MixPath: A Unified Approach for One-shot Neural Architecture Search

Xiangxiang Chu   Shun Lu   Xudong Li   Bo Zhang

cxxgtxy@gmail.com, lushun19s@ict.ac.cn, lixudong16@mails.ucas.edu.cn, zhangboyd@qq.com

Abstract

Blending multiple convolutional kernels is proved advantageous in neural architecture design. However, current two-stage neural architecture search methods are mainly limited to single-path search spaces. How to efficiently search models of multi-path structures remains a difficult problem. In this paper, we are motivated to train a one-shot multi-path supernet to accurately evaluate the candidate architectures. Specifically, we discover that in the studied search spaces, feature vectors summed from multiple paths are nearly multiples of those from a single path. Such disparity perturbs the supernet training and its ranking ability. Therefore, we propose a novel mechanism called Shadow Batch Normalization (SBN) to regularize the disparate feature statistics. Extensive experiments prove that SBNs are capable of stabilizing the optimization and improving ranking performance. We call our unified multi-path one-shot approach as MixPath, which generates a series of models that achieve state-of-the-art results on ImageNet.

1. Introduction

Complete automation in neural network design is one of the most important research directions of automated machine learning [32, 22]. Among various mainstream paradigms, one-shot approaches [13, 8, 19] make use of a weight-sharing mechanism that reduces a large amount of computational cost, but these approaches mainly focus on searching for single-path networks. Observing that a multi-path structure is beneficial for model performance as in Inception [30] and ResNeXT [37], it is necessary to incorporate multi-path structure into the search space. Noticeably, exploring multi-path search space is made possible in one-stage approaches like [9] and [26]. However, it poses a challenge for two-stage methods to train a one-shot supernet that can accurately predict the performance of its multi-path submodels.

Although FairNAS [8] largely alleviates the ranking difficulty in the single-path case with a fairness strategy, it is inherently difficult to apply the same method in a multi-path scenario. It should be emphasized that the most critical point of the two-stage NAS is that the supernet can rank submod-
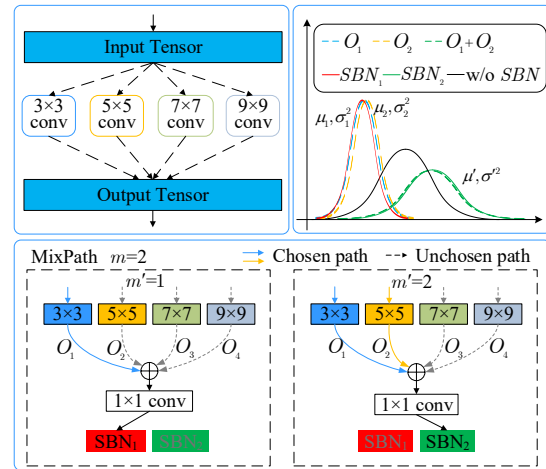

Figure 1: **Top Left**: Options in a demo block, where at most m paths can be chosen. **Bottom**: An example of MixPath supernet training with Shadow Batch Normalizations (SBNs). Note 1x1 Conv is not a must. SBNs are standard BNs, except that SBNs are applied w.r.t the number of activated paths. E.g., SBN_1 is used whenever only $m' = 1$ path is activated, SBN_2 for $m' = 2$ paths. **Top Right**: SBNs catch each of the statistics in two modes (red for $m' = 1$, green for $m' = 2$), however, a single BN (black) can't capture both.

els as accurately as possible. However, we find that a vanilla training of a multi-path supernet (e.g. randomly pick a multi-path submodel to train at each step) is not stable to provide a confident ranking because of changing statistics during the sampling process. Therefore, we dive into its real causes and undertake a unified approach to tackle this issue. Our contributions can be summarized as follows.

- We propose a unified approach for multi-path (say at most m paths are allowed) one-shot NAS, as opposed to the current single-path methods. From this perspective, current one-shot methods [13, 8] become one of our special cases when $m = 1$.

- We disclose why a vanilla multi-path training fails, and we propose a novel yet lightweight mechanism, called *shadow batch normalization* (SBN, see Fig. 1), to stabi-

lize the supernet with a neglectable cost. By exploiting feature similarity from different paths, we further reduce the number of SBNs to be m, otherwise it would be exponential.

- We reveal that our multi-path supernet is trained with stability due to SBNs. It also comes with boosted ranking performance. Together with post BN calibration, it obtains a high Kendall tau (0.597) on a subset of NAS-Bench-101 [40].

- We use extensive experiments to verify the validness of our method across 4 spaces. We search proxylessly on ImageNet using10 GPU days. The searched models obtain state-of-the-art performances, which are comparable with MixNet [34] models searched with 200× more computing powers. Moreover, our MixPath-B makes use of multi-branch feature aggregation and reaches higher accuracy than EfficientNet-B0 with fewer FLOPS and parameters.

2. Related work

Conditional batch normalization Batch Normalization (BN) [16] has greatly facilitated the training of neural networks by normalizing layer inputs to have fixed means and variances. Slimmable neural networks[42] introduces a shared supernet that can run with switchable channels at four different scales (1×, 0.75×, 0.5×, 0.25×). Due to feature inconsistency, independent BNs are applied for each switch configuration to encode conditional statistics. However, it is impractical because it requires an increased number of BNs when it has arbitrary channel widths. BatchQuant [1] utilizes BNs to stabilize the mixed-precision supernet training in a single-path MobileNetV3 search space.

Model ranking correlation It should be emphasized that the ranking ability for the family of two-stage algorithms is of the uttermost importance [2], whose sole purpose is to evaluate networks. To quantitatively analyze their ranking ability, previous works like [43, 8, 20, 46] have applied a Kendall tau measure [17]. To this end, various proxies [32, 47, 48], explicit or implicit performance predictors [23, 21] are developed to avoid the intractable evaluation. Recent one-shot approaches [2, 13, 8] utilize a supernet where each submodel can be rapidly assessed with inherited weights. One-Shot [2] uses dropout (a very sensitive hyperparameter)to simulate sampling and evaluating submodel using a supernet. However, its poor ranking performance prevents it from robustly finding good models [8].

3. MixPath: a unified approach

3.1. Motivation

There is a call for an efficient multi-path one-shot approach. Informally, existing weight-sharing approaches

[22, 9, 13, 8] can be classified into four categories based on two dimensions: stage levels and multi-path support, as shown in Fig. 2. Specifically, DARTS [22] and Fair DARTS [9] both are one-stage methods while the latter allows multiple paths between any two nodes. One-shot methods [13, 8] typically train a supernet in the first stage and evaluate submodels in the second stage for model selection. So far, the two-stage one-shot methods mainly consider a single-path search space except [2] which involves strong regularization and careful hyper-parameter tuning for supernet training. It is thus natural to study this multi-path counterpart further. MPNAS [35] resorts to reinforcement learning as in [26] for a multi-path searching but verified only on small datasets. ScaleNAS [6] adopts an evolutionary algorithm in a multi-scale search space but it is designed for downstream tasks like segmentations and human pose estimation.

Moreover, it's reasonable to design such a search space regarding two factors. **First**, multi-path feature aggregation is proven to be useful, such as Inception series [30, 16, 31, 29] and ResNeXt [37]. **Second**, it has the potential to balance the trade-off between performance and cost.
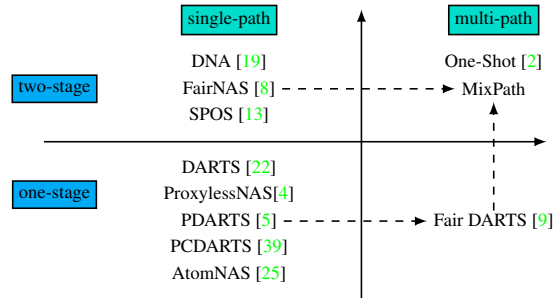

Figure 2: A brief taxonomy of weight-sharing NAS. MixPath fills the empty space of two-stage multiple-path methods.

Without loss of generality, say an inverted bottleneck has an input feature of $C_{in} \times H \times W$, whose number of intermediate and output channels are C_{mid} and C_{out} (same as C_{in}) respectively. Its computational cost can be formulated as $c_{total} = 2HWC_{in}C_{mid} + k^2 HWC_{mid} = 2HWC_{mid}(C_{in} + \frac{k^2}{2})$, where k is the kernel size of the depthwise convolution. Usually, the value of k is set to 3 or 5. Since C_{in} typically dominates $\frac{k^2}{2}$, we can boost the representative power of depthwise transformation by mixing more kernels with a neglectable increased cost. This design can be regarded as a straightforward combination of MixConv [34] and ResNeXt [37].

Multi-path supernet is very hard to train. Vanilla training of the multi-path supernet suffers from severe training difficulties. We can simply train a multi-path supernet by randomly activating a multi-path model at a single step. Here we randomly activate or deactivate each operation. However, this training process is very unstable according to our pi-

lot experiments conducted with the MixPath supernet on ImageNet [11], shown by the blue line in Fig. 3 (a).
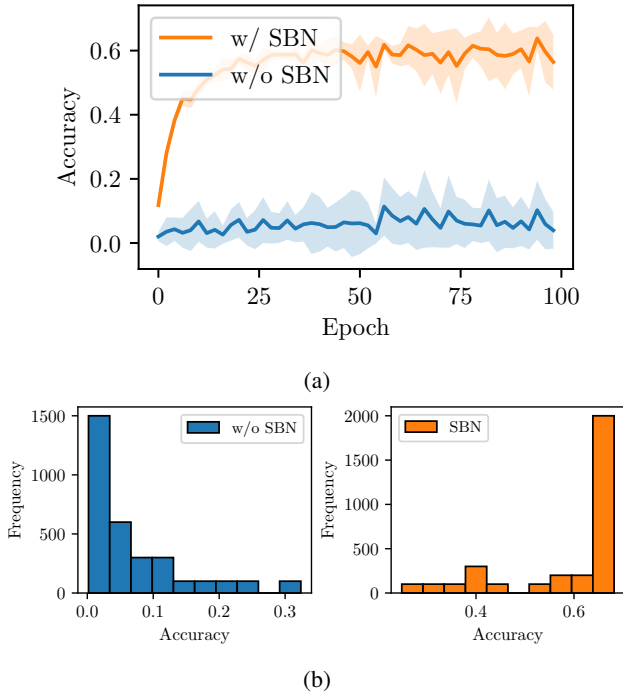

(a)

(b)

Figure 3: **(a)** Average one-shot validation accuracies (solid line) and their variances (shade) when training the MixPath supernet in S_2 (activating at most $m = 2$ paths in each MixPath block) on ImageNet. Twenty models are randomly sampled every two epochs. **(b)** Histogram of accuracies of randomly sampled 3k one-shot models. One-shot models generally come with good performance when SBNs are enabled during supernet training.

What might have caused such a problem? It has been pointed out that high cosine similarity of features from different operations is important for training stability [8]. To verify the issue, we draw the cosine similarity matrix of different features from our trained supernet in the top row of Fig. 4 (a). Surprisingly, not only are the features from single-path similar to each other, but also the multi-path features where they get added up. We discover that cosine similarity is limited because it measures the orientation only. The addition of high-dimensional feature vectors does not change their angles too much (as they are similar), but their magnitudes get scaled up, see blue arrows in Fig.4 (b). We can further make a bold postulation that *a simple superposition of multiple vectors renders very dynamic statistics, hence it causes training instability*. With this motivation in mind, we next scale the feature vectors down to the same magnitude to verify if we can make it stabilized.

3.2. Regularizing Multi-path Supernet

First of all, we can think of using a large amount of BNs to track the changing statistics. For a thorough regularization, we adopt multiple standard BNs to regularize feature statistics from different combinations of m paths. We call them *shadow batch normalizations* (SBNs) since they follow the activated paths combinations as shadows. Strictly speaking, in order to regularize all feature combinations in a multi-path setting, the number of SBNs has to grow exponentially.[1]

3.2.1 Reduce the complexity from 2^m to m.

The above regularization is too costly. As [8] indicates for the single path case, the outputs of different operations at the same layer have higher similarities. It motivates us to reduce the number of shadow BN by diving into the underlying mechanisms further. Informally, if the features of every single path are similar (both magnitude and angles), we can derive that their combinations have similar statistics too, then the number of BNs can be reduced to m. Formally, this can be formulated as follows.

Definition 1. *Zero-order condition: Let two high-dimensional random variables $\mathbf{y} = \mathbf{f}(\mathbf{x}) \in \mathbf{R^{M \times d_1}}$ and $\mathbf{z} = \mathbf{g}(\mathbf{x}) \in \mathbf{R^{M \times d_1}}$, we say that $\mathbf{y}$ and $\mathbf{z}$ satisfy the zero-order condition if $|\mathbf{y} - \mathbf{z}|_{\mathbf{2}} \leq \epsilon$ for any valid sample $\mathbf{x} \in \mathbf{R^{N \times d}}$, where ϵ is a very small positive float number.*

Lemma 3.1. *Let $\mathbf{m}$ be the maximum number of the activable paths, and each pair of operations satisfy the zero-order condition, we can use $\mathbf{m}$ kinds of expectation and variance to approximate all combinations (2^m).*

The proof is provided in A.1 (appendix). Note that the magnitude of ϵ largely affects the approximation error. It is pretty straightforward as the expectations and variances are simply scaled up according to the number of additions. For instance, $\mathbb{E}[\mathbf{y} + \mathbf{z}] \approx \mathbb{E}[\mathbf{u} + \mathbf{v}] \approx 2\mathbb{E}[\mathbf{y}]$. Lemma 3.1 assures us that we can take $SBN_i (i = 1, 2, \cdots, m)$ to track the combination that contains i paths. For the multi-path setting, we can observe similar results in Figure 4, which means the requirement of Definition 1 is also satisfied.

Statistical analysis of Shadow Batch Normalization. We look into the statistics from the supernet trained on CIFAR-10[2] in search space S_1 (elaborated later in Section 4.3) (appendix). Without loss of generality, we set $m = 2$ and collect statistics of the four parameters of SBNs across all channels for the first choice block in Fig. 5. Note SBN_1 captures the statistics of one path, so does SBN_2 for two paths. Based on the theoretical analysis of Section 3.2, we should have $\mu_{bn_1} \approx 0.5\mu_{bn_2}$ and $\sigma_{bn_1}^2 \approx 0.25\sigma_{bn_2}^2$. This

---

[1] $C_m^1 + C_m^2 + \cdots + C_m^m = 2^m - 1$

[2] It's cheaper to train the supernet on CIFAR-10, and similar observations are also reproducible on ImageNet.
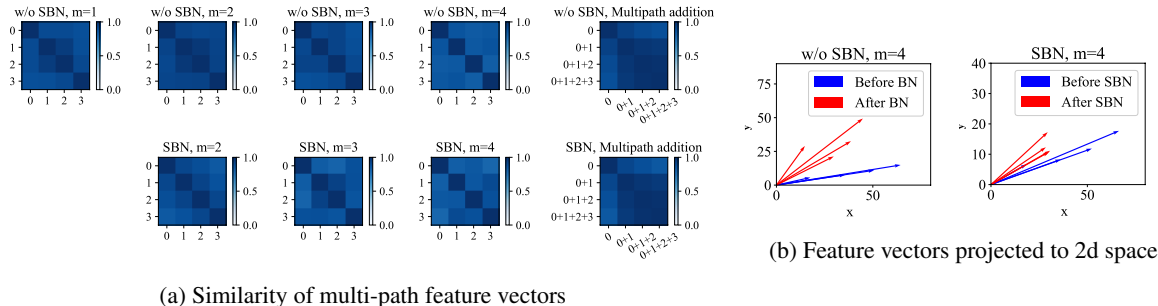
(a) Similarity of multi-path feature vectors

(b) Feature vectors projected to 2d space

Figure 4: **(a)** Features from various path combinations are quite alike (w/ and w/o SBNs). Shown here is the cosine similarity map of first-block feature vectors from the supernets (in S_1). Darker means higher similarity. **(b)** SBNs regularize the magnitudes of feature vectors (projected into 2-dimensional space here) while vanilla BN cannot. Blue arrows represent the feature vectors obtained before SBNs or a single BN, and red ones after them. Here SBNs transform the magnitudes (19.6, 37.2, 53.7, 68.7) to around 32. BN normalized magnitudes (32.6, 36.6, 50.6, 66.7) are still diverse.

is consistently observed in the first two columns of Fig. 5. It's also interesting to see that the other two learnable parameters β and γ are quite similar for SBN_1 and SBN_2. This is explainable as SBNs have transformed different statistics to a similar distribution, the parameters of β and γ become very close. Similar results are observed when $m = 3, 4$ and in different layers in Fig. 3 and Fig. 4 in C.6 (appendix).

3.3. MixPath Neural Architecture Search

Based on the above analysis, we train our multi-path supernet with *shadow batch normalizations*, which we call the MixPath supernet. Following One-Shot [2], the supernet is used to evaluate models in the predefined search space, which makes our approach a two-stage pipeline: 1) training supernet with SBNs and 2) searching for competitive models. Specifically, in the training stage, we apply random sampling so that multiple paths can be activated at each step, where they add up to a mixed output. We then attach a corresponding SBN to the current path combination. This process is detailed in Algorithm 1 (appendix). Next, similar to [9], we progress with a well-known evolutionary algorithm NSGA-II [10] for searching. In particular, our objectives are to maximize classification accuracy while minimizing the FLOPS. In addition, batch normalization calibration [2, 13, 25], as an orthogonal post-processing trick, is verified effective to further improve the ranking ability.

4. Experiments

4.1. Search spaces

We investigate four search spaces in this paper, from S_1 to S_4. For searching on CIFAR-10, we use S_1 that contains 12 inverted bottleneck blocks. For ImageNet, we search in S_2 and S_3 that are similar to MnasNet [32] and MixNet [34]. For Kendall Tau analysis, we devise a subset of a common benchmark NAS-Bench-101 [40], denoted as S_4. The details

---

Algorithm 1 : Stage 1 Supernet training with SBN.

Input: search space $S_{(n,L)}$ with n choice paths per layer and L layers in total, maximum optional paths m, supernet parameters $\Theta(n, L)$, training epochs N, training data loader D, loss function $Loss$.

Initialize every $\theta_{j,l}$ in $\Theta_{(m,L)}$, SBN set $\{SBN_1, SBN_2, \cdots, SBN_n\}$ in each layer.

for $i = 1$ **to** N **do**
  for $data, labels$ **in** D **do**
    for $l = 1$ **to** L **do**
      Randomly sample a number m' between 1 and m, then randomly sample m' paths without replacement, where each has output $\mathbf{o}_i^l$, and use $SBN_{m'}$ to act on the sum of outputs: $\mathbf{o}^l = SBN_{m'}\left(\sum_i \mathbf{o}_i^l\right)$
    end for
    Select $model$ from above sampled index.
    Clear gradients recorder for all parameters.
    Calculate gradients for $model$ based on $Loss$, $data$, $labels$, update $\theta_{(m,L)}$ by gradients.
  end for
end for

---

of these search spaces are listed in Table 1 (appendix).

4.2. Ranking ability analysis on NAS-Bench-101

To show that SBNs can stabilize the supernet training and improve its ranking, we score our method on a subset of a NAS-Bench-101 [40]. Each model is stacked by 9 cells, and every cell has at most 5 internal nodes. We make some adaptations to our method. The first 4 nodes are used to make candidate paths, each has 3 possible operations: 1×1 Conv, 3×3 Conv, and 3×3 Maxpool. The outputs of selected paths are summed up to give input to the fifth node (assigned as 1×1 Conv), after which the proposed SBNs are used.
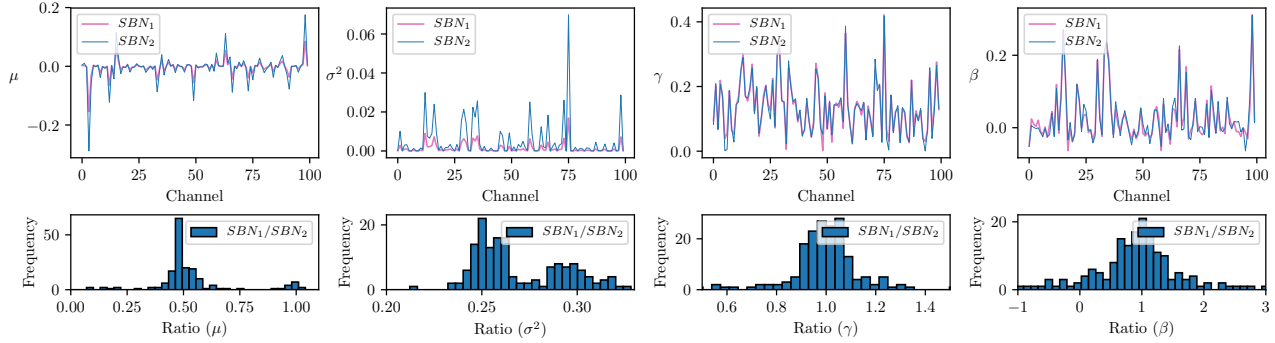
Figure 5: **Top**: Parameters $(\mu, \sigma^2, \gamma, \beta)$ of the first-layer SBNs in MixPath supernet trained on CIFAR-10 when at most $m = 2$ paths are activated. Specifically, SBN_1 and SBN_2 represent SBNs for a single path and two paths. **Bottom**: Histogram of ratios (SBN_1/SBN_2) for each BN parameter. They center around (0.5, 0.25, 1, 1). Note the mean of SBN_2 is roughly twice as that of SBN_1. It's interesting that we have previously forecasted this result.
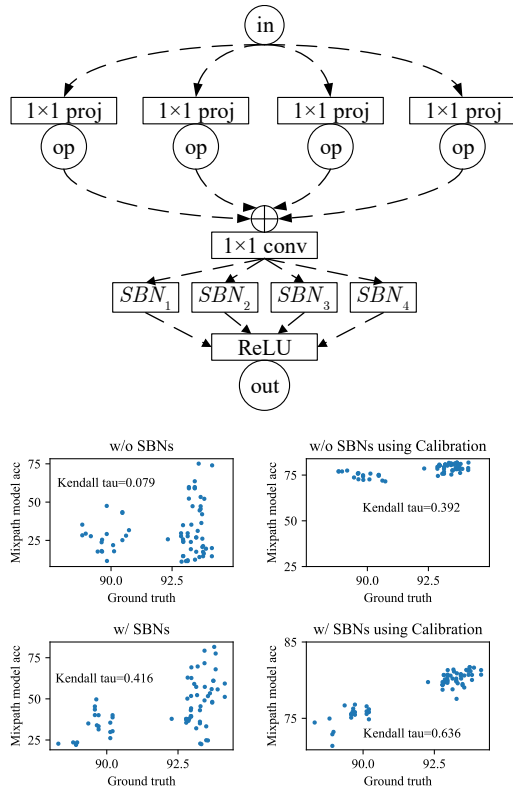
The designed cell space is shown in Fig. 6.

Figure 6: **(a)** Cell space with SBNs in NAS-Bench-101 [40] **(b)** Ground truth accuracies vs. predictions by one-shot MixPath supernets ($m = 4$) trained with various strategies. We sample 70 models in NAS-Bench-101 [40] (sampling one time) to get the ground-truth. The supernet trained SBN with post BN calibration (top right) gives the best prediction. A vanilla BN alone (w/ SBNs) wrongly pushes models into a narrow range under 0.2.

Firstly, we conduct the baseline experiments on NAS-Bench-101 [40], training with and without SBNs. For the latter, the features from the multi-path are summed up and then divided by the number of activated paths, following a 1×1 Conv-BN-ReLU (i.e., a single BN).

To deeply analyze the model ranking capacity of our method and to understand the effects of SBNs, we conduct two pairs of experiments: (i) vanilla BN vs. SBN (ii) post–calibrated vanilla BN vs. post-calibrated SBN. For all experiments, we train the supernet for 100 epochs with a batch size of 96 and a learning rate of 0.025 on three different seeds. We randomly sample 70 models to find their ground-truth top-1 accuracy from NAS-Bench-101 to obtain Kendall tau [17]. Compared with FairNAS [8] and SGAS [20] utilizing about 10-20 models to calculate Kendall-tau, 70 models are adequate enough to evaluate the ranking performance. We use $m = 4$ in this section.[3] The results are shown in Table 1 and Fig. 6 (b).

Table 1: Comparison of Kendall taus between MixPath supernets ($m = 4$) trained with various strategies. For each control group, we sampled 70 models from NAS-Bench-101 [40] and repeated it 3 times on different seeds. [†]: after BN calibration.

Strategy	w/o Calibration	w/ Calibration
w/o SBN	0.117 ± 0.027	0.382 ± 0.007
w/ SBN	0.393 ± 0.037	$\mathbf{0.597 \pm 0.037}$

Notably, BN post-calibration can boost Kendall tau in each case, which confirms the validity of this post-processing trick [2, 13, 25]. Compared with baseline experiments, the supernet trained with SBNs has a much stronger ranking capacity. Even without calibration, the proposed method

---

[3]We place $m = 3$ case in ablation study.

still ranks architectures better than the calibrated vanilla BN (+0.011). The composition of two tricks can further boost the final score to 0.597. Next, we will search for models in the search space commonly used for CIFAR-10 [18] and ImageNet [11] dataset respectively.

4.3. Search on CIFAR-10

We conduct experiments on CIFAR-10 dataset [18] with a designed search space S_1 containing 12 inverted bottlenecks [27], each of which has four choices of kernel size (3, 5, 7, 9) for the depthwise layer and two choices (3, 6) for expansion rates. Hence, the size of search space S_1 ranges from 8^{12} (m=1) to 162^{12} (m=4). For each case, we train the supernet on CIFAR-10 for 200 epochs. The batch size is set to 96 and we use the SGD optimizer with a momentum of 0.9 and 3×10^{-4} weight decay. We also use a cosine scheduler for the learning rate strategy with an initial value of 0.025. It only takes about 6 GPU hours on a single Tesla V100 GPU in total. The comparison with recent state-of-the-art models is listed in Table 2. Our searched model MixPath-c obtains 97.4 % top-1 accuracy using 493M FLOPS on CIFAR-10, whose architecture is shown in Fig. 5 in C.8 (appendix).

Table 2: Comparison on CIFAR-10. †: MultAdds computed from the provided code. *: transferred from ImageNet (search cost is on ImageNet). RL: Reinforcement learning, GD: Gradient-based, OS: one-shot, TF: Transferred.

Models	Params (M)	$\times+$ (M)	Top-1 (%)	Type	Cost $G \cdot d$
NASNet-A [48]	3.3	608	97.4	RL	1800
NASNet-A Large [48]*	85.0	12030	98.0	TF	1800
ENAS [26]	4.6	626†	97.1	RL	0.5
DARTS [22]	3.3	528	97.1	GD	0.4
RDARTS [44]	-	-	97.1	GD	1.6
SNAS [38]	2.9	422	97.0	GD	1.5
GDAS [12]	3.4	519	97.1	GD	0.2
P-DARTS [5]	3.4	532	97.5	GD	0.3
PC-DARTS [39]	3.6	558	97.4	GD	0.1
DARTS- [7]	3.5	568	97.5	GD	0.4
Few-shot DARTS [45]	3.8	-	97.6	GD	1.3
MixPath-c (ours)	5.4	493	97.4	OS	0.25
MixNet-M [34]*	3.5	360	97.9	TF	≈3k
MixPath-a (ours)*	3.5	348	98.1	TF	10
MixPath-b (ours)*	3.6	377	98.2	TF	10

4.4. Proxyless Search on ImageNet

We search proxylessly on ImageNet [11] in the search space of MnasNet [32], as this layer-wise search space with inverted bottleneck blocks [27] is also very commonly used [32, 33, 36, 8, 34]. We fix the expansion rate as in [34] and search for the combinations of various depthwise convolutions. Specifically, we search from the kernel sizes (3, 5,

7, 9) and their combinations. We call this search space S_2. Moreover, we also construct a search space S_3 to incorporate architectures like MixNet [34], where we evenly divide the channel dimension of the depthwise layer into 4 groups and search from the kernel sizes $(3, 5, 7, 9)$ for each group. We set $m = 2$ (in S_2) and $m = 1$ (in S_3). In each case, we utilize the same hyper-parameters: a batch size of 1024 and the SGD optimizer with 0.9 momentum and a weight decay of 10^{-5}. The initial learning rate is 0.1 with cosine decay and it decreases to 0 within 120 epochs. This takes about 150k times of backpropagation (10 GPU days on Tesla V100). Once the supernet is trained, we adopt NSGA-II [10] for model selection.

As per training stand-alone models, we use the same tricks as MixNet-M [34], the results are listed in Table 3. To make fair comparisons, we disable distillation in NAS approaches because it can marginally boost the performance of found models. Our cost in Table 3 includes training supernet (10 GPU days) and NSGA-II (0.3 GPU day). MixPath-A, sampled from the Pareto front in S_3 (so is MixNet), uses 349M multiply-adds to obtain 76.9% accuracy on ImageNet. MixPath-B, in S_2, uses fewer FLOPS and parameters w.r.t EfficientNet-B0 to obtain 77.2%. Note SPOS, FBNet, ProxylessNAS are also in S_2 or similar. In the appendix we list the **detailed structures** of two models in Fig. 5 of C.8.

Transferability. We also transfer these models to CIFAR-10 and *COCO* detection, please refer to Table 2 and C.4.

5. Ablation Study and Discussions

5.1. Supernet ranking with different strategies

To study the individual contribution of SBN and BN post-calibration, we profit from NAS-Bench-101 [40] for ranking calculation since it provides ground truths for all models. Moreover, it allows a fair comparison with other approaches in the same search space. Specifically, we investigate the case $m = 3$ and report the result of each control group using 3 different seeds. Kendall tau [17] is calculated by randomly sampled 70 models.

Supernet with and without SBNs. Table 4 shows that the supernet trained without SBN is inadequate to evaluate sampled models ($\tau = 0.05$), which is much lower than the one with a linear number of SBNs ($\tau = 0.32$). We further quantify the effect of the additional post BN calibration, and both taus can be boosted. Particularly for SBN, it's improved to 0.59 (+0.27). Therefore, the combination of SBN and post BN calibration delivers the best ranking performance.

Linear number of SBNs vs. exponential. Through theoretical analysis, we conclude that a linear number of SBNs are enough to match all the combinations of activable paths, while we can still use its exponential counterpart where every single combination follows an SBN. Table 4 shows that

Table 3: Comparison with state-of-the-art models on ImageNet. For fair comparisons, we disable distillations for all methods. Search cost is computed by GPU days. [†]: TPU days. [‡]: w/ distillation.

Models	×+ (M)	#P (M)	Top-1 (%)	Top-5 (%)	Cost G·d
MobileNetV2 [27]	300	3.4	72.0	91.0	-
ShuffleNetV2 [24]	299	3.5	72.6	-	-
GhostNet 1.3× [14]	226	7.3	75.7	92.7	-
DARTS [22]	574	4.7	73.3	-	0.5
P-DARTS[5]	557	4.9	75.6	92.6	0.3
PC-DARTS [39]	586	5.3	74.9	92.2	0.1
SGAS (Cri.2 best) [20]	598	5.4	75.9	92.7	0.25
FairDARTS-C [9]	380	4.2	75.1	92.4	0.4
MnasNet-A2 [32]	340	4.8	75.6	92.7	≈4k
One-Shot Small (F=32) [2]	-	5.1	74.2	-	4[†]
FBNet-B [36]	295	4.5	74.1	-	9
MobileNetV3 [15]	219	5.4	75.2	92.2	≈3k
Proxyless-R [4]	320	4.0	74.6	92.2	8.3
FairNAS-A [8]	388	4.6	75.3	92.4	12
Single Path One Shot [13]	328	3.4	74.9	92.0	12
Single-Path NAS [28]	365	4.3	75.0	92.2	1.25
OFA w/o PS [3]	235	4.4	72.4	-	50
OFA [‡]n [3]	230	-	76.0	-	50
BigNAS-M [‡] [41]	418	5.5	77.4	93.5	-
MixNet-M [34]	360	5.0	77.0	93.3	≈3k
MixPath-A (ours)	349	5.0	76.9	93.4	10.3
MixPath-B (ours)	378	5.1	77.2	93.5	10.3

Table 4: Kendall taus' comparison between MixPath supernets ($m = 3$) trained with multiple strategies. (same settings as Table 1). ESBN: Exponential SBNs, LSBN: Linear SBNs

τ	w/o SBN	ESBN	LSBN
w/o Calibration	0.05±0.06	0.37±0.06	0.32±0.03
w/ Calibration	0.43±0.03	0.34±0.08	**0.59±0.02**

linear SBN with BN calibration works best (with the highest τ), while Exp SBN (exponential) performs best if no calibration is applied. Why is it so? Linear SBNs make use of the zero-order condition which is an approximate relationship. Exp SBNs are more accurate since they catch statistics from each of the combinations. However, this makes a wider distribution of learnable parameters γ, β, which is difficult for BN calibration to fit them well. Therefore, we choose Linear SBNs with BN calibration instead. Regarding the Occam's Razor, the linear growing BN generalizes better than the exponential counterpart.

One-shot accuracy distribution of candidate models w.r.t. m We run four experiments with $m = 1, 2, 3, 4$ to investigate the effect of SBNs in S_1, other settings are kept the same. In total, 1k models are randomly sampled to plot their test accuracy distributions on CIFAR-10 for each m

in Fig. 7. When $m = 1$, MixPath falls back to single path. Whereas, SBN begins to demonstrate its power for $m > 1$, whose absence leads to a bad supernet with lower predicted accuracies and a much larger gap. This indicates the supernet without SBNs severely underestimates the performance of a large proportion of architectures.

Simple scaling vs. SBNs. Intuitively from the above motivation, simply scaling the magnitudes of all features to the same level in each block could also stabilize the supernet training. We hereby treat *simple scaling* as a baseline experiment. However, it is somewhat limited because it adjusts magnitudes only. We find that ranking ability after the calibration of the simple scaling case is higher than that of vanilla BNs and exponential SBNs, but it is lower than that of linear SBNs. This is consistent with the baseline experiment in Table 5. Our proposed method with linear SBNs is still the best among all the compared strategies. Note that the simple scaling approach can only roughly scale the magnitudes of the output. In contrast, SBNs can adjust the magnitudes and angles together to better capture the changing distributions.

Table 5: Comparison of Kendall taus between MixPath supernets ($m = 4$) trained with various strategies. We sampled 70 models from NAS-Bench-101 [40] and repeated it 3 times on different seeds. [†]: after BN calibration

Strategy	Simple Scaling	w/ SBNs
w/o Calibration	0.293±0.016	0.393±0.017
Calibration	0.472±0.021	**0.597±0.037**

Applying SPOS on multi-path search space. We design a "branch composition" experiment $m = 4$ in NAS-Bench-101 with 3 candidate operations, leading to 34 possible blocks. We use SPOS [13] to train this supernet with the same hyper-parameters as ours, but we only get a Kendall tau of **-0.225** while our method obtains **0.504** in terms of ranking the same sub-models (see Fig. 8). This result is not surprising. We blame the failure of the former approach for the use of an excessive number of blocks, which brings nearly **4×** more trainable parameters in the supernet (199.7M vs. ours 52.9M). In this case, it's impractical and infeasible for SPOS to work under such setting because it's hard to make candidate blocks fully optimized. In effect, thus-trained supernet is unable to ensure a reliable rank. For example, on average, one block receives an update every 34 iterations, resulting in poor ranking performance. Besides, this approach will encounter memory explosion at a growing m. Even if we can solve the ranking issue, it requires much more GPU memory to save the trainable parameters, which prevents proxyless search on large datasets or big models. In contrast, our method doesn't suffer from these issues.
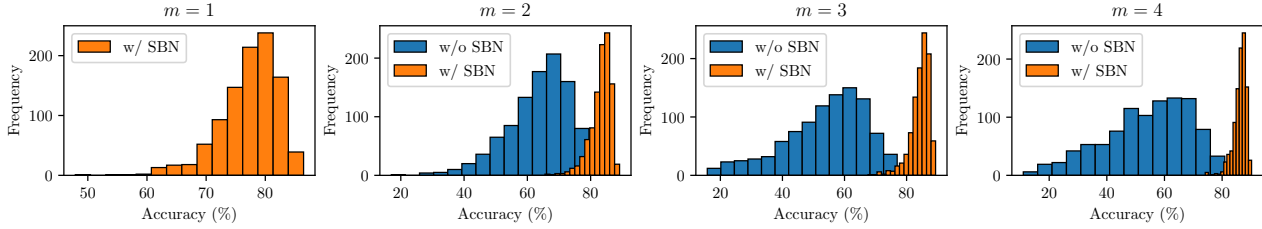
Figure 7: Histogram of one-shot model accuracies sampled from MixPath supernets trained on CIFAR-10 in S_1, activating at most $m = 1, 2, 3, 4$ paths respectively. In all cases, the supernet trained without SBNs severely underestimates a large proportion of candidate architectures. Note when $m = 1$ it falls back to a vanilla BN.
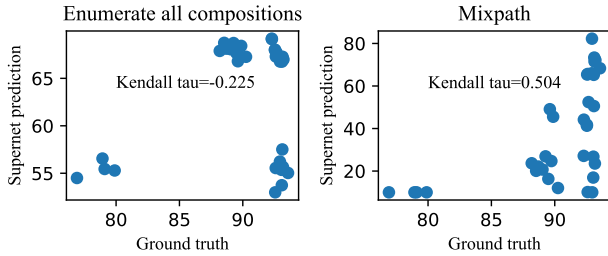

Figure 8: Rank performance comparison between 'enumerating all compositions' and our method.

5.2. Comparison of search strategies

We use the search space S_3 on ImageNet to compare the adopted NSGA-II search algorithm [10] with random search by charting the Pareto-front of models found by both methods in Fig. 1 (appendix). NSGA-II has an advantage that the elite models have higher accuracies and fewer FLOPs.

5.3. Pipeline component analysis

Apart from the ablation studies on NAS-Bench-101 [40], we make a component analysis of our pipeline in S_1. We focus on the supernet training with and without SBN by fixing the second-stage searching algorithm. Each pipeline is run three times on different random seeds. Our goal is to find the best models under 500M FLOPS. As a result, with SBN enabled, we obtain 97.44% top-1 accuracy, higher than the best 97.12% when SBN is disabled. This confirms that it mainly benefits from our novel training mechanism.

5.4. Discussions

Why can SBNs stabilize supernet training and improve ranking ability? We have verified that high cosine similarity is not the only factor that guarantees the stable training of a multi-path supernet. It is more important to ensure that the feature distributions are consistent. Otherwise, the training is disrupted because a single BN is not enough to capture changing statistics from multiple paths. The proposed SBNs are able to track a variety of distribu-

tions, alleviating this instability. Empirically, better training of the supernet also gives more appropriate weights for each subnetwork, with which we can better estimate the ground-truth performance of the stand-alone models.

Why can SBNs work together with BN post-calibration? According to the analysis in Section 3.2, the parameters μ, σ of SBNs that follow a multi-path combination are multiples of those of a single-path, as shown in Fig. 5. But not all the parameters fit well into this relationship, post BN calibration can then readjust the mean and variance of the inputs to fit the learned parameters γ and β. To some extent, it can make up for the gap from the supernet to submodels. Noticeably, SBN is a type of regularization technique during the supernet training, while BN calibration is a post-processing trick. They can work in an orthogonal way and both contribute to the ranking.

Comparison with Switchable Batch Normalization Switchable BN [42] is only applied to learn the statistics of a limited number (K) of sub-architectures with $K = \{4, 8\}$ channel configurations. Our SBN is designed to match the changing feature statistics from a more flexible combination of different choice paths, while the number of channels is fixed. The former is used for network pruning while our SBN is to stabilize the training of the multi-path supernet.

6. Conclusion

In this paper, we propose a unified approach for two-stage one-shot neural architecture search, with a multi-path search space enabled. Existing single-path approaches can thus be regarded as a special case. The proposed method uses SBNs to catch the changing features from various branch combinations, which successfully solves two difficulties of vanilla multi-path methods: the unstable training of supernet and the unbearable weakness of model ranking. Moreover, by exploiting feature similarities from different path combinations, we reduced the number of required exponential SBNs to m, identical to the number of allowed paths. Our method can enjoy the model ranking capacity of the one-shot supernet and robustly search good models across various benchmarks.

References

[1] Haoping Bai, Meng Cao, Ping Huang, and Jiulong Shan. Batchquant: Quantized-for-all architecture search with robust quantizer. *Advances in Neural Information Processing Systems*, 34:1074–1085, 2021. 2

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and Simplifying One-Shot Architecture Search. In *ICML*, pages 549–558, 2018. 2, 4, 5, 7

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. 7

[4] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *ICLR*, 2019. 2, 7

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. In *ICCV*, 2019. 2, 6, 7

[6] Hsin-Pai Cheng, Feng Liang, Meng Li, Bowen Cheng, Feng Yan, Hai Li, Vikas Chandra, and Yiran Chen. Scalenas: Multi-path one-shot nas for scale-aware high-resolution representation. In *First Conference on Automated Machine Learning (Main Track)*, 2022. 2

[7] 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. 6

[8] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. In *ICCV*, 2021. 1, 2, 3, 5, 6, 7

[9] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. In *ECCV*, 2020. 1, 2, 4, 7

[10] 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. 4, 6, 8

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, pages 248–255. IEEE, 2009. 3, 6

[12] Xuanyi Dong and Yi Yang. Searching for a Robust Neural Architecture in Four GPU Hours. In *CVPR*, pages 1761–1770, 2019. 6

[13] 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 *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 1, 2, 4, 5, 7

[14] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. GhostNet: More Features from Cheap Operations. In *CVPR*, 2020. 7

[15] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *ICCV*, 2019. 7

[16] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, pages 448–456, 2015. 2

[17] Maurice G Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1/2):81–93, 1938. 2, 5, 6

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009. 6

[19] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely Supervised Neural Architecture Search with Knowledge Distillation. *arXiv preprint arXiv:1911.13053*, 2019. 1, 2

[20] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Müller, Ali Thabet, and Bernard Ghanem. SGAS: Sequential Greedy Architecture Search. In *CVPR*, 2020. 2, 5, 7

[21] 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*, pages 19–34, 2018. 2

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *ICLR*, 2019. 1, 2, 6, 7

[23] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural Architecture Optimization. In *NIPS*, pages 7816–7827, 2018. 2

[24] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical Guidelines for Efficient CVV Architecture Design. In *ECCV*, pages 116–131, 2018. 7

[25] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Yang Jianchao. AtomNAS: Fine-Grained End-to-End Neural Architecture Search. In *ICLR*, 2020. 2, 4, 5

[26] 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, 6

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*, pages 4510–4520, 2018. 6, 7

[28] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-Path NAS: Designing Hardware-Efficient ConvNets in less than 4 Hours. In *ECML PKDD*, 2019. 7

[29] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI*, 2017. 2

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, pages 1–9, 2015. 1, 2

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, pages 2818–2826, 2016. 2

[32] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-Aware Neural Architecture Search for Mobile. In *CVPR*, 2019. 1, 2, 4, 6, 7

[33] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, 2019. 6

[34] Mingxing Tan and Quoc V. Le. MixConv: Mixed Depthwise Convolutional Kernels. In *BMVC*, 2019. 2, 4, 6, 7

[35] Qifei Wang, Junjie Ke, Joshua Greaves, Grace Chu, Gabriel Bender, Luciano Sbaiz, Alec Go, Andrew Howard, Ming-Hsuan Yang, Jeff Gilbert, et al. Multi-path neural networks for on-device multi-domain visual classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3019–3028, 2021. 2

[36] 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. 6, 7

[37] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In *CVPR*, pages 1492–1500, 2017. 1, 2

[38] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic Neural Architecture Search. In *ICLR*, 2019. 6

[39] 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 *ICLR*, 2020. 2, 6, 7

[40] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *ICML*, pages 7105–7114, 2019. 2, 4, 5, 6, 7, 8

[41] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pages 702–717. Springer, 2020. 7

[42] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable Neural Networks. In *ICLR*, 2019. 2, 8

[43] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating The Search Phase of Neural Architecture Search. In *ICLR*, 2020. 2

[44] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 6

[45] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *International Conference on Machine Learning*, pages 12707–12718. PMLR, 2021. 6

[46] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial Distribution Learning for Effective Neural Architecture Search. In *ICCV*, pages 1304–1313, 2019. 2

[47] Barret Zoph and Quoc V Le. Neural Architecture Search with Reinforcement Learning. In *ICLR*, 2017. 2

[48] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. In *CVPR*, volume 2, 2018. 2, 6