Subnet-Aware Dynamic Supernet Training for Neural Architecture Search - Supplementary Material

In the supplementary material, we provide detailed implementation descriptions (Sec. A), and additional results, including comparisons to the state-of-the-art methods, and experiments conducted on additional NAS spaces (Sec. B). We also present discussions on the ablation studies and the design choices of our method (Sec. C). Furthermore, we explain extensions to various N-shot NAS methods with detailed algorithmic explanations and pseudocode (Sec. D). Lastly, we visualize the structures of the subnets searched with our method (Sec. E).

A. Implementation details

MobileNet. We follow the standard approach [3, 7] for supernet training, evolutionary search, and retraining the retrieved subnet. For training supernet on ImageNet [4] in the MobileNet space [1], we use the SGD optimizer with a momentum coefficient of 0.9 and a weight decay of 4e-5. The batch size and initial LR are set as 512 and 0.045, respectively. We apply our dynamic supernet training to three baselines, including SPOS [7], FairNAS [3], and FSNAS [18], where they are trained for 150, 75, and 100 epochs, respectively. Note that FSNAS utilizes multiple sub-supernets, and each sub-supernet is trained for 100 epochs. We set the maximum and minimum decay ratio of CaLR as $\gamma_{\text{max}} = 1/\gamma_{\text{min}} = \gamma'$, and perform a grid search to determine $\gamma' \in \{2, 3, 4\}$, and we determine $\gamma' = 3$ for all baselines. For layer selection in MS, we choose the first one of the searchable layers. We then perform the evolutionary search for 20 epochs, with a population number of 50 to find the best subnet. For retraining the retrieved subnets, we use the same setting of [10, 12]. We use 4 A5000 GPUs for training the supernet and 8 A5000 GPUs for retraining the searched subnets.

NAS-Bench-201. We train the supernet with the SGD optimizer with a momentum coefficient of 0.9 and a weight decay of 5e-4. The batch size and initial LR are set as 64 and 0.025, respectively. Our framework is applied on SPOS [7], FairNAS [3], and FSNAS [18], where they are trained for 250, 200, and 300 epochs, respectively. Similar to the MobileNet space, we perform the grid search to determine the maximum and minimum decay ratio of CaLR in the same candidate, and we set $\gamma' = 4$. We use a randomly selected edge for subnet clustering in MS. We use a single RTX 2080Ti for training the supernet.

Table A. Quantitative comparison of the search performance on CIFAR-10 [9] in the NAS-Bench-201 space [6].

Methods	Top-1 Acc.
DARTS [13]	86.23 ± 4.93
GDAS [5]	93.26 ± 0.32
NSAS [17]	92.23 ± 0.10
Cream [15]	92.83 ± 0.67
SUMNAS [8]	93.09 ± 0.12
PA&DA [14]	93.33 ± 0.22
SPOS [7]	93.12 ± 0.03
SPOS + Ours	**93.50** ± 0.33
FairNAS [3]	92.13 ± 0.18
FairNAS + Ours	**93.52** ± 0.50
FSNAS [18]	93.43 ± 0.24
FSNAS + Ours	**93.63** ± 0.21

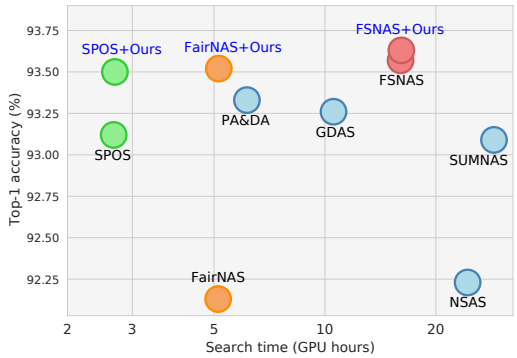


Figure A. Visual comparison of the top-1 accuracy and search time for the subnets retrieved from the NAS-Bench-201 space [6] for CIFAR-10 [9].

Table B. Quantitative results of the top-1 accuracy on CIFAR-10 in the DARTS space.

Methods	Top-1 Acc. (%)
SPOS	96.1
SPOS + Ours	**96.3**

B. More experiments

In this section, we provide a comparison to the state-of-the-art NAS methods, and results on additional NAS spaces, including DARTS [13] and Autoformer spaces [2].

Table C. Quantitative comparisons on ImageNet in the Autoformer space. We report the top-1 validation accuracy, together with the number of parameters limit and the number of parameters.

Methods	# Params Limit	# Params	Top-1 Acc. (%)
Autoformer	≤6M	5.97M	74.5
Autoformer + Ours	≤6M	5.98M	**74.8**
Autoformer	≤7.5M	7.49M	76.0
Autoformer + Ours	≤7.5M	7.49M	**76.3**
Autoformer	≤9M	8.91M	76.5
Autoformer + Ours	≤9M	8.96M	**76.7**

Table D. Quantitative comparisons of CB and C^3 on CIFAR-10, CIFAR-100 [9], and ImageNet16-120 [4] in the NAS-Bench-201 space [6].

Method	CIFAR-10		CIFAR-100		ImageNet16-120	
	CB	C^3	CB	C^3	CB	C^3
SPOS	0.76	-0.19	0.84	-0.25	0.81	-0.22
+CaLR	0.63	-0.09	0.7	-0.11	0.68	-0.1
FairNAS	0.72	-0.13	0.69	-0.16	0.73	-0.15
+CaLR	0.62	-0.06	0.6	-0.1	0.65	-0.08
FSNAS	0.79	-0.21	0.82	-0.24	0.8	-0.22
+CaLR	0.63	-0.1	0.66	-0.11	0.65	-0.05

B.1. Comparison to the state of the art

We present in Table A the quantitative comparison of our method and state-of-the-art NAS methods for NAS [3, 5, 7, 8, 13–15, 17, 18] on CIFAR-10 [9] in the NAS-Bench-201 space [6]. We can see that our method outperforms state-of-the-art methods in terms of top-1 accuracy. To further validate the efficiency of our method, we compare in Fig. A the search performance and retrieval time. We can observe that ours provides better compromise between the search performance and search time compared to other methods. Moreover, our method can improve the performance of various NAS approaches, without a significant increase in the search time. For example, applying a dynamic supernet training to FairNAS [3] boosts the top-1 accuracy of the searched subnets from 92.13% to 93.52%, with a negligible additional time, demonstrating the efficiency of our method.

B.2. Results on additional spaces

To show that our method is generally applicable to various NAS spaces, we provide other benchmark results, including DARTS [13] and Autoformer spaces [2]. The DARTS space is a cell-based search space, searching for normal and reduction cells, each consisting of 14 edges and 7 candidate operations. We train a supernet with SPOS [7] and SPOS with our dynamic supernet training on CIFAR-10. We follow the setting of [11] for training a supernet and searching subnets. Specifically, the supernet is trained for 50 epochs, and the most promising subnet is selected by evaluating 1,000 subnets randomly chosen. We retrain the searched subnet with 200 epochs.

The Autoformer space [2] is designed for building ViT architectures, with search parameters including Q-K-V dimension, embedding dimension, number of heads, MLP ratio, and network depth. We follow the experimental setting of [2] for supernet training and subnet searching. We use the Supernet-tiny of the Autoformer space as our supernet, and train the supernet on ImageNet using the Autoformer algorithm [2] with and without our approach. We then search the subnets using an evolutionary algorithm with various constraints on the number of parameters. Note that the Autoformer algorithm does not require a retraining step.

We show in Tables B and C quantitative comparisons of baseline NAS methods with and without our approach. We can see that our method can generally enhance these baselines in various search spaces. Note that these improvements come from marginal overheads, demonstrating that our method can be applied efficiently to various scenarios.

C. More Discussions

In this section, we provide additional discussions on the ablation studies and the design choices of our method.

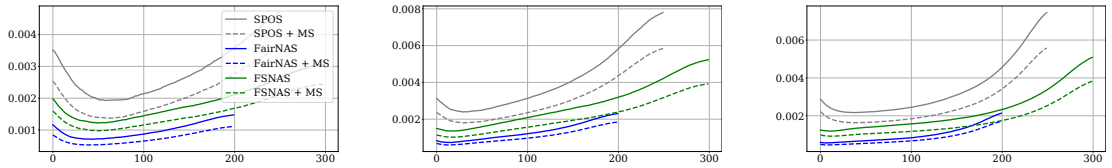
C.1. Ablations on CaLR and MS

To further validate the effectiveness of our method in various NAS methods and datasets beyond Table 4 and Fig. 6 of the main paper, we show in Table D and Fig. B additional ablation studies of CaLR and MS, conducted on various methods (SPOS [7], FairNAS [3], and FSNAS [18]) and datasets (CIFAR-10, CIFAR-100 [9], and ImageNet16-120 [4]) in the NAS-Bench-201 space [6]. We can see that applying CaLR and MS alleviates the unfairness and noisy momentum problems, respectively, across different methods and datasets, demonstrating the generalizability of our method.

C.2. Variants of the design of decay ratio in CaLR

To compute the decay ratio $\gamma(\alpha)$ of a subnet α, we first feed the complexity score $\mathcal{C}(\alpha)$ of the subnet α into a logarithmic function, followed by applying an affine transformation to $\log(\mathcal{C}(\alpha))$ (see Eq. (4)). We consider the different design options for $\gamma(\alpha)$. To this end, we first reformulate Eq. (4) as follows:

$$\gamma(\alpha) = \omega g(\mathcal{C}(\alpha)) + \tau, \tag{i}$$



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet16-120

Figure B. Plots of standard deviations for gradients of the supernet on CIFAR-10, CIFAR-100 [9], and ImageNet16-120 [4] in the NAS-Bench-201 space [6].

Table E. Quantitative results of the ranking consistency on CIFAR-10 [9] in the NAS-Bench-201 space [6] with different choices of $g(x)$.

Baselines	CaLR	MS	$g(x)$	Kendall's Tau
SPOS	-	✓	-	0.772 ± 0.007
	✓	✓	e^x	0.774 ± 0.018
	✓	✓	x	0.783 ± 0.013
	✓	✓	$\log(x)$	$\mathbf{0.814} \pm 0.007$
FairNAS	-	✓	-	0.784 ± 0.007
	✓	✓	e^x	0.794 ± 0.016
	✓	✓	x	0.813 ± 0.011
	✓	✓	$\log(x)$	$\mathbf{0.828} \pm 0.020$
FSNAS	-	✓	-	0.750 ± 0.024
	✓	✓	e^x	0.755 ± 0.015
	✓	✓	x	0.763 ± 0.013
	✓	✓	$\log(x)$	$\mathbf{0.767} \pm 0.010$

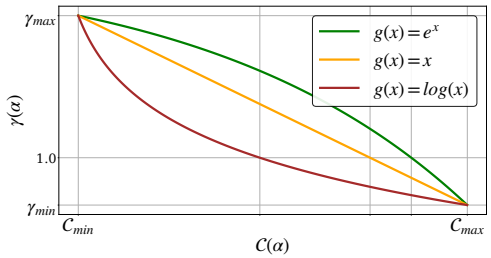


Figure C. Visualization of the decay ratio $\gamma(\alpha)$ based on the complexity score $\mathcal{C}(\alpha)$ with different choices of $g(x)$.

where ω and τ are coefficients for the affine transformation:

$$\omega = -\frac{\gamma_{\max} - \gamma_{\min}}{g(\mathcal{C}_{\max}) - g(\mathcal{C}_{\min})}, \tag{ii}$$

$$\tau = \gamma_{\min} - \omega g(\mathcal{C}_{\max}). \tag{iii}$$

We show in Table E the results of the ranking consistency on CIFAR-10 [9] in the NAS-Bench-201 space [6] with different functions for $g(x)$, including exponential e^x, linear x, and logarithmic $\log(x)$. From the results, we observe that $g(x) = \log(x)$ outperforms other options. Note that $g(x)$ determines the degree of the differentiation of the decay ratio $\gamma(\alpha)$ for the subnets with different complexities. The logarithmic function allows the subnet with a complexity score close to the medium (around the center point between $\mathcal{C}_{\min}$ and $\mathcal{C}_{\max}$) to have a linear decaying LR, *i.e.*, the decay ratio $\gamma(\alpha)$ of 1 (Fig. C). This ensures LR of the high-complexity subnets to be slowly decayed (*i.e.*, $\gamma_{\min} < \gamma(\alpha) < 1$), while decaying it faster for low-complexity ones (*i.e.*, $1 < \gamma(\alpha) < \gamma_{\max}$). This suggests that designing $g(x)$ in a logarithmic function is effective in terms of differentiating the LR decay for various subnets. Other options, such as x, and e^x, fail to differentiate LR decay for varying complexities of subnets (Fig. C), since they do not satisfy the aforementioned criteria, resulting in an inferior performance. For these reasons, we adopt $g(x) = \log(x)$ for computing the decay ratio $\gamma(\alpha)$.

Table F. Quantitative results of the ranking consistency on CIFAR-10 [9] in the NAS-Bench-201 space [6] with different choices of metrics for complexity. $\mathcal{C}(\alpha)$ denotes the complexity score of a subnet α.

Baselines	CaLR	MS	$\mathcal{C}(\alpha)$	Kendall's Tau
SPOS	-	-	-	0.751 ± 0.008
	✓	✓	Params	$\mathbf{0.814} \pm 0.007$
	✓	✓	FLOPs	0.783 ± 0.015
FairNAS	-	-	-	0.766 ± 0.015
	✓	✓	Params	$\mathbf{0.828} \pm 0.020$
	✓	✓	FLOPs	0.812 ± 0.017
FSNAS	-	-	-	0.729 ± 0.019
	✓	✓	Params	$\mathbf{0.767} \pm 0.010$
	✓	✓	FLOPs	0.756 ± 0.010

C.3. Metrics for complexity

We show in Table F the results of the ranking consistency on CIFAR-10 [9] in the NAS-Bench-201 space [6] with different metrics for complexity. We consider the number of parameters (Params), and FLOPs as metrics for complexity. We observe that using the number of parameters achieves slightly better performance than FLOPs. Moreover, since the number of parameters is a more straightforward to compute than FLOPs, which requires a forward pass, we adopt the number of parameters as a metric for complexity in our framework.

Table G. Quantitative comparison of Kendall's Tau on CIFAR-10 in the NAS-Bench-201 space using different subnet sampling strategies and LR schedulers for supernet training. The subnet sampling strategies include uniform sampling and sampling proportional to the number of parameters (#Params). The LR schedulers include the Cosine Annealing Scheduler (CS) and Complexity-aware LR scheduler (CaLR). The momentum separation technique (MS) is used as a default for all methods. We report the average and standard deviations for 3 runs.

Baselines	Subnet Sampling	Scheduler (Supernet Training)	Kendall's Tau
SPOS [7]	Uniform	CS	0.772 ± 0.007
SPOS [7]	#Params	CS	0.784 ± 0.012
SPOS [7]	Uniform	CaLR	**0.814** ± 0.007
FairNAS [3]	Uniform	CS	0.784 ± 0.007
FairNAS [3]	#Params	CS	0.811 ± 0.013
FairNAS [3]	Uniform	CaLR	**0.828** ± 0.020
FSNAS [18]	Uniform	CS	0.750 ± 0.024
FSNAS [18]	#Params	CS	0.761 ± 0.011
FSNAS [18]	Uniform	CaLR	**0.767** ± 0.010

C.4. Using non-uniform sampling method in supernet training

CaLR is designed to address the unfairness problem in supernet training by adjusting the LR based on the complexity of each subnet. One possible alternative to address the unfairness problem is to adjust the sampling probabilities of subnets based on their complexities. To compare these two approaches, we show in Table G the results of using non-uniform sampling strategy, which samples the subnets with probabilities proportional to their complexities, instead of using CaLR in supernet training. We can see that adjusting the sampling probabilities improves the baselines, but shows inferior results compared to CaLR. Adjusting the sampling probabilities can cause a part of subnets to be sampled frequently while others are neglected, suggesting that large numbers of subnets would not be trained. In contrast, our CaLR maintains uniform sampling probabilities for all subnets but adjusts the LR based on the complexity of each subnet. This approach helps to balance the training across different subnets more effectively, ensuring that a wider range of subnets receive sufficient training.

C.5. Using CaLR in the retraining stage

Once we train a supernet with CaLR, and search the subnets from the trained supernet, we retrain the searched subnets with a standard scheduler (*i.e.*, cosine scheduler). To investigate the impact of using CaLR during retraining phase, we compare in Table H quantitative results between using CaLR and a standard scheduler in the retraining phase. We can see that using CaLR during a retraining stage does not improve performance compared to using a regular scheduler. The primary reason is that retraining aims to optimize a single subnet, and the cosine annealing scheduler is effective for this purpose, as it gradually lowers the LR, allowing the model to converge smoothly to a local minimum. In contrast, CaLR adjusts LRs based on the complexities of subnets, *e.g.*, keeping high LR for high-complexity subnets, which is effective for supernet training to address the unfairness problem, but not optimal for retraining a single subnet.

Table H. Quantitative results of top-1 accuracies of the searched subnets on CIFAR-10 in the NAS-Bench-201 space using different schedulers for the retraining stage. The schedulers include the Cosine Annealing Scheduler (CS) and Complexity-aware LR scheduler (CaLR). For supernet training, we use consistent settings, applying CaLR and MS to baselines. We report the average and standard deviations for 3 runs.

Baselines	Scheduler (Retraining)	Top-1 Acc.
SPOS [7]	CS	**93.50** ± 0.33
SPOS [7]	CaLR	90.51 ± 0.48
FairNAS [3]	CS	**93.52** ± 0.50
FairNAS [3]	CaLR	90.75 ± 0.27
FSNAS [18]	CS	**93.63** ± 0.21
FSNAS [18]	CaLR	90.24± 0.47

To explain in more detail, our CaLR method is specifically designed to address the challenges of supernet training, where the main difficulty stems from the vast number of subnets within the search space (*e.g.*, $\sim 7^{21}$ for the MobileNet space). In this scenario, fully training each subnet to convergence within a limited number of iterations is impractical. Instead of extending the training iterations across all subnets, CaLR adjusts the LR dynamically based on the complexity of each subnet. This approach ensures that higher-complexity subnets, which have more parameters to optimize, receive relatively more training amounts compared to lower-complexity subnets. By doing so, CaLR balances the training process by giving more attention to subnets that need it due to their inherent complexity, alleviating the unfairness problem in supernet training.

In contrast, the retraining phase operates under different conditions where each subnet can be trained to full convergence. Applying CaLR during retraining may lead to suboptimal performance: maintaining a high LR for high-complexity subnets could result in overshooting, which hampers convergence, while a rapidly decaying LR for low-complexity subnets might require more iterations to achieve convergence. Hence, while CaLR is effective for supernet training, it may not be suitable for the retraining phase where a gradual LR decay, like the cosine annealing scheduler, is more appropriate.

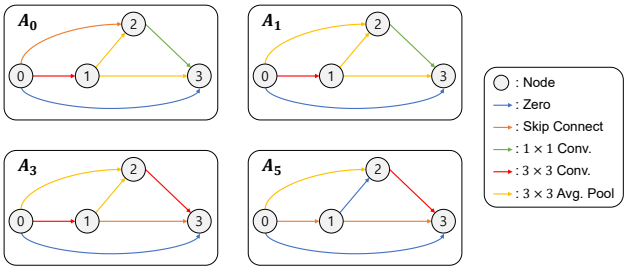


Figure D. Visualization of the cell architectures of the generated subnets.

Table I. Quantitative results of mean gradient cosine similarity (MGCS) with respect to A_0 on CIFAR-10 in NAS-Bench-201.

Subnet	MGSC
A_1	**0.6**
A_3	0.27
A_5	0.09

C.6. Relationship between structural similarity and gradient consistency

We have shown that our MS, which clusters subnets based on their structural similarities, provides better results in terms of gradient consistencies, compared to the baseline NAS methods and a random clustering approach, in Fig. 2(c) and Fig. 6 of the main paper. This demonstrates that subnets with similar structures tend to make more consistent gradients.

To further validate the statement, we perform a quantitative evaluation by computing cosine similarities between gradients, while varying network structures in terms of structural similarities in NAS-Bench-201. Specifically, we select the base subnet A_0. We then randomly change the operations of n edges. By setting n as 1, 3, and 5, the generated subnets are shown in Fig. D. Note that A_1 is the most similar network for A_0 in terms of the structural similarity, while A_5 is the most distinct one.

The supernet, consisting of subnets A_0, A_1, A_3, and A_5, is trained using the accumulated gradients from all subnets at each iteration. During the last epoch of training, we compute the cosine similarities between gradients of the first layer of the subnets for every iteration. Specifically, we compute the similarities between the base network A_0 and A_n, $n = 1, 3, 5$, *i.e.*, $cos(g(A_0), g(A_n))$, where $cos()$ and $g(A_n)$ represent the cosine similarities and the gradients of the first layer of subnet A_n, respectively. We then report the mean gradient cosine similarities (MGCS), by averaging $cos(g(A_0), g(A_n))$ for all iterations. We show in Table I the results of MGSC. We can observe that subnets with higher structural similarities to A_0 show higher MGCS, suggesting that subnets with similar structures tend to generate similar gradients.

Table J. Quantitative results of the ranking consistency on CIFAR-10 [9] in the NAS-Bench-201 space [6]. We compare our framework with different momentum settings.

Baselines	MS	β	Kendall's Tau
SPOS	-	0	0.615 ± 0.046
	-	0.9	0.751 ± 0.008
	✓	0.9	**0.772** ± 0.007
FairNAS	-	0	0.743 ± 0.022
	-	0.9	0.766 ± 0.015
	✓	0.9	**0.784** ± 0.007
FSNAS	-	0	0.616 ± 0.016
	-	0.9	0.729 ± 0.019
	✓	0.9	**0.750** ± 0.024

Table K. Quantitative results of the ranking consistency on CIFAR-10 [9] in the NAS-Bench-201 space [6] with different γ'.

Baseline	CaLR	MS	γ'	Kendall's Tau
SPOS	-	-	-	0.751 ± 0.008
	✓	✓	2	0.787 ± 0.001
	✓	✓	3	0.797 ± 0.010
	✓	✓	4	**0.814** ± 0.007

C.7. Impact of excluding momentum in supernet training

We have shown that sharing a single optimizer for all subnets makes momentum updates noisy. The natural question is that what the impact of excluding momentum in supernet training (*i.e.*, $\beta = 0$ in Eq. (8)) would be. To this end, we compare the performance of the baselines under three different conditions: without momentum ($\beta = 0$), with momentum ($\beta = 0.9$), and with MS in Table J. We observe that the baseline without momentum achieves inferior performance compared to the baseline with momentum. This indicates that despite the noisy nature of momentum updates in supernets, momentum still plays a crucial role in stabilizing the optimization process. Utilizing MS with momentum, however, can achieve better performance, since it can reduce the noise in momentum updates by clustering the subnets with similar structural characteristics. We also observe that the performance degradation of FairNAS [3] without momentum is smaller than that of SPOS [7] and FSNAS [18]. The reason for this is the design of FairNAS, which involves the use of multiple subnets in each iteration. The process of averaging the gradients from these subnets effectively mimics the smoothing effect of momentum, thereby mitigating the impact of its absence and resulting in a smaller performance drop for FairNAS without momentum.

Table L. Quantitative results of the search performance on ImageNet [4] in the MobileNet space [1] with different γ'.

Baseline	CaLR	MS	γ'	Top-1 (%)
SPOS-S	-	-	-	74.3
	✓	✓	2	74.4
	✓	✓	3	**74.6**
	✓	✓	4	74.5

C.8. Hyperparameter search

We show in Tables K and L the results of the hyperparameter search on NAS-Bench-201 [6] and MobileNet [1] spaces[1], respectively. Our framework has two hyperparameters, including the maximum and minimum decay ratio of CaLR, γ_{max} and γ_{min}, respectively, and we set $\gamma_{\text{max}} = 1/\gamma_{\text{min}} = \gamma'$ for simplicity. We perform the grid search to determine the maximum and minimum decay ratio of CaLR; $\gamma' \in \{2, 3, 4\}$. From the results, we observe that our framework achieves the best performance when $\gamma' = 4$ and 3 for NAS-Bench-201 and MobileNet spaces, respectively. This variation in optimal hyperparameter settings can be attributed to the differences in the distribution of the number of parameters among subnets across these search spaces. Furthermore, our framework consistently outperforms the baselines across various hyperparameter configurations, demonstrating its robustness and adaptability to different hyperparameter choices.

C.9. Layer selection in MS

We show in Table M an ablation study on layer selection for MS in the MobileNet space. We compare two cases for selecting a layer in MS (the first layer and the last layer). We observe that choosing the first layer achieves superior performance compared to selecting the last layer. We attribute this to the distinct characteristics of the layers in the subnets. As suggested in [16], the feature distributions of each layer highly affect the gradient flow in the subnet. The features in the deeper layers are likely to be more diverse across the subnets, due to the cumulative effect of operation choices made in all preceding layers. This diversity in feature distributions leads to more inconsistent gradients. Therefore, selecting the deep layer for MS may not significantly contribute to the stabilization of momentum updates. In contrast, sharing the same operation in the shallower layer results in more consistent feature distributions across the subnets, due to the property of the shallow layers that learn more general features such as edges and textures. Selecting the first layer for MS thus results in more consistent gradients within the cluster than selecting the deeper layers, which can stabilize the momentum updates.

[1]We retrain the searched subnet in MobileNet space for 240 epochs when searching hyperparameters, for efficiency.

Table M. Quantitative results of the search performance on ImageNet [4] in the MobileNet space [1]. We compare the layer selection strategy in MS. We report the top-1 validation accuracy of the searched subnets.

Baseline	CaLR	MS	Layer selection	Top-1 (%)
SPOS-S	-	-	-	74.6
	✓	✓	First layer	**74.8**
	✓	✓	Last layer	74.7

D. Extensions to various *N*-shot NAS methods

We provide overall processes of applying our framework on various NAS methods, including SPOS [7], FairNAS [3], and FSNAS [18] in Algorithms 1, 2, and 3, respectively. In the following, we detail the extensions to various NAS methods other than SPOS.

FairNAS. For FairNAS, we sample multiple subnets at each iteration, and compute the gradients of each subnet. We then update the parameters through a weighted sum of the gradients, where the weights assigned to each gradient are determined based on the corresponding subnet using Eq. (4) of the main paper. This approach ensures that the influence of each subnet's gradient on the overall update is proportional to their complexities. Additionally, to facilitate the update of momentum with the accumulated gradients, we sample the subnets from the same cluster at each iteration.

FSNAS. FSNAS utilizes multiple sub-supernets partitioned from a supernet. We apply SPOS [7] with our dynamic supernet training framework to optimize each sub-supernet. Unlike SPOS and FairNAS, FSNAS retrains the retrieved subnet of each sub-supernet and selects the best subnet among them.

E. Subnets

For reproducibility, we provide in Figs. E, F, and G the searched subnets on ImageNet [4] in the MobileNet space [1] using SPOS [7] + Ours, FairNAS [3] + Ours, and FSNAS [18] + Ours as search algorithms, respectively. Note that MBx $k \times k$ represents a MobileNet block with an expansion ratio of x and a kernel size of $k \times k$. The numbers over the blocks denote the number of output channels.

Algorithm 1 Dynamic supernet training on SPOS [7]

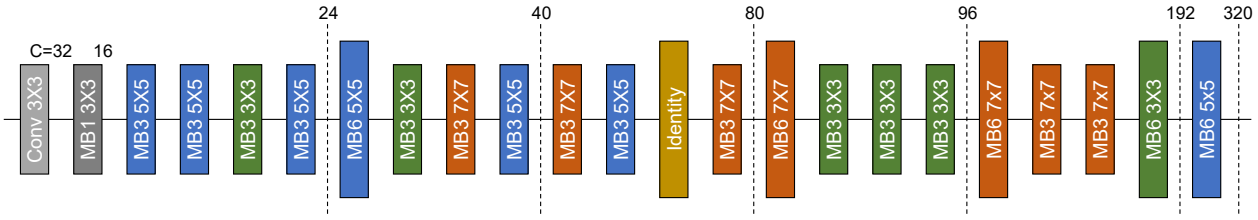
Input: Supernet $\mathcal{N}$, weights of supernet $\mathcal{W}$, momentum of supernet μ, training set $\mathcal{D}_{train}$, total number of iterations T.
Initialize each cluster's momentum: $\mu^0 = 0$.
for $t = 1$ to T **do**
  Sample a mini-batch from training set $\mathcal{D}_{train}$.
  Randomly sample a subnet α from a supernet $\mathcal{N}$.
  Compute LR η^t of current iteration t, considering a complexity score of the subnet $\mathcal{C}(\alpha)$ using Eq. (3).
  Obtain momentum μ_i from cluster i where subnet α is located using Eq. (7).
  Compute gradients g^t of the subnet α w.r.t. train loss.
  Update momentum: $\mu_i^t = \beta \cdot \mu_i^{t-1} + g^t$, where β is a coefficient of moving average.
  Update weights: $\mathcal{W}^t(\alpha) = \mathcal{W}^{t-1}(\alpha) - \eta^t \cdot \mu_i^t$.
end for

Algorithm 2 Dynamic supernet training on FairNAS [3]

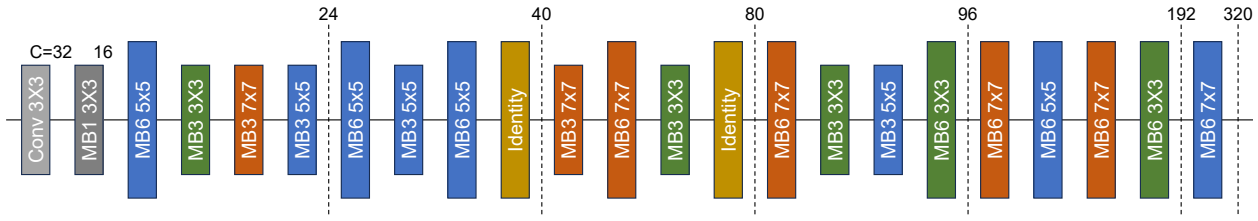
Input: Supernet $\mathcal{N}$, weights of supernet $\mathcal{W}$, momentum of supernet μ, set of clusters S, supernet depth L, number of candidate operations n, training set $\mathcal{D}_{train}$, total number of iterations T.
Initialize each cluster's momentum: $\mu^0 = 0$.
for $t = 1$ to T **do**
  Sample a mini-batch from training set $\mathcal{D}_{train}$.
  Randomly select a cluster S_i from the set of clusters S.
  for $l = 1$ to L **do**
    c_l = a uniform permutation of index for n candidate operations of layer l.
  end for
  Initialize gradients: $g^t = 0$.
  for $k = 1$ to n **do**
    $c_{e_k} = i$, where e is a selected layer for MS. // To ensure that the sampled subnets are within the cluster.
    Sample a subnet α_k = $(c_{1_k}, c_{2_k}, \cdots, c_{L_k})$.
    Compute η_k^t of current iteration t, considering a complexity score of the subnet $\mathcal{C}(\alpha_k)$ using Eq. (3).
    Compute gradients g_k^t of the subnet α_k w.r.t train loss.
    Accumulate gradients: $g^t \leftarrow g^t + g_k^t \cdot \eta_k^t$.
  end for
  Update momentum: $\mu_i^t = \beta \cdot \mu_i^{t-1} + g^t$, where β is a coefficient of moving average.
  Update weights: $\mathcal{W}^t(\alpha) = \mathcal{W}^{t-1}(\alpha) - \mu_i^t$.
end for

Algorithm 3 Dynamic supernet training on FSNAS [18]

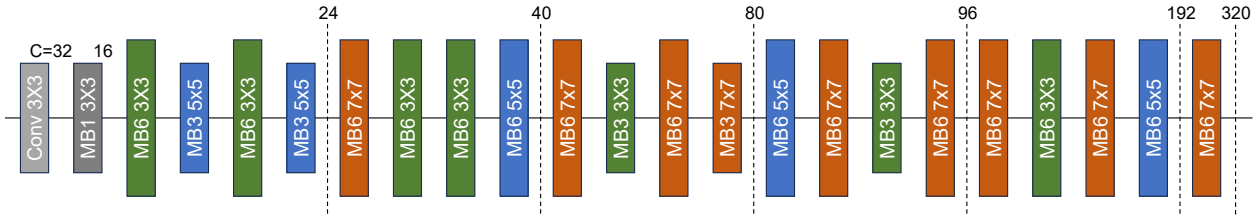
Input: Supernet $\mathcal{N}$, weights of supernet $\mathcal{W}$, momentum of supernet μ, training set $\mathcal{D}_{train}$, total number of iterations T.
Randomly split supernet $\mathcal{N}$ into K sub-supernets: $\{\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_K\}$.
for $k = 1$ to K **do**
  Initialize each cluster's momentum within a sub-supernet $\mathcal{N}_k$: $\mu_k^0 = 0$.
  for $t = 1$ to T **do**
    Sample a mini-batch from training set $\mathcal{D}_{train}$.
    Randomly sample a subnet α from a supernet $\mathcal{N}_k$.
    Compute LR η^t of current iteration t, considering a complexity score of the subnet $\mathcal{C}(\alpha)$ using Eq. (3).
    Obtain momentum μ_{k_i} from cluster k_i where α is located using Eq. (7).
    Compute gradients g^t of the subnet α w.r.t train loss.
    Update momentum: $\mu_{k_i}^t = \beta \cdot \mu_{k_i}^{t-1} + g^t$, where β is a coefficient of moving average.
    Update weights: $\mathcal{W}_k^t(\alpha) = \mathcal{W}_k^{t-1}(\alpha) - \eta^t \cdot \mu_{k_i}^t$.
  end for
end for



(a) SPOS-S + Ours.

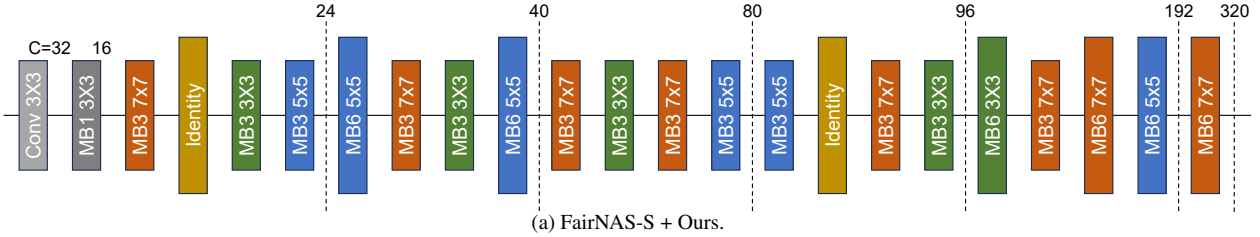


(b) SPOS-M + Ours.

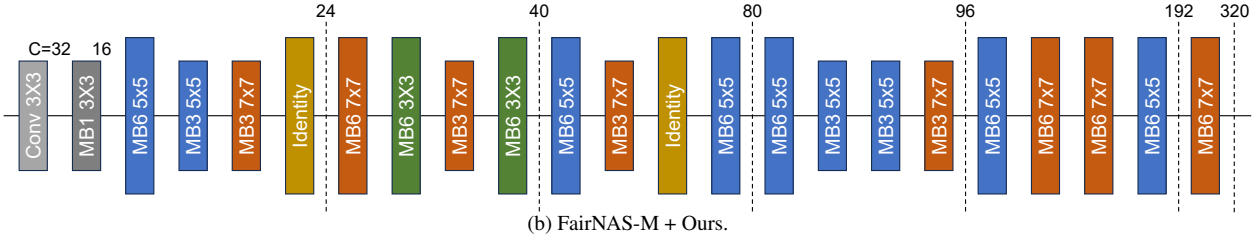


(c) SPOS-L + Ours.

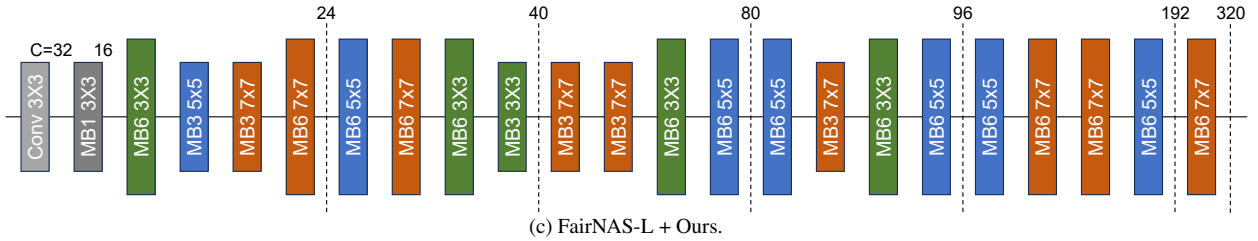
Figure E. Visualizations of our searched subnets on ImageNet [4] in the MobileNet space [1] using SPOS [7] + Ours.



(a) FairNAS-S + Ours.

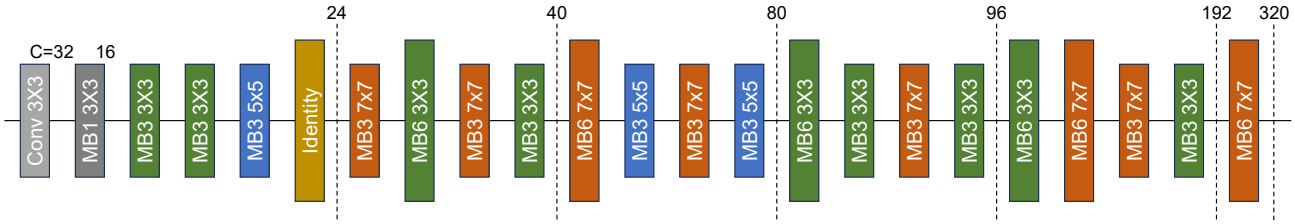


(b) FairNAS-M + Ours.

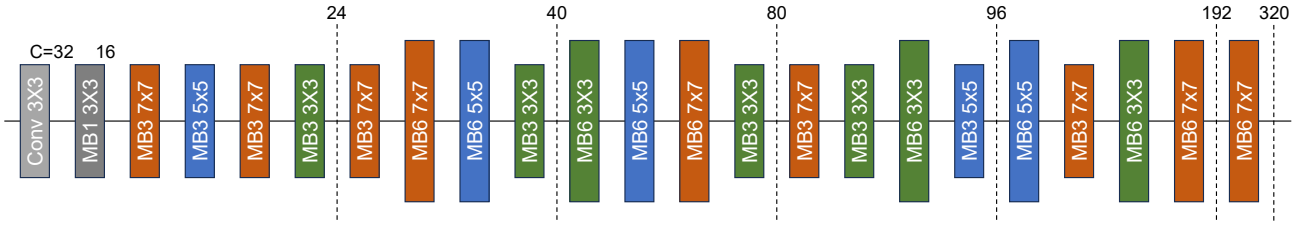


(c) FairNAS-L + Ours.

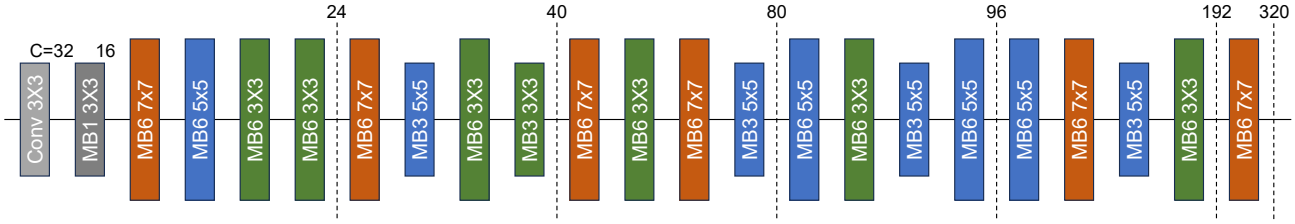
Figure F. Visualizations of our searched subnets on ImageNet [4] in the MobileNet space [1] using FairNAS [3] + Ours.



(a) FSNAS-S + Ours.



(b) FSNAS-M + Ours.



(c) FSNAS-L + Ours.

Figure G. Visualizations of our searched subnets on ImageNet [4] in the MobileNet space [1] using FSNAS [18] + Ours.

References

[1] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 1, 6, 8, 9

[2] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *ICCV*, 2021. 1, 2

[3] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. In *ICCV*, 2021. 1, 2, 4, 5, 6, 7, 8

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2, 3, 6, 8, 9

[5] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 2019. 1, 2

[6] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 1, 2, 3, 5, 6

[7] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 1, 2, 4, 5, 6, 7, 8

[8] Hyeonmin Ha, Ji-Hoon Kim, Semin Park, and Byung-Gon Chun. SUMNAS: Supernet with unbiased meta-features for neural architecture search. In *ICLR*, 2022. 1, 2

[9] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1, 2, 3, 5

[10] Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. ZICO: Zero-shot NAS via inverse coefficient of variation on gradients. In *ICLR*, 2023. 1

[11] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2020. 2

[12] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-NAS: A zero-shot NAS for high-performance image recognition. In *ICCV*, 2021. 1

[13] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 1, 2

[14] Shun Lu, Yu Hu, Longxing Yang, Zihao Sun, Jilin Mei, Jianchao Tan, and Chengru Song. PA&DA: Jointly sampling path and data for consistent NAS. In *CVPR*, 2023. 1

[15] Houwen Peng, Hao Du, Hongyuan Yu, Qi Li, Jing Liao, and Jianlong Fu. Cream of the Crop: Distilling prioritized paths for one-shot neural architecture search. In *NeurIPS*, 2020. 1, 2

[16] Jiefeng Peng, Jiqi Zhang, Changlin Li, Guangrun Wang, Xiaodan Liang, and Liang Lin. Pi-NAS: Improving neural architecture search by reducing supernet training consistency shift. In *ICCV*, 2021. 6

[17] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su. Overcoming multi-model forgetting in one-shot NAS with diversity maximization. In *CVPR*, 2020. 1, 2

[18] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *ICML*, 2021. 1, 2, 4, 5, 6, 7, 9