MergeNAS: Merge Operations into One for Differentiable Architecture Search

Xiaoxing Wang\textsuperscript{1*}, Chao Xue\textsuperscript{3}, Junchi Yan\textsuperscript{2,1†}, Xiaokang Yang\textsuperscript{1}, Yonggang Hu\textsuperscript{4} and Kewei Sun\textsuperscript{4}

\textsuperscript{1}MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
\textsuperscript{2}Department of Computer Science and Engineering, Shanghai Jiao Tong University
\textsuperscript{3}IBM Research – China
\textsuperscript{4}IBM System

Abstract

Differentiable architecture search (DARTS) has been a promising one-shot architecture search approach for its mathematical formulation and competitive results. However, besides its caused high memory utilization and a large computation requirement, many research works have shown that DARTS also often suffers notable over-fitting and thus does not work robustly for some new tasks. In this paper, we propose a one-shot neural architecture search method referred to as MergeNAS by merging different types of operations, e.g., convolutions into one operation. This merge-based approach not only reduces the search cost (about half a GPU day), but also alleviates over-fitting by reducing the redundant parameters. Extensive experiments on different search space and various datasets have been conducted to verify our approach, showing that MergeNAS can converge to a stable architecture and achieve better performance with fewer parameters and search cost. For test accuracy and its stability, MergeNAS outperforms all NAS baseline methods implemented on NAS-Bench-201, including DARTS, ENAS, RS, BOHB, GDAS and hand-crafted ResNet.

1 Introduction

Neural Architecture Search is a process to automatically search for an efficient neural network instead of manually design, which need a large amount of trials and expert knowledge. There are several basic approaches for neural architecture search: reinforcement learning based NAS, evolutionary algorithm based NAS, and differentiable NAS (DARTS). Many works \cite{Baker2017, Zoph2017, Zhong2018, Zoph2018} frame NAS as a reinforcement learning problem, and consider the generation of an architecture as the agent's action. ENAS \cite{Pham2018} proposed weight sharing strategy among the same operations in different architectures during the search phase, which can significantly reduce the search cost and obtain a great performance. Other works \cite{Real2017, Liu2018b, Real2019, Miikkulainen2019, Xie2017, Elsken2019} encode the neural architecture and use genetic algorithms to propose a group of architectures as a population. New individuals are generated based on the rules of crossover and mutation, and the next population is selected according to the validation accuracy of each architecture. Unlike the first two frameworks that regards the searching process as a black box, DARTS \cite{Liu2019} introduces architecture parameters to indicate the saliency of the edges and thus trains a single model that contains all possible operations. By solving a bi-level optimization problem, DARTS can obtain an efficient neural architecture in a few GPU days.

However, DARTS suffers a severe over-fitting problem, which is also referred as the problem of stability \cite{Bi2019}. Multiple experiments indicate that the saliency of none and skip-connect operations will dominate the architecture parameters after a hundred of epochs, and thus the accuracy of the chosen architecture declines. Recent work \cite{Zela2020} points out that involving early stop strategy or increasing the weight decay can improve the stability of searching process. In this work, we consider another solution to overcome over-fitting problem by merging different types of convolutions into one and sharing weights among those convolutional operations. Specifically, we maintain a single convolution with a large kernel size as the only parametric operation in the search space, and kernels of other convolutions can be derived from the large kernel.

The contributions of our approach are as follows:

1) MergeNAS Approach. We propose MergeNAS based on DARTS to merge the convolutions with different kernel sizes and dilation rates into one. We also provide weight merge strategy for separable convolutions. It should be noticed that weight merge strategy can be applied to one-shot based NAS approaches, including but not limited to DARTS.

2) Strong Robustness. According to the experiments, we observe the unstable searching process of DARTS, which is also the over-fitting problem for one-shot based NAS. MergeNAS is able to alleviate the over-fitting problem by reducing redundant parameters in one-shot model. For test accu-
racy and stability, MergeNAS achieves state-of-the-art performance on NAS-Bench-201.

3) Strong Cost-effectiveness. We evaluate MergeNAS on two typical search spaces: the micro search space of ENAS [Pham et al., 2018] and the search space in NAS-Bench-201 [Dong and Yang, 2020]. Compared to DARTS, our approach can converge to an efficient architecture with fewer memory cost and searching time.

2 Related Work

One-shot Architecture Search. One-shot NAS [Bender et al., 2018] searched for an efficient architecture by regarding the neural architecture as a Directed Acyclic Graph (DAG), and constructing a large graph which integrates all types of operations and connections. A neural architecture in the search space can be seen as a sub-graph of the one-shot model. Based on one-shot NAS, the authors in [Liu et al., 2019] introduced the architecture parameters indicating the importances of the operations and connections, which were optimized iteratively with the network weights based on gradient decent algorithm. Another work [Nayman et al., 2019] regarded the one-shot NAS as an online selection task, and utilized the prediction with experts advice (PEA) theory to select the operations and connections. Some other approaches [Chen et al., 2019; Zhou et al., 2019] were proposed to reduce the memory consuming by pruning the redundant connections in the search phase, which is also a problem of one-shot NAS due to the over-parameterized model.

Weight Sharing. ENAS [Pham et al., 2018] first used weight sharing to train the weights of network selected by an LSTM controller in searching process, which can significantly reduce the search time with a good performance. Another work [Chu et al., 2019] devised FairNAS to randomly sample $M$ architectures from a one-shot model, and update the network weights belonging to the selected operations in the $M$ networks. Based on DARTS, ProxylessNAS [Cai et al., 2019] and SNAS [Xie et al., 2019] proposed to sample a single path of the one-shot model and only trained weights of the selected small model in one iteration during the search phase. ProxylessNAS proposed to binarize the connections between every two nodes by sampling according to the softmax of the architecture parameters. While SNAS utilized the Gumbel softmax trick to sample selected operations.

Discussion. Unlike weight sharing that reuse the weights of the same operations in various trials, our approach proposes weight merge for one-shot NAS by merging the convolutions with different kernel size and dilation rate into one. Weight merge can not only share the parameters, but also share the computations, which is the distinction between weight sharing and weight merge. Recent work [Stamoulis et al., 2019] is also related to our approach. However, they fixed the backbone of the architecture as a specific architecture based on MobileNetV2 [Sandler et al., 2018], and only mixed up different kernels in the depth-wise convolution. Consequently, the search space only contains the kernel size of depth-wise convolutions, which is much smaller than ours. In contrast, we apply weight merge strategy on one-shot DARTS, and search for not only the types of operations but also the connections.

3 The Proposed MergeNAS

To reduce the number of parameters of one-shot model, we expand weight sharing strategy by sharing weights among the convolutions with different kernel sizes and dilation rates on the same edge. We introduce three propositions, and demonstrate that the convolutions can be merged into one operation when they share weights. Furthermore, we also introduce how to merge parameterless operations, such as skip-connect and average pooling, into the one operation.

3.1 Theoretical Study

Based on one-shot NAS, we denote $e_{ij}$ as the edge from node $i$ to node $j$, and $o_{ij}$ as the operation associated with edge $e_{ij}$. Similar to some one-shot based works [Liu et al., 2019; Bender et al., 2018], we limit the selection of operations to a basic set denoted as $O$. We also include the architecture parameters to control the saliency of different operations in edges, denoted as $o_{ij}^o$. The output of the edge $e_{ij}$ is denoted as $o_j(z_i)$, which is the weighted average of the operations $o_{ij}$.

$$o_j(z_i) = \sum_{o\in O} o_{ij}^o o_{ij}(z_i)$$

where $z_i$ is the output of node $i$, and the output of node $j$ can be obtained by

$$z_j = \sum_{i<j} o_j(z_i)$$

In one-shot NAS, the search space consists of parameterless and parametric operations. On the one hand, zero operation, identity operation and pooling operation are categorized into parameterless operations. On the other hand, multiple types of convolutions with different kernel sizes and dilation rates representing various receptive fields belong to parametric operations which are trainable. We denote the set of parametric operations as $O^p \subset O$, and the weighted average of all the parametric (convolutional) operations in the edge $e_{ij}$ is denoted as $o_j^p(z_i)$, where $z_i$ is the output of node $i$, $W_{ij}^p$ is the parameters belonging to the operation $o_{ij}$, and $*^p$ denotes the convolutional operation.

$$o_j^p(z_i) = \sum_{o\in O^p} o_{ij}^o o_{ij}(z_i) = \sum_{o\in O^p} o_{ij}^o [z_i * W_{ij}^p]$$

In the following, we omit the subscript $(i, j)$ for conciseness. To derive new techniques of reducing the computation of the parametric operations, we first present the following propositions.

**Proposition 1.** The convolution operation with kernel $W$ is equivalent to\(^1\) a set of convolution operations whose kernel $W$ is derived from $W$ by padding $2n$ $(n \geq 0)$ zeros around.

**Proposition 2.** The dilated convolution operation with kernel size $k^{(1)}$ and dilation rate $r$ is equivalent to a normal convolution operation with kernel size $k^{(2)} = r(k^{(1)} - 1) + 1$. In the

\(^1\)Equivalence in this paper means the output of two convolutions are the same.
kernel of the normal convolution, there exits $r$ zeros between two non-zero weights.

Proof. Suppose a convolution with dilution rate $r$ has weight tensor $W_r^{(0)} \in \mathbb{R}^{C_o \times C_i \times k_h^{(0)} \times k_w^{(0)}}$, where $C_o, C_i$ are the number of output and input channels, respectively. For simplification, we denote $W_{i,j,0,0}$ the central value of the $j^{th}$ kernel in the $i^{th}$ filter of the tensor $W$, and thus $W_{i,j,-1,0}$ indicates the nearest value above the central value. A tensor $W_r^{(1)}$ can be derived as $W_r^{(0)}$ with kernel size $k_h^{(1)} = rk_h^{(0)}, k_w^{(1)} = rk_w^{(0)}$. We then define the tensor $W_r^{(2)}$ with a larger kernel $k_h^{(2)} \times k_w^{(2)}$ by padding $2n$ ($n \geq 0$) zeros around $W_r^{(1)}$.

\[
W_{c,m,p,q}^{(2)} = \begin{cases} 
W_{c,m,p/r,q/r}^{(0)} & \text{p, q mod } r = 0 \\
0 & \text{otherwise} 
\end{cases} 
\]  

(4)

Proposition 1 holds if the output of the convolution $o_1$ with weight $W_r^{(2)}$ equals to the convolution with weight $W_r^{(1)}$, and Proposition 2 holds if the convolution with $W_r^{(1)}$ equals to the convolution with $W_r^{(0)}$. The output tensor $Y$ of a normal convolution, whose input and weight are $X$ and $W^{(2)}$, can be computed as Eq. 5. For conciseness, we define $u_h^{(i)} = (k_h^{(i)} - 1)/2, u_w = (k_w^{(i)} - 1)/2, i \in \{0, 1, 2\}$

\[
Y_{n,c,h,w} = \sum_{m=0}^{C_i-1} \sum_{p=-u_h^{(2)}}^{u_h^{(2)}} \sum_{q=-u_w^{(2)}}^{u_w^{(2)}} X_{n,m,h+p,w+q} W^{(2)}_{c,m,p,q}
\]  

(5)

The result of Eq. 5 remains unchanged by removing the terms that equal to zero. As a consequence, The above convolution is equivalent to the normal convolution with the weight $W^{(1)}$, which omits the zero padding of $W^{(2)}$. Furthermore, by omitting the zero values inside the weight tensor, we can compute the tensor $Y$ as Eq. 6, which is the dilation convolution with the weight $W^{(0)}$.

\[
Y_{n,c,h,w} = \sum_{m=0}^{C_i-1} \sum_{p=-u_h^{(0)}}^{u_h^{(0)}} \sum_{q=-u_w^{(0)}}^{u_w^{(0)}} X_{n,m,h+p,w+q} W^{(1)}_{c,m,p,q}
\]  

(6)

As a consequence, we can learn that the output of the dilation convolution equals to a normal convolution, which is also equivalent to a set of normal convolutions by padding zero around the kernels\(^2\). This ends the proof. \qed

According to Proposition 1 and 2, we can obtain the equivalent convolutions with a large kernel size for all the parametric operations in the operation set $O$. As shown in Figure 1, the white block indicates zero, and the red block indicates the weights for original operations. Based on the homogeneity and linearity of convolution, we can then simplify the Equation 3 as follows, where $\tilde{W}^{o}$ is the equivalent weights with large kernel size for convolutions in the search space, and $\tilde{W}$ is the merged weights of the single convolution.

\[
o^{o}(z) = \sum_{o \in O^{o}} \alpha^{o} [z \ast W^{o}] = z \ast \left[ \sum_{o \in O^{o}} \alpha^{o} \tilde{W}^{o} \right] = z \ast \tilde{W}
\]  

(7)

According to Eq. 7, various convolutions can be merged into one. We can obtain the weighted average of the convolutions by conducting convolutional operation once, which reduces the forward computation.

However, the number of parameters in one-shot model is unchanged, and the model still demands a large amount of GPU memories. To reduce the memory cost and further accelerate the searching process, we propose an approach (MergeNAS) to share the weights among different convolutions.

We first describe MergeNAS for normal convolution. Then we derive the formula of weight merge strategy for separable convolution. We also introduce the merge of parameterless operations which can also be merged into one convolution.

### 3.2 Merge of Normal Convolutions

To further reduce the parameters in one-shot model, we first construct a tensor $W$ with a large kernel size as the merged weights. Then all convolutions’ weights can be derived from the single tensor $W$. Specifically, we use mask matrices $M^{o}, o \in O^{o}$ to denote the receptive field of different parametric operations as shown in Figure 1, and thus $\tilde{W}^{o} = M^{o} \cdot W$, where the convolution with weights $\tilde{W}^{o}$ is equivalent to the convolution with weights $W^{o}$ according to Proposition 1 and 2. Therefore, the weights of the merged convolution $\tilde{W}$ in Eq. 7 can be simplified as:

\[
\tilde{W} = \sum_{o \in O^{o}} \alpha^{o} \tilde{W}^{o} \left[ \sum_{o \in O^{o}} \alpha^{o} M^{o} \right] \cdot W = \tilde{M} \cdot W
\]  

(8)

The matrix $\tilde{M} = \sum \alpha^{o} M^{o}$ denotes the saliency of different receptive field, as shown in Figure 1(e). By sharing the weights among convolutions, MergeNAS can reduce not only the number of parameters but also the amount of computation. Multiple experiments on NAS-Bench-201 [Dong and Yang, 2020] have been conducted in Section 4 to show the efficiency and efficacy of MergeNAS for normal convolutions.

### 3.3 Merge of Separable Convolutions

Many approaches [Liu et al., 2019; Bender et al., 2018] used separable convolutions as parametric operations because of large GPU memory costs in one-shot NAS. In order to apply MergeNAS approach on separable convolutions, we have the following proposition to reveal the relationship between separable convolution and normal convolution.

Proposition 3. Given no activation and normalization layers between the depth-wise and point-wise convolution, the separable convolution is equivalent to a normal convolution, whose weight tensor is the Hadamard Product of the weight tensors of depth-wise and point-wise convolution.

\[^2\]The spatial size of $Y$ in Eq. 5 and Eq. 6 is guaranteed to be the same by padding zeros around $X$
Figure 1: (a)-(d) is the equivalent mask $M^n$ of the receptive field for different convolutions (red denotes non-zero weights). (e) is the weighted average mask $\bar{M}$. Different color indicate various combinations of architecture parameters $\alpha$. In the micro space search of DARTS, there are four kinds of convolutions (see Section 4).

**Proof.** Noted that the condition of Proposition 3, that no activation and normalization layers exist between the depth-wise and point-wise convolution, is satisfied in the search space of DARTS [Liu et al., 2019] and ENAS [Pham et al., 2018].

We then prove that the proposition holds for dilated separable convolution, which degrades to a normal separable convolution when the dilation rate $r = 1$. Suppose a dilated separable convolution with dilation rate $r$ has the depth-wise weight $W^d \in \mathbb{R}^{C \times 1 \times k \times k_x \times k_w}$ and point-wise weight $W^p \in \mathbb{R}^{C \times C \times 1 \times 1}$. The output tensor $Y$ of the normal convolution with the input $X$ can be computed by Eq. 10, where $u_h = (k_h - 1)/2$, $u_w = (k_w - 1)/2$.

$$
Y^d_{n,m,h,w} = \sum_{p=-u_h}^{u_h} \sum_{q=-u_w}^{u_w} X_{n,m,h+rp,w+rq} W^d_{m,1,p,q} \tag{9}
$$

$$
Y_{n,c,h,w} = \sum_{m=0}^{C-1} Y^d_{n,m,h,w} W^p_{c,m,1,1}
$$

$$
= \sum_{p=-u_h}^{u_h} \sum_{q=-u_w}^{u_w} \sum_{m=0}^{C-1} X_{n,m,h+rp,w+rq} [W^d_{m,1,p,q} W^p_{c,m,1,1}]
$$

$$
= \sum_{p=-u_h}^{u_h} \sum_{q=-u_w}^{u_w} \sum_{m=0}^{C-1} X_{n,m,h+rp,w+rq} W^s_{c,m,p,q} \tag{10}
$$

Thus, the separable convolution is equivalent to a normal convolution with weights $W^s$. This ends the proof.

To unify the notation, we denote $W^{d,o}, W^{p,o}(o \in \mathcal{O})$ as the weights of the depth-wise and the point-wise convolution, and the weights of the equivalent convolution is denoted as $W^{s,o}$, which can be obtained by Eq. 11, where $[W^{d,o}]^T \in \mathbb{R}^{1 \times C \times k \times k_x \times k_w}$ is the transposed version of $W^{d,o}$ along the first two dimensions, and $\cdot^T$ denotes Hadamard product.

$$
W^{s,o} = W^{d,o} \ast W^{p,o} = [W^{d,o}]^T \cdot W^{p,o} \tag{11}
$$

According to Eq. 8, we can compute the weights of the merged convolution as follows by sharing the weights of depth-wise convolution denoted as $W^d$.

$$
\tilde{W} = \sum_{o \in \mathcal{O}} \alpha^o M^o \cdot [W^d]^T \cdot W^{p,o} \tag{12}
$$

The weights of point-wise convolution, denoted as $W^p$ can be further shared, and the merged weights of the single convolution is as follows:

$$
\tilde{W} = [W^d]^T \cdot \sum_{o \in \mathcal{O}} \alpha^o M^o \cdot W^p = \left[\tilde{M} \cdot W^d\right]^T \cdot W^p = W^d \ast W^p \tag{13}
$$

As such, the weighted average of separable convolutions can be merged into a single separable convolution whose depth-wise weight is $W^d$ and point-wise weight is $W^p$. The specific algorithm is shown in Alg. 1

**Algorithm 1 MergeNAS: Weight Merge for NAS**

**Input:**
1) Operation set $\mathcal{O}$;
2) Parametric operation set $\mathcal{O}^p \subset \mathcal{O}$;
3) Parameterless operation set $\mathcal{O}^d = \mathcal{O} \setminus \mathcal{O}^p$;
4) The input of node $j$, $z_i$;

**Parameter:**
1) The sharing weights for edge $e_{ij}$ of one-shot model ($W_{ij}$ for normal, and $W^d_{ij}, W^p_{ij}$ for separable convolution);
2) The architecture parameters for edge $e_{ij}, \alpha^o_{ij}, o \in \mathcal{O}$;
3) Mask matrix for convolutional operations $M^o, o \in \mathcal{O}^p$;

**Output:**
The output of node $j$, $z_j$;

**Procedure:**
1: Compute the saliency matrix $\bar{M}_{ij} = \sum_{o \in \mathcal{O}^p} \alpha^o_{ij} M^o$;
2: if normal convolution in $\mathcal{O}^p$ then
3: $\tilde{W}_{ij} = \bar{M}_{ij} \cdot W_{ij}$
4: Compute the single convolution with weights $\tilde{W}_{ij}$:
5: if separable convolution in $\mathcal{O}^p$ then
6: $W^d_{ij} = \bar{M}_{ij} \cdot W^d_{ij}$;
7: Compute single separable convolution with depth-wise weights $W^d_{ij}$ and point-wise weights $W^p_{ij}$:
8: end if
9: $o_j(z_i) = \alpha^o_j (z_i) + \sum_{o \in \mathcal{O}^d} \alpha^o_{ij} o_{ij}(z_i)$
10: return the output of edge $e_{ij}, o_j(z_i)$;
3.4 Merge of Parameterless Operations

To further reduce the memory cost for MergeNAS, we can also merge the skip connect operation and average pooling into the single convolution. Without changing spatial size and number of channels, the skip connect operation is equivalent to a point-wise convolution whose weights can be derived from an Identity Matrix. By Proposition 1, we can obtain the equivalent convolution with a larger kernel size. Average pooling can be computed by a separable convolution whose input channel $C_i$ equals to the output channel $C_o$. The depth-wise weights are filled with $1/(k_h \times k_w)$, where $k_h$, $k_w$ are the height and width of the kernel. While the point-wise weights is an identity matrix. Based on Proposition 3, we can obtain the equivalent normal convolution for average pooling.

Based on Eq. 7, the average pooling and skip-connect can be merged with a normal convolution. However, it is hard to merged them with a separable convolution, which needs to share the weights of both depth-wise and point-wise convolutions.

4 Experiments

The efficiency and stability of MergeNAS is evaluated on two cell search spaces: the micro search space of ENAS [Pham et al., 2018] and DARTS [Liu et al., 2019], as well as the search space in NAS-Bench-201 [Dong and Yang, 2020].

In the micro search space of ENAS, the operation set $O$ contains the separable convolutions with different kernel size. DARTS replaced the single separable convolutions by a pair of separable convolutions and expanded the search space by involving dilated convolutions. In our settings, we directly expand the micro search space of ENAS by involving dilated convolutions and still use single separable convolutions with kernel size $3 \times 3$ and $5 \times 5$ as the previous work [Pham et al., 2018], due to their good performance on multiple experiments. Therefore, there are 8 operations in our experiments: zero, identity, max pooling, average pooling, $3 \times 3$ separable convolution, $5 \times 5$ separable convolution, $3 \times 3$ dilated separable convolution, and $5 \times 5$ dilated separable convolution.

NAS-Bench-201 is a benchmark for neural architecture search. It designed a specific search space for only normal cells and maintained the reduced cell as the bottleneck structure of ResNet [He et al., 2016] to limit the size of search space. The set $O$ contains 5 basic operations: none, skip connect, average pooling, $1 \times 1$ convolution, and $3 \times 3$ convolution. NAS-Bench-201 has evaluated validation and test accuracy of all the possible architectures in the search space, so we can focus on the searching process and directly index for the information of the architecture obtained by our approach.

4.1 Search on Micro Cells of ENAS and DARTS

Training Settings. We set the training settings similar as DARTS. In search phase, we train the one-shot model stacked
by 8 cells for 50 epochs, and optimize the architecture parameters and the network weights based on the two-step iteration of DARTS [Liu et al., 2019]. SGD optimizer with momentum 0.9 and initial learning rate 0.025 is used to optimize the network weights. Adam optimizer with initial learning rate $10^{-4}$ is used to optimize the architecture parameters. In this search space, we can not merge the average pooling and skip-connection due to the reason explained in 3.4.

**Search Results.** The normal and reduced cell found by MergeNAS is shown in Figure 2. In the evaluation phase, a large network stacked by 20 cells, where the $6^{th}$ and $13^{th}$ cells are the reduced cell, is trained from scratch with batch size 96. Noted that the convolutions in the final network are single separable convolutions in our operation set. Similar to DARTS [Liu et al., 2019], we use the cutout augmentation strategy and auxiliary classifier in the evaluation phase. We search for the architecture based on MergeNAS, and optimize the network weights and architecture parameters iteratively based on 1st and 2nd DARTS [Liu et al., 2019]. The comparison with state-of-the-art NAS approaches is given in Table 1. MergeNAS (1st order) is able to find an efficient architecture with 97.27% ultimate accuracy on CIFAR10 in 0.2 GPU-days. And the architecture searched by MergeNAS (2nd order) achieves 97.32% ultimate accuracy in 0.6 GPU-days. Compared to DARTS and SNAS, MergeNAS achieves better performance with fewer parameters and shorter searching time. Additionally, due to the limitation of GPU memory, DARTS [Liu et al., 2019] had to search the architecture of normal and reduced cells on a shallow one-shot model (8 cells), but evaluate the ultimate performance on a deep network (20 cells). However, the performance gap of the depth transfer in one-shot NAS can be large. Our approach can reduce the memory cost in the search phase, making it possible to search on a deep one-shot model (20 cells) directly. To accelerate the search phase, we optimize the architecture parameters based on the first order of DARTS. Other hyper-parameters are set similar to DARTS. Results are presented in Table 3, showing that searching on a deep one-shot network directly is able to obtain a better architecture than transferring from a shallow neural architecture.

### 4.2 Search on NAS-Bench-201

**Training Settings.** We follow the settings in NAS-Bench-201 [Dong and Yang, 2020]. In search phase, we train the one-shot model stacked by 17 cells (including 5 normal cells in each resolution) for 50 epochs and only search the structure of the normal cell. We optimize the network weights by SGD optimizer with momentum 0.9, and the architecture parameters by Adam optimizer whose $\beta$ equals to (0.5, 0.999). Inspired by the recent work [Zela et al., 2020], we increase the weight decay up to $2e - 2$ for both DARTS and MergeNAS. For NAS-Bench-201, the convolutions in the operation set are normal ones, so we can merge average pooling into parametric operation, to further reduce the memory cost.

**Search Results.** NAS-Bench-201 provides the detailed information of all possible architectures belonging to the specific search space, including the accuracy and latency.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>test accuracy (%) on CIFAR10</th>
<th>CIFAR100</th>
<th>ImageNet-16-120</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>trial1</td>
<td>trial2</td>
<td>trial1</td>
</tr>
<tr>
<td>RS</td>
<td>90.77</td>
<td>91.14</td>
<td>91.11</td>
</tr>
<tr>
<td>GNAS</td>
<td>92.36</td>
<td>91.93</td>
<td>91.93</td>
</tr>
<tr>
<td>SETN</td>
<td>92.48</td>
<td>91.58</td>
<td>93.30</td>
</tr>
<tr>
<td>DARTS-V2 (wd=1e - 2)</td>
<td>80.57</td>
<td>93.31</td>
<td>88.21</td>
</tr>
<tr>
<td>DARTS-V2 (wd=2e - 2)</td>
<td>87.50</td>
<td>87.50</td>
<td>85.21</td>
</tr>
<tr>
<td>MergeNAS-V2 (wd=1e - 2)</td>
<td>90.65</td>
<td>90.65</td>
<td>94.36</td>
</tr>
<tr>
<td>MergeNAS-V2 (wd=2e - 2)</td>
<td>94.36</td>
<td>94.36</td>
<td>94.36</td>
</tr>
</tbody>
</table>

Table 2: Comparison with state-of-the-art image classifiers on NAS-Bench-201. Except DARTS, Numbers of other compared methods are directly cited from the raw paper. The optimal is provided by the NAS-Bench-201, which indicates the best architecture in the search space.

<table>
<thead>
<tr>
<th>Number of Cells</th>
<th>Acc</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Evaluation</td>
<td>(%)</td>
<td>(M)</td>
</tr>
<tr>
<td>8 (1st order)</td>
<td>97.27</td>
<td>2.87</td>
</tr>
<tr>
<td>8 (2nd order)</td>
<td>97.33</td>
<td>2.92</td>
</tr>
<tr>
<td>20 (1st order)</td>
<td>97.44</td>
<td>2.96</td>
</tr>
</tbody>
</table>

Table 3: Results on CIFAR10 of the networks with 20 cells whose structure is searched by MergeNAS on the one-shot network stacked by 8/20 cells.
on three datasets: CIFAR10 [Krizhevsky et al., 2009], CIFAR100 [Krizhevsky et al., 2009], and ImageNet-16-120 [Dong and Yang, 2020]. ImageNet-16-120 contains 120 different classes in ImageNet dataset [Russakovsky et al., 2015], and all images are reshaped to 16 × 16. The training settings and split protocol can be found in [Dong and Yang, 2020]. We can directly index for the accuracy of the network searched by MergeNAS instead of training from scratch. We search for the architecture based on DARTS and MergeNAS for three times with different random seed. Because the information of the architectures is indexed from the benchmark, the accuracy is the same if the approach converges to the same architecture. The results are given in Table 2, showing that our method achieves state-of-the-art.

Stability of MergeNAS. The stability of NAS approaches can be evaluated by compare the ultimate performance of the searched architecture in each epochs. A good NAS approach should be able to converge to an efficient architecture, whose ultimate performance should not drop a lot. However it needs a large amount of computation resource and time to obtain the ultimate performance. Fortunately, NAS-Bench-201 benchmark provides the ultimate performance for all the architectures in the specific search space. Therefore, we can evaluate the stability of MergeNAS by plotting the learning curve for accuracy over epochs. Figure 3 compares the stability of DARTS and our approach at different weight decay. MergeNAS converges at around 40 epochs, while DARTS has the trend of decline at 20 epoch. Though the search process of DARTS with large weight decay also converges, the ultimate performance is worse.

5 Conclusion

We have proposed an efficient and stable differentiable NAS approach referred to as MergeNAS. By sharing weights among the convolutions with different kernel size and dilation rate, we can merge all the convolutions associated to the same edge into one. We also propose merge strategy for separable convolutions, average pooling, and skip connect operation. Compared to vanilla DARTS [Liu et al., 2019], MergeNAS is able to reduce the memory cost in search phase, and obtain an efficient architecture with fewer GPU-days. Furthermore, MergeNAS shows strong robustness and cost-effectiveness on two different search spaces. Especially, MergeNAS achieves state-of-the-art on NAS-Bench-201 for both test accuracy and stability.

Acknowledgements

Xiaoxing Wang, Junchi Yan, Xiaokang Yang were supported by National Key Research and Development Program of China (2018AAA0100704, 2016YFB1001003), and NSFC (61972250, U19B2035), STCSM (18DZ1112300).

References


