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

Xiangxiang Chu[1], **Xudong Li**[2†*], **Yi Lu**[2‡*], **Bo Zhang**[1*], **Jixiang Li**[1]

[1]Xiaomi AI Lab

[2]University of Chinese Academy of Sciences

[1]{chuxiangxiang, zhangbo11, lijixiang}@xiaomi.com [†]{lixudong16@mails.ucas.edu.cn}
[‡]{mmmr.lu@outlook.com}

Abstract

The expressiveness of search space is a key concern in neural architecture search (NAS). Previous approaches are mainly limited to searching for single-path networks. Incorporating multi-path search space with the one-shot doctrine remains untackled. In this paper, we investigate the supernet behavior under the multi-path setting, which we call MixPath. For a sampled training, simply switching multiple paths on and off incurs severe feature inconsistency which deteriorates the convergence. To remedy this effect, we employ what we term as *shadow batch normalizations* (SBN) to follow various path patterns. Experiments performed on CIFAR-10 show that our approach is effective regardless of the number of allowable paths. Further experiments are conducted on ImageNet to have a fair comparison with the latest NAS methods. Our code will be available here[1].

1 Introduction

Carrying a high promise of complete automation in network design across various domains [6, 8, 14, 23, 24, 26, 29, 31], the research on neural architecture search now enters into a stage of fierce competition [5, 22, 25, 32, 33].

Among various mainstream paradigms, one-shot approaches [1, 2, 7, 15, 22] make use of weight-sharing mechanism that reduces a large amount of computational cost. Typically in its first stage, a supernet is trained to convergence to serve as an evaluator for sub-models' performance. The second searching stage can either be done with EA or RL, even random sampling. It is thus of utter importance for the supernet evaluator to have accurate ranking ability. FairNAS [7] discusses throughly on this regard, arguing that fairness training for each sampled blocks contributes to the final ranking. As each of its searchable cells is independently supervised by the corresponding block in the teacher network, randomness is largely reduced, also each cell is well-trained, both improves its ranking skill.

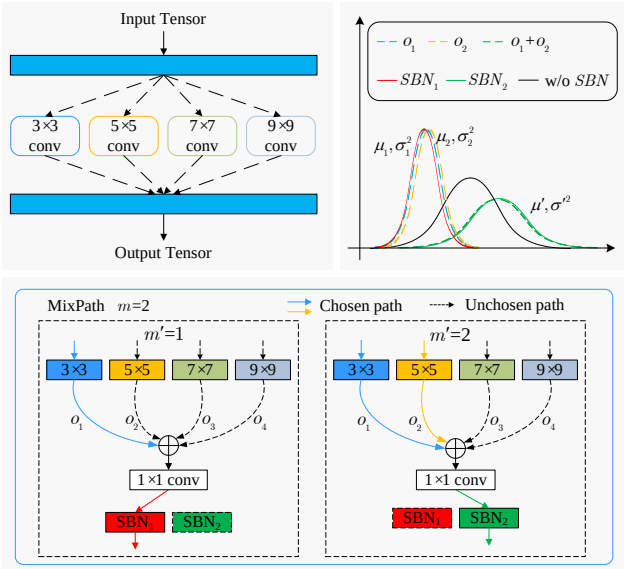

Figure 1: Our MixPath framework for supernet training. **Left**: macro architecture. **Middle**: the input tensor is split into four groups, and at most m paths are chosen in each groupfor thhere will be m SBNs: $\{SBN_1, SBN_2, \ldots, SBN_m\}$. Here we take $m = 2$ for example. Actually, we will choice $m' \leq m$ paths in each MixPath, thus $SBN_{m'}$ will be chosen. **Right**: shadow BN can correctly catch the statistics of varying activated paths with two modes, however, vanilla BN wrongly catches the mixed version.

Exploring multi-path search space is made possible in a differentiable method Fair DARTS [9]. However, it poses a challenge to think of its one-shot counterpart. It is a non-trivial problem. One-shot [1] can be thought of multi-path training as it dynamically drops paths from the supernet, which reportedly comes with instability even regularization tricks and recalibration of Batch Normalization statistics don't help much. In this paper, we dive into its real causes and undertake a unified approach, which we call MixPath, to incorporate most of the preceding one-shot works, as well as increased multi-path capability.

Our contributions can be summarized into four aspects.

- We propose a uniform approach for one-shot NAS to step out of the existing single-path limitations and em-

*Equal Contribution.

[1]https://github.com/xiaomi-automl/MixPath.git

power multi-path (at most m paths) expressiveness, which bridges the gap between one shot and multi branch searching. From this perspective, existing single-path weight-sharing approaches become a special case of $m = 1$ path .

- We disclose the obstacles that make vanilla multi-path approaches fail and propose a new and light-weight mechanism, shadow batch normalization, to stabilize the training process of the over-parameterized supernet with neglect-able costs. Moreover, it can boost the most critical capacity of supernet: model ranking consistence, which breaks a record on NAS-bench-101 search space.
- We prove that the number of shadow batch normalizations can be greatly reduced to grow linearly with the number (m) of maximum activable paths instead of exponentially, exploiting the underlying mechanics that secures weight-sharing.
- We search proxylessly on ImageNet at a cost of 10 GPU days. The searched models obtain new state-of-the-art results on the ImageNet 1k classification task, which can be compared with MixNet models searched with 300× more computing powers. Moreover, our model MixPath-B, makes use of multi-branch feature aggregation and obtains higher accuracy than EfficientNetB0 at the cost of fewer FLOPS and parameters.

2 Related Works

Weight-sharing Mechanism. In recent one-shot neural architecture search methods which apply the weight-sharing paradigm, intermediate features learned by different operations exhibit high similarities [5, 7]. Ensuring such similarity is important to train a supernet to better convergence with improved stability. For instance, paralleling skip connections with other inverted bottleneck blocks [28] creates large feature discrepancy that harms supernet training. This is rectified by appending an equivariant learnable stabilizer [5] to each skip connection, which is dedicated to boost feature similarities and in turn supernet performance. We can draw a rule of thumb to design advanced one-shot search algorithms: *when a supernet fails to converge, we should first look into its feature similarities*.

Mixed Depthwise Convolution. MixNet [33] proposes a MixConv operator that processes equally-partitioned channels with different depthwise convolutions, which is proved to be effective for image classification. Still, MixNet follows MnasNet [31] for architectural search that comes with immense cost, which is infeasible in practice. AtomNAS [25] incorporates MixConv with variable channel sizes in its search space. To amortize the search cost, it applies the differentiable method DARTS [24] removing dead blocks on the fly. Despite its high performance of the resulted models, such fine-grained channel partitions leads to large incongruence which requires specific treatment on mobile end.

Multi-branch Feature Aggregation. To our knowledge, the first multi-branch neural architecture dates back to ResNet [16] with a skip connection branch for image classification. ResNeXt pushes the multi-branch design further [35], in which homogeneous convolutions are aggregated by addition. Therefore, the combination of mixed depth-wise convolution and multi branch design is reasonable.

Conditional Batch Normalization. Batch Normalization [19] has greatly facilitated the training of neural networks by normalizing layer inputs to have fixed means and variances. In the case of training supernets, a single batch normalization has difficulty to capture dynamic inputs from various heterogenous operations. Slimmable neural networks [40] introduces a shared supernet that can run with switchable channels at four different scales (1×, 0.75×, 0.5×, 0.25×). Training such a network suffers from feature inconsistency at different switches, therefore, they apply independent batch normalizations for each switch configuration to encode conditional statistics. Thus jointly trained supernet enjoy improved accuracies at four scales. However, it requires an increased number of batch normalizations when it comes to arbitrary channel widths, which is impractical because of intensive computation. The following work US-Nets [39] circumvents this issue with distributed computing. In addition, post-statistics for networks of different channel widths are computed on a subset of the target dataset to save more time.

The Model Ranking Correlation. It should be emphasized that the ranking ability for one-shot algorithms is of the uttermost importance, whose sole purpose is to evaluate networks. Previous works like [7, 41] have applied Kendall tau [20] for a clear measure of it.

3 MixPath: A Unified Approach

3.1 Motivation

Informally, existing weight-sharing approaches [5,7,9,15,24] can be classified into four categories based on two dimensions: prior-learning type and multi-path support, as shown in Figure 2. Specifically, DARTS [24] and Fair DARTS [9] both learn priors towards a promising network while the latter allows multiple paths between any two nodes. One-shot methods [5, 7, 15] don't learn priors but train a supernet to evaluate submodels instead. So far, they only consider single-path search space. It is thus natural to devise their multi-path counterpart.

Mixture has potentials to balance the trade-off between the performance and model cost better than the monotonous one. Without loss of generality, the computational cost of an inverted bottleneck with C_{in} input of $H \times W$ features, C_{mid} middle and $C_{out} = C_{in}$ output channels can be formulated as $c_{total} = 2HWC_{in}C_{mid} + k^2HWC_{mid} = 2HWC_{mid}(C_{in}+\frac{k^2}{2})$, where k is the kernel size of the depthwise convolution. Usually the value of k is set 3 or 5. When C_{in} dominates $\frac{k^2}{2}$, we can boost the representative power of depth wise transformation by mixing more kernels with neglect-able cost increase. This design can be regarded as a straightforward combination by MixConv and ResNeXt.

3.2 Multi-path Training Instability and Poor Evaluation Performance

Multi-path support is not as easy as thought to be. We expect to train a supernet that can accurately predict the performance

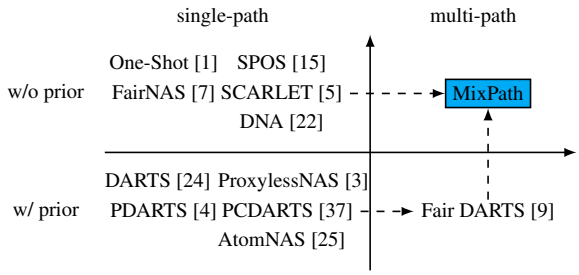

Figure 2: A NAS taxonomy according to multi-path capability and prior-based learning.

of multi-path submodels. To do so, we can think of training the supernet by randomly activating a multi-path model at a single step. This is based on the assumption that weights from multi-path training fit well in a multi-path submodel. Here we apply Bernoulli sampling to independently activate or deactivate each operation. It is advantageous to have a steady training like in single-path methods [7, 15]. However, it is not true according to our pilot experiments conducted in MixPath supernet on the ImageNet dataset [12], where its one-shot models have low accuracies swinging back and forth, as shown by the blue line in Figure 3.

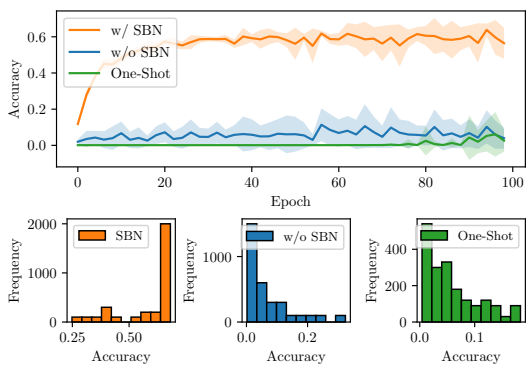

Figure 3: **Top**: One-shot average validation accuracies (solid line) and their variances (shadow) when training the MixPath supernet (activating at most $m = 2$ paths in each MixPath block) and One-Shot [1] on ImageNet. Twenty models are randomly sampled every two epochs. **Bottom**: Histogram of randomly sampled 3k one-shot models' accuracies. Enabling shadow batch normalization improves supernet training and one-shot performance.

One-Shot [1] is also a similar case of multi-path training. By gradually dropping out paths, some operations are activated and others are not, it can be seen as a way of multi-path sampling. Also exhibited in Figure 3, it suffers more severe training difficulty compared to vanilla training of MixPath. In a long term of early epochs, One-Shot fails to learn any useful information to pass onto its one-shot submodel.

3.3 Restore Stability with Shadow Batch Normalization

According to [5, 7], similarities of intermediate features learned by different operations is crucial for the stability of supernet training. In order to solve above problem, an intuitive solution is using variable number of BNs to track the changing features from the combinations of different operations. Take the case where $m = 2$ paths at most, the outputs of two paths are $\mathbf{x}$ and $\mathbf{y}$, and there are five kind of combitions: $\{\mathbf{x}, \mathbf{y}, \mathbf{x}+\mathbf{x}, \mathbf{x}+\mathbf{y}, \mathbf{y}+\mathbf{y}\}$. That is to say there are five different kind of feature combination, thus five BNs are need to track these statistics respectively. We call such BN as Shadow Batch Normalization (SBN), which means shadowing the different features. However, the number of SBNs in such naive multi-path is exponential. Let $\mathcal{S}_l = \{\mathcal{O}_i^l | i = 1, 2, \ldots, n\}$ be the alternative path set in choice block l, $\mathbf{o}_i^l$ be the output of $\mathcal{O}_i^l$, m be the maximum of selected paths. There will be $\sum_i^m C_n^i = \sum_i^m \frac{n!}{i!(n-i)!}$ possible combinations in a choice block, which grows exponentially with m. In fact, if some specific conditions are met, the number of SBNs can be reduced to m. The followings are complete explanations and related proofs, all taking m = 2 as an example.

Let $\mathbf{z}$ be the input images, $\mathbf{x}$ and $\mathbf{y}$ be the outputs of selective path $\mathcal{O}_1$ and $\mathcal{O}_2$ in choice block l respectively. Firstly, two important definitions are given.

Definition 1. ***Condition of Zero Order***: *Given two high-dimension functions,* $\mathbf{x}$=$f(\mathbf{z})$ *and* $\mathbf{y}$=$g(\mathbf{z})$*, we say that* $\mathbf{x}$ *and* $\mathbf{y}$ *satisfy the condition of zero order if* $\mathbf{x} \approx \mathbf{y}$ *for any valid* $\mathbf{z}$*.*

Note that both $\mathbf{x}$ and $\mathbf{y}$ are CNN feature maps with high dimensions such as $H \times W \times C$. Therefore, we reshape them as $1 \times HWC$ and use their cosine similarity to measure the degree of approximation. For Definition 1, considering the mechanism that weight-sharing works, the cosine similarity between the channel-wise feature maps are very high in a steadily trained supernet [5, 7]. Above conclusion leads that $\mathbf{x} \approx \mathbf{y}$, that is to say satisfy the condition of zero order, which also hold in our case.

With above discussion, we can draw the following two lemmas.

Lemma 1. *If* $\mathbf{x}$ *and* $\mathbf{y}$ *satisfy the condition of zero order, the expectation and variance are approximate, i.e.* $\mathbb{E}[\mathbf{x}] \approx \mathbb{E}[\mathbf{y}], Var[\mathbf{x}] \approx Var[\mathbf{y}]$.

Proof. Let $\mathbf{x} \sim p_{\mathbf{x}}(\mathbf{x})$, $\mathbf{y} \sim p_{\mathbf{y}}(\mathbf{y})$, $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$. First we consider the expectation of $\mathbf{x}$ and $\mathbf{y}$:

Because $\mathbf{x}$ and $\mathbf{y}$ are obtained by two functions $f(\mathbf{z})$ and $f(\mathbf{z})$, above equations can be written as:

$$\begin{aligned} \mathbb{E}[\mathbf{x}] &= \mathbb{E}[f(\mathbf{z})] = \int p_{\mathbf{z}}(\mathbf{z}) f(\mathbf{z}) \mathrm{d}\,\mathbf{z} \\ \mathbb{E}[\mathbf{y}] &= \mathbb{E}[g(\mathbf{z})] = \int p_{\mathbf{z}}(\mathbf{z}) g(\mathbf{z}) \mathrm{d}\,\mathbf{z} \end{aligned} \tag{1}$$

According to the condition of zero order, we have $f(\mathbf{z}) \approx g(\mathbf{z})$. And $p(\mathbf{z})$ is same for both $\mathbf{x}$ and $\mathbf{y}$. So we have $\mathbb{E}[\mathbf{x}] \approx \mathbb{E}[\mathbf{y}]$.

Now we prove $Var[\mathbf{x}] \approx Var[\mathbf{y}]$. Note that $Var[\mathbf{x}] = \mathbb{E}\left[\mathbf{x}^2\right] - (\mathbb{E}[\mathbf{x}])^2$ and $Var[\mathbf{y}] = \mathbb{E}\left[\mathbf{y}^2\right] - (\mathbb{E}[\mathbf{y}])^2$, thus we only need to prove $\mathbb{E}\left[\mathbf{x}^2\right] \approx \mathbb{E}\left[\mathbf{y}^2\right]$. It's similar to the prove

of expectation, $\mathbb{E}\left[\mathbf{x}^2\right]$ and $\mathbb{E}\left[\mathbf{y}^2\right]$ can be written as:

$$\begin{aligned} \mathbb{E}\left[\mathbf{x}^2\right] &= \int p_{\mathbf{x}}(\mathbf{x})\mathbf{x}^2 \mathrm{d}\mathbf{x} = \int p_{\mathbf{z}}(\mathbf{z}) f^2(\mathbf{z}) \mathrm{d}\mathbf{z} \\ \mathbb{E}\left[\mathbf{y}^2\right] &= \int p_{\mathbf{y}}(\mathbf{y})\mathbf{y}^2 \mathrm{d}\mathbf{y} = \int p_{\mathbf{z}}(\mathbf{z}) g^2(\mathbf{z}) \mathrm{d}\mathbf{z} \end{aligned} \tag{2}$$

According to the condition of zero order, we can prove than $Var[\mathbf{x}] \approx Var[\mathbf{y}]$

□

In summary, we can draw the conclusion that $\mathbf{x}$ and $\mathbf{y}$ have approximately the same expectation and variance. Similarly, it can be proved that this conclusion holds when $m \in [0, 1, \cdots, n]$. Based on Lemma 1, we can further get the next lemma.

Lemma 2. *If m is the maximum of selected paths and each pair of the outputs $\mathbf{o}_i^l$ of all parallel selective paths in choice block l meet Definition 1, there are m kingd of expectations and variances in all possible combinations of chosen paths.*

Proof. This is obviously true when $m = 1$. For the case of $m = 2$, we have $\mathbb{E}(\mathbf{x}) \approx \mathbb{E}(\mathbf{y})$. When the two paths are both selected, the output becomes $\mathbf{x} + \mathbf{y}$, it's expectation can be written as:

$$\mathbb{E}[\mathbf{x} + \mathbf{y}] = \mathbb{E}[\mathbf{x}] + \mathbb{E}[\mathbf{y}] \approx 2\mathbb{E}[\mathbf{x}] \tag{3}$$

For variance, we have $Var[\mathbf{x}] \approx Var[\mathbf{y}]$. Let $(\mu_{\mathbf{x}} = \mathbb{E}[\mathbf{x}]) \approx (\mu_{\mathbf{y}} = \mathbb{E}[\mathbf{y}])$, the variance of $\mathbf{x} + \mathbf{y}$ can be written as:

$$Var[\mathbf{x} + \mathbf{y}] \approx Var[2\mathbf{x}] = 4Var[\mathbf{x}] \tag{4}$$

Therefore, there are two kind of expectations and variances: $\mathbb{E}[\mathbf{x}]$ and $Var[\mathbf{x}]$ for $\{\mathbf{x}, \mathbf{y}\}$, and $2\mathbb{E}[\mathbf{x}]$ and $4Var[\mathbf{x}]$ for $\{\mathbf{x}+\mathbf{y}, \mathbf{x}+\mathbf{x}, \mathbf{y}+\mathbf{y}\}$.

Similarly, in the case where $m \in [1, n]$, there will be m kinds of expectations and variances. □

Until here, the number of SBNs has been reduced to m. $SBN_i (i = 1, 2, \cdots, m)$ will track the combination that contains i paths. Compared to Switchable Batch Normalization [40]: Switchable BN is applied to catch the statistics of limited number (K) of sub-architectures with K channel configurations[2]. Shadow BN is designed to catch the changing statistics from the flexible combination (exponential) of different search-able paths while keeping the fixed number of channels.

We take a popular search space for example to illustrate our unified approach for one-shot neural architecture search, *MixPath*, which mixes up different number of paths in Figure 1. SBNs ensure the stability for supernet training in MixPath. Particularly, the input tensor is divided into four groups, where at most m paths with various kernel sizes are randomly chosen. Note that the actual number of paths m' in MixPath is randomly selected, *i.e.* $m' \sim U(1, m)$. For example, m' is equally sampled from $\{1, 2, 3\}$ if $m = 3$.

[2]The value of K is 4 or 8 in the original paper.

Algorithm 1 MixPath Supernet Training.

Input: search space $S_{(n,L)}$ with n choice paths per layer and L layers in total, maximum selective paths m, supernet parameters $\Theta(n, L)$, training epochs N, training data loader D, loss function $Loss$
initialize every $\theta_{j,l}$ in $\Theta_{(m,L)}$, shadow BN 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**
      apply Bernoulli sampling to select $m' \leq m$ paths, each has output $\mathbf{o}_i^l$, and select $SBN_{m'}$ to act on the sum of outputs: $\mathbf{o}^l = SBN_{m'}\left(f_{1\times1conv}\left(\sum_i \mathbf{o}_i^l\right)\right)$
    end for
    Build $model$ from above sampled index.
    Clear gradients recorder for all parameters.
    $\nabla\theta_{j,l} = 0$ where $j = 1, 2, ..., m$ and $l = 1, 2, ..., L$
    Calculate gradients for $model$ based on $Loss$, $data$, $labels$.
    update $\theta_{(m,L)}$ by gradients.
  end for
end for

Then the selected paths are added and followed a SBN. According to above analysis, the output of each path is approximately identically distributed, and the number of SBN increases linearly with m. Therefore, each MixPath contains m SBNs, $\{SBN_1, SBN_2, \ldots, SBN_m\}$. For the actual number of paths m', $SBN_{m'}$ will be activated.

3.4 The NAS pipeline

To summarize the overall pipeline of our approach, we illustrate the details of MixPath supernet training in Algorithm 1. Next we progress with the well-known evolutionary algorithm NSGA-II [11] for searching. In particular, our objectives are maximizing the classification accuracy while minimizing the FLOPs.

4 Experiments

4.1 Search on CIFAR10

With the guidance of MixPath and shadow BN, we design a search space S_1 containing 12 inverted bottlenecks, each of which has 4 kernel size choices of (3, 5, 7, 9) for depth-wise layer and 2 choices of (3, 6) for expansion rate. Hence, a huge search space named S_1 can be obtained specifically in the range of 8^{12} (m=1) to $(8^{12} + 12^{12} + 8^{12} + 2^{12})$ (m=4).

As for each case, we directly train the supernet on CIFAR-10 for the same 600 epochs till it converges. Batch size is set to 256 and use SGD optimizer with 0.9 momentum and 10^{-4} weight decay. In the training process, we set the initial learning rate to 0.06 and the cosine scheduler is applied. It takes about 19 GPU hours on the single Tesla V100 card for training and random search. After the training of supernet, random search algorithm is applied to sample 1000 models to obtain the model accuracy distribution.The comparisons with recent state of the art models on CIFAR-10 are listed in Table 1.

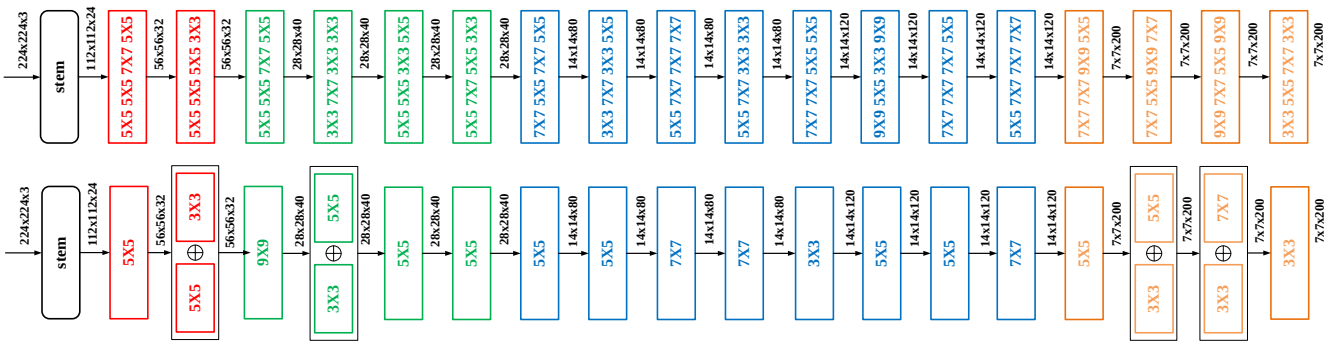

Figure 4: The architecture of MixPath-A (top) and MixPath-B (bottom).

Models	Params (M)	×+ (M)	Test Error (%)	Type
NASNet-A [42]	3.3	608	2.65	RL
DARTS [24]	3.3	528	2.86	GD
SNAS [36]	2.9	422	2.98	GD
GDAS [13]	3.4	519	2.93	GD
P-DARTS [4]	3.4	532	2.50	GD
PC-DARTS [37]	3.6	558	2.57	GD
FairDARTS-a [9]	2.8	371	2.54	GD
MixNet-M [33]	5.1	360	2.10	TF
MixPath-a (ours)	5.3	473	2.60	OS
MixPath-b (ours)	3.5	299	2.17	TF

Table 1: Comparison of architectures on CIFAR-10. GD: gradient based, OS: one shot, TF: transfer from ImageNet.

4.2 Search on ImageNet

Layer-wise search based on inverted bottleneck block is another commonly used space [7, 31–34]. Therefore, we also search on ImageNet proxylessly based on the search space of MnasNet [31]. However, we fix the expansion rate as [33] and focus29693 on searching the depth-wise convolution layer of the inverted bottleneck block and their combinations (18 layers in total). Specifically, we search the kernel size (3, 5, 7 ,9) and their combinations of the depth-wise layer, thus building a search space S_2 with capacity about $(2^4)^{18} = 16^{18}$. Moreover, we also construct a group based kernel search as MixNet to make fair comparisons. Particularly, we evenly categorize the depth wise layer aligned channel dimension by 4 groups and search the kernel size $(3, 5, 7, 9)$ and their combinations within each group, which forms the search space S_3.

We search under two settings of $m = 1, 2$. For each case, we utilize the same hyper-parameters. We use batch size of 1024 and SGD optimizer with 0.9 momentum and 10^{-5} weight decay. The initial learning rate is 0.1 and scheduled to zero by cosine decay strategy within 120 epochs, which involves about 150k times of back propagation and takes about 10 GPU days on Tesla V100 machines.

We use the same training tricks as MnasNet [31] to train searched models. Unlike EfficientNet [27], we don't use auto-augment tricks [10]. The performances of our searched models and comparisons with state-of-the-art architectures are listed in Table 2.

Models	×+ (M)	Params (M)	Top-1 (%)	Top-5 (%)
MobileNetV2 [28]	300	3.4	72.0	91.0
DARTS (2nd order) [24]	595	4.9	73.1	-
PDARTS (CIFAR-10)	557	4.9	75.6	92.6
PCDARTS [37]	586	5.3	74.9	92.2
FairDARTS-C [9]	380	4.2	75.1	92.4
FBNet-B [34]	295	4.5	74.1	-
MnasNet-A2 [31]	340	4.8	75.6	92.7
MobileNetV3 [17]	219	5.4	75.2	92.2
Proxyless-R [3]	320	4.0	74.6	92.2
FairNAS-A [7]	388	4.6	75.3	92.4
Single-Path [30]	365	4.3	75.0	92.2
SPOS [15]	328	3.4	74.9	92.0
MixNet-M [33]	360	5.0	76.6 † (77)	93.2
AtomNAS-B [25]	326	4.4	75.5	92.6
EfficientNet B0 [32]	390	5.3	76.3	93.2
SCARLET-A [5]	365	6.7	76.9	93.4
MixPath-A (ours)	349	5.0	76.9	93.4
MixPath-B (ours)	378	5.1	76.7	93.3

Table 2: Comparison with state-of-the-art models on ImageNet. ‡: model trained from scratch by us.

MixPath-A from S_3 uses 349M multiply-adds to obtain 76.9% top-1 accuracy on ImageNet validation dataset. By contrast, MixNet-M uses 10M more flops and 300 times more GPU days [3] to obtain such level of accuracy.

Compared with EfficientNet-B0, MixPath-B from S_2 uses fewer FLOPS and number of parameters to obtain higher top-1 validation accuracy (76.7%). It makes extensive uses of larger kernel (60% 5×5 and 22% 7×7) instead of small ones. 3×3 kernels are mainly used in parallel with large one to balance the trade off between FLOPS and accuracy. We attribute the high accuracy performance to the feature aggregation of multi branches. Moreover, this light weight model benefits from the analysis in Section 3.1, where multi-branch is promising in balancing accuracy and inference complexity.

[3] We approximate it by MnasNet (3000+ GPU days).

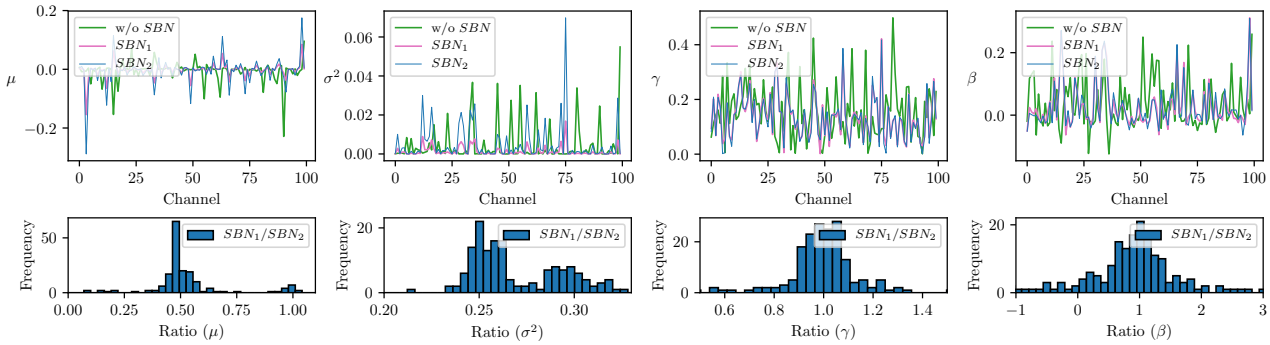

Figure 5: **Top**: Parameters $(\mu, \sigma^2, \gamma, \beta)$ of first-layer shadow batch normalization in MixPath supernet trained on CIFAR-10 when at most $m = 2$ paths can be activated. Specifically, SBN_1 and SBN_2 represent shadow BN for single path and two paths. **Bottom**: Histogram of ratios (SBN_1/SBN_2) for each BN parameter. They center around (0.5, 0.25, 1, 1) respectively. The mean of SBN_2 is roughly twice as that of SBN_1.

4.3 MixPath Transfer to CIFAR-10

We also evaluated the transfer ability of MixPath on CIFAR-10 dataset, as shown in Table 1. We fine-tune the model on CIFAR-10 dataset, which is trained on ImageNet from scratch. The settings are referred to [18] and [21]. Compared with MixNet-M [33], MixPath achieved 97.83% with only 3.5M number of parameters and 299M FLOPs on CIFAR-10 dataset. It shows that the searched model using our proposed method also have a strong ability to transfer learning.

4.4 Batch Normalization Analysis

The theoretical analysis about parameters of batch normalization can be verified by experiments. Without loss of generality, we set m = 2 and make statistics on the four parameters of shadow BN across all channels for the first choice block. While SBN_1 goes like a shadow of one branch, SBN_2 does for the case of two paths. The histogram of four parameters is shown in Figure 5. Based on the theoretical analysis of Section 3.3, $\mu_{bn_1} \approx 0.5\mu_{bn_2}$ and $\sigma^2_{bn_1} \approx 0.25\sigma^2_{bn_2}$, which can be observed in Figure 5. It's interesting to see that the other two learn-able parameters β and γ are quite similar for SBN_1 and SBN_2. In such case, vanilla batch normalization plus post calibration for μ and σ^2 is promising [1].

5 Ablation Study

5.1 Vanilla Batch Normalization vs. Shadow Batch Normalization

We run four experiments with different m ($m = 1, 2, 3, 4$) to investigate the effect of shadow batch normalization, where all other variables are controlled. Each experiment randomly samples 1k models and reports the test accuracy distributions on CIFAR10 in Figure 6. It's interesting to see that MixPath falls backs to single path when $m = 1$. Whereas, shadow batch normalization begins to demonstrate its power for $m > 1$, whose absence leads to a bad supernet with lower accuracy and much larger gap. This means the supernet severely under-estimates the performance of a large proportion of architectures.

5.2 NSGA-II vs Random Search

We compare the adopted NSGA-II search algorithm with random search by charting the Pareto-front of models found by both methods in Figure 7. NSGA-II has a clear advantage in that the final Pareto-font models have higher accuracies and fewer multi-adds.

6 Model Ranking Capacity Analysis

As mentioned above, the most critical role of the one-shot supernet is to differentiate good and bad architectures. NAS-bench-101 [41] is a good benchmark to model ranking evaluation and also used to score our methods in this paper. The shadow BN is placed after the input edge of Node 5.

Batch normalization calibration is a kind of BN post processing trick, which is used in [1, 15, 25] to correct the biased mean and variance with extra data and computational resources. As [41], we utilize Kendall Tau to evaluate the model ranking performance for four comparison groups: shadow BN with/without post BN recalibration and vanilla BN with/without post BN re-calibration. We randomly sample 70 models from S_1 and look up their top-1 accuracy from NAS-bench 101 table and calculate the metrics. Particularly, we run two experiments for $m = 3, 4$ across three seeds and train the supernet for 100 epochs with batchsize 96 and learning rate 0.025, the result is shown in Table 3.

Type	Kendall Tau (m=4)	Kendall Tau (m=3)
Shadow BN	0.393 ± 0.017	0.318 ± 0.034
Shadow BN + CA	**0.597 ± 0.037**	**0.592 ± 0.024**
wo Shadow BN	0.167 ± 0.038	0.045 ± 0.060
wo Shadow BN + CA	0.368 ± 0.134	0.430 ± 0.031

Table 3: Comparison of Kendall tau ranking between MixPath supernets trained with shadow BN and vanilla BN based on 70 sampled models from NAS-Bench 101 [38]. Each control group is repeated 3 times on different seeds. CA is short for calibration

It's notable to see that BN post calibration can boost Kendall Tau in each case, which indicates the validity of this

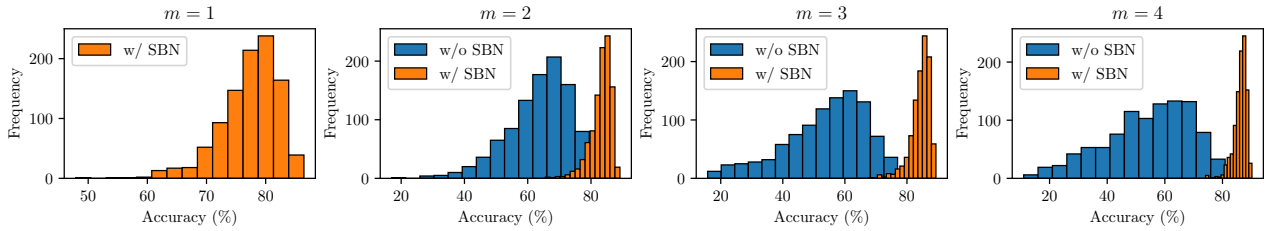

Figure 6: Histogram of one-shot model accuracies sampled from MixPath supernets trained on CIFAR-10, activating at most $m = 1, 2, 3, 4$ paths respectively. In all cases, the supernet trained with Shadow Batch Normalization (SBN) boosts one-shot performance. Note when $m = 1$ it falls back to vanilla batch normalization.

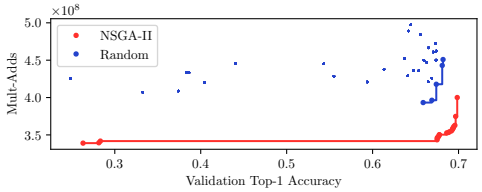

Figure 7: Pareto-front of models by NSGA-II vs. random search.

Type	mean	variance
Feature Map	0.9342	6e-5
Jacobian Matrix	0.9042	1e-5

Table 4: Cosine similarities of first searchable layer feature maps and Jacobian matrices of MixPath supernet ($m = 2$) trained on CIFAR-10, averaged on 10 input images.

post processing work [1]. For $m = 4$, even without calibration, Shadow BN still ranks architectures better than Vanilla BN with 0.025 higher Kendall Tau value. The composition of two tricks can further boost the score to 0.597, which breaks a new record for model ranking on NAS-bench-101. From theoretical analysis, shadow BN can degrade as vanilla when $m = 1$, the Kendall Tau gap between Shadow BN and vanilla is narrowed when we decease m from 4 to 3.

7 Discussion

7.1 One More Step for the Weight Sharing Mechanism

The statement that highly similar features of competitive choices play a critical role in supporting weight sharing under the one shot setting seems promising [7]. We also calculate the similarities of the feature maps between one path and two paths of MixPath for $m = 2$ in Table 4. We use the features $(32 \times 32 \times 48)$ of the first searchable block from the supernet in S_1 and 10 images to obtain the mean and variance of the similarity value. The mean of cosine similarity is above 0.93, thus meeting the condition of zero order.

We further calculate the Jacobi matrix of these features over input images, shown in Table 4 . Particularly, we forward the neural network for a given image to obtain two feature maps counterpart, one is from single path and another is summation of two paths. Then we use auto-grad to calculate the Jacobi matrix and obtain their cosine similarities. Since network weights are updated by the back propagation algorithm, such high similarity above 0.9 ensure stable update for network weights regardless of the randomly alternated number of paths.

8 Conclusion

In this paper, we propose a unified approach for one shot neural architecture search, which bridges the gap between one-shot and multi-path. Existing single-path approaches can be regarded as a special case of ours. The proposed method uses shadow batch normalization to catch the changing feature from various branch combinations, which successfully solves two difficulties of vanilla multi-path: the unstable training of supernet and the unbearable weakness of model ranking. Moreover, we can reduce the number of shadow BN to be linear with m-paths under some cases. Extensive experiments on NAS-bench-101 show that our method can boost the model ranking capacity of one-shot supernet with clear margins.

Based on thorough theoretical reasoning of weight-sharing mechanism and batch normalization's functionality, we are able to offer practical guidelines that might shed lights on future design of one-shot NAS algorithms.

9 Acknowledgement

We are grateful to Deli Zhao for his insightful discussion about feature aggregations.

References

[1] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and Simplifying One-Shot Architecture Search. In *International Conference on Machine Learning*, pages 549–558, 2018.

[2] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. SMASH: One-Shot Model Architecture. Search Through HyperNetworks. *International Conference on Learning Representations*, 2018.

[3] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations*, 2019.

[4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. In *International Conference on Computer Vision*, 2019.

[5] Xiangxiang Chu, Bo Zhang, Jixiang Li, Qingyuan Li, and Ruijun Xu. Scarletnas: Bridging the gap between scalability and fairness in neural architecture search. *arXiv preprint arXiv:1908.06022*, 2019.

[6] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. *arXiv preprint arXiv:1901.07261*, 2019.

[7] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. *arXiv preprint. arXiv:1907.01845*, 2019.

[8] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Hailong Ma. Multi-Objective Reinforced Evolution in Mobile Neural Architecture Search. *arXiv preprint. arXiv:1901.01074*, 2019.

[9] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. *arXiv preprint arXiv:1911.12126*, 2019.

[10] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning Augmentation Policies from Data. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[11] 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.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[13] Xuanyi Dong and Yi Yang. Searching for a Robust Neural Architecture in Four GPU Hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

[14] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.

[15] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single Path One-Shot Neural Architecture Search with Uniform Sampling. *arXiv preprint. arXiv:1904.00420*, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[17] 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. *International Conference on Computer Vision*, 2019.

[18] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. *Advances in Neural Information Processing Systems*, 2019.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 448–456, 2015.

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

[21] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do Better Imagenet Models Transfer Better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[22] 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.

[23] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019.

[24] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*, 2019.

[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. *International Conference on Learning Representations*, 2020.

[26] Junran Peng, Ming Sun, ZHAO-XIANG ZHANG, Tieniu Tan, and Junjie Yan. Efficient neural architecture transformation search in channel-level for object detection. In *Advances in Neural Information Processing Systems*, pages 14290–14299, 2019.

[27] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient Neural Architecture Search via Parameter Sharing. In *International Conference on Machine Learning*, 2018.

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[29] David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886, 2019.

[30] 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. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2019.

[31] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-Aware Neural Architecture Search for Mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[32] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, 2019.

[33] Mingxing Tan and Quoc V. Le. MixConv: Mixed Depthwise Convolutional Kernels. *The British Machine Vision Conference*, 2019.

[34] 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. *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[35] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[36] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic Neural Architecture Search. *International Conference on Learning Representations*, 2019.

[37] 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 *International Conference on Learning Representations*, 2020.

[38] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114, 2019.

[39] Jiahui Yu and Thomas Huang. Universally Slimmable Networks and Improved Training Techniques. In *International Conference on Computer Vision*, 2019.

[40] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable Neural Networks. In *International Conference on Learning Representations*, 2019.

[41] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2020.

[42] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.