One-Shot Neural Ensemble Architecture Search by Diversity-Guided Search Space Shrinking

Minghao Chen[1]*, Houwen Peng[2]†, Jianlong Fu[2], Haibin Ling[1]
[1]Stony Brook University [2]Microsoft Research Asia
{minghao.chen,haibin.ling}@stonybrook.edu, {hopeng,jianf}@microsoft.com

Abstract

Despite remarkable progress achieved, most neural architecture search (NAS) methods focus on searching for one single accurate and robust architecture. To further build models with better generalization capability and performance, model ensemble is usually adopted and performs better than stand-alone models. Inspired by the merits of model ensemble, we propose to search for multiple diverse models simultaneously as an alternative way to find powerful models. Searching for ensembles is non-trivial and has two key challenges: enlarged search space and potentially more complexity for the searched model. In this paper, we propose a one-shot neural ensemble architecture search (NEAS) solution that addresses the two challenges. For the first challenge, we introduce a novel diversity-based metric to guide search space shrinking, considering both the potentiality and diversity of candidate operators. For the second challenge, we enable a new search dimension to learn layer sharing among different models for efficiency purposes. The experiments on ImageNet clearly demonstrate that our solution can improve the supernet's capacity of ranking ensemble architectures, and further lead to better search results. The discovered architectures achieve superior performance compared with state-of-the-arts such as MobileNetV3 and EfficientNet families under aligned settings. Moreover, we evaluate the generalization ability and robustness of our searched architecture on the COCO detection benchmark and achieve a 3.1% improvement on AP compared with MobileNetV3. Codes and models are available here.

1. Introduction

The emergence of deep neural networks greatly relieves the need for feature engineering. Previous studies have shown that the design of neural network architecture [11, 25, 32, 41] is essential to the performance for varied tasks in computer vision. However, the number of possible architectures is enormous, making the manual design very difficult. Neural Architecture Search (NAS) [48] aims to automate the design process. Recently, NAS methods have achieved state-of-the-arts on varied tasks such as image classification [48], semantic segmentation [23], object detection [4], *etc*. Despite great progress achieved, most of the NAS methods focus on searching for optimal architectures of single models. However, the generalization ability and performance of single models are usually affected by different initialization, noisy data, and training recipe modification.

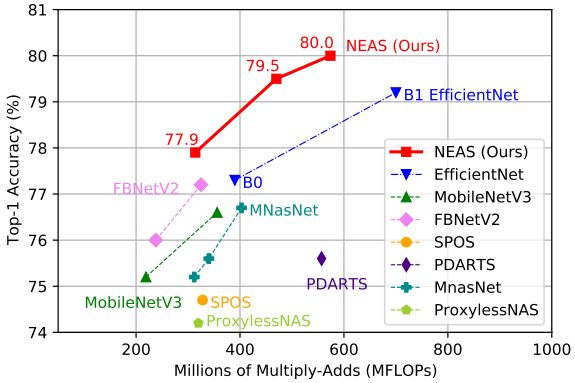

Figure 1. Comparison of our method with state-of-the-art approaches on ImageNet under mobile settings.

Model ensemble has been proved to be a universally effective method to build more robust and accurate models compared with single models. Implicit ensemble methods like Dropout [34], Dropconnect [40], StochDepth [15], Shake-Shake [9] are already widely used in neural architecture design. On the contrary, although explicit ensemble methods like averaging, bagging, boosting, and stacking have been commonly adopted in large competitions and real-world scenarios. The use of explicit ensemble methods in designing efficient models is not fully explored due to the extra computation they brought.

*This work is done when Minghao is an intern at Microsoft.
†Corresponding author

Inspired by the effectiveness of ensemble, we propose to search for multiple models instead of one simultaneously to form a robust, accurate and efficient ensemble model. However, the combination of NAS and ensemble faces two challenges: (1) efficient search and supernet optimization over a large search space (2) reducing the extra complexity brought by model ensemble. Addressing these challenges, in this paper, we propose a one-shot *neural ensemble architecture search* (NEAS) approach searching for lightweight ensemble models.

To solve the first challenge caused by the enlarged space of ensemble models compared with single models, we propose a novel metric called *diversity score* to progressively drop inferior candidates during the supernet training process, thus reduce the difficulty of finding promising ensemble models. This metric explicitly quantifies the diversity between the operators, which is commonly believed to be a key factor in building models with better feature expression capability.

To solve the second challenge, we introduce the layer sharing mechanism to reduce the model complexity. We allow the ensemble components share some shallow layers and search for the best architectures of the shared layers together with the architectures of the rest layers. We further introduce a new search dimension called *split point* to automatically find optimal layers for sharing under a given FLOPs constraint.

Comprehensive experiments verify the effectiveness of the proposed diversity score and layer sharing strategy. They improve the ranking ability of trained supernet and lead to better searched architectures under same complexity constraint. The searched architectures generate new state-of-the-art performance on ImageNet [8]. For instance, as shown in Fig. 1, our search algorithm finds a 314M FLOPs model that achieves 77.9% top-1 accuracy on ImageNet, which is 19% smaller and 1.6% better than EfficientNet-B0 [37]. The architecture discovered by NEAS transfers well to downstream object detection task, suggesting the generalization ability of the searched models. We obtain an AP of 33.0 on COCO validation set, which is superior to the state-of-the-art backbone, MobileNetV3 [12].

In summary, we make the following contributions:

- We propose a pipeline, NEAS, searching for diverse models under certain resource constraints. Our approach could search for both homogeneous and heterogeneous ensemble models.
- We design a new metric, diversity score, to guide the shrinking process of search space. We evaluate its superiority on supernet training and the performance of searched models by enormous experiments.
- We propose a layer-sharing strategy to reduce the complexity of ensemble models and enlarge the search space to search for an optimal split point.
- We compare the searched architectures to state-of-the-art NAS methods on the image classification task and achieve state-of-the-art results. Furthermore, we evaluate our searched model on the downstream object detection task, showing their generalization ability.

2. Related works

Neural Architecture Search. Early NAS approaches search the architectures using either reinforcement learning [48, 49, 45] or evolution algorithms [31, 35]. These methods have demonstrated that NAS can find architectures that surpass hand-crafted ones on a variety of tasks. However, these approaches require training thousands of architecture candidates from scratch, leading to unaffordable computation overhead. Most recent works resort to the weight sharing strategy to amortize the searching cost. Those approaches train a single over-parameterized supernet and then share the weights across subnets. They could be further categorized as two types: path-based [10, 6, 5] and gradient-based methods [24, 3, 42]. Path-based methods sample paths in each iteration to optimize the weights of supernets. Once the training process is finished, the subnets can be ranked by the shared weights. On the other hand, gradient-based methods relax the discrete search space to be continuous, and optimize the search process by the efficient gradient descent.

Ensemble Learning. Ensemble methods are widely used to boost the performance of neural networks [46, 34, 40, 14, 47, 33]. Strategies for building ensembles could be mainly divided into two categories. The first ones train different models independently and then apply ensemble methods to form a more robust model, such as boosting, bagging, and stacking [47]. The other methods train only one model with specific strategies to achieve implicit ensemble [46, 34, 40, 14]. Different from the above methods, we perform explicit ensemble without separate training and search for diverse model architectures to build ensemble models with great feature expression ability.

Search Space Shrinking. Recent works have shown search space shrinking is effective in boosting the ranking ability of NAS methods, especially when the search space is huge. [13, 20, 28, 27]. These methods could be classified into different types according to their evaluation metrics. There are three basic types: accuracy-based, magnitude-based, and angle-based metrics. For example, PCNAS [20] drop unpromising operators layer by layer using accuracy and shows that it improves candidate networks' quality. AngleNAS [13] uses the angles between weights of models to guide the search process. However, existing shrinking techniques only consider operators independently. Therefore, they can't directly adapt to search for ensemble models. We

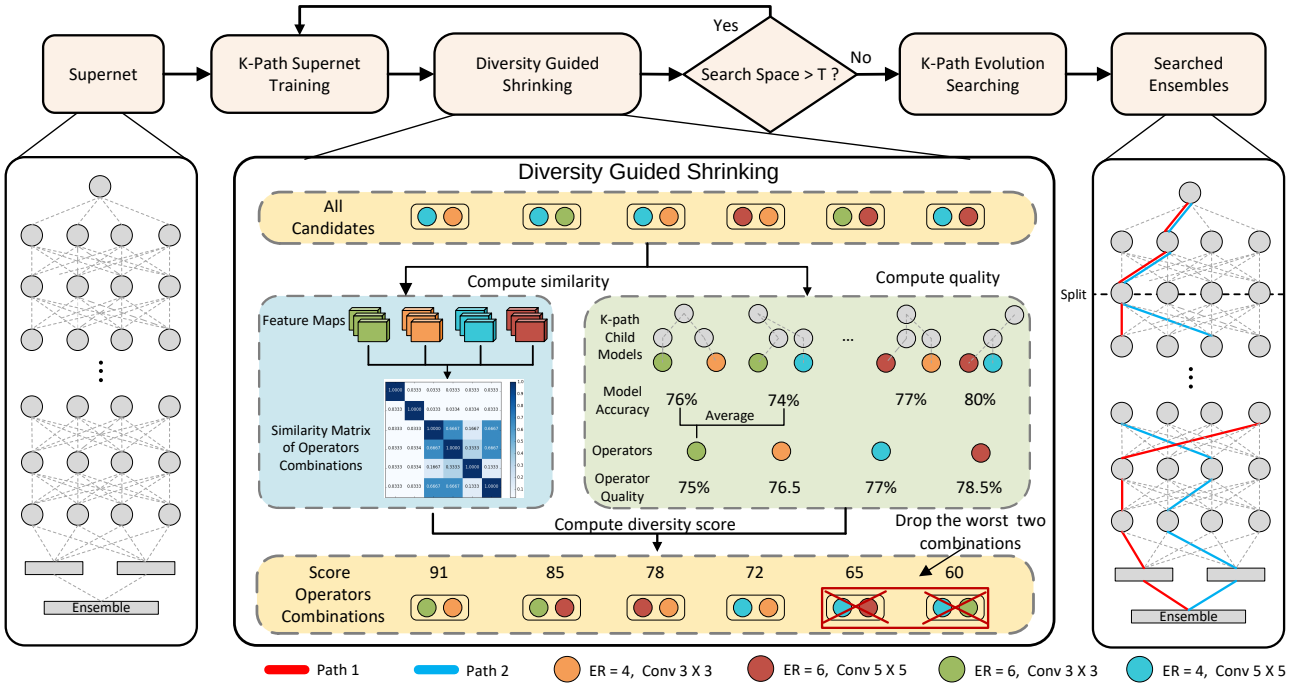

Figure 2. NEAS contains mainly two steps: K-path Supernet Training with Diversity-Guided Shrinking, and K-path Evolution Searching. It takes the search space as the input and outputs an ensemble model with shared shallow layers. We set the number of paths in the searched models to 2 and choice operators to 4 for explanation. The overlapping upper lines in the right graph indicate that the two paths share the first two layers. They then branch to two different paths. ER means the expansion ratio for the mobile inverted residual block.

design a new metric considering both the performance of single operators and the diversity across them.

3. Approach

In Section 3.1, we give the formulation of NEAS. In Section 3.2, we present the definition of the diversity score and the space shrinking pipeline. In Section 3.3, we introduce the layer sharing mechanism and the new search dimension *Split Point*. In Section 3.4, we give the detailed pipeline of NEAS which allows to search under different resource constrains. The overall framework is visualized in Fig. 2.

3.1. NEAS Formulation

Given the search space Ω of single deep neural networks, denote $\mathcal{A} = \{\phi_k \in \Omega : k = 1, \ldots, K\}$ as a set of K architectures with corresponding parameters $\mathcal{W} = \{\omega_k : k = 1, \ldots, K\}$, $\Phi(\cdot; \mathcal{A}, \mathcal{W})$ as the ensemble model, and $\mathcal{S} = \Omega^K$ as the search space of ensemble models. The goal of NEAS is to find an optimal architectures set $\mathcal{A}^*$ that maximizes the overall validation accuracy. To reduce the search cost, we constrain Ω to a certain architecture family, specifically, the subnetworks induced by a predefined supernet. In our work, we specify $\Phi(\cdot; \mathcal{A}, \mathcal{W})$ as:

$$\Phi(\cdot; \mathcal{A}, \mathcal{W}) = \sum_{i=1}^{K} \frac{\phi_i(\cdot; w_i)}{K}. \tag{1}$$

We then formulate NEAS as a two-stage optimization problem like other one-shot methods (*e.g.*, [10]). The first-stage is to optimize the weight of the supernet by:

$$W_{\mathcal{S}} = \arg\min_{W} \mathcal{L}_{\text{train}}(\Phi(\cdot; \mathcal{A}, W(\mathcal{A}))), \tag{2}$$

where $\mathcal{L}_{\text{train}}$ is the loss function on the training set, $W(\mathcal{A})$ means architectures in $\mathcal{A}$ inherit weights from W.

This step is done by uniformly sampling an ensemble architecture Φ from $\mathcal{S}$ and performing backpropagation to update the weight of the corresponding blocks in the supernet for each iteration. Please refer to Section 3.4 for details.

The second step is to search for an optimal architecture set $\mathcal{A}^*$ via ranking the performance based on learned weight $W_{\mathcal{S}}$ of supernet, which is formulated as

$$\begin{aligned} \mathcal{A}^* &= \arg\max_{\mathcal{A}} \text{ACC}_{\text{val}}(\Phi(\cdot; \mathcal{A}, W_{\mathcal{S}}(\mathcal{A}))), \\ \text{s.t.} \quad & \sum_{i}^{K} g_i(\phi_i) < C, \end{aligned} \tag{3}$$

where g and C are the resource computation functions and the resource constraints. Typical constraints include FLOPs, parameters size, and run-time latency.

Since it is difficult to enumerate all ensemble architectures for evaluation, we resort to a specific K-path evolu-

tion algorithms to find the most promising one. The details are presented in *Appendix B* and Section 3.4.

3.2. Diversity-Guided Search Space Shrinking

Since we search directly for the ensemble models, the search space for each layer increases exponentially from N to $A_N^K = \frac{N!}{(N-K)!}$ compared with single path methods, where N is number of the alternative operators for each layer. The large search space causes inefficiency search and supernet optimization problem. Search space shrinking is a feasible solution to alleviate the problem by discarding inferior operators progressively with a specific metric. Since diversity plays a key role in building a robust ensemble model, we design a new metric to explicitly quantify the diversity across operators inspired by fixed-size determinantal point processing (K-DPP) [17], a popular sampling model with great ability to measure the global diversity and quality within a set. In the following section, we first define the diversity score of an operator combination and then present the diversity-guided search space shrinking pipeline.

Definition of Diversity Score. Assume we have an ensemble model $\Phi(.;\mathcal{A}, W(\mathcal{A}))$, wherer $\mathcal{A} = \{\phi_1, \phi_2, \cdots, \phi_K\}$ consisting of K different paths. Since we fix the depth of the search space, we can slice $\mathcal{A}$ into operator combinations by layer. Then, $\mathcal{A}$ can be reshaped as $\{h_m | h_m = (o_{1,m}, o_{2,m}, \cdots, o_{K,m}), m = 1, 2, \cdots, d\}$, where m and d are the index of the layer and number of total layers, $o_{i,m}$ denotes the operator on layer m of the path i. Now our goal changes to find the optimal operator combination for each layer.

Given that layer m has N alternative operators $O = O_{1,m}, O_{2,m}, \cdots, O_{N,m}$, we construct a DPP kernel $L \in \mathbb{R}^{N\times N}$ for layer m as:

$$L_m = \mathrm{diag}(r^m) \cdot S^m \cdot \mathrm{diag}(r^m), \tag{4}$$

where the kernel is formed by two components: a similarity matrix $S^m \in \mathbb{R}^{N\times N}$ and a quality matrix $r^m \in \mathbb{R}^N$.

Let $v_1, \cdots, v_K$ denote the feature maps output from the K different paths $\phi_1, \cdots, \phi_K$ of the ensemble model $\Phi(.;\mathcal{A}, W(\mathcal{A}))$. We define the similarity $S_{i,j}^m$ of two operators $O_{i,m}$ and $O_{j,m}$ as expected the similarity between paths that contains the two operators, respectively:

$$S_{i,j}^m = \mathbb{E}_{\mathcal{A}\subseteq\mathcal{S}}\Big(\sum_{p,q} \mathbb{I}(i,j,p,q)\exp(-\beta\|v_p - v_q\|_2)\Big), \tag{5}$$

where $1 \le p, q \le K$, β is a scaling factor, and the indicator function is defined as:

$$\mathbb{I}(i,j,p,q) = \begin{cases} 1, & O_{i,m} \in \phi_p, \\ 0, & O_{j,m} \in \phi_q. \end{cases} \tag{6}$$

The quality of operator $O_{i,m}$ is computed by taking expected accuracy of paths containing it. The formal definition is:

$$r_i^m = \gamma\, \mathbb{E}_{\mathcal{A}\subseteq\mathcal{S}}\left(\frac{\sum_{\phi_q | O_{i,m}\in\phi_q} \mathrm{ACC}_{\mathrm{train}'}(\phi_\mathrm{q})}{\#\{\phi_\mathrm{p} | \mathrm{O_{i,m}} \in \phi_\mathrm{p}\}}\right), \tag{7}$$

where $\phi_q, \phi_p \in \mathcal{A}$, $\mathrm{ACC}_{\mathrm{train}'}$ is the accuracy evaluated on a small part of training dataset.

In practice, we do not calculate the exact expectation of similarity matrix and quality matrix. Instead, we randomly sample a finite number of ensemble models and use the mean as an approximate of the expectation.

The diversity score of a certain operator combination h_m of layer m is defined as following:

$$Score(h_m) = \det(L_m^y), \tag{8}$$

where L_m^y is the submatrix of L_m that contains all operators of h_m. The trade-off between similarities and accuracy is controlled by the hyperparameter γ.

According to the the definition of diversity score, we have the following property:
For h_m and h_m' that are different by only the i_{th} operator, if $S_{i,j}^m < S_{i',j}^m$ for $j = 1, 2, \cdots, K$ and $r_i^m > r_{i'}^m$, then

$$Score(h_m) > Score(h_m^{'}). \tag{9}$$

This property suggests that the metric will drop similar and unpromising operator combinations while keep diverse and accurate operator combinations. We refer to *Appendix A* for a proof.

Diversity-Guided Search Space Shrinking. Based on the diversity score, we present Algorithm 1 to describe the diversity-guided search space shrinking pipeline shown in middle of Fig. 2. Note that during the shrinking process, at least one operator combination is preserved, since our method does not change the connectivity of the supernet.

Algorithm 1 Diversity-Guided Search Space Shrinking

Input: A search space $\mathcal{S}$, threshold of search space size $\mathcal{T}$, number of operators dropped out each shrinking k, supernet $\mathcal{G}$, number of ensembles sampled each shrink Z, training epochs between each shrink E.
Output: A shrunk search space $\tilde{\mathcal{S}}$.

```
Let S̃ = S
while |S̃| > T do
    Training the supernet G for E epochs following Section 3.4;
    Sample Z ensemble models Φ_1, Φ_2, ··· , Φ_Z randomly;
    Compute diversity score of each operator combination from S̃ using Eq. 5,7,8;
    Removing k operator combination from S̃ with the lowest k scores
end while
```

3.3. Layer Sharing Among Ensemble Components

The challenge of potential massive complexity of searched ensemble models is handled by the layer sharing mechanism. This mechanism is inspired by several recent studies [16, 26, 30]. These works find that both the same neural architectures with different initialization and different architectures learn similar features in their lower layers. Therefore, we consider to share the shallow layers of different ensemble components. We propose to search for diverse ensemble components with shared shallow layers and different deep layers to reduce the computation cost. To automatically find which layers should be shared, we design a new search dimension called *split point*. The split point defines where the ensemble model will have heterogeneous architectures. It also handles the trade-off between diversity and computation constrain. A comparison between the architectures searched by NEAS and other NAS methods such as [10, 12] is presented in Fig. 3.

3.4. Neural Ensemble Architecture Search

As state in Section 3.1 and in Fig. 2, NEAS includes two sequential phases: K-path supernet training with diversity-guide search space shrinking, and K-path evolution search.

Phase 1: K-Path Supernet Training with Diversity-Guide Search Space Shrinking. For each training iteration, an ensemble model $\Phi(.; \mathcal{A}, W(\mathcal{A}))$ is randomly sampled. In specific, we randomly sample the split point s, the architecture of sharing layers $\mathcal{A}_{sharing} = \{o_1, o_2, \cdots, o_s\}$, and the operator combinations $\mathcal{A}_{split} = \{h_{s+1}, h_{s+2}, \cdots, h_d\}$ for the rest of layers from the shrunk search space. The loss $\mathcal{L}_i$ of each path ϕ_i is computed independently while the backpropagation is performed using the combined loss $\mathcal{L} = \sum_i^K \mathcal{L}_i$ to update the weights of corresponding blocks in the supernet. Following this updating process, the whole network is still trained in an end-to-end style. After training the supernet for several epochs, we follow the steps in Algorithm 1 to shrink the search space. The shrinking and training are conducted alternatively.

During inference, these selected paths make predictions independently, and our ensemble network's output is the average of predictions from all paths.

Phase 2: K-Path Evolution Search. After obtaining the trained supernet, we perform evolution search on it to obtain an optimal ensemble model. These models are evaluated and picked according to the manager of the evolution algorithm. It is worth noting that, before evaluating an ensemble model, we first need to recalculate the batch normalization (BN) statistics for each block. This is because, during the supernet training, the BN statistics of different blocks are optimized simultaneously. These statistics are usually not applicable to the subnets. We randomly extract a part of the ImageNet training set to recalculate the BN statistics.

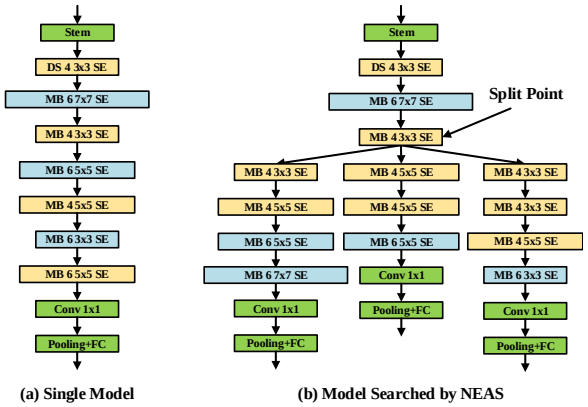

Figure 3. (a) The architecture searched by classical NAS methods (*e.g.*, [10, 6]). (b) The architecture searched by NEAS. Different color means different expansion ratio while the length of the block represents the kernel size.

At the beginning of the evolution search, we pick N_{seed} random architecture as seeds. The top k architectures are picked as parents to generate the next generation by crossover and mutation. In one crossover, two randomly selected candidates are picked and crossed to produce a new one during each generation. We drop the architecture got by crossover if the corresponding architecture is not in the shrunk search space or exceeds the FLOPs constraint. In one mutation, a candidate mutates its split point with a probability P_s. If the split point increases, the number of sharing layers increases with the same number. We randomly pick one path and move its corresponding architectures to the sharing architecture. Otherwise, if the split point decreases, we cut the sharing architecture and add it to each path's architecture. At last, the candidate mutates its layers with a probability of P_m to produce a new candidate. It is worth noting that the operation combinations are only picked from the shrunk search space. We perform crossover and mutation several times to generate new candidates. We generate some random architectures after crossover and mutation to meet the given population demanding. We provide the detailed algorithm in the *Appendix B*.

4. Experiment

In this section, we first give details of our search space and implementation. We then present ablation studies dissecting our method, followed by a comparison with previous state-of-the-art NAS methods. At last, we evaluate the generalization ability and robustness of the searched architecture on COCO object detection benchmark.

4.1. Implementation Details

Search Space. Consistent with previous NAS methods [10, 6, 37], our search space includes a stack of mobile inverted bottleneck residual blocks (MBConv). We also add

Table 1. Comparison of state-of-the-art NAS methods on ImageNet. †: TPU days, ⋆: reported by [10], ‡: searched on CIFAR-10, "-" means not reported. ♢: Tested on *NVIDIA GTX 1080Ti*.

	Methods	Top-1 (%)	Top-5 (%)	FLOPs (M)	Memory cost	Superne train (GPU days)	Search cost (GPU days)	Retrain epochs
200 – 350M	MobileNetV3$_{Large1.0}$ [12]	75.2	-	219	single path	288†	-	150
	OFA [1]	76.9	-	230	two paths	53	2	-
	MobileNetV2 [32]	72.0	91.0	300	-	-	-	-
	MnasNet-A1 [36]	75.2	92.5	312	single path	288†	-	350
	FairNAS-C [6]	74.7	92.1	321	single path	10	2	-
	FBNetV2-L1 [39]	77.2	-	325	-	-	-	400
	SPOS [10]	74.7	-	328	single path	12$^{\diamondsuit}$	< 1	240
	NEAS-S (Ours)	**77.9**	**93.9**	314	K paths	12	< 1	350
350 – 500M	GreedyNAS-A [43]	77.1	93.3	366	single path	7	< 1	300
	EfficientNet-B0 [37]	77.1	93.3	390	-	-	-	350
	FBNetV2-L2 [39]	78.2	-	422	-	-	-	400
	ProxylessNAS[2]	75.1	-	465	two paths	15*	-	300
	Cream-M [29]	79.2	94.2	481	two paths	12	0.02	500
	NEAS-M (Ours)	**79.5**	**94.6**	472	K paths	12	< 1	350
500 – 600M	DARTS [24]	73.3	91.3	574	whole supernet	4‡	-	250
	BigNASModel-L [44]	79.5	-	586	two paths	96†	-	-
	OFA$_{Large}$ [1]	80.0	-	595	two paths	53	2	-
	DNA-d [19]	78.4	94.0	611	single path	24	0.6	500
	EfficientNet-B1 [37]	79.2	94.5	734	-	-	-	350
	NEAS-L (Ours)	**80.0**	**94.8**	574	K paths	12	< 1	350

squeeze-excitation modules to each block following EfficientNet [37] and MobileNetV3 [12]. For details, there are 7 basic operators for each layer, including MBConv with kernel sizes of 3,5,7, expansion rates of 4,6 and skip connect for elastic depth. The split point space is set to range $(9, 20)$ to handle different complexity constrains. In total we have $7^{20K} \times 12 \geq 7 \times 10^{33}$ $(K \geq 2)$ architectures, which is much larger than most NAS methods. A more detailed description of search space could be found in *Appendix A*.

Supernet Training. We train the supernet for 120 epochs using the settings similar to SPOS [10]: SGD optimizer with momentum 0.9 and weight decay 4e-5, initial learning rate 0.5 with a linear annealing. The shrinking process is conducted every 20 epochs. The number of operators dropped each time is empirically set to 20. β in the computing the similarity matrix is set to 1e-3 according to experimental results.

Evolution Search. We set the population N_{seed} of evolution search to 50 with the size of top candidates pool k equals to 10. The number of generations is 20. P_s and P_m are both 0.1. The number of candidates performs mutation and crossover are set to 25 in each generation. We recalculate the BN statistics on a subset of ImageNet.

Retrain. We retrain the discovered architectures for 350 epochs on ImageNet using similar settings as EfficientNet [37]: RMSProp optimizer with momentum 0.9 and decay 0.9, weight decay 1e-5, dropout ratio 0.2, initial learning rate 0.064 with a warmup in the first 10 epochs and a cosine annealing. AutoAugment [7] and exponential moving average are also used for training. We retrain the models with a batch size of 2,048 on 16 Nvidia Tesla V100 GPUs.

Table 2. Comparison of different shrink metrics. Baseline means no search space shrinking during the supernet training. †: average accuracy use weight inherits from supernet. The accuracies are evaluated on ImageNet.

Metric	Kendall Tall	Top-1 (%)	Top-5 (%)	Top-1† (%)
Baseline	0.45	77.3	93.3	67.8
Accuracy	0.42	77.2	93.2	67.2
Diversity	0.65	77.9	93.9	68.3

4.2. Ablation Study

Effectiveness of Diversity Score. We set the baseline as NEAS without diversity-guided shrinking. In addition, we compare the diversity score with the accuracy metric to further verify its efficacy. Since the accuracy-based methods only consider the accuracy of single operators in each layer. We adapt the definition of accuracy to the accuracy of operator combinations. Other methods like Angle-based metric can not easily adapt to search for ensembles.

We first perform correlation analysis to evaluate whether the training process with diversity shrinking can improve the ranking ability of supernet. We randomly sample 30 subnets and calculate the rank correlation between the weight sharing performance and the true performance of training from scratch. Training many such subnets on ImageNet is very computationally expensive. We follow the

Table 3. Comparison of architectures with homogeneous (homo) and heterogeneous (hetero) paths. Both architectures have two paths. The numbers in the columns 2,3,4 are the top-1 accuracies.

	Path 1 (%)	Path 2 (%)	Ens.(%)	FLOPs(M)
homo (first)	79.0	79.1	79.4	566
homo (second)	79.2	79.3	79.6	586
hetero (baseline)	78.9	79.0	80.0	574

Table 4. Comparison of split point in different searched models. Baseline: ensemble model (2 models) with no shared layers. Top-1 and Top-5 represents the top-1 and top-5 accuracy on ImageNet.

Model	Split point	Top-1 (%)	Top-5 (%)	FLOPs (M)
NEAS-L	16	80.0	94.8	574
NEAS-M	16	79.5	94.6	472
NEAS-S	20	77.9	93.9	314
baseline	-	78.5	94.2	605

setting of Cream [29], which constructs a subImageNet dataset consisting of 100 classes randomly sampled from ImageNet. Each class has 250 training images and 50 validation images. We use Kendall Tau to show the ranking capacity of supernet. The second column of Table 2 suggests that our diversity score effectively helps supernet to rank the ensemble architectures in the supernet.

We also retrain the searched architectures by the three methods under the same FLOPs constraint. The top-1 and top-5 accuracy results on the ImageNet dataset are shown in the third and fourth columns in Table 2. We could see that the diversity-guided shrinking is 0.6% better than the baseline and 0.7% better than the accuracy-based method. We further compare the average accuracies of the architectures in the last generation of evolution search, displaying in the fourth columns. Our diversity-guided shrinking surpass the baseline and accuracy-based method by 0.5% and 1.1% top-1 accuracy on ImageNet in the supernet. The results suggest that the diversity score helps remove unpromising candidates and enhance the convergence of supernet.

Impact of Heterogeneous Path Architectures. Ensembling models of homogeneous (homo) architectures are known to be an effective way of building powerful models [18]. We here compare the ensemble models of homogeneous and heterogeneous architectures to show the importance of heterogeneous ensemble. We use our searched two-path ensemble model as the baseline. Then we mirror one path of the searched architecture to form two homogeneous ensemble models for comparison. Fig. 4 gives the visualization of the final hidden features of baseline and the homogeneous network fine-tuned on CIFAR-10. We can see that the homo paths have a similar feature distribution. However, the hetero paths have varied feature distribution and a clearer margin between the clusters.

In table 3, we compare the performance of these three models on ImageNet. This table shows an interesting fact

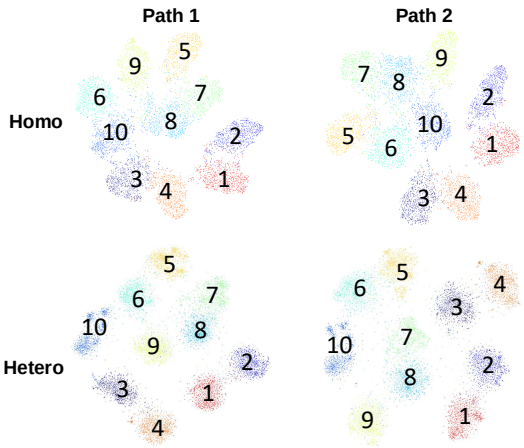

Figure 4. t-SNE visualization on the final hidden features of two different ensemble model. The first row denotes the model with two homogeneous path, while the second has two heterogeneous paths. The inputs are the test set of CIFAR-10.

Table 5. Impact of number of paths predefined for NEAS. Top-1 and Top-5 represents the top-1 and top-5 accuracy on ImageNet.

#paths	Top-1 (%)	Top-5 (%)	FLOPs (M)
2	80.0	94.8	574
3	79.5	94.6	564
5	78.5	94.1	570

that even the stand-alone performance of homo paths are better than hetero. However, the performance of the homo ensemble is worse than the heterogeneous one, indicating that the two paths of the searched model are complementary.

Impact of Layer Sharing. Layer sharing plays a significant role in reducing the complexity of an ensemble model. Here, we explore the effectiveness of layer sharing. The baseline is the ensemble model with no shared layers searched by our method. In Table 4, we could see that layer sharing will help to reduce the complexity of ensemble models largely while keeping outstanding performance. Besides, we observed that in our searched models, the larger model attempts to share fewer layers. One reason could be that the feature expression ability of stand-alone paths in larger models is already strong since it is more complicated. Therefore, they prefer to share fewer layers and get more diverse paths.

Impact of Number of Paths for Ensemble. The number of paths K used to form the ensemble model is a hyperparameter we define at first. We compare the performance of the searched model under the mobile setting ($\leqslant$600M FLOPs) using different K. From Table 5, we can see that when the number of paths is equal to 2, we achieve the best results. One likely reason could be that if a network has too many paths, each path's stand-alone feature expression ability decreases a lot due to complexity constraints.

Table 6. Object detection results of various drop-in backbones on COCO `val2017`. Top-1 accuracies are on ImageNet. †: reported by [6].

Backbones	FLOPs (M)	AP (%)	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Top-1 (%)
MobileNetV3† [12]	219	29.9	49.3	30.8	14.9	33.3	41.1	75.2
MobileNetV2† [32]	300	28.3	46.7	29.3	14.8	30.7	38.1	72.0
FairNAS-C [6]	325	31.2	50.8	32.7	16.3	34.4	42.3	76.7
MnasNet-A2† [36]	340	30.5	50.2	32.0	16.6	34.1	41.1	75.6
MixNet-M† [38]	360	31.3	51.7	32.4	17.0	35.0	41.9	77.0
SPOS† [10]	365	30.7	49.8	32.2	15.4	33.9	41.6	75.0
NEAS-S	314	**33.0**	**53.3**	**34.4**	**17.9**	**36.2**	**43.8**	**78.0**

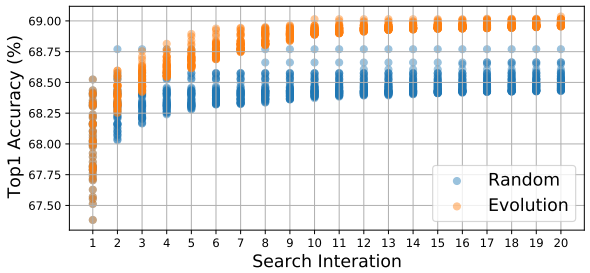

Figure 5. Random search versus evolution algorithm.

Impact of Search Algorithm. Random search is known to be a competitive baseline in NAS methods. We compare random search with evolution search to evaluate the effectiveness of evolution search. We demonstrate the performance of architectures using the weights inherited from supernet on the validation dataset during the search. Top 50 candidates until the current iteration are depicted at each iteration. Fig. 5 illustrates that evolution search is better for searching on supernet.

4.3. Comparisons with State-of-the-Art Methods

Table 1 presents the comparison of our method with state-of-the-arts under mobile settings on ImageNet. It shows that when considering models with FLOPs smaller than 600M, our method consistently outperforms the recent MobileNetV3 [12] and EfficientNet-B0/B1 [37]. In particular, NEAS-L achieves 80.0% top-1 accuracy with only 574M FLOPs, which is 160M FLOPs smaller and 0.8% better than EfficientNet-B1. NEAS-M obtains 79.5% top-1 accuracy with 472M FLOPs. NEAS-S achieves 77.9% accuracy using only 314M FLOPs, which is 0.8% better and 19% smaller than EfficientNet-B0. We also provide results of other state-of-the-art NAS methods in Table 1. It is worth noting that some NAS methods like OFA [1], BigNAS [44], DNA [19] use knowledge distillation to boost the training process and also improve the accuracy of searched models. However, even compared with these methods, our searched ensemble architectures, which do not use knowledge distillation, still achieve superior performance.

4.4. Generalization Ability and Robustness

To further evaluate the generalization ability of the architectures found by NEAS, we transfer the architectures to the downstream COCO [22] object detection task. We use the NEAS-S (pre-trained 500 epochs on ImageNet) as a drop-in replacement for the backbone feature extractor in RetinaNet [21] and compare it with other backbone networks. We perform training on the train2017 set (around 118k images) and evaluation on the val2017 set (5k images) with 32 batch sizes using 8 V100 GPUs. Following the settings in [6], we train the detection model with 12 epochs, an initial learning rate of 0.04, and multiply the learning rate by 0.1 at epochs 8 and 11. The optimizer is SGD with 0.9 momentum and 1e-4 weight decay. As shown in Table 6, our method surpasses MobileNetV2 by 4.7% using similar FLOPs. Compared with MnasNet [36], our method utilizes 7% fewer FLOPs while achieving 2.5% higher performance, suggesting the architecture has good generalization ability when transferred to other vision tasks.

5. Conclusion

In this work, we propose a novel approach to search for lightweight ensemble models based on one-shot NAS. We design a new metric, called *diversity score*, to guide search space shrinking. We further use the layer-sharing mechanism to reduce the complexity of ensemble models and introduce a new search dimension, called *split point*, to handle the trade-off between diversity and complexity constraint. Extensive experiments demonstrate that the proposed new metric is effective and improves the weight sharing supernet's ranking ability. Our searched architectures do achieve not only state-of-the-art performance on ImageNet but also have great generalization ability and robustness.

References

[1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *ICLR*, 2019.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2018.

[3] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019.

[4] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *NeurIPS*, 2019.

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

[6] 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.

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *CVPR*, 2018.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[9] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.

[10] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[12] 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.

[13] Yiming Hu, Yuding Liang, Zichao Guo, Ruosi Wan, Xiangyu Zhang, Yichen Wei, Qingyi Gu, and Jian Sun. Angle-based search space shrinking for neural architecture search. *ECCV*, 2020.

[14] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016.

[16] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019.

[17] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *ICML*, 2011.

[18] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

[19] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020.

[20] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. In *CVPR*, 2020.

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[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 *CVPR*, 2019.

[24] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2018.

[25] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.

[26] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *NeurIPS*, 2018.

[27] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. In *NeurIPS*, 2019.

[28] Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. Asap: Architecture search, anneal and prune. In *AISTATS*. PMLR, 2020.

[29] Houwen Peng, Hao Du, Hongyuan Yu, Qi Li, Jing Liao, and Jianlong Fu. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. *NeurIPS*, 2020.

[30] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NeurIPS*, 2017.

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

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

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[35] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *GECOO*, 2017.

[36] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnas-

net: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

[37] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[38] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. *BMVC*, 2019.

[39] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *CVPR*, 2020.

[40] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[41] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[42] 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 *ICML*, 2020.

[43] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *CVPR*, 2020.

[44] 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. *ECCV*, 2020.

[45] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *CVPR*, 2018.

[46] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[47] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *AI*, 2002.

[48] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2016.

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

Appendix A

In this appendix, we include: (I) proof of the property stated in Section 3.2, (II) the detailed supernet structure and search space.

A-I: Proof of Diversity Score Property

In this section, we show a more detailed formula of the property stated in Section 3.2 and the proof of the property.

Property: Assume that $h_m := (o_{1,m}, \cdots, o_{j,m}, \cdots, o_{K,m})$ and $h_m^{'} := (o_{1,m}, \cdots, o_{j,m}^{'}, \cdots, o_{K,m})$ are different only by j_{th} operator. Denote the indexes of operators in h_m and $h_m^{'}$ as $\sigma_1, \sigma_2, \cdots, \sigma_K$ and $\sigma_1^{'}, \sigma_2^{'}, \cdots, \sigma_K^{'}$. If $S_{i,k}^m < S_{i^{'},k}^m$ for $k = 1, 2, \cdots, K$ and $r_i^m > r_{i^{'}}^m$, then we have:

$$Score(h_m) > Score(h_m^{'}), \tag{10}$$

where σ_j and $\sigma_j^{'}$ equal to i and $i^{'}$.

Proof: Given the property of matrix determinant and definition of L_m^y, the diversity score of h_m could be expressed as:

$$Score(h_m) = \prod_{i=1}^{K} r_{\sigma_i}^2 \cdot \det(S_m^y). \tag{11}$$

where S_m^y are the corresponding submatrixs of h_m in S_m.

According to the assumption, we know that $\prod_{i=1}^{K} r_{\sigma_i}^2 > \prod_{i=1}^{K} r_{\sigma_i^{'}}^2$. Now, if $\det(S_m^y)$ is greater than $\det(S_m^{y^{'}})$ then the property holds easily. Because h_m and $h_m^{'}$ are only different by the j_{th} operator and S_m is a symmetry matrix, the number of total different entries between S_m^y and $S_m^{y^{'}}$ is less than $2K$. We could construct a series of matrixs $B_i \in \mathbb{R}^{K \times K}, i = 0, 1, 2, \cdots, K$ as following:

$$B_i(k,l) = \begin{cases} S_m^y(k,l), & k < i, l = j, \\ S_m^y(k,l), & l < i, k = j, \\ S_m^{y^{'}}(k,l), & Otherwise, \end{cases} \tag{12}$$

where $B_i(k,l)$ is the entry in row k column l. We then prove the following inequality by induction:

$$\det(B_i) \leq \det(B_{i+1}), i = 0, 1, 2, \cdots K-1. \tag{13}$$

For $i = 0$, consider matrix A defined as follow:

$$A(k,l) = \begin{cases} \dfrac{S_m^y(1,j)}{S_m^{y^{'}}(1,j)}, & k = 1, l = j, \\ \dfrac{S_m^y(j,1)}{S_m^{y^{'}}(j,1)}, & l = 1, k = j, \\ 1, & Otherwise. \end{cases} \tag{14}$$

Given the assumption that $S_{i,k}^m < S_{i^{'},k}^m$ for $k = 1, 2, \cdots, K$, we have $S_m^y(1,j) < S_m^{y^{'}}(1,j)$. Then we could get A is a positive define matrix easily using the definition of positive define matrixs. Regarding A and B_0 are both semi-positive define matrix, we have following statement using **Oppenheim's inequality**:

$$\det(A \circ B_0) = \det(B_1) \geq \det(B_0) \prod_{i}^{K} A(i,i) = \det(B_0), \tag{15}$$

where $A \circ B_0$ is the **Hadamard product** (element-wise product) of A and B_0. Besides, B_1 is also a semi-positive define matrix according to **Schur product theorem**.

For $i = 1, 2, \cdots, K-1$, it is easy to construct A with similar definition like above and get the statement that $\det(B_i) \leq \det(B_{i+1})$. Now, combining the chain of inequality, we have:

$$\det(S_m^y) = \det(B_{K-1}) \geq \det(B_0) = \det(S_m^{y^{'}}). \tag{16}$$

Using Eq. (11)(16), the property holds easily.

A-II: Supernet Structure and Search Space

In this section we give the detailed supernet structer and space of the new dimension *Splint Point*.

Input Shape	Operators	Channels	Repeat	Stride
$224^2 \times 3$	3×3 Conv	16	1	2
$112^2 \times 16$	3×3 Depthwise Separable Conv	16	1	2
$56^2 \times 16$	MBConv / SkipConnect	24	4	2
$28^2 \times 24$	MBConv / SkipConnect	40	4	2
$14^2 \times 40$	MBConv / SkipConnect	80	4	1
$14^2 \times 80$	MBConv / SkipConnect	112	4	2
$7^2 \times 112$	MBConv / SkipConnect	160	4	1
$7^2 \times 160$	1×1 Conv	960	1	1
$7^2 \times 960$	Global Avg. Pooling	960	1	-
960	1×1 Conv	1,280	1	1
$1,280$	Fully Connect	1,000	1	-
Split Point		(9, 20, 1)		

Table 7. The structure of the supernet. The "MBConv" contains 6 inverted bottleneck residual block MBConv [32] (kernel sizes of {3,5,7}) with the squeeze and excitation module (expansion rates {4,6}). The "Repeat" represents the maximum number of repeated blocks in a group. The "Stride" indicates the convolutional stride of the first block in each repeated group. (9, 20, 1) means space starts from 9 to 20 with a step of 1.

Appendix B

In appendix B, we show the detailed evolution algorithm, with the detailed algorithm of K-path evolution search below. Specific steps of $\mathrm{Crossover}, \mathrm{Mutation}$ are presented in Section 3.4.

Algorithm 2 K-Path Evolution Search

Input:

Shrunk search space $\tilde{\mathcal{S}}$, weights $W_{\tilde{\mathcal{S}}}$, population size P, resources constraints C, number of generation iteration $\mathcal{T}$, validation dataset D_{val}, training dataset D_{train}, Mutation probability of split point P_s, Mutation probability of layer combination P_m.

Output: The most promising ensemble architecture $\mathcal{A}^*$.

$G_{(0)} :=$ Random sample P ensemble architectures $\{\mathcal{A}_1, \mathcal{A}_2, \cdots \mathcal{A}_P\}$ from $\tilde{\mathcal{S}}$ with constrain C;
while search step $t \in (0, \mathcal{T})$ **do**
  while $\mathcal{A}_i \in G_{(t)}$ **do**
    Recalculate the statistics of BN on D_{train};
    Obtain the accuracy of $\Phi(\cdot; \mathcal{A}_i, W_{\tilde{\mathcal{S}}})$ on D_{val}.
  end while
  $G_{\mathrm{topk}} :=$ the Top K candidates by accuracy order;
  $G_{\mathrm{crossover}} := \mathrm{Crossover}(G_{\mathrm{topk}}, \tilde{\mathcal{S}}, C)$;
  $G_{\mathrm{mutation}} := \mathrm{Mutation}(G_{\mathrm{topk}}, P_s, P_m, \tilde{\mathcal{S}}, C)$;
  $G_{(t+1)} = G_{\mathrm{crossover}} \cup G_{\mathrm{mutation}}$
end while