal{D}}$ is the parameters of the PD, $\mathcal{L}_{\mathcal{D}}$ is the loss function, and $y_{\boldsymbol{a}}(W_{\mathcal{A}})$ is the queried label of path $\boldsymbol{a}$ by the supernet. We will elaborate the details of how to train the PD in the next subsection.

Stochastic space shrinkage. As illustrated previously, the search space will be ideally shrunk by the introduced PD if the supernet can ideally reflect the space partition as Eq.(4) and the PD can ideally inherit the discrimnation ablity of supernet as Eq.(5). Though the approximation holds excellently in practice (*e.g.*, relative ranking correlation of supernet is 0.8+ and binary classification of PD is 95%+), it is still at the risk that the Oracle optimal paths might get lost during shrinking the space. In this way, we encourage a stochastic space shrinkage manner that also samples paths randomly from the entire space $\mathcal{A}$ with a small probability [37] such as 0.1. Then the sampling in Eq.(7) and (8) is extended to

$$\boldsymbol{a} \sim p(\mathcal{A}, \mathcal{D}) = (1 - \varepsilon) \cdot U(\mathcal{A}_{\mathcal{D}}^+) + \varepsilon \cdot U(\mathcal{A}), \tag{9}$$

each path is sampled either from the estimated shrunk space $\mathcal{A}_{\mathcal{D}}^+$ with high probability or from the entire space $\mathcal{A}$ occasionally. From this point, we can see a theoretical tradeoff in space shrinkage. For the entire space, though it contains all good paths, however, its size is super huge, which undermines the ranking ability of supernet and is detrimental for searching algorithms to locate some good paths. For a shrunk search space, tough some good paths might not exist in this space, the ranking ability of supernet will be boosted significantly and benefits the searching algorithms to locate promising paths. Our introduced PD balances this tradeoff by leading the search to a good estimated shrunk space with most good paths identified and preserved in the space.

Algorithm 1 Training of supernet with a PD.

```
Warm-up stage: train Supernet via uniform sampling
Shrinkage stage: continuously shrink search space via PD for T times
for i = 1, .., T do
    if i = 1 then
        Sample k paths from original search space
    else
        Sample k paths from the (i−1)-th PD search space as Eq.(9)
    end if
    Evaluate the accuracy of paths on supernet
    Sort path by accuracy and label the top r paths as +1 while the rest as -1
    Train PD via ranking loss and unlabeled paths using Algorithm 2.
    Train Supernet by sampling from the i-th PD search space as Eq.(9)
end for
```

As a result, training of supernet and PD will be implemented in an alternate manner. The search space will be eventually shrunk into the target space $\mathcal{A}_N$ estimated by the final PD $\mathcal{D}^*$ with $\mathcal{A}_N \approx \mathcal{A}_{\mathcal{D}^*}^+$. The framework of using PD to shrink the search space is presented in Algorithm 1. Note that in Figure 1, the correlation of relative ranking is not that significant in the early stage of supernet training. In practice, we propose to first warm up the supernet training with uniform sampling as Eq.(1) for a period of time, so that the predictability of supernet gets strong before we initialize a PD.

3.2. Training of Path Discriminator

Given a supernet, a path discriminator is to predict whether a path is from $\mathcal{P}^+$ or $\mathcal{P}^-$, which serves essentially as a binary classifier. Details of training PD are presented in Algorithm 2.

We first illustrate our used structure for this binary classifier. For the ShuffleNet-like and MobileNet-like search space with M layers and K candidate operators per layer, we use $\{0, 1\}^{M \times K}$ one-hot vectors to encode it and adopt MLP as the backbone network of the binary classifier. While for the NASnet-like search space, the connection relationship of nodes can be expressed by adjacency matrix and feature matrix using a graph convolution network (GCN). Detailed information can be found in the supplementary materials.

Algorithm 2 Training of a PD.

Input: Labeled and unlabeled path dataset A and $\tilde{A}$
Output: A trained PD to discriminate paths

1: Obtain the dataset A including path structure and label
2: Train PD via classification loss $\mathcal{H}(\cdot,\cdot)$ and ranking loss $\mathcal{L}_r$.
3: Augmented dataset $\tilde{A}$ is judged the same by PD according to the similar editing distance.
4: Retrain PD via Eq.(11) $\alpha \cdot \mathcal{L}_{\mathcal{D}}(A) + (1-\alpha) \cdot \mathcal{L}_{\mathcal{D}}(\tilde{A})$

3.2.1 Binary classification with paired ranking loss

To train the PD, we need to collect its training data. To do so, we sample k paths from $\mathcal{A}_{\mathcal{D}}^{+}$[2], which is a reduced space indicated by the previous PD. Then we rank their performance by the supernet on the validation dataset, and specify a threshold $r \in (0,1)$[3] Then we label the top r ratio of paths with $+1$ and the rest with -1. Denote the obtained dataset as $A = \{\boldsymbol{a}_i, y_i\}$. Then the PD can be learned with a binary cross entropy loss. Besides, since the labels are queried by ranking, we also leverage a pair-wise ranking loss to further strengthen the consistence of relative ranking, which tends to benefit the optimization empirically [14].

Concretely, suppose the $+1$ and -1 paths in A are indexed by $\mathcal{I}^+$ and $\mathcal{I}^-$, respectively, with $|\mathcal{I}^+| = k*r$ and $|\mathcal{I}^-| = k*(1-r)$. Then we encourage the prediction scores $s_i = s(\boldsymbol{a}_i)$ for $\mathcal{I}^+$ are much larger than that of $\mathcal{I}^-$ by the following paired ranking loss,

$$\mathcal{L}_{\mathcal{D}}(A) = \frac{1}{k}\sum_{i=1}^{k}\mathcal{H}(\boldsymbol{a}_i, y_i) + \lambda\mathcal{L}_r = \frac{1}{k}\sum_{i=1}^{k}\mathcal{H}(\boldsymbol{a}_i, y_i) - \frac{\lambda}{r(1-r)k^2}\sum_{i\in\mathcal{I}^+, j\in\mathcal{I}^-}\log\left(\frac{\exp(s_i)}{\exp(s_i)+\exp(s_j)}\right), \tag{10}$$

where $\mathcal{H}(\cdot,\cdot)$ is the binary cross entropy loss, and $\lambda > 0$ is a parameter to balance $\mathcal{H}(\cdot,\cdot)$ and ranking loss $\mathcal{L}_r$.

3.2.2 Improving path labeling efficiency

Training the PD needs to query the label for the sampled paths, which requires the supernet to evaluate on the validation dataset (*e.g.*, 50K images on ImageNet). This labeling cost can harm the training efficiency if the volume of paths is large and each path is evaluated on a large dataset as well.

We thus improve the labeling efficiency by targeting at both sides. First, for the evaluation cost, following [37], we use a small subset (*e.g.*, 1K images for ImageNet) of validation dataset to evaluate since the correlation of ranking can be high empirically. As for the volume of labeled paths, inspired by that architectures with similar structures[4] tend to have similar performance [36], we can use structural augmentation to generate unlabeled paths, and use them to boost training in a semi-supervised manner.

Specifically, we use the simple self-training method [31] to fully exploit the path augmentation. First, we train the PD on the labeled dataset $A = \{\boldsymbol{a}_i, y_i\}$ as above. Then we sample paths that lie in a neighborhood of labeled paths under the Edit distance [36]. If a labeled path and its augmented (unlabeled) path correspond to the same prediction of the PD, then the unlabeled path is labeled with the same label as well. Finally, both labeled A and unlabeled $\tilde{A}$ paths are mixed to obtain the data set (A, $\tilde{A}$), so the final training loss for the PD can be expressed as

$$\mathcal{L}_{total} = \alpha \cdot \mathcal{L}_{\mathcal{D}}(A) + (1-\alpha) \cdot \mathcal{L}_{\mathcal{D}}(\tilde{A}), \tag{11}$$

where $\alpha \in [0,1]$ is a trade-off parameter, and $\alpha = 0.5$ suffices in our experiment.

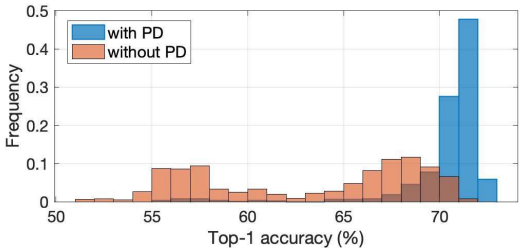


Figure 3: Histogram of accuracy of searched paths on supernet by evolutionary searching method in ImageNet (with or without PD).

3.3. Conditioned Searching with Path Discriminator

After the space shrinkage, target space $\mathcal{A}_N$ will be estimated by the final PD $\mathcal{D}^*$. Then we can implement various searching methods (*e.g.*, random or evolutionary search) on this shrunk space as Eq.(9), *i.e.*,

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}\sim p(\mathcal{A},\mathcal{D}^*)}{\operatorname{argmax}} \operatorname{ACC}_{\text{val}}\left(\mathcal{N}\left(\boldsymbol{a}, W_{\mathcal{A}}(\boldsymbol{a})\right)\right). \tag{12}$$

As Figure 3 shows, searching on this shrunk space (with PD) can significantly boost the searching performance in contrast of that searching on the original huge space (without PD).

4. Experimental Results

In this section, we first perform experiments on ImageNet dataset [6], to verify the superiority of our proposed

[2]For the first PD, we just sample from the complete space $\mathcal{A}$, since the space has not been shrunk yet.

[3]$r = 0.5$ works in our experiments. We empirically find r= 0.5 suffices better from [0.2,0.8]. Too large or small value of thresh-old r may be prone to the class imbalance problem of path labels for PD.

[4]NAS-Bench-101 [36] shows that architectures with Edit distance within 6 have a strong performance correlation.

Table 1: Overall comparison on the ImageNet dataset. In the same search space, our method is significantly better than SPOS with the same training settings. The training and search costs refer to [37], and our method only brings a little training cost and reduces the search cost.

Search space	Method	Top-1 (%)	Top-5 (%)	FLOPs (M)	Params (M)	Search Method	Training cost (GPU days)	Search cost (GPU days)
	MobileNetV2(1.0x) [21]	72.0	91.0	300	3.4	Manual	–	–
	ShuffleNetV2(1.5x) [19]	72.6	90.6	299	3.5	Manual	–	–
MobileNetV2	FairNAS-A [4]	75.3	92.4	365	4.3	EA	10.0	2.0
	FBNet-C [30]	74.9	–	375	5.5	Gradient	9.0	NA
	ProxylessNAS(GPU) [2]	75.1	92.5	465	5.3	Gradient	15	NA
	SPOS [11]	74.8	–	465	–	EA	12.0	<1
	PDNAS(Ours)	76.1	93.1	469	5.9	Shrinking	12.4	<0.25
MobileNetV2+	EfficientNet-B0 [28]	76.3	93.2	390	5.3	RL	–	–
	SCARLET-A [3]	76.9	93.4	365	6.7	EA	10.0	12
	GreedyNAS-A [37]	77.1	93.9	366	6.5	Shrinking	7.0	<1
	PDNAS(Ours)	77.2	94.1	364	6.5	Shrinking	7.0	<0.25
ShuffleNetV2	Random Search	73.8	91.8	323	–	Random	12.0	<1
	DSNAS [12]	74.3	91.9	324	–	Gradient	8.0	NA
	SPOS [11]	74.3	–	319	–	EA	12.0	<1
	PDNAS(Ours)	74.9	92.1	345	3.5	Shrinking	12.4	<0.25
ShuffleNetV2+	SPOS [11]	77.1	93.3	360	6.7	EA	12.0	<1
	PDNAS(Ours)	77.5	93.7	367	6.7	Shrinking	12.4	<0.25

method in both MobileNet-like and ShuffleNet-like search spaces; then we directly implement searching on the NAS-Bench-201 dataset [7] to comprehensively show our effectiveness with the given ground-truth performance of all architectures; we finally present ablation studies to investigate the improvement of our introduced PD.

4.1. Searching on ImageNet Dataset

Search space. We use the same ShuffleNetV2 search space (*i.e.*, supernet) with SPOS [11], which contains 20 layers and 4 building blocks per layer with kernel size $\{3, 5, 7\}$ and an additional Xception-like block, namely `choice_3`, `choice_5`, `choice_7` and `choice_x`. Thus the size of search space amounts to 4^{20}. Besides, we also expand the search space with a Squeeze-and-Excitation (SE) module and the size of search space reaches 8^{20}, which is named ShuffleNetV2+. We also benchmark our experiments on MobileNetV2 search space used in other one-shot NAS methods, such as ProxylessNAS [2] and GreedyNAS [37]. The size of MobileNetV2 space and the corresponding MobileNetV2+ space amounts to 7^{21} and 13^{21}, respectively. Details of supernet structure refer to supplementary materials.

Training and evaluation. We randomly sample 50K images from the original training set as validation set, and use the rest for training. The final accuracy is reported on the original validation set. For ShuffleNetV2 search space, the supernet is trained for 120 epochs with batch size 1024. We first train the supernet using uniform sampling for 20 epochs as warm-up following SPOS. For each epoch, we optimize each sampled path per mini-batch. Then we implement PDNAS to sample and evaluate $k = 1000$ structures every 20 epochs, choose the top 50% $r = 0.5$ architectures as +1 paths via validation set. In this way, the PD is trained for $T = 5$ times using Algorithm 2 with Adam optimizer (decayed by consine schedule). In the searching stage, we use NSGA-II [5] for evolutionary search with population size 50 and generation number 20. Then we select the architecture with highest validation accuracy for retraining from scratch. Other search spaces follow the same training strategy as comparison methods. Details can be found in supplementary materials.

Comparison of results with state-of-the-art methods. First we can see after the last PD is trained, the search space has been shrunk into a small space filled with promising paths, and the searched paths usually have significantly better performance than those by uniform sampling as shown in Figure 3. Under the same search space and training strategy, the performance of our searched models is significantly improved compared with that uniformly-sampled SPOS [11]. For example, for MobileNetV2 search space, as shown in Table 1, our searched architecture reaches 76.1% Top-1 accuracy and improves by 1.3% compared with SPOS under almost the same FLOPs, which is a very significant margin for NAS methods on ImageNet dataset. For ShufflenetV2 and ShufflenetV2+ search space, although these search space are more challenging, our method still obtains architectures with 74.9% and 77.5% Top-1 accu-

Table 2: Searching results on NAS-Bench-201 dataset [7]. Our method obtains 1.32%, 2.69% and 1.12% absolute accuracy improvement compared with SPOS on the test sets of CIFAR-10, CIFAR-100 and ImageNet-16-120, respectively. Note that results for other weight-sharing methods are reported by NAS-Bench-201 benchmark [7], which only use the labels of the dataset as the true performance of the searched architecture for fair comparison.

Method	CIFAR-10 Acc. (%)		CIFAR-100 Acc. (%)		ImageNet-16-120 Acc. (%)	
	Validation	Test	Validation	Test	Validation	Test
RSPS [16]	84.16±1.69	87.66±1.69	59.00±4.60	58.33±4.34	31.56±3.28	31.14±3.88
GDAS [9]	90.00±0.21	93.51±0.13	71.14±0.27	70.61±0.26	41.70±1.26	41.84±0.90
SETN [8]	84.04±0.28	87.64±0.00	58.86±0.04	59.05±0.20	33.06±0.02	32.51±0.21
DSNAS [12]	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
PC-Darts [33]	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
SPOS [11]	88.40±1.07	92.24±1.16	67.84±2.00	68.07±2.25	39.28±3.00	40.28±3.00
PDNAS(Ours)	89.94±0.96	93.56±0.59	69.00±1.82	70.76±1.39	41.50±1.18	41.41±0.70
Optimal	91.61	94.37	73.49	73.51	46.77	47.31

racy, surpassing other baselines by 0.3%∼0.9%. For MobileNetV2+ search space, we compare with another space shrinkage method GreedyNAS [37], and our PDNAS is slightly better than it.

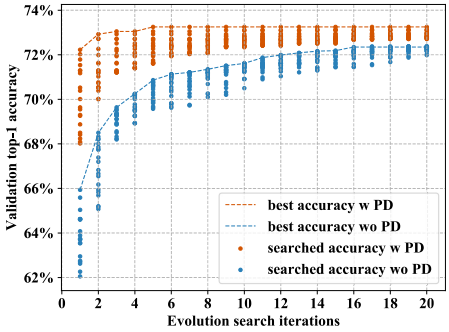


Figure 4: Top-1 accuracy of evolutionary searching results on ShuffleNetV2 search space by in ImageNet (with or without PD).

Comparison on search efficiency. In addition to accuracy gains, our PDNAS can significantly improve the efficiency of the search stage, which can be at least 2 times faster in searching since the search space is reduced by PD. As shown in the Figure 4, we can see with the PD, the search space is shrunk, and the best architecture can be obtained just in the 5th iteration with PD, and in the 16th iteration without PD. In summary, The PD helps to search faster and for better architectures.

4.2. Searching on NAS-Bench-201 Dataset

As SPOS [11], we also implement searching on the recent NAS-Bench-201 [7] dataset for more fair and comprehensive comparison with other baseline methods. We constructed a supernet with the same search space as NAS-Bench-201. The supernet is trained for 250 epochs, then we search the optimal architecture using the evolutionary algorithm. As for the baseline SPOS [11], we implement

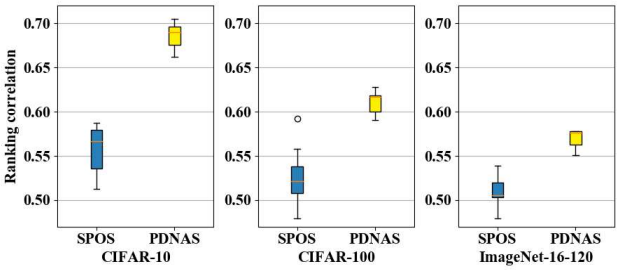


Figure 5: Correlation comparison between SPOS [11] and PDNAS. The Y-axis represents the range of ranking correlation from 10 repeated experiments with different random seeds on NAS-Bench-201 dataset [7].

uniform sampling during training supernet while for our method, we use GCN to encode the network architectures. After 40 epochs of supernet training as warm-up, 200 structures are sampled to train the PD every 20 epochs, and the evolutionary search is performed conditionedly on the last PD. Details refer to the supplementary materials.

Searching results. Results are shown in Table 2. We can see that on the test sets of CIFAR-10, CIFAR-100 and ImageNet-16-120, our method obtains 1.32%, 2.69% and 1.12% absolute accuracy improvement compared with SPOS, which further validates that PDNAS enables to boost the performance of one-shot NAS methods by shrinking the search space. Even compared with the recent NAS methods (*e.g.* PC-Darts [33] and DSNAS [12]), our method has achieved state-of-the-art performance on the CIFAR-10 and CIFAR-100 of NAS-Bench-201 dataset.

Correlation of ranking with ground-truth accuracy. For weight-sharing one-shot methods, the evaluation ability of supernet can be reflected by the ranking correlation of architectures between the accuracy on the supernet and accuracy by training from scratch. In this way, after the supernet is trained well, for the baseline SPOS [11] method, we randomly sample 200 paths from the original search space.

Table 3: Effect of paired ranking loss and unlabeled path augmentation on the searching performance with a PD. Accuracy refers to the ImageNet dataset in MobileNetV2 search space.

Method	Top-1	Top-5
SPOS [11]	74.80%	–
PDNAS w/o Unlabel-Aug	75.80%	92.60%
PDNAS w/o Rank-Loss	75.50%	92.30%
PDNAS	76.10%	93.10%

Then we evaluate the accuracy of these paths and calculate the ranking correlation with their ground-truth accuracy. In contrast, the correlation of our method is calculated by sampling 200 architectures from the reduced space maintained by the last PD. The results of repeating 10 times is shown in Figure 5. We can find the correlation of our PDNAS is slightly higher than that of SPOS. It shows that even in a subset of the original paths and with good performance, our method can still maintain a good correlation.

4.3. Ablation Studies

Effect of pairwise ranking loss and unlabeled augmentation. In our method, the ability of PD to distinguish good and weak paths is very important for the effectiveness of search space shrinkage. Then paired ranking loss and unlabeled path augmentation are used for boosting its training. We investigate their effect on MobileNetV2 search space as Section 4.1. As shown in Table 3, we can find that ranking loss and unlabeled path augmentation have 1.0% and 0.7% performance improvements, respectively.

Besides, we also straightforwardly analyze their effect on the classification accuracy of the PD. To do so, we sample 1000, 500, 200 paths from the supernet, then take 90% of them for training PD and 10% to test the classification accuracy of PD, which are reported in Table 4. We can find that both ranking loss and unlabeled path augmentation can better improve the accuracy under the different number of sampled paths. The ranking loss tends to achieve better performance when the number of sampled paths is large. And unlabeled path augmentation plays a more important role when the number of sampled paths is small.

Table 4: Effect of paired ranking loss and unlabeled path augmentation on the accuracy of PD. Results refer to MobileNetV2 search space w.r.t. ImageNet dataset.

Method		Number of sampled paths		
Rank-Loss	Unlabel-Aug	200	500	1000
×	×	79.02%	87.78%	90.36%
✓	×	83.19%	91.16%	94.12%
×	✓	83.43%	90.40%	92.76%
✓	✓	83.89%	91.78%	95.29%

Effect of number of PD for shrinking space. Since the performance of different paths changes during supernet training, we need to periodically resample paths and train the PD accordingly. In this way, we investigate the effect of number of trained PD on the searching performance. On NAS-Bench-201 dataset, we train PDs in every 10, 20, and 40 epochs, and for each PD, we sample 100, 200, 300 paths for training. The corresponding searching results for CIFAR-10 and CIFAR-100 are reported in Figure 6. We can find that introducing more PDs (smaller epoch interval) will bring better performance (*e.g.*, 10 vs 40 epochs), but beyond a certain range, the gain will be relatively small (*e.g.*, 10 vs 20 epochs). Similarly, the results of sampling 300 and 200 paths for training PD are relatively close, better than that by sampling 100 paths. Considering that introducing more PDs and sampling more paths will bring additional computational cost, in practice, we thus suggest that sampling 200 path in every 20 epochs to train a PD is a good option.

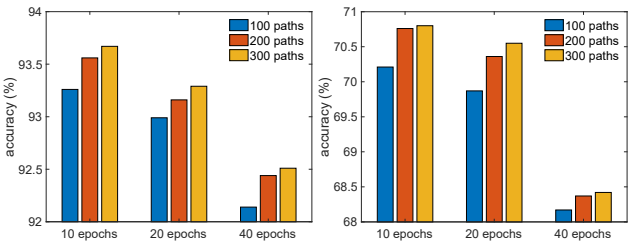


Figure 6: Effect of training a PD by every 10, 20, 40 epochs with different amount of sampled paths. Results refer to the accuracy for CIFAR-10 (left) and CIFAR-100 (right) on NAS-Bench-201 dataset.

5. Conclusion

We proposed PDNAS, which uses a binary path discriminator (PD) for good and weak paths. Then a PD is capable of encoding the relative ranking knowledge in supernet, and shrinking the search space. By doing this, we bridge the supernet training and searching via the PD, and promising architectures are more likely to be searched on a small space. Extensive experiments on ImageNet and NAS-Bench-201 datasets validate the effectiveness of our introduced PD, and the superiority of our proposed method.

Acknowledgment

This work is funded by the National Key Research and Development Program of China (No. 2018AAA0100701) and the NSFC 61876095. Shan You is supported by Beijing Postdoctoral Research Foundation. Qingyi Gu is supported by the National Key R&D Program of China 2019YFB1311100, the Scientific Instrument Developing Project of the Chinese Academy of Sciences (No. YJKYYQ20200045). We thank Yiming Hu, Shizheng Qin, Zichao Guo and Tao Huang for valuable help.

References

[1] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559, 2018.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[3] Xiangxiang Chu, Bo Zhang, Jixiang Li, Qingyuan Li, and Ruijun Xu. Scarlet-nas: bridging the gap between stability and scalability in weight-sharing neural architecture search. *arXiv preprint arXiv:1908.06022*, 2019.

[4] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[8] Xuanyi Dong and Yezhou Yang. One-shot neural architecture search via self-evaluated template network. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3680–3689, 2019.

[9] Xuanyi Dong and Yezhou Yang. Searching for a robust neural architecture in four gpu hours. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770, 2019.

[10] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graphnas: Graph neural architecture search with reinforcement learning. *ArXiv*, abs/1904.09981, 2019.

[11] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

[12] Shoukang Hu, S. Xie, Hehui Zheng, C. Liu, Jianping Shi, Xunying Liu, and D. Lin. Dsnas: Direct neural architecture search without parameter retraining. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12081–12089, 2020.

[13] Yiming Hu, Xingang Wang, Lujun Li, and Qingyi Gu. Improving one-shot nas with shrinking-and-expanding supernet. *Pattern Recognition*, 118:108025, 2021.

[14] Mahmut Kaya and H. Ş. Bilge. Deep metric learning: A survey. *Symmetry*, 11:1066, 2019.

[15] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *ArXiv*, abs/1902.07638, 2019.

[16] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2019.

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[18] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *ArXiv*, abs/1710.07535, 2017.

[19] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.

[20] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

[21] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[22] Xiu Su, Tao Huang, Yanxi Li, Shan You, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Prioritized architecture sampling with monto-carlo tree search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10968–10977, 2021.

[23] Xiu Su, Shan You, Tao Huang, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Locally free weight sharing for network width search. In *ICLR*. OpenReview.net, 2021.

[24] Xiu Su, Shan You, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Bcnet: Searching for network width with bilaterally coupled network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2175–2184, 2021.

[25] Xiu Su, Shan You, Jiyang Xie, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Vision transformer architecture search. *arXiv preprint arXiv:2106.13700*, 2021.

[26] Xiu Su, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. K-shot NAS: learnable weight-sharing for NAS with k-shot supernets. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 9880–9890. PMLR, 2021.

[27] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *GECCO*, 2017.

[28] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[29] Colin. White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. *ArXiv*, abs/1910.11858, 2019.

[30] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[31] Qizhe Xie, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with noisy student improves imagenet classification. *ArXiv*, abs/1911.04252, 2019.

[32] Saining Xie, Alexander Kirillov, Ross B. Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1284–1293, 2019.

[33] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: partial channel connections for memory-efficient architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[34] Yibo Yang, Hongyang Li, Shan You, Fei Wang, Chen Qian, and Zhouchen Lin. Ista-nas: Efficient and consistent neural architecture search by sparse coding. *Advances in Neural Information Processing Systems*, 33, 2020.

[35] Yibo Yang, Shan You, Hongyang Li, Fei Wang, Chen Qian, and Zhouchen Lin. Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6667–6676, 2021.

[36] Chris Ying, Aaron Klein, Esteban Real, Eric L. Christiansen, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. *ArXiv*, abs/1902.09635, 2019.

[37] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[38] Yizhou Zhou, Xiaoyan Sun, Chong Luo, Zheng-Jun Zha, and Wen-Jun Zeng. Posterior-guided neural architecture search. In *AAAI 2020*, 2019.

[39] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.