PA&DA: Jointly Sampling PAth and DAta for Consistent NAS

Shun Lu[1,2], Yu Hu[1,2]*, Longxing Yang[1,2], Zihao Sun[1,2], Jilin Mei[1], Jianchao Tan[3], Chengru Song[3]
[1] Research Center for Intelligent Computing Systems,
Institute of Computing Technology, Chinese Academy of Sciences
[2] School of Computer Science and Technology, University of Chinese Academy of Sciences
[3] Kuaishou Technology
{lushun19s, huyu, yanglongxing20b, sunzihao18z, meijilin}@ict.ac.cn,
{jianchaotan, songchengru}@kuaishou.com

Abstract

*Based on the weight-sharing mechanism, one-shot NAS methods train a supernet and then inherit the pre-trained weights to evaluate sub-models, largely reducing the search cost. However, several works have pointed out that the shared weights suffer from different gradient descent directions during training. And we further find that large gradient variance occurs during supernet training, which degrades the supernet ranking consistency. To mitigate this issue, we propose to explicitly minimize the gradient variance of the supernet training by jointly optimizing the sampling distributions of **PA**th and **DA**ta (PA&DA). We theoretically derive the relationship between the gradient variance and the sampling distributions, and reveal that the optimal sampling probability is proportional to the normalized gradient norm of path and training data. Hence, we use the normalized gradient norm as the importance indicator for path and training data, and adopt an importance sampling strategy for the supernet training. Our method only requires negligible computation cost for optimizing the sampling distributions of path and data, but achieves lower gradient variance during supernet training and better generalization performance for the supernet, resulting in a more consistent NAS. We conduct comprehensive comparisons with other improved approaches in various search spaces. Results show that our method surpasses others with more reliable ranking performance and higher accuracy of searched architectures, showing the effectiveness of our method. Code is available at https://github.com/ShunLu91/PA-DA.*

1. Introduction

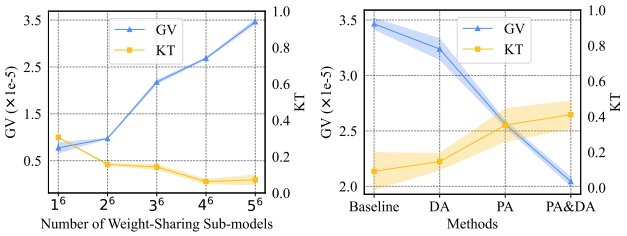
Neural architecture search (NAS) aims to automate the process of designing architectures. Conventional NAS methods [2,58] separately train each sub-model to guide the controller for better architectures, demanding prohibitive computational complexity. ENAS [34] explores the weight-sharing mechanism across sub-models and One-Shot NAS [3] proposes to train a supernet to share weights for sub-models to achieve higher efficiency. Later on, many follow-ups [9, 18, 30, 49] adopt this vein to perform NAS.

Though the weight-sharing mechanism greatly improves NAS efficiency, many works [19,20,41,50,53,56,57] point out that the shared weights suffer from different gradient descent directions in different sub-models, leading to large gradient variance and poor ranking consistency. To mitigate this issue, they [20, 41, 56, 57] propose to maintain multi-copies of supernet weights to decrease the weight-sharing extent, manually elaborate a better path sampling strategy [9, 50], or introduce additional loss regularizations [19, 50, 53]. However, they typically require multiple computation burdens for the supernet training and obtain unsatisfying results, motivating us to explore a better solution.

Notice that significant efforts [13, 17, 22, 36, 38, 40] have been dedicated to reducing the variance of the stochastic gradient descent (SGD) for minimizing finite sums. Many works [1, 5, 16, 23, 25, 45, 55] focus on optimizing the data sampling distribution to reduce the gradient variance (GV) for training deep models. These methods generally enjoy faster convergence and better generalization performance, inspiring us to improve the supernet training from the perspective of gradient variance reduction.

We conduct a toy experiment on NAS-Bench-201 [15] using CIFAR-10 to investigate the GV for the supernet training. We use SPOS [18] algorithm to train the supernet and gradually increase the candidate operations on each edge to change the weight-sharing extent. We record the average GV of all candidate operation weights during training and evaluate the supernet performance by measuring the ranking results of the same 64 sub-models (corresponding to the smallest search space with 2 candidate operations on

*Corresponding author.



(a) Different weight-sharing extent (b) Different methods

Figure 1. (a) KT and GV with different weight-sharing extents by changing the number of sub-models exponentially. (b) Comparison between the baseline and our method. KT: Kendall's Tau, GV: Gradient Variance. GV calculation is in the supplements.

6 edges). As illustrated in Fig.1(a), with more sub-models sharing weights from the supernet, GV becomes larger and the ranking consistency becomes worse. These results indicate that a larger GV during training harms the supernet ranking consistency, which prompts us to reduce the GV to improve the supernet performance.

In this paper, we aim to reduce the supernet GV during training to improve the convergence rate and generalization performance. We first theoretically derive the relationship between the supernet GV and the sampling distributions of path and data. Then we explicitly minimize the GV by optimizing the sampling distributions. We find that the optimal sampling probability is proportional to the normalized gradient norm of the path and training data. Thus we use the normalized gradient norm as the importance indicator and adopt an importance sampling strategy for path and data during the supernet training. As exemplified in Fig.1(b), by using our proposed PAth importance sampling (PA) and DAta importance sampling (DA), we can reduce the supernet GV and improve its ranking consistency.

In summary, our contributions are as follows:

- We validate that the weight-sharing mechanism for supernet training induces large GV, harming the supernet performance and worsening its ranking consistency.
- By deriving the relationship between the supernet GV and sampling distributions, we propose to explicitly minimize the GV by **jointly optimizing path and data sampling distributions** during supernet training. We reveal that the optimal sampling probability is proportional to the normalized gradient norm of path and data, and adopt an importance sampling for them during the supernet training.
- Our method only requires negligible computation to perform the importance sampling for path and data, and does not need tediously hyper-parameter tuning. We obtain the highest Kendall's Tau [39] 0.713 on NAS-Bench-201 and achieve superior performance on DARTS and ProxylessNAS search spaces.

2. Related Work

2.1. One-Shot Neural Architecture Search

Early NAS methods [2, 58] are time-consuming due to the training of each sub-model from scratch. ENAS [34] leverages the weight-sharing mechanism to share weights across sub-models, greatly speeding up the NAS process. One-Shot NAS [3] proposes to train the supernet using the path dropout technique and then conduct the architecture search by inheriting the pre-trained supernet weights to sub-models for fast evaluation. Follow-up works can be roughly divided into gradient-based approaches [8, 30, 31, 49] and sampling-based methods [7, 18, 42, 51]. Gradient-based approaches relax the architectural parameters to be continuous and jointly optimize the architectural parameters and supernet weights using the gradient descent, which requires multiple memory costs and often suffers from instability. While sampling-based methods train the supernet weights by sampling different paths (i.e. sub-models) and only optimize one path at each training step, which is more efficient and robust in practice. For example, the widely used SPOS [18] algorithm adopts a uniform sampling strategy to sample path and training data during the supernet training. In this work, we focus on the sampling-based one-shot NAS methods and utilize SPOS as our baseline.

2.2. Improved Sampling-based One-Shot NAS

To enhance the consistency of one-shot NAS, some works [20, 41, 56, 57] maintain multiple copies of supernet weights to decrease the weight-sharing extent. They focus on how to split the supernet and how to select appropriate weights for a sampled sub-model, again complicating the supernet training. Several works try to seek a more reasonable gradient direction for better convergence. NSAS [53] utilizes the loss regularization to prevent the performance of other sub-models from degrading and SUMNAS [19] computes the reptile gradient during supernet training. Both of them are motivated by multi-model forgetting and consume multiple computations during training. Other works manually elaborate a better path sampling strategy to handle this issue. FairNAS [10] samples and trains candidate operations without replacement and accumulate the gradients until all of them are activated, ensuring strict fairness to benefit the supernet training. MAGIC-AT [50] increases the gradient similarity between sampled architectures by substituting only one candidate operation across consecutively sampled paths and employing an alignment loss for supernet training. They require lots of human experience and more computation resources. In this work, we optimize the sampling distributions of path and data according to the normalized gradient norm, thus maintaining the efficiency and improving the consistency of one-shot NAS without the need for complex manual design.

2.3. Variance Reduction

Many approaches [13, 17, 22, 36, 38, 40] achieve linear convergence on empirical risk minimization problems by reducing the variance of SGD. They enjoy faster convergence and better generalization performance than vanilla SGD. Other works [1, 5, 16, 23, 25, 45, 55] optimize the data sampling distribution during training to reduce the stochastic gradient variance. They [33, 55] derive a clear relationship that the optimal sampling distribution is proportional to the per-sample gradient norm and use an importance sampling [11, 21, 23, 25] for the training data. As common deep learning frameworks only provide the average gradient of a mini-batch, it is prohibitively expensive to compute the per-sample gradient norm. To solve this problem, [16] exploits the side information to model the sampling distribution per class instead of per sample and [25] approximates the upper bound of the per-sample gradient norm efficiently. Our work is motivated by these methods. Differently, we focus on sampling different paths for supernet training using the normalized gradient norm, which can be efficiently acquired across mini-batches. Besides, we utilize the approximation from [25] to perform an importance sampling for training data during the supernet optimization.

3. Method

3.1. Sampling-based One-Shot NAS

One-Shot NAS [3, 34] has recently become mainstream due to its efficiency and simplicity. Particularly, sampling-based one-shot NAS approaches [9, 18] demonstrate superior performance in searching for top-performing architectures. These methods generally have two stages, i.e., supernet training and sub-model searching.

In the first stage, a supernet $\mathcal{N}$ with weights $\mathcal{W}$ is built by encoding the whole search space $\mathcal{A}$. During training, a sub-model α is sampled according to the discrete distribution $\mathbf{p}(\mathcal{A})$ and we only train the weights $\mathcal{W}_\alpha$ included in the sampled sub-model at each step. We aim to obtain the optimal supernet weights $\mathcal{W}^\star$ by iteratively sampling and training the sampled sub-models with the training loss $\mathcal{L}$,

$$\mathcal{W}^\star = \underset{\mathcal{W}}{\operatorname{argmin}} \mathbb{E}_{\substack{\alpha\sim\mathbf{p}(\mathcal{A}) \\ (x,y)\sim\mathbf{q}(\mathbb{D}_T)}} [\mathcal{L}(\mathcal{N}(x,\alpha;\mathcal{W}_\alpha),y)] \tag{1}$$

where (x, y) is sampled from the training dataset $\mathbb{D}_T$ according to the distribution $\mathbf{q}(\mathbb{D}_T)$.

In the second stage, we inherit the optimal supernet weights $\mathcal{W}^\star$ for each sub-model to efficiently evaluate their performance $\mathcal{P}$ on the validation dataset $\mathbb{D}_V$. A heuristic search algorithm is often applied to search for the top-performing sub-model $\alpha^\star$,

$$\alpha^\star = \underset{\alpha\in\mathcal{A}}{\operatorname{argmax}} \mathbb{E}_{(x,y)\sim\mathbf{q}(\mathbb{D}_V)} [\mathcal{P}(\mathcal{N}(x,\alpha;\mathcal{W}^\star_\alpha),y)] \tag{2}$$

The performance of each sub-model is measured using the supernet weights, thus the supernet ranking consistency becomes essential for the ultimate NAS performance. We try to reduce the supernet gradient variance during training to improve the supernet convergence and ranking consistency. We propose to jointly optimize the sampling distributions of $\mathbf{p}(\mathcal{A})$ and $\mathbf{q}(\mathbb{D}_T)$ during the supernet training, which implies a bi-level optimization problem with $\mathcal{W}$ as the upper-level variable and $\mathbf{p}, \mathbf{q}$ as the lower-level variable

$$\begin{aligned} &\mathcal{W}^\star = \underset{\mathcal{W}}{\operatorname{argmin}} \mathbb{E}[\mathcal{L}(\mathcal{N}(x,\alpha;\mathcal{W}_\alpha),y)] \\ &\text{s.t.} \begin{cases} \alpha\sim\mathbf{p}^\star(\mathcal{A}),\ (x,y)\sim\mathbf{q}^\star(\mathbb{D}_T), \\ \mathbf{p}^\star = \underset{\mathbf{p}}{\operatorname{argmin}} \mathbb{V}[d(\mathbf{p})], \\ \mathbf{q}^\star = \underset{\mathbf{q}}{\operatorname{argmin}} \mathbb{V}[d(\mathbf{q})] \end{cases} \end{aligned} \tag{3}$$

where $d(\mathbf{p})$ and $d(\mathbf{q})$ are the gradient variance function regarding the path and data sampling distributions. In the following, we introduce how to derive their relationship and alternatively optimize both sampling distributions.

3.2. Path Importance Sampling

For the sake of simplicity, we first explain how to seek the optimal $\mathbf{p}^\star$ with the data sampling distribution $\mathbf{q}$ fixed. Given total training steps N, we can re-formulate the expectation in Eq.1 as below

$$\mathcal{W}^\star = \arg\min_{\mathcal{W}} \sum_{i=1}^{N} p_i \mathcal{L}(\mathcal{N}(x_i,\alpha_i;\mathcal{W}_{\alpha_i}),y_i) \tag{4}$$

At i-th training step, when we sample a sub-model α_i from the distribution $\mathbf{p}(\mathcal{A})$ with the probability p_i, the resulting stochastic gradient is given by

$$d_i(p_i) = \frac{1}{Np_i} \nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i,\alpha_i;\mathcal{W}_{\alpha_i}),y_i) \tag{5}$$

where the scaling factor $(Np_i)^{-1}$ ensures the gradient $d_i(p_i)$ is an unbiased approximation of the true quantity. While in previous works [9, 18] using the uniform sampling strategy, the sampling probability of α_i is $p_i = \frac{1}{N}$. We expect to minimize the gradient variance in Eq.5 by optimizing the sampling distribution $\mathbf{p}$, which can be formulated as the following optimization problem

$$\min_{\mathbf{p}} \mathbb{V}[d(\mathbf{p})] = \mathbb{E}\left[d^\top d\right] - \mathbb{E}\left[d\right]^\top \mathbb{E}\left[d\right] \tag{6}$$

By introducing Eq.5 into Eq.6, we have

$$\begin{aligned} \mathbb{E}[d^\top d] &= \sum_{i=1}^{N} p_i \frac{1}{N^2} \frac{1}{p_i^2} \left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i,\alpha_i;\mathcal{W}_{\alpha_i}),y_i)\right\|^2 \\ \mathbb{E}[d] &= \sum_{i=1}^{N} p_i d_i = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i,\alpha_i;\mathcal{W}_{\alpha_i}),y_i) \end{aligned} \tag{7}$$

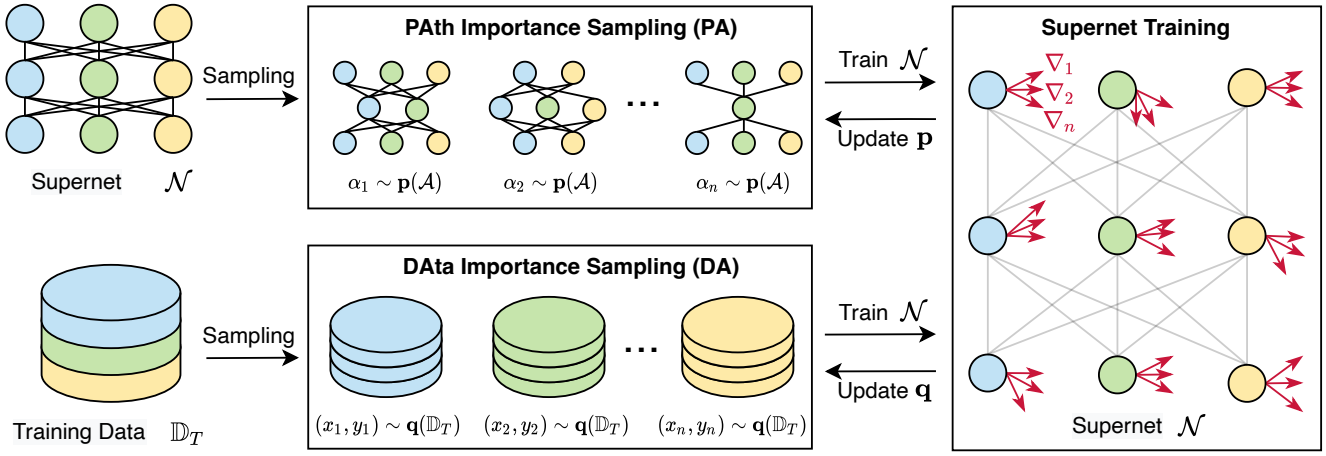


Figure 2. Our supernet training framework includes path importance sampling and data importance sampling. Red arrows show similar gradients of candidate operations via our method. We use the normalized gradient norm to update the path and data sampling distributions.

We can see that $\mathbb{E}[d]$ is independent of the path sampling distribution $\mathbf{p}$, so we can reformulate the problem in Eq.6 as a constrained optimization problem

$$\min_{\mathbf{p}} \sum_{i=1}^{N} \frac{1}{N^2}\frac{1}{p_i} \left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|^2$$
$$\text{s.t.} \sum_{i=1}^{N} p_i = 1 \quad \text{and} \quad p_i \geq 0 \quad \forall i = 1, 2, \dots N$$

Since each $\left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|^2 \geq 0$, the optimal sampling probability p_i must satisfy the inequality constraint and thus we can only consider the equality constraint. By introducing the Lagrangian multiplier λ, we have the Lagrangian function $\Psi(\mathbf{p}, \lambda)$ as below

$$\Psi(\mathbf{p}, \lambda) = \sum_{i=1}^{N} \frac{1}{N^2}\frac{1}{p_i} \left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|^2 + \lambda(\sum_{i=1}^{N} p_i - 1) \tag{8}$$

By setting $\frac{\partial \Psi(\mathbf{p}, \lambda)}{\partial p_i} = 0$, we can get

$$p_i = \frac{\left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|}{N\sqrt{\lambda}} \tag{9}$$

Applying the equality constraint, we have $\sqrt{\lambda} = \sum_{i=1}^{N} \frac{\left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|}{N}$, and further derive the optimal sampling distribution $\mathbf{p}^\star$ when

$$p_i^\star = \frac{\left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|}{\sum_{i=1}^{N} \left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|} \tag{10}$$

Consequently, we can conclude that the optimal path sampling probability $p_i^\star$ is proportional to the normalized gradient norm of the sub-model α_i, saying that sampling the sub-model with a larger gradient norm can reduce the GV for the supernet training.

In practice, we measure the gradient norm of the sub-model α_i as the sum of the gradient norm of its contained candidate operations and use the normalized gradient norm of each candidate operation as their sampling probability. We calculate the gradient norm after backward and update the sampling probability after each epoch. Hence, our optimization for the path sampling distribution $\mathbf{p}$ only requires negligible computation and is particularly efficient.

3.3. Data Importance Sampling

Now we consider the optimal data sampling distribution $\mathbf{q}^\star$ with the path distribution $\mathbf{p}$ fixed. The solution for the $\mathbf{q}^\star$ is off-the-shelf as in previous works [1, 33, 55]. They have demonstrated that sampling training data according to their normalized gradient norm is helpful to reduce GV for deep models training, which can be formally expressed as

$$q_i^\star \propto \left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\| \tag{11}$$

However, computing the per-sample gradient norm is computationally prohibitive, especially in the context of training deep models, where common deep learning frameworks generally provide the average gradient in a batch-wise manner instead of per-sample-wise. Several works [21, 24, 25, 52] have delved into this problem and [25] designs an efficient method to approximate the upper bound of the gradient norm for each training data. Specifically, they propose that the gradient of the loss function regarding the pre-activation outputs of the last layer ∇_L can be deemed an effective estimate of the upper bound, that is

$$\sup\{\left\|\nabla_{\mathcal{W}} \mathcal{L}(\mathcal{N}(x_i, \alpha_i; \mathcal{W}_{\alpha_i}), y_i)\right\|\} \leq \nabla_L \tag{12}$$

In this way, we can easily measure the importance of each training data by accessing their upper bound. Take the im-

Algorithm 1 Supernet training algorithm of PA&DA

Input: Input training data $\mathbb{D}_T$, supernet $\mathcal{N}$ with weights $\mathcal{W}$, training epochs n_{epochs}, training steps n_{steps} per epoch.
Output: Optimized supernet weights $\mathcal{W}^\star$.

for $j = 1$ **to** n_{epochs} **do**
  for $k = 1$ **to** n_{steps} **do**
    Sample a path based on the distribution $\mathbf{p}(\mathcal{A})$;
    Sample a mini-batch training data based on the distribution $\mathbf{q}(\mathbb{D}_T)$;
    Train supernet weights $\mathcal{W}$ by gradient descent;
    Record gradient norm of the sampled path after back-propagation;
    Approximate and record gradient norm of the sampled data using Eq.13.
  end for
  Linearly increase smoothing parameters δ and τ;
  Update the path sampling distribution $\mathbf{p}(\mathcal{A})$ according to Eq.10 and add it to uniform distribution;
  Update the data sampling distribution $\mathbf{q}(\mathbb{D}_T)$ according to Eq.11 and add it to uniform distribution;
end for

age classification task as an example, the pre-activation outputs of the last layer y_L are usually followed by a $\mathrm{softmax}$ layer. When using the cross-entropy loss, we can derive the gradient expression for ∇_L in advance and conveniently compute it during training as below

$$\nabla_L = \mathrm{softmax}(y_L) - \mathbb{1}(y_i) \tag{13}$$

The above computation only requires an additional line of code and can be efficiently executed in a mini-batch manner. Therefore, we use this approximation to estimate the importance of training data and adopt the normalized results to update the sampling distribution $\mathbf{q}$ after each epoch.

3.4. Importance Sampling NAS

Our method aims to improve the supernet ranking consistency by reducing the gradient variance during training. We propose a novel and effective importance-based sampling strategy for training the supernet, including path importance sampling and data importance sampling. Though we fix one of the sampling distributions in the above derivation, we jointly optimize them in practice. We summarize our supernet training algorithm in Algo.1.

PAth Importance Sampling (PA) Following the derived relationship in Eq.10, we record and accumulate the gradient norm for each sampled path after the back-propagation. We use the normalized gradient norm of each candidate operation as their importance and update the sampling distribution after each epoch. To handle those parameter-free operations and avoid the meaningless gradient information at early epochs, we employ a smoothing parameter δ to add our importance sampling distribution and the uniform sampling distribution. We simply linearly increase δ from 0 to 1 during training and provide a discussion about other changing schemes in our experiments.

DAta Importance Sampling (DA) We use the upper bound in Eq.13 as the importance indicator for our training data. After each epoch, we utilize the normalized importance to update the data sampling distribution and adopt an importance sampling strategy with replacement to sample the training indices for the coming epoch. Note that if a training instance is not sampled in the current epoch, it will have zero gradient norm in the update, leading to zero sampling probability. Analogously, we use a smoothing parameter τ to add our importance sampling distribution and the uniform sampling distribution together to tackle the above problem. We linearly increase τ from 0 to 1 during training and compare other strategies in our ablation studies.

4. Experiments

To demonstrate the effectiveness of PA&DA in reducing gradient variance during supernet training, we conduct two types of evaluations. The first is developed on the NAS-Bench-201 [15] using the CIFAR-10 dataset [26] and we provide a comprehensive ranking comparison with other methods. The second type is based on the widely-used public search spaces DARTS [30] and ProxylessNAS [4], using CIFAR-10 and ImageNet [27] datasets, respectively. We conduct an architecture search on these search spaces and compare our search performance with other state-of-the-art methods. At the end of this section, we further provide extensive ablation studies to analyze our method in depth.

4.1. Evaluation of Supernet Ranking Consistency

Search Space NAS-Bench-201 [15] is a popular NAS benchmark and provides the training and test performance of CIFAR-10, CIFAR-100, and ImageNet-16 for each sub-model in this search space. Sub-models are composed of repeated stacking cells with the same structures. Each cell has four nodes and six edges, and each edge has five candidate operations, leading to 5^6 architectures in total.

Settings We construct the supernet with default settings as NAS-Bench-201 and train it on the CIFAR-10 dataset. The smoothing parameters δ and τ are linearly increased from 0 to 1. We employ total training epochs of 256 with a mini-batch size of 256. The SGD optimizer is adopted with an initial learning rate of 0.05, a momentum of 0.9, and a cosine decay strategy. After training the supernet, we evaluate the performance of all sub-models on the test dataset by inheriting the pre-trained supernet weights.

Method	Cost	KT	P@Top5%
SPOS [18]	**1.6**	0.639 ± 0.030	0.211 ± 0.168
FairNAS† [9]	5.4	0.541 ± 0.023	0.160 ± 0.034
Magic-AT† [50]	4.4	0.547 ± 0.059	0.019 ± 0.011
NSAS [53]	14.6	0.653 ± 0.051	0.064 ± 0.028
SUMNAS† [19]	22.6	0.505 ± 0.039	0.145 ± 0.061
Few-Shot-25 [56]	18.6	0.696	-
GM†-8 [20]	18.0	0.656 ± 0.011	0.153 ± 0.006
CLOSE [57]	2.5	0.643 ± 0.050	0.031 ± 0.021
PA&DA	1.8	$\mathbf{0.713 \pm 0.002}$	$\mathbf{0.301 \pm 0.018}$

Table 1. Ranking results on NAS-Bench-201. Cost: we report the supernet training time in terms of the GPU hours. †: they did not release code thus we implement them following their paper strictly. Few-Shot-25 and GM-8 denote splitting the one-shot supernet into 25 and 8 sub-supernets, respectively.

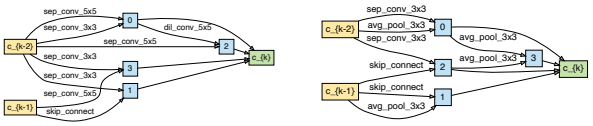
To compare with other methods, we calculate Kendall's Tau (KT) and Precision@Top5% (P@Top5%) metrics. KT indicates the proportion of correct ranking pairs in all ranking pairs, which measures the supernet overall ranking consistency. P@Top5% is the proportion of predicted top-5% sub-models in real top-5% sub-models, showing the ability to identify superb architectures.

Results We summarize the results in Tab.1. The regularization-based or manually-designed methods such as FairNAS, Magic-AT, and SUMNAS not only consume more training time but also perform worse than SPOS. NSAS obtains higher KT but lower P@Top5% and spends nearly an order of magnitude more training time than SPOS. Although the splitting methods such as Few-Shot-NAS, GM, and CLOSE achieve better KT, they generally need several times more cost than SPOS. In contrast, PA&DA only requires 0.2 more GPU hours than SPOS and reaches the highest KT and P@Top5% when compared with others, demonstrating that our training paradigm is effective and beneficial to improving the supernet ranking consistency.

4.2. Search Performance on CIFAR-10

Search Space We use the CIFAR-10 dataset to search for superior cells in the DARTS [30] search space. The supernet is composed of six normal cells and two reduction cells. Normal cells process the feature map without down-sampling, while reduction cells perform down-sampling on the feature map with stride $= 2$ and are located at the $1/3$ and $2/3$ of the total depth of the supernet. Each cell consists of seven nodes with four intermediate nodes and fourteen edges with eight candidate operations on each edge. We search for the most two powerful operations for each edge to get the final searched cell.

Settings We follow the settings in NSAS [53] to combine our method with RandomNAS [28]. We use the SGD optimizer with momentum 0.9 and weight decay 3e-4 to train the supernet for 50 epochs. The initial learning rate is 0.025 and is then decayed to 0.001 by a cosine strategy. After the supernet training, we randomly search for 60 rounds and evaluate 100 sub-models at each round to select the most promising architecture. By re-training the searched architecture, we compare the top-1 classification accuracy with other methods. We visualize the best searched cell in Fig.3, and provide other cells and re-training details in our Supp.



(a) Normal Cell (b) Reduction Cell

Figure 3. Our best searched cells in the DARTS search space.

Results We report the best and average test accuracy from repeated experiments with three random seeds in Tab.2. As can be seen, our method achieves the highest average test accuracy 97.52 ± 0.07, surpassing the original DARTS and its advanced variants. When compared with other improved one-shot NAS methods such as NSAS, Few-Shot-NAS, GM, and CLOSE, our method consistently outperforms them with the least search cost.

Our best cells are shown in Fig.3. We can observe that both the normal cell and the reduction cell have the $skip\_connect$ operation from the input nodes, leading to a residual link with other operations. As pointed out in [44], such a ResNet-style residual link is helpful for achieving state-of-the-art performance, demonstrating that our method excels in identifying excellent architectures.

4.3. Search Performance on ImageNet

Search Space We use the chain-like search space as proposed in ProxylessNAS [4], including 21 searchable layers in the supernet. We search for lightweight MobileNet [37] blocks by exploring the kernel sizes $\{3, 5, 7\}$ and expansion rates $\{3, 6\}$ for the searchable blocks. A searchable $skip\_connect$ is added for the blocks without down-sampling, leading to 7 or 6 candidate operations per layer.

Settings We utilize our method to train the supernet on 8 GPU cards for 120 epochs, with a total batch size of 2048. SGD optimizer is adopted with the weight decay 4e-5 and momentum 0.9. The initial learning rate is 0.5 and is decayed to 5e-4 by a cosine strategy. After training the supernet, we use an evolutionary search algorithm to search for top-performing architectures with the FLOPs constraint 400 M. The evolutionary search lasts for 20 epochs in total. At each epoch, we maintain a population with 50 sub-models,

Method	Test Accuracy		Parameters (M)	Search Cost (GPU Days)	Search Method
	Best(%)	Average(%)			
NASNet-A [59]	97.35	-	3.3	1,800	RL
ENAS [34]	97.11	-	4.6	0.5	RL
DARTS [30]	-	97.00 ± 0.14	3.3	0.4	Gradient
GDAS [14]	97.07	-	3.4	**0.3**	Gradient
RandomNAS [28]	-	97.15 ± 0.08	4.3	2.7	Random
DARTS-PT [46]	97.52	97.39 ± 0.08	**3.0**	0.8	Gradient
BaLeNAS [54]	-	97.50 ± 0.07	3.8	0.6	Gradient
AGNAS [42]	97.54	97.47 ± 0.003	3.6	0.4	Gradient
ZARTS [47]	-	97.46 ± 0.07	3.7	1.0	Gradient
GDAS-NSAS [53]	97.27	-	3.5	0.4	Gradient
RandomNAS-NSAS [53]	97.36	-	3.1	0.7	Random
Few-Shot-NAS† [56]	97.42	97.37 ± 0.06	3.8	2.8	Gradient
GM [20]	97.60	97.51 ± 0.08	3.7	2.7	Gradient
CLOSE [57]	-	97.28 ± 0.04	4.1	0.6	Gradient
PA&DA	**97.66**	$\mathbf{97.52 \pm 0.07}$	3.9	0.4	Random

Table 2. Comparison with other state-of-the-art methods on the CIFAR-10 dataset using DARTS search space. We report the best and average test accuracy of repeated experiments.†: reported by GM [20].

Method	Params. (M)	FLOPs (M)	Top-1 (%)	Top-5 (%)
AmoebaNet-A [35]	5.1	555	74.5	92.0
MnasNet-A1 [43]	**3.9**	**312**	75.2	92.5
PNAS [29]	5.1	588	74.2	91.9
TNASP-C [32]	5.3	497	75.8	92.7
DA-NAS [12]	-	389	74.6	-
SPOS [18]	5.4	472	74.8	-
FBNet-C [48]	5.5	375	74.9	-
ProxylessNAS [4]	7.1	465	75.1	92.3
FairNAS-A [9]	4.6	388	75.3	-
MAGIC-AT [50]	6.0	598	76.8	93.3
Few-Shot NAS [56]	4.9	521	75.9	-
GM [20]	4.9	530	76.6	93.0
PA&DA	5.3	399	**77.3**	**93.5**

Table 3. Comparison with other state-of-the-art methods on the ImageNet dataset using the ProxylessNAS search space.

including 25 sub-models from mutation and crossover respectively. We retrain our searched architecture on the ImageNet training dataset and evaluate its performance on the validation dataset. The detailed retraining configuration and our searched architecture are provided in our Supp.

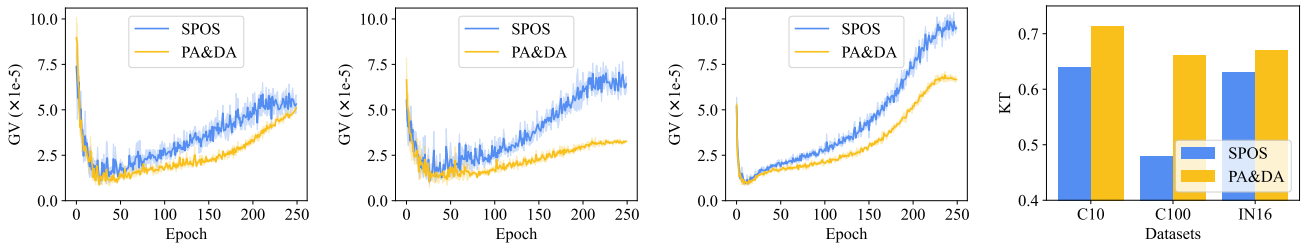
Results The performance comparison is summarized in Tab.3. Our PA&DA surpasses DA-NAS and FairNAS-A with a bit more FLOPs. When compared with SPOS or other improved one-shot NAS methods such as ProxylessNAS, MAGIC-AT, Few-Shot NAS, and GM, our searched architecture is smaller and obtains the highest top-1 accuracy 77.3, suffice to demonstrate the efficacy of our method.

4.4. Ablation Studies

Effect of batch size As larger batch sizes (BS) can stabilize the training of deep models with lower GV, we use the SPOS [18] to ablation BS from 16 to 512 to validate this phenomenon during supernet training. To ensure sufficient convergence, we use double epochs for BS 512. As shown in Fig.5(a), GV decreases and KT increases monotonically as BS becomes larger, and BS 512 obtains the best KT 0.670 $\pm$ 0.029. This further confirms that lower GV benefits the supernet training, which exactly meets our idea. However, PA&DA does not require more epochs than BS 512 and gets higher KT, thus is more efficient and effective.

Effect of schedules for smoothing parameters To preload data indices for efficiency, we update the sampling probability of DA after each epoch. We explore two changing styles for τ: linearly decrease and increase, and evaluate the distribution granularity sample-wise or class-wise in Tab.4. Notice that using a sample-wise distribution and linearly increasing τ yields the best result. As for PA, we investigate the update frequency for the sampling distribution and also two changing styles for δ. Results show that updating the sampling probability per epoch and linearly increasing δ is better. Both results suggest a linearly increase schedule for smoothing parameters, showing that the importance sampling is preferable in the late of training.

Effect of DA and PA We conduct the ablation study for DA and PA in Tab.5. When DA and PA are both disabled, our method degenerates to the baseline method SPOS [18]. When either one is applied, we can obtain higher KT and P@Top5%. Furthermore, both modules cooperate well with



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet16-120 (d) Ranking Comparison

Figure 4. KT and GV on NAS-Bench-201 using three datasets. C10: CIFAR-10, C100: CIFAR-100, IN16: ImageNet16-120.

Module	Freq.	Style	Gran.	KT
DA	PE	↓	Class	0.635 ± 0.029
	PE	↑	Class	0.637 ± 0.024
	PE	↓	Instance	0.643 ± 0.021
	PE	↑	Instance	**0.644 ± 0.014**
PA	PS	↓	Path	0.663 ± 0.008
	PS	↑	Path	0.698 ± 0.007
	PE	↓	Path	0.667 ± 0.003
	PE	↑	Path	**0.699 ± 0.004**

Table 4. Ranking performance w.r.t the smoothing parameters and update schedules for DA and PA. PE: update the sampling distribution per epoch, PS: update the sampling distribution per step.

each other, and using them together yields the best result. Besides, we empirically observe that PA contributes more performance gains than DA.

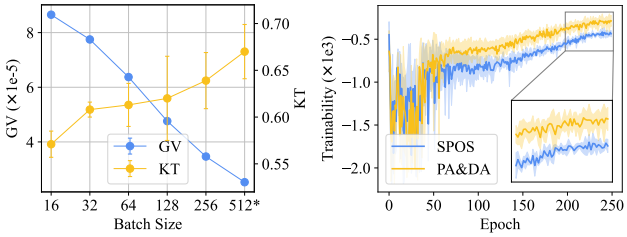
DA	PA	KT	P@Top5%
-	-	0.639 ± 0.030	0.211 ± 0.168
✓	-	0.644 ± 0.014	0.225 ± 0.049
-	✓	0.699 ± 0.004	0.299 ± 0.008
✓	✓	**0.713 ± 0.002**	**0.301 ± 0.018**

Table 5. Ablation study for PA and DA.

5. Analysis and Discussions

5.1. Gradient Variance Comparison

To show the benefit of PA&DA for reducing GV during supernet training, we compare PA&DA with the baseline method SPOS [18] on CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets using NAS-Bench-201 [15]. We repeat the experiments with three different seeds and record the supernet GV of each epoch in Fig.4. As the training goes on and smoothing parameters linearly increases, the original uniform sampling strategy for path and data gradually shifts to the biased sampling, making PA&DA achieve lower GV than the baseline. Due to this advantage, the supernet trained by PA&DA obtains higher ranking consistency in three datasets as shown in Fig.4(d).



(a) KT and GV w.r.t Batch Size (b) Trainability comparison

Figure 5. Effect of various batch sizes and trainability comparison.

5.2. Trainability of Paths from PA

As analyzed above, PA plays a more critical role than DA. To explore its key advantage, we adopt the trainability from TE-NAS [6] to measure the sampled path at each training step. Results are shown in Fig.5(b). We can see that the sampled path of PA constantly enjoys a higher trainability than the baseline, especially at the end of the training, which explains the faster convergence and better generalization performance from PA.

6. Conclusion

In this work, we reduce the gradient variance for the supernet training by jointly optimizing the path and data sampling distributions to improve the supernet ranking consistency. We derive the relationship between the gradient variance and the sampling distributions and use the normalized gradient norm to update both distributions. Extensive experiments demonstrate the effectiveness of our method. In the future, we will further explore more effective methods to reduce gradient variance for supernet training.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under grant No. 2018AAA0102701 and in part by the National Natural Science Foundation of China under Grant No. 62176250 and No.62203424.

References

[1] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015. 1, 3, 4

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017. 1, 2

[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 1, 2, 3

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 5, 6, 7

[5] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NIPS*, 2017. 1, 3

[6] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *ICLR*, 2021. 8

[7] Xiangxiang Chu, Xudong Li, Shun Lu, Bo Zhang, and Jixiang Li. Mixpath: A unified approach for one-shot neural architecture search. *arXiv preprint arXiv:2001.05887*, 2020. 2

[8] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: robustly stepping out of performance collapse without indicators. In *ICLR*, 2021. 2

[9] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *ICCV*, 2021. 1, 3, 6, 7

[10] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In *ECCV*, 2020. 2

[11] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *ICLR*, 2020. 3

[12] Xiyang Dai, Dongdong Chen, Mengchen Liu, Yinpeng Chen, and Lu Yuan. Da-nas: Data adapted pruning for efficient neural architecture search. In *ECCV*, 2020. 7

[13] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, 2014. 1, 3

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

[15] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 1, 5, 8

[16] Siddharth Gopal. Adaptive sampling for sgd by exploiting side information. In *ICML*, 2016. 1, 3

[17] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *AISTATS*, 2020. 1, 3

[18] 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. In *ECCV*, 2020. 1, 2, 3, 6, 7, 8

[19] Hyeonmin Ha, Ji-Hoon Kim, Semin Park, and Byung-Gon Chun. Sumnas: Supernet with unbiased meta-features for neural architecture search. In *ICLR*, 2022. 1, 2, 6

[20] Shoukang Hu, Ruochen Wang, Lanqing Hong, Zhenguo Li, Cho-Jui Hsieh, and Jiashi Feng. Generalizing few-shot nas with gradient matching. In *ICLR*, 2022. 1, 2, 6, 7

[21] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019. 3, 4

[22] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013. 1, 3

[23] Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In *NIPS*, 2018. 1, 3

[24] Angelos Katharopoulos and François Fleuret. Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*, 2017. 4

[25] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *ICML*, 2018. 1, 3, 4

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

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 5

[28] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2020. 6, 7

[29] 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. 7

[30] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 1, 2, 5, 6, 7

[31] Shun Lu, Yu Hu, Longxing Yang, Zihao Sun, Jilin Mei, Yiming Zeng, Xiaowei Li, and Tecent ADlab. Du-darts: Decreasing the uncertainty of differentiable architecture search. In *BMVC*, 2021. 2

[32] Shun Lu, Jixiang Li, Jianchao Tan, Sen Yang, and Ji Liu. Tnasp: A transformer-based nas predictor with a self-evolution framework. In *NeurIPS*, 2021. 7

[33] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *NIPS*, 2014. 3, 4

[34] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 1, 2, 3, 7

[35] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 7

[36] Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *NIPS*, 2012. 1, 3

[37] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 6

[38] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, 2017. 1, 3

[39] Pranab Kumar Sen. Estimates of the regression coefficient based on kendall's tau. *Journal of the American statistical association*, 1968. 2

[40] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *arXiv preprint arXiv:1209.1873*, 2012. 1, 3

[41] 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*, 2021. 1, 2

[42] Zihao Sun, Yu Hu, Shun Lu, Longxing Yang, Jilin Mei, Yinhe Han, and Xiaowei Li. Agnas: Attention-guided micro and macro-architecture search. In *ICML*, 2022. 2, 7

[43] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019. 7

[44] Xingchen Wan, Binxin Ru, Pedro M Esperança, and Zhenguo Li. On redundancy and diversity in cell-based neural architecture search. In *ICLR*, 2022. 6

[45] Linnan Wang, Yi Yang, Renqiang Min, and Srimat Chakradhar. Accelerating deep neural network training with inconsistent stochastic gradient descent. *Neural Networks*, 93:219–229, 2017. 1, 3

[46] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *ICLR*, 2021. 7

[47] Xiaoxing Wang, Wenxuan Guo, Junchi Yan, Jianlin Su, and Xiaokang Yang. Zarts: On zero-order optimization for neural architecture search. In *NeurIPS*, 2021. 7

[48] 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*, 2019. 7

[49] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *ICLR*, 2019. 1, 2

[50] Jin Xu, Xu Tan, Kaitao Song, Renqian Luo, Yichong Leng, Tao Qin, Tie-Yan Liu, and Jian Li. Analyzing and mitigating interference in neural architecture search. In *ICML*, 2022. 1, 2, 6, 7

[51] Longxing Yang, Yu Hu, Shun Lu, Zihao Sun, Jilin Mei, Yinhe Han, and Xiaowei Li. Searching for burgerformer with micro-meso-macro space design. In *ICML*, 2022. 2

[52] Jiong Zhang, Hsiang-Fu Yu, and Inderjit S Dhillon. Autoassist: A framework to accelerate training of deep neural networks. In *NeurIPS*, 2019. 4

[53] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *CVPR*, 2020. 1, 2, 6, 7

[54] Miao Zhang, Shirui Pan, Xiaojun Chang, Steven Su, Jilin Hu, Gholamreza Reza Haffari, and Bin Yang. Balenas: Differentiable architecture search via the bayesian learning rule. In *CVPR*, 2022. 7

[55] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, 2015. 1, 3, 4

[56] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *ICML*, 2021. 1, 2, 6, 7

[57] Zixuan Zhou, Xuefei Ning, Yi Cai, Jiashu Han, Yiping Deng, Yuhan Dong, Huazhong Yang, and Yu Wang. Close: Curriculum learning on the sharing extent towards better one-shot nas. In *ECCV*, 2022. 1, 2, 6, 7

[58] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 1, 2

[59] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 7