A. Proofs

A.1. Proof of Lemma 3.1

**Proof.** Let \( y_{p_x}(y) = f(x), z_{p_x}(z) = g(x), x \sim p_x(x) \). For the case \( m = 1 \), the expectation of \( y \) and \( z \) can be written respectively as:

\[
\begin{align*}
\mathbb{E}[y] &= \mathbb{E}[f(x)] = \int p_x(x)f(x)\,dx \\
\mathbb{E}[z] &= \mathbb{E}[g(x)] = \int p_x(x)g(x)\,dx
\end{align*}
\]

(1)

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

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

\[
\begin{align*}
\mathbb{E}[y^2] &= \int p_y(y)y^2\,dy = \int p_x(x)f^2(x)\,dx \\
\mathbb{E}[z^2] &= \int p_z(z)z^2\,dz = \int p_x(x)g^2(x)\,dx
\end{align*}
\]

(2)

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

For the case \( m = 2 \), when the two paths are both selected, the output becomes \( y + z \), its expectation can be written as:

\[
\mathbb{E}[y + z] = \mathbb{E}[y] + \mathbb{E}[z] \approx 2\mathbb{E}[y]
\]

(3)

And the variance of \( y + z \) is,

\[
\text{Var}[y + z] \approx \text{Var}[2y] = 4\text{Var}[y]
\]

(4)

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

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 \( C_{\text{acc}}, C_{\text{FLOPs}} \).

for \( i = 0 \) to \( N - 1 \)

\( R_i = P_i \cup Q_i \)

\( F = \text{non-dominated-sorting}(R_i) \)

Pick \( n \) individuals to form \( P_{i+1} \) by ranks and the crowding distance weighted by \( w \).

\( Q_{i+1} = \emptyset \)

while \( \text{size}(Q_{i+1}) < n \)

\( M = \text{tournament-selection}(P_{i+1}) \)

\( q_{i+1} = \text{crossover}(M) \)

if \( \text{FLOPs}(q_{i+1}) > \text{FLOPs}_{\text{max}} \) then

continue

end if

Evaluate model \( q_{i+1} \) with \( S \) (BN calibration is recommended) on \( D \)

if \( \text{acc}(q_{i+1}) > \text{acc}_{\text{min}} \) 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%)
There are 12 stacked inverted bottleneck blocks. Kernel sizes are in (3, 5, 7, 9), and expansion rates are in (3, 6).

There are 18 stacked inverted bottleneck blocks. Kernel sizes are in (3, 5, 7, 9). Expansion rate is fixed following MixNet.

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.

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>
<thead>
<tr>
<th>Space</th>
<th>Dataset</th>
<th>m</th>
<th>Size</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>CIFAR-10</td>
<td>1</td>
<td>8¹²</td>
<td>There are 12 stacked inverted bottleneck blocks. Kernel sizes are in (3, 5, 7, 9), and expansion rates are in (3, 6).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>36¹²</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>92¹²</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>16²¹²</td>
<td></td>
</tr>
<tr>
<td>S₂</td>
<td>ImageNet</td>
<td>2</td>
<td>10¹⁸</td>
<td>There are 18 stacked inverted bottleneck blocks. Kernel sizes are in (3, 5, 7, 9). Expansion rate is fixed following MixNet.</td>
</tr>
<tr>
<td>S₃</td>
<td>ImageNet</td>
<td>1</td>
<td>256¹¹⁸</td>
<td>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.</td>
</tr>
<tr>
<td>S₄</td>
<td>CIFAR-10</td>
<td>4</td>
<td>255</td>
<td>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).</td>
</tr>
</tbody>
</table>

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

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.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 Acc (%)</th>
<th>Search Cost (GPU Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARTS [6]</td>
<td>79.42±0.23</td>
<td>7</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>94.22±0.06</strong></td>
<td>5</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Backbones</th>
<th>× +</th>
<th>$F$ (M)</th>
<th>Acc (%)</th>
<th>AP (%%)</th>
<th>AP [50] (%%)</th>
<th>AP [75] (%%)</th>
<th>AP S (%%)</th>
<th>AP M (%%)</th>
<th>AP L (%%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV3</td>
<td>219</td>
<td>5.4</td>
<td>75.2</td>
<td>29.9</td>
<td>49.3</td>
<td>30.8</td>
<td>14.9</td>
<td>33.3</td>
<td>41.1</td>
</tr>
<tr>
<td>MnasNet-A2</td>
<td>340</td>
<td>4.8</td>
<td>75.6</td>
<td>30.5</td>
<td>50.2</td>
<td>32.0</td>
<td>16.6</td>
<td>34.1</td>
<td>41.1</td>
</tr>
<tr>
<td>SingPath NAS</td>
<td>365</td>
<td>4.3</td>
<td>75.0</td>
<td>30.7</td>
<td>49.8</td>
<td>32.2</td>
<td>15.4</td>
<td>33.9</td>
<td>41.6</td>
</tr>
<tr>
<td>MixNet-M</td>
<td>360</td>
<td>5.0</td>
<td>77.0</td>
<td>31.3</td>
<td>51.7</td>
<td>32.4</td>
<td>17.0</td>
<td>35.0</td>
<td>41.9</td>
</tr>
<tr>
<td>MixPath-A</td>
<td>349</td>
<td>5.0</td>
<td>76.9</td>
<td>31.5</td>
<td>51.3</td>
<td>33.2</td>
<td>17.4</td>
<td>35.3</td>
<td>41.8</td>
</tr>
</tbody>
</table>

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


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.
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 supernet trained on NAS-Bench-101 with and without SBNs. (b) Feature vectors projected into 2-dimensional space.