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

Minghao Chen\textsuperscript{1,} Jianlong Fu\textsuperscript{2}, Haibin Ling\textsuperscript{1}
\textsuperscript{1}Stony Brook University \textsuperscript{2}Microsoft Research Asia
\{minghao.chen, haibin.ling\}@stonybrook.edu, 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 \url{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 \cite{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) \cite{48} aims to automate the design process. Recently, NAS methods have achieved state-of-the-arts on varied tasks such as image classification \cite{48}, semantic segmentation \cite{23}, object detection \cite{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.

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 \cite{34}, Dropconnect \cite{40}, StochDepth \cite{15}, Shake-Shake \cite{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.

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

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 search dimension Split Point. In Section 3.3, we introduce the layer sharing mechanism and the new search dimension Score Operators Combinations. 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 $\Omega$ 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 $\Omega$ 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:

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

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:

$$W_S = \arg \min_W \mathcal{L}_{\text{train}}(\Phi(\cdot; \mathcal{A}, W(\mathcal{A}))),$$

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 $\Phi$ 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 $W_S$ of supernet, which is formulated as

$$\mathcal{A}^* = \arg \max_{\mathcal{A}} \text{ACC}_{\text{val}}(\Phi(\cdot; \mathcal{A}, W_S(\mathcal{A}))),$$

s.t. $\sum_i g_i(\phi_i) < C,$

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 evolution 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 $A_N^K = N^{K \times N}$ compared with single path methods, where $N$ is the number of the alternative operators for each layer. The large search space causes inefficiency search and superenrollment 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 diversity, we present Algorithm 1 diversity-guided search space shrinking pipeline shown in following section. First, we 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 $\Phi(\cdot; A, W(A))$, where $A = \{\phi_1, \phi_2, \ldots, \phi_K\}$ consisting of $K$ different paths. Since we fix the depth of the search space, we can slice $A$ into operator combinations by layer. Then, $A$ can be reshaped as $[h_m]_m = (\phi_{1,m}, \phi_{2,m}, \ldots, \phi_{K,m}), m = 1, 2, \ldots, d$, where $m$ and $d$ are the index of the layer and number of total layers, $\phi_{i,m}$ 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 $O = O_{1,m}, O_{2,m}, \ldots, O_{N,m}$, we construct a DPP kernel $L \in \mathbb{R}^{N \times N}$ for layer $m$ as:

$$L_m = \text{diag}(r_m^m) \cdot S_m \cdot \text{diag}(m),$$

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

$$S^m_{ij} = \mathbb{E}_{A \subseteq S} \left( \sum_{p,q} I(i,j,p,q) \exp(-\beta \|v_p - v_q\|_2) \right),$$

(5)

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

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

(6)

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

$$r^m_i = \gamma \mathbb{E}_{A \subseteq S} \left( \sum_{\phi \in \Phi} \frac{ACCTrain(\phi_i)}{\#(\phi|O_{i,m} \in \phi)} \right),$$

(7)

Algorithm 1 Diversity-Guided Search Space Shrinking

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

Output: A shrunk search space $\tilde{S}$.

1. Let $\tilde{S} = S$
2. while $|\tilde{S}| > T$ do
3. Training the supernet $G$ for $E$ epochs following Section 3.4;
4. Sample $Z$ ensemble models $\Phi_1, \Phi_2, \ldots, \Phi_Z$ randomly;
5. Compute diversity score of each operator combination from $\tilde{S}$ using Eq. 5.7.8;
6. Removing $k$ operator combination from $\tilde{S}$ with the lowest $k$ scores
7. end while

where $\phi_q, \phi_p \in A$, ACC$\text{Train}^{'(h)}$ 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 approximation of the expectation.

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

$$Score(h_m) = \det(L^m_u),$$

(8)

where $L^u_m$ 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 $\gamma$.

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 $\iota_{th}$ operator, if $S^m_{ij} < S^m_{ij'}$ for $j = 1, 2, \ldots, K$ and $r^m_i > r^m_i'$ then

$$Score(h_m) > Score(h_m'),$$

(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.

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 stated 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 \( \Phi(\cdot; A, W(A)) \) is randomly sampled. In specific, we randomly sample the split point \( s \), the architecture of sharing layers \( A_{\text{sharing}} = \{a_1, a_2, \ldots, a_s\} \), and the operator combinations \( A_{\text{split}} = \{h_{s+1}, h_{s+2}, \ldots, h_d\} \) for the rest of layers from the shrunk search space. The loss \( L_i \) of each path \( \phi_i \) is computed independently while the backpropagation is performed using the combined loss \( L = \sum_i^K L_i \) 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.

At the beginning of the evolution search, we pick \( N_{\text{seed}} \) 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 se-

![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.](image-url)
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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>200 – 350M</th>
<th>350 – 500M</th>
<th>500 – 600M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top-1 (%)</td>
<td>Top-5 (%)</td>
<td>FLOPs (M)</td>
</tr>
<tr>
<td>MobileNetV3_{large} [12]</td>
<td>75.2</td>
<td>219</td>
<td>-</td>
</tr>
<tr>
<td>OFA [1]</td>
<td>76.9</td>
<td>230</td>
<td>-</td>
</tr>
<tr>
<td>MobileNetV2 [32]</td>
<td>72.0</td>
<td>91.0</td>
<td>300</td>
</tr>
<tr>
<td>MnasNet-A1 [36]</td>
<td>75.2</td>
<td>92.5</td>
<td>312</td>
</tr>
<tr>
<td>FairNAS-C [6]</td>
<td>74.7</td>
<td>92.1</td>
<td>321</td>
</tr>
<tr>
<td>FBNetV2-L1 [39]</td>
<td>77.2</td>
<td>325</td>
<td>-</td>
</tr>
<tr>
<td>SPOS [10]</td>
<td>74.7</td>
<td>328</td>
<td>-</td>
</tr>
<tr>
<td>NEAS-S (Ours)</td>
<td>77.9</td>
<td>93.9</td>
<td>314</td>
</tr>
<tr>
<td>GreedyNAS-A [43]</td>
<td>77.1</td>
<td>93.3</td>
<td>366</td>
</tr>
<tr>
<td>EfficientNet-B0 [37]</td>
<td>76.3</td>
<td>93.2</td>
<td>390</td>
</tr>
<tr>
<td>FBNetV2-L2 [39]</td>
<td>78.2</td>
<td>422</td>
<td>-</td>
</tr>
<tr>
<td>ProxylessNAS[2]</td>
<td>75.1</td>
<td>465</td>
<td>-</td>
</tr>
<tr>
<td>Cream-M [29]</td>
<td>79.2</td>
<td>94.2</td>
<td>481</td>
</tr>
<tr>
<td>NEAS-M (Ours)</td>
<td>79.5</td>
<td>94.6</td>
<td>472</td>
</tr>
<tr>
<td>DARTS [24]</td>
<td>73.3</td>
<td>91.3</td>
<td>574</td>
</tr>
<tr>
<td>BigNASModel-L [44]</td>
<td>79.5</td>
<td>586</td>
<td>-</td>
</tr>
<tr>
<td>OFA_{large} [1]</td>
<td>80.0</td>
<td>595</td>
<td>-</td>
</tr>
<tr>
<td>DNA-d [19]</td>
<td>78.4</td>
<td>94.0</td>
<td>611</td>
</tr>
<tr>
<td>EfficientNet-B1 [37]</td>
<td>79.2</td>
<td>94.5</td>
<td>734</td>
</tr>
<tr>
<td>NEAS-L (Ours)</td>
<td>80.0</td>
<td>94.8</td>
<td>574</td>
</tr>
</tbody>
</table>

nect for elastic depth. The split point space is set to range (9, 20) to handle different complexity constraints. In total we have $24K \times 12 \geq 7 \times 10^{31} (K \geq 2)$ 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. $\beta$ 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.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Kendall Tall</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
<th>Top-1 † (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.45</td>
<td>77.3</td>
<td>93.3</td>
<td>67.8</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.42</td>
<td>77.2</td>
<td>93.2</td>
<td>67.2</td>
</tr>
<tr>
<td>Diversity</td>
<td>0.65</td>
<td>77.9</td>
<td>93.9</td>
<td>68.3</td>
</tr>
</tbody>
</table>

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 cannot 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 setting of Cream [29], which constructs a subImageNet dataset consisting of 100 classes randomly sampled from
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.

<table>
<thead>
<tr>
<th></th>
<th>Path 1 (%)</th>
<th>Path 2 (%)</th>
<th>Ens. (%)</th>
<th>FLOPs (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>homo (first)</td>
<td>79.0</td>
<td>79.1</td>
<td>79.4</td>
<td>566</td>
</tr>
<tr>
<td>homo (second)</td>
<td>79.2</td>
<td>79.3</td>
<td>79.6</td>
<td>586</td>
</tr>
<tr>
<td>hetero (baseline)</td>
<td>78.9</td>
<td>79.0</td>
<td>80.0</td>
<td>574</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Split point</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
<th>FLOPs (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEAS-L</td>
<td>16</td>
<td>80.0</td>
<td>94.8</td>
<td>574</td>
</tr>
<tr>
<td>NEAS-M</td>
<td>16</td>
<td>79.5</td>
<td>94.6</td>
<td>472</td>
</tr>
<tr>
<td>NEAS-S</td>
<td>20</td>
<td>77.9</td>
<td>93.9</td>
<td>314</td>
</tr>
<tr>
<td>baseline</td>
<td>-</td>
<td>78.5</td>
<td>94.2</td>
<td>605</td>
</tr>
</tbody>
</table>

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 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 ($\leq 600$M 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.

Impact of Search Algorithm. Random search is known to be a competitive baseline in NAS methods. We com-
Table 6. Object detection results of various drop-in backbones on COCO val2017. Top-1 accuracies are on ImageNet. †: reported by [6].

<table>
<thead>
<tr>
<th>Backbones</th>
<th>FLOPs (M)</th>
<th>AP (%)</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_S$</th>
<th>AP$_M$</th>
<th>AP$_L$</th>
<th>Top-1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV3†</td>
<td>219</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>
<td>75.2</td>
</tr>
<tr>
<td>MobileNetV2†</td>
<td>300</td>
<td>28.3</td>
<td>46.7</td>
<td>29.3</td>
<td>14.8</td>
<td>30.7</td>
<td>38.1</td>
<td>72.0</td>
</tr>
<tr>
<td>FairNAS-C</td>
<td>325</td>
<td>31.2</td>
<td>50.8</td>
<td>32.7</td>
<td>16.3</td>
<td>34.4</td>
<td>42.3</td>
<td>76.7</td>
</tr>
<tr>
<td>MnasNet-A†</td>
<td>340</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>
<td>75.6</td>
</tr>
<tr>
<td>MixNet-M†</td>
<td>360</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>
<td>77.0</td>
</tr>
<tr>
<td>SPOS†</td>
<td>365</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>
<td>75.0</td>
</tr>
<tr>
<td>NEAS-S</td>
<td>314</td>
<td>33.0</td>
<td>53.3</td>
<td>34.4</td>
<td>17.9</td>
<td>36.2</td>
<td>43.8</td>
<td>78.0</td>
</tr>
</tbody>
</table>

Figure 5. Random search versus evolution algorithm.

pare 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 1.6% 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.

Acknowledgments

We are very grateful to Minghao’s mentor Dr. Houwen Peng at MSRA for his great and thorough contributions in all aspects including idea proposals, intensive discussion, experiment design and analysis, and writing, etc. Ling was supported in part by US National Science Foundation (1814745 and 2006665).
References


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


