MergeNAS: Merge Operations into One for Differentiable Architecture Search

Xiaoxing Wang[1*], **Chao Xue**[3], **Junchi Yan**[2,1†], **Xiaokang Yang**[1], **Yonggang Hu**[4] and **Kewei Sun**[3]
[1]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[2]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[3]IBM Research – China
[4]IBM System
figure1_wxx@sjtu.edu.cn, xuechao@cn.ibm.com, yanjunchi@sjtu.edu.cn, xkyang@sjtu.edu.cn, yhu@ca.ibm.com, sunkewei@cn.ibm.com

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 trails 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 [Baker *et al.*, 2017; Zoph and Le, 2017; Zhong *et al.*, 2018; Zoph *et al.*, 2018] frame NAS as a reinforcement learning problem, and consider the generation of an architecture as the agent's action. ENAS [Pham *et al.*, 2018] 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 [Real *et al.*, 2017; Liu *et al.*, 2018b; Real *et al.*, 2019; Miikkulainen *et al.*, 2019; Xie and Yuille, 2017; Elsken *et al.*, 2019] 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 are selected according to the validation accuracy of each architecture. Unlike the first two frameworks that regards the searching process as a black box, DARTS [Liu *et al.*, 2019] 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 [Bi *et al.*, 2019]. 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 [Zela *et al.*, 2020] 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-

*This work was done when the first author was an intern at IBM Research – China.

†Junchi Yan is the corresponding author.

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 $\mathcal{O}$. We also include the architecture parameters to control the saliency of different operations in edges, denoted as α_{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}.

$$o_j(z_i) = \sum_{o \in \mathcal{O}} \alpha_{ij}^o o_{ij}(z_i) \tag{1}$$

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

$$z_j = \sum_{i<j} o_j(z_i) \tag{2}$$

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 $\mathcal{O}^p \subset \mathcal{O}$, 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, $\mathbf{W}_{ij}^o$ is the parameters belonging to the operation o_{ij}, and '$*$' denotes the convolutional operation.

$$o_j^p(z_i) = \sum_{o \in \mathcal{O}^p} \alpha_{ij}^o o_{ij}(z_i) = \sum_{o \in \mathcal{O}^p} \alpha_{ij}^o \left[z_i * \mathbf{W}_{ij}^o \right] \tag{3}$$

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* $\mathbf{W}$ *is equivalent to*[1] *a set of convolution operations whose kernel* $\tilde{\mathbf{W}}$ *is derived from* $\mathbf{W}$ *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 dilation rate r has weight tensor $\mathbf{W}^{(0)} \in \mathbb{R}^{C_o \times C_i \times k_h^{(0)} \times k_w^{(0)}}$, where C_o, C_i are the number of output and input channels, respectively. For simplification, we denote $\mathbf{W}_{i,j,0,0}$ the central value of the j^{th} kernel in the i^{th} filter of the tensor $\mathbf{W}$, and thus $\mathbf{W}_{i,j,-1,0}$ indicates the nearest value above the central value. A tensor $\mathbf{W}^{(1)}$ can be derived as Eq. 4 with kernel size $k_h^{(1)} = rk_h^{(0)}, k_w^{(1)} = rk_w^{(0)}$. We then define the tensor $\mathbf{W}^{(2)}$ with a larger kernel $k_h^{(2)} \times k_w^{(2)}$ by padding $2n$ $(n \geq 0)$ zeros around $\mathbf{W}^{(1)}$.

$$\mathbf{W}^{(1)}_{c,m,p,q} = \begin{cases} \mathbf{W}^{(0)}_{c,m,p/r,q/r} & p, q \ \mathbf{mod}\ r = 0 \\ 0 & otherwise \end{cases} \tag{4}$$

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

$$\mathbf{Y}_{n,c,h,w} = \sum_{p=-u_h^{(2)}}^{u_h^{(2)}} \sum_{q=-u_w^{(2)}}^{u_w^{(2)}} \sum_{m=0}^{C_i-1} \mathbf{X}_{n,m,h+p,w+q} \mathbf{W}^{(2)}_{c,m,p,q} \tag{5}$$

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 $\mathbf{W}^{(1)}$, which omits the zero padding of $\mathbf{W}^{(2)}$. Furthermore, by omitting the zero values inside the weight tensor, we can compute the tensor $\mathbf{Y}$ as Eq. 6, which is the dilation convolution with the weight $\mathbf{W}^{(0)}$.

$$\begin{aligned} \mathbf{Y}_{n,c,h,w} &= \sum_{p=-u_h^{(0)}}^{u_h^{(0)}} \sum_{q=-u_w^{(0)}}^{u_w^{(0)}} \sum_{m=0}^{C_i-1} \mathbf{X}_{n,m,h+rp,w+rq} \mathbf{W}^{(1)}_{c,m,rp,rq} \\ &= \sum_{p=-u_h^{(0)}}^{u_h^{(0)}} \sum_{q=-u_w^{(0)}}^{u_w^{(0)}} \sum_{m=0}^{C_i-1} \mathbf{X}_{n,m,h+rp,w+rq} \mathbf{W}^{(0)}_{c,m,p,q} \end{aligned} \tag{6}$$

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. □

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 $\mathcal{O}$. 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 $\tilde{\mathbf{W}}^o$ is the equivalent weights with large kernel size for convolutions in the search space, and $\tilde{\mathbf{W}}$ is the merged weights of the single convolution.

$$o^p(z) = \sum_{o \in \mathcal{O}^p} \alpha^o \left[z * \mathbf{W}^o \right] = z * \left[\sum_{o \in \mathcal{O}^p} \alpha^o \tilde{\mathbf{W}}^o \right] = z * \tilde{\mathbf{W}} \tag{7}$$

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

$$\tilde{\mathbf{W}} = \sum_{o \in \mathcal{O}^p} \alpha^o \tilde{\mathbf{W}}^o = \left[\sum_{o \in \mathcal{O}^p} \alpha^o \mathbf{M}^o \right] \cdot \mathbf{W} = \tilde{\mathbf{M}} \cdot \mathbf{W} \tag{8}$$

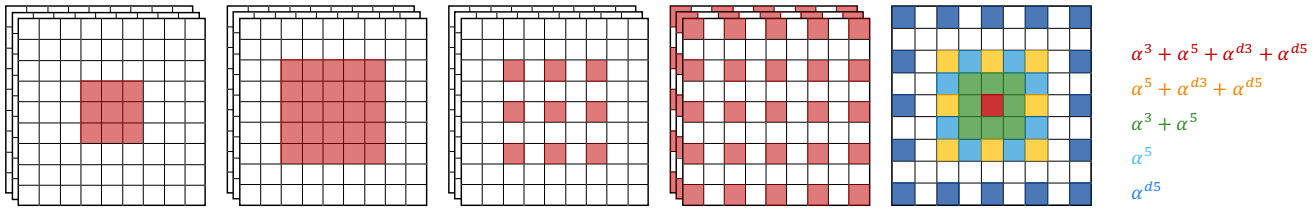
The matrix $\tilde{\mathbf{M}} = \sum \alpha^o \mathbf{M}^o$ denotes the saliency of different receptive field, as shown in Figure 1(e). By sharing the weights among convolutions, MergeNAS can reduce not only the number of parameters but also the amount of computation. Multiple experiments on NAS-Bench-201 [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 $\mathbf{Y}$ in Eq. 5 and Eq. 6 is guaranteed to be the same by padding zeros around $\mathbf{X}$



(a) 3×3 convolution (b) 5×5 convolution (c) 3×3 dilated conv. (d) 5×5 dilated conv. (e) mask matrix $\tilde{M}$

Figure 1: (a)-(d) is the equivalent mask $\mathbf{M}^o$ of the receptive field for different convolutions (red denotes non-zero weights). (e) is the weighted average mask $\tilde{\mathbf{M}}$. Different color indicate various combinations of architecture parameters α. In the micro search space 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 $\mathbf{W}^d \in \mathbb{R}^{C_i \times 1 \times k_h \times k_w}$ and point-wise weight $\mathbf{W}^p \in \mathbb{R}^{C_o \times C_i \times 1 \times 1}$. The output tensor $\mathbf{Y}$ of the normal convolution with the input $\mathbf{X}$ can be computed by Eq. 10, where $u_h = (k_h - 1)/2, u_w = (k_w - 1)/2$.

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

$$\begin{aligned}\mathbf{Y}_{n,c,h,w} &= \sum_{m=0}^{C_i-1} \mathbf{Y}^d_{n,m,h,w} \mathbf{W}^p_{c,m,1,1} \\ &= \sum_{p=-u_h}^{u_h} \sum_{q=-u_w}^{u_w} \sum_{m=0}^{C_i-1} \mathbf{X}_{n,m,h+rp,w+rq} \left[\mathbf{W}^d_{m,1,p,q} \mathbf{W}^p_{c,m,1,1}\right] \\ &= \sum_{p=-u_h}^{u_h} \sum_{q=-u_w}^{u_w} \sum_{m=0}^{C_i-1} \mathbf{X}_{n,m,h+rp,w+rq} \mathbf{W}^s_{c,m,p,q} \end{aligned} \tag{10}$$

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

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

$$\mathbf{W}^{s,o} = \mathbf{W}^{d,o} * \mathbf{W}^{p,o} = [\mathbf{W}^{d,o}]^\top \cdot \mathbf{W}^{p,o} \tag{11}$$

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 $\mathbf{W}^d$.

$$\begin{aligned}\tilde{\mathbf{W}} &= \sum_{o \in \mathcal{O}^p} \alpha^o \mathbf{M}^o \cdot [\mathbf{W}^d]^\top \cdot \mathbf{W}^{p,o} \\ &= [\mathbf{W}^d]^\top \cdot \sum_{o \in \mathcal{O}^p} \alpha^o \mathbf{M}^o \cdot \mathbf{W}^{p,o} \end{aligned} \tag{12}$$

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 $\mathcal{O}^l = O \backslash O^p$;
4) The input of node j, z_i;
Parameter:
1) The sharing weights for edge e_{ij} of one-shot model ($\mathbf{W}_{ij}$ for normal, and $\mathbf{W}^d_{ij}, \mathbf{W}^p_{ij}$ for separable convolution);
2) The architecture parameters for edge e_{ij}, $\alpha^o_{ij}, o \in \mathcal{O}$;
3) Mask matrix for convolutional operations $\mathbf{M}^o, o \in \mathcal{O}^p$;
Output:
The output of node j, z_j;
Procedure:

```
Compute the saliency matrix M̃_ij = Σ_{o∈O^p} α^o_ij M^o;
if normal convolution in O^p then
    W̃_ij = M̃_ij · W_ij
    Compute the single convolution with weights W̃_ij:
    o^p_j(z_i) = z_i * W̃_ij
else if separable convolution in O^p then
    W^{d'}_ij = M̃_ij · W^d_ij;
    Compute single separable convolution with depth-wise
    weights W^{d'}_ij and point-wise weights W^p_ij:
    o^p_j(z_i) = z_i * W^{d'}_ij * W^p_ij
end if
o_j(z_i) = o^p_j(z_i) + Σ_{o∈O^l} α^o_ij o_ij(z_i)
return the output of edge e_ij, o_j(z_i);
```

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

$$\begin{aligned}\tilde{\mathbf{W}} &= [\mathbf{W}^d]^\top \cdot \sum_{o \in \mathcal{O}^p} \alpha^o \mathbf{M}^o \cdot \mathbf{W}^p \\ &= \left[\tilde{\mathbf{M}} \cdot \mathbf{W}^d\right]^\top \cdot \mathbf{W}^p = \mathbf{W}^{d'} * \mathbf{W}^p \end{aligned} \tag{13}$$

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

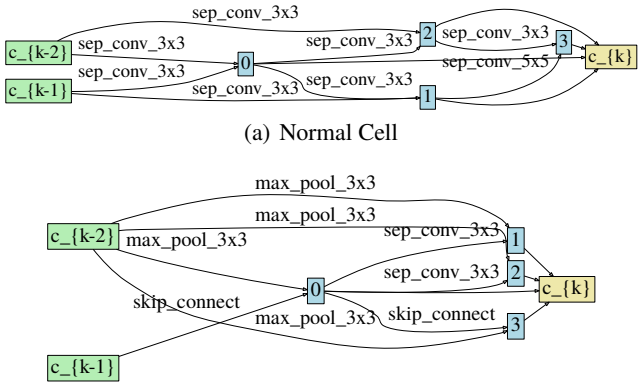
Architecture	Test Error (%)	Params (M)	#ops	Search Cost (GPU-days)	Search Method
DenseNet-BC [Huang *et al.*, 2017]	3.46	25.6	-	-	manual
NASNet-A + cutout [Zoph *et al.*, 2018]	2.65	3.3	13	1800	RL
AmoebaNet-B + cutout [Real *et al.*, 2019]	2.55±0.05	2.8	19	3150	evolution
Hierarchical Evo [Liu *et al.*, 2018b]	3.75±0.12	15.7	6	300	evolution
PNAS [Liu *et al.*, 2018a]	3.41±0.09	3.2	8	225	SMBO
ENAS + cutout [Pham *et al.*, 2018]	2.89	4.6	6	0.5	RL
DARTS (1st order) + cutout [Liu *et al.*, 2019]	3.00±0.14	3.3	7	0.4	gradient-based
DARTS (2nd order) + cutout [Liu *et al.*, 2019]	2.76±0.09	3.4	7	1.0	gradient-based
SNAS (moderate) + cutout [Xie *et al.*, 2019]	2.85±0.02	2.8	7	1.5	gradient-based
ProxylessNAS + cutout [Cai *et al.*, 2019]	2.08	5.7	7	4.0	gradient-based
P-DARTS + cutout [Chen *et al.*, 2019]	2.50	3.4	7	0.3	gradient-based
PC-DARTS (1st order) + cutout [Xu *et al.*, 2020]	2.57±0.07	3.6	7	0.1	gradient-based
BayseNAS + cutout [Zhou *et al.*, 2019]	2.81±0.04	3.4	7	0.2	gradient-based
MergeNAS (1st order) + cutout	2.73±0.02	2.9	7	0.2	gradient-based
MergeNAS (2nd order) + cutout	2.68±0.01	2.9	7	0.6	gradient-based

Table 1: Comparison with state-of-the-art image classifiers on CIFAR10 (lower error rate is better). Similar to DARTS, the search cost for MergeNAS does not include the selection cost or the final evaluation cost. Numbers of compared methods are excerpted from the raw paper. The search cost only includes the time of the searching process on NVIDIA 1080 Ti.

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.



(a) Normal Cell

(b) Reduced Cell

Figure 2: The structure of normal and reduced cell searched by MergeNAS (2nd order) on CIFAR10 dataset under the micro search space of ENAS and DARTS. The deep architecture stacked by 20 cells based on the above cell structures achieves 97.33% accuracy with 2.92M parameters.

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 $\mathcal{O}$ 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 3×3 and 5×5 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, 3×3 separable convolution, 5×5 separable convolution, 3×3 dilated separable convolution, and 5×5 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 $\mathcal{O}$ contains 5 basic operations: none, skip connect, average pooling, 1×1 convolution, and 3×3 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

Architecture	test accuracy (%) on CIFAR10				CIFAR100		ImageNet-16-120	
	trial_1	trail_2	trail_3	average	valid	test	valid	test
RSPS	90.77	91.91	93.11	91.93	67.45	67.63	39.58	39.80
GDAS	92.28	91.93	91.93	92.05	66.81	67.25	39.42	38.92
SETN	92.48	91.58	93.30	92.45	69.02	69.03	42.19	42.29
ENAS	39.13	39.13	39.13	39.13	15.62	15.62	15.87	15.87
DARTS-V2 (wd=$1e-2$)	80.57	93.31	88.21	87.36	59.90	59.80	36.06	35.58
DARTS-V2 (wd=$2e-2$)	87.50	87.50	85.21	86.74	57.67	57.38	31.09	30.78
MergeNAS-V2 (wd=$1e-2$)	90.65	90.65	94.36	91.89	69.28	69.44	42.38	41.87
MergeNAS-V2 (wd=$2e-2$)	94.36	94.36	94.36	94.36	73.49	73.51	46.37	46.34
RS	93.19±0.72				69.73	69.89	43.15	43.09
REA	92.86±0.86				69.07	69.25	42.32	42.34
REINFORCE	93.31±0.58				70.14	70.26	42.62	43.03
BOHB	93.14±0.74				69.58	69.79	43.04	42.96
ResNet	93.97				70.42	70.86	44.53	43.63
Optimal	94.37				73.49	73.51	46.77	47.31

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. We evaluate DARTS and MergeNAS for three times under different random seed and provide the test accuracy of each trial on CIFAR10, CIFAR100, and ImageNet-16-120. MergeNAS is able to converge to the same architecture under different random seed, so the accuracy of the three trials are the same (We directly index for the detailed information of each architecture).

Number of Cells		Acc	Params
Search	Evaluation	(%)	(M)
8 (1st order)	20	97.27	2.87
8 (2nd order)	20	97.33	2.92
20 (1st order)	20	97.44	2.96

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.

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-connect 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 is 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 give 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 β 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

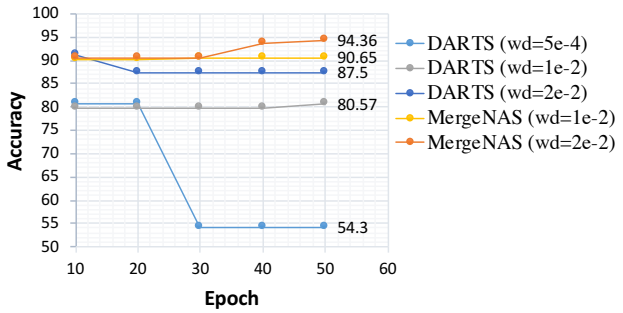


Figure 3: Stability of DARTS and MergeNAS. The accuracy is the ultimate test accuracy of the architecture found at various epochs in the search phase.

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

[Baker *et al.*, 2017] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2017.

[Bender *et al.*, 2018] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 549–558, 2018.

[Bi *et al.*, 2019] Kaifeng Bi, Changping Hu, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. Stabilizing darts with amended gradient estimation on architectural parameters. *arXiv preprint arXiv:1910.11831*, 2019.

[Cai *et al.*, 2019] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

[Chen *et al.*, 2019] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019.

[Chu *et al.*, 2019] 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.

[Dong and Yang, 2020] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

[Elsken *et al.*, 2019] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations*, 2019.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[Liu *et al.*, 2018a] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018.

[Liu *et al.*, 2018b] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

[Liu *et al.*, 2019] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[Miikkulainen *et al.*, 2019] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier, 2019.

[Nayman *et al.*, 2019] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik-Manor. XNAS: neural architecture search with expert advice. In *NeurIPS*, pages 1975–1985, 2019.

[Pham *et al.*, 2018] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, 2018.

[Real *et al.*, 2017] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911, 2017.

[Real *et al.*, 2019] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, pages 4780–4789, 2019.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[Sandler *et al.*, 2018] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[Stamoulis *et al.*, 2019] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. *arXiv preprint arXiv:1904.02877*, 2019.

[Xie and Yuille, 2017] Lingxi Xie and Alan L. Yuille. Genetic CNN. In *IEEE International Conference on Computer Vision, ICCV*, pages 1388–1397, 2017.

[Xie *et al.*, 2019] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.

[Xu *et al.*, 2020] 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 *International Conference on Learning Representations*, 2020.

[Zela *et al.*, 2020] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020.

[Zhong *et al.*, 2018] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2423–2432, 2018.

[Zhou *et al.*, 2019] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *International Conference on Machine Learning*, pages 7603–7613, 2019.

[Zoph and Le, 2017] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

[Zoph *et al.*, 2018] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.