Supplementary of "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

A. Proofs

A.1. Proof of Lemma 3.1

Proof. Let $\mathbf{y}_{p_{\mathbf{y}}(\mathbf{y})} = f(\mathbf{x})$, $\mathbf{z}_{p_{\mathbf{z}}(\mathbf{z})} = g(\mathbf{x})$, $\mathbf{x} \sim p_{\mathbf{x}}(\mathbf{x})$. For the case $m = 1$, the expectation of $\mathbf{y}$ and $\mathbf{z}$ can be written respectively as:

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

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

Now we prove $Var[\mathbf{y}] \approx Var[\mathbf{z}]$. Note that $Var[\mathbf{y}] = \mathbb{E}\left[\mathbf{y}^2\right] - (\mathbb{E}[\mathbf{y}])^2$ and $Var[\mathbf{z}] = \mathbb{E}\left[\mathbf{z}^2\right] - (\mathbb{E}[\mathbf{z}])^2$, thus we only need to prove $\mathbb{E}\left[\mathbf{y}^2\right] \approx \mathbb{E}\left[\mathbf{z}^2\right]$. It can be similarly proved as follows:

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

According to the zero-order condition, we have $Var[\mathbf{y}] \approx Var[\mathbf{z}]$.

For the case of $m = 2$, when the two paths are both selected, the output becomes $\mathbf{y} + \mathbf{z}$, its expectation can be written as:

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

And the variance of $\mathbf{y} + \mathbf{z}$ is,

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

Therefore, there are two kinds of expectations and variances: $\mathbb{E}[\mathbf{y}]$ and $Var[\mathbf{y}]$ for $\{\mathbf{y}, \mathbf{z}\}$, and $2\mathbb{E}[\mathbf{y}]$ and $4Var[\mathbf{y}]$ for $\{\mathbf{y} + \mathbf{z}\}$. Similarly, in the case where $m \in [1, n]$, there will be m kinds of expectations and variances. $\square$

B. Algorithms

Algorithm 1 : Stage 2-NSGA-II search strategy.

Input: Supernet S, the number of generations N, population size n, validation dataset D, constraints C, objective weights w.
Output: A set of K individuals on the Pareto front.
Uniformly generate the populations P_0 and Q_0 until each has n individuals satisfying C_{acc}, C_{FLOPs}.
for $i = 0$ **to** $N - 1$ **do**
  $R_i = P_i \cup Q_i$
  $F =$ non-dominated-sorting(R_i)
  Pick n individuals to form P_{i+1} by ranks and the crowding distance weighted by w.
  $Q_{i+1} = \emptyset$
  while $size(Q_{i+1}) < n$ **do**
    $M =$ tournament-selection(P_{i+1})
    $q_{i+1} =$ crossover(M)
    if $FLOPs(q_{i+1}) > FLOPs_{s_{max}}$ **then**
      continue
    end if
    Evaluate model q_{i+1} with S (BN calibration is recommended) on D
    if $acc(q_{i+1}) > acc_{min}$ **then**
      Add q_{i+1} to Q_{i+1}
    end if
  end while
end for
Select K equispaced models near Pareto-front from P_N

C. Experiments details

C.1. Search Spaces

We show the list of used search spaces in Table 1.

C.2. More experiments

We further search directly in S_4. To be comparable, this case is formulated as a single objective optimization problem: finding the best model with known ground truth (94.29%)

Space	Dataset	m	Size	Details
S_1	CIFAR-10	1 2 3 4	8^{12} 36^{12} 92^{12} 162^{12}	There are 12 stacked inverted bottleneck blocks. Kernel sizes are in (3, 5, 7, 9), and expansion rates are in (3, 6).
S_2	ImageNet	2	10^{18}	There are 18 stacked inverted bottleneck blocks. Kernel sizes are in (3, 5, 7, 9). Expansion rate is fixed following MixNet.
S_3	ImageNet	1	256^{18}	There are 18 stacked mobile inverted bottlenecks. Depthwise layer channels are divided into 4 groups and there are 4 choice kernel sizes (3, 5, 7, 9) for each group. Expansion rate is also fixed as above.
S_4	CIFAR-10	4	255	There are 9 stacked cells, each with 5 internal nodes, where the first 4 nodes are candidate paths. Each node has 3 operation choices (1×1 Conv, 3×3 Conv, 3×3 Maxpool). See Fig. 6 (main text).

Table 1: Four search spaces used in this paper. m is the maximum number of allowed paths per layer

in the space. As a strong baseline, we run DARTS [6] three times using different seeds. The result is shown in Table 2. Our method obtains 94.22% within 5 GPU hours, which outperforms DARTS with a large margin.

Table 2: Search results on the reduced NAS-Bench-101. The accuracy of known optimal is 94.29%.

Method	Top-1 Acc (%)	Search Cost (GPU Hours)
DARTS [6]	79.42±0.23	7
Ours	**94.22±0.06**	5

We also use $m = 3$ and perform multi-path supernet training in DARTS space. Then we search for the optimal sub-model with 60 generations. The total search cost is 0.5 GPU days and MixPath achieves a competitive 97.5% test accuracy with only 3.6M parameters on CIFAR10.

C.3. Transferring to CIFAR-10

We also evaluated the transferability of MixPath models on CIFAR-10 dataset, as shown in Table 3 (main text). The settings are the same as [2] and [3]. Specifically, MixPath-b achieved 98.2% top-1 accuracy with only 377M FLOPS.

C.4. Transferring to object detection

We further verify the transferability of our models on object detection tasks and we only consider mobile settings. Particularly, we utilize the RetinaNet framework [4] and use our models as drop-in replacements for the backbone component. Feature Pyramid Network (FPN) is enabled for all experiments. The number of the FPN output channels is 256. The input features from the backbones to FPN are the output of the depth-wise layer of the last bottleneck block in four stages, which covers 2 to 5.

Backbones	$\times+$ (M)	P (M)	Acc (%)	AP (%)	AP_{50} (%)	AP_{75} (%)	AP_S (%)	AP_M (%)	AP_L (%)
MobileNetV3	219	5.4	75.2	29.9	49.3	30.8	14.9	33.3	41.1
MnasNet-A2	340	4.8	75.6	30.5	50.2	32.0	16.6	34.1	41.1
SingPath NAS	365	4.3	75.0	30.7	49.8	32.2	15.4	33.9	41.6
MixNet-M	360	5.0	77.0	31.3	51.7	32.4	17.0	35.0	41.9
MixPath-A	349	5.0	76.9	31.5	51.3	33.2	17.4	35.3	41.8

Table 3: COCO Object detection with various drop-in backbones

All the models are trained and evaluated on the MS COCO dataset [5] (train2017 and val2017 respectively) for 12 epochs with a batch size of 16. We use the SGD optimizer with 0.9 momentum and 0.0001 weight decay. The initial learning rate is 0.01 and multiplied by 0.1 at epochs 8 and 11. Moreover, we use the MMDetection toolbox [1] based on PyTorch [7]. Table 3 shows that MixPath-A gives competitive results.

C.5. Comparison of search strategies

We show the Pareto front of models searched by NSGA-II vs. Random in Fig. 1.

C.6. More statistics analysis on SBNs

To further confirm our early postulation, we train MixPath supernet in the search space S_1 on CIFAR-10, allowing the number of activable paths $m = 3$ and $m = 4$. Other settings

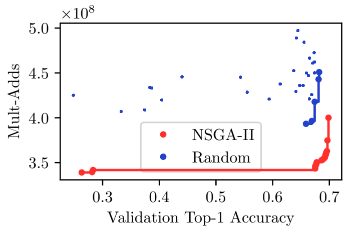


Figure 1: Pareto-front of models by NSGA-II vs. random search.

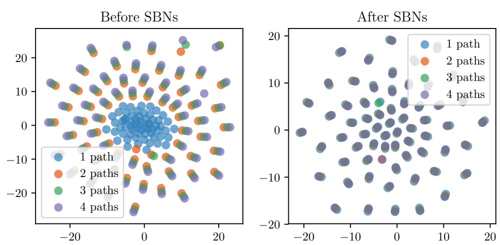


Figure 2: A t-SNE visualization of first-layer multi-path features before and after SBNs. We randomly sample 64 samples to get these features. Dots of the same color indicate the same multi-path combination. SBNs make distant features from multi-path combinations similar to each other (see closely overlapped dots on the right). Best viewed in color.

are kept the same as the case $m = 2$. The relationship of parameters in SBNs is shown in Fig. 3. As expected, SBNs capture feature statistics for different combinations of path activation. For instance, the mean of SBN_3 is three times that of SBN_1. The similar phenomenon can be observed in other layers as well, for instance, the statistics of the 6-th and 11-th layer are shown in Fig. 4.

SBNs have a strong impact on regularizing features from multiple paths This is more obvious when we draw a t-SNE visualization [8] of first-layer feature maps from our MixPath supernet ($m = 4$) trained on CIFAR-10 in Fig. 2. Before applying SBNs, features from different path combinations are quite distant from each other, while SBNs close up this gap and make them quite similar.

C.7. Cosine similarity and feature vectors on NAS-Bench-101

We also plot the cosine similarity of features from different operations along with their projected vectors before/after SBNs and vanilla BNs on NAS-Bench-101 in Fig. 6. We can see that not only are the features from different operations similar, but so are the summations of features from multiple paths. At the same time, SBNs can transform the amplitudes of different vectors to the same level, while vanilla BNs can't. This is similar to the situation in the search space S_1 and matches with our theoretical analysis.

C.8. Searched architectures on CIFAR-10 and ImageNet

The architectures of MixPath-c, MixPath-A and MixPath-B are shown in Fig 5.

References

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 2

[2] 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. *NeurIPS*, 2019. 2

[3] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do Better Imagenet Models Transfer Better? In *CVPR*, pages 2661–2671, 2019. 2

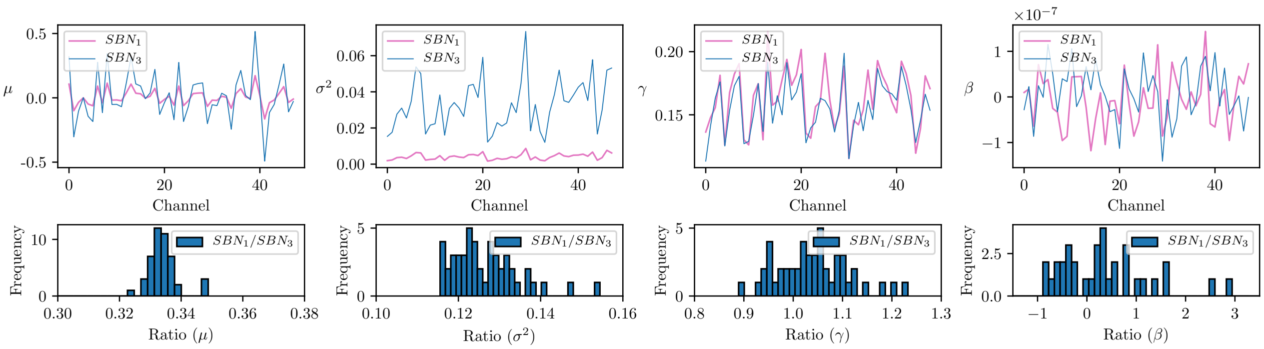
[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *ICCV*, pages 2980–2988, 2017. 2

[5] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014. 2

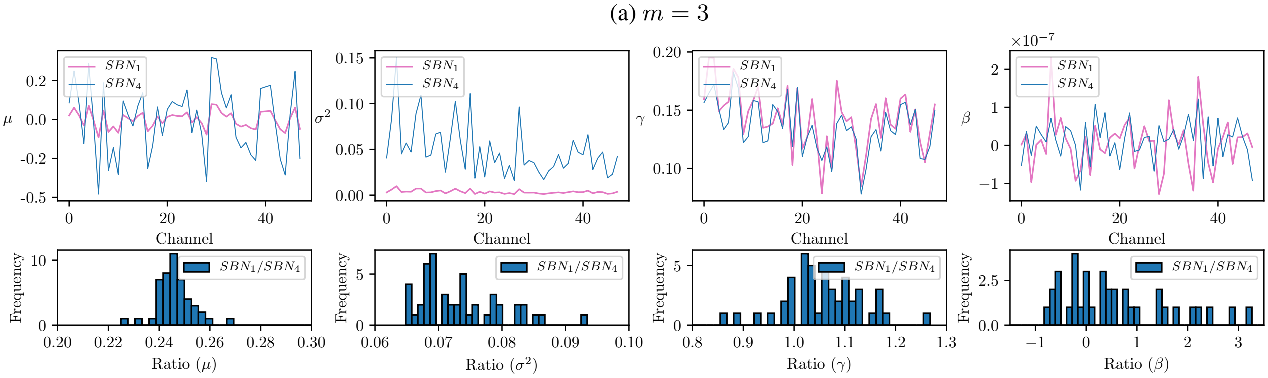
[6] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *ICLR*, 2019. 2

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019. 2

[8] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 3

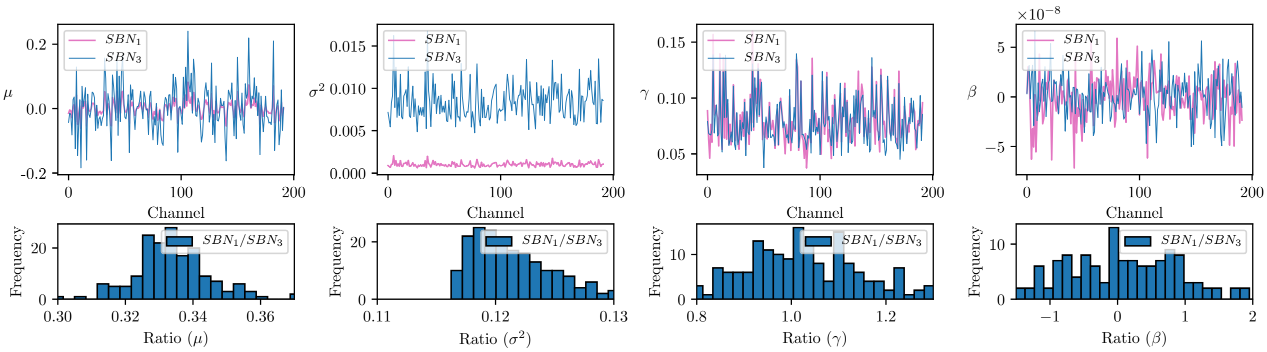


(a) $m = 3$

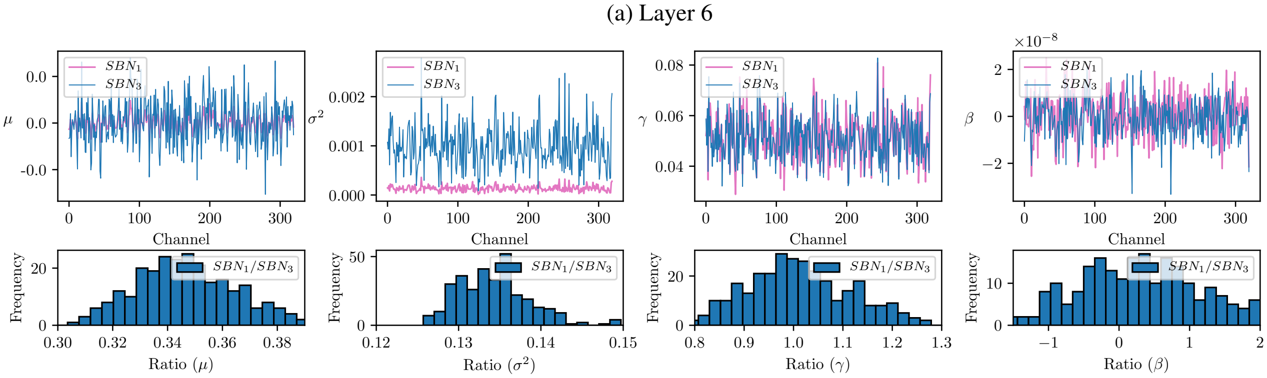


(b) $m = 4$

Figure 3: Parameters ($\mu, \sigma^2, \gamma, \beta$) of the first-layer SBNs in MixPath supernet (in S_1) trained on CIFAR-10 when at most $m = 3, 4$ paths can be activated. SBN_n refers to the one follows n-path activations. The parameters of SBN_3 and SBN_4 are multiples of SBN_1 as expected.



(a) Layer 6



(b) Layer 11

Figure 4: Parameters ($\mu, \sigma^2, \gamma, \beta$) of the SBNs of the 6th and 11th layer in MixPath supernet (in S_1) trained on CIFAR-10 when at most $m = 3$ paths can be activated. SBN_n refers to the one follows n-path activations. The parameters of SBN_3 are still multiples of SBN_1 as expected.

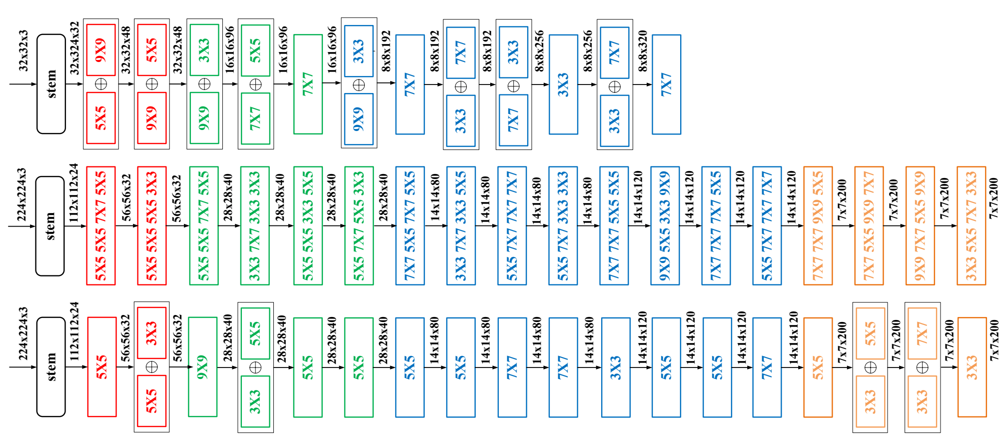


Figure 5: The architecture of MixPath-c (top), MixPath-A (middle) and MixPath-B (bottom). MixPath-B makes use of feature aggregation and outperforms EfficientNet-B0 with fewer FLOPS and parameters.

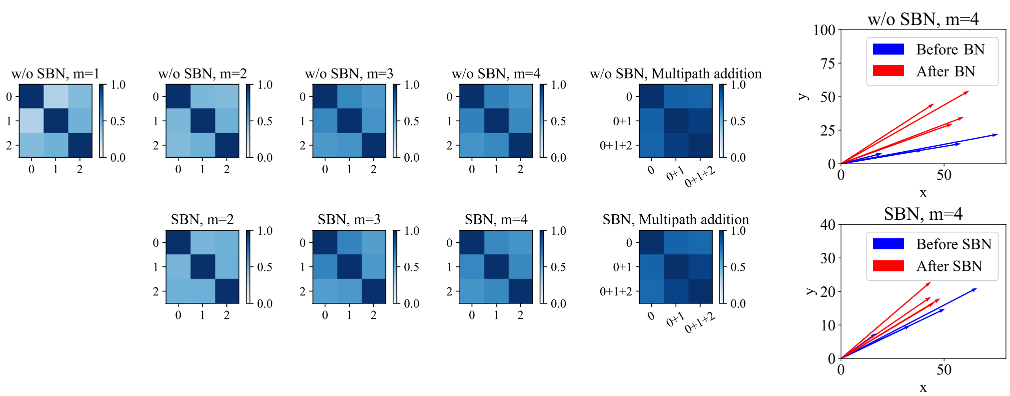


Figure 6: **(a)** Cosine similarity of first-block features from the supernets trained on NAS-Bench-101 with and without SBNs **(b)** Feature vectors projected into 2-dimensional space.