GrowTAS: Progressive Expansion from Small to Large Subnets for Efficient ViT Architecture Search

Supplementary Material

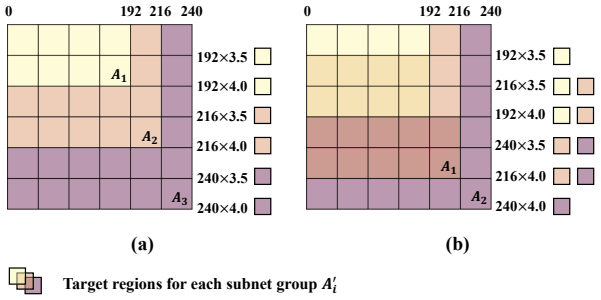


Figure A. Comparison between (a) the ideal and (b) the actual target weight regions for each subnet group $\mathcal{A}_i'$, based on the AutoFormer-T configuration. While the ideal case assumes a clean inclusion relationship between subnet groups based on increasing embed dimensions, the actual case shows that differences in MLP ratios can lead to overlapping or reversed inclusion relationships.

Detailed Definition of Subspace. To realize progressive training across disjoint subnet groups, we must define a criterion that enables a clear and consistent partitioning of the search space. Among the tunable architectural attributes such as embedding dimension, MLP ratio, number of attention heads, and depth, only the embedding dimension and depth remain constant across all layers of a subnet. Other attributes may vary layer-wise, which leads to ambiguity when assigning subnets to a unique group. Since our focus lies in analyzing the interference between subnets from a width-oriented perspective (for example, weight matrix shape and reuse), we exclude depth from our group definition. Therefore, we choose the embedding dimension as the primary axis to construct disjoint subnet groups, as it directly determines the width of projection matrices and governs the layout of the shared weights. In AutoFormer's search space, each variant (T, S, and B) provides exactly three discrete choices for the embedding dimension. This implies that the maximum number of disjoint groups $K_{\max}$ is inherently bounded by 3.

Why We Set $K = 2$. While embedding dimension serves as a reliable basis for defining disjoint subnet groups, the actual weight usage can be distorted by variations in the MLP ratio. In particular, a subnet with a smaller embedding dimension but a larger MLP ratio may consume more parameters than a neighboring subnet with a larger embedding dimension. Such inconsistencies break the intended inclusion relationship across progressively growing subnet groups, thereby weakening the weight transferability and stability during training.

To formalize this, we define $\mathcal{Q}_i$ as the subgroup of subnets that share the same embedding dimension d_i. Although our main training space is progressively constructed as $\mathcal{A}_k = \bigcup_{i=1}^{k+1} \mathcal{Q}_i$, this inclusion may not always translate to a superset relationship in weight usage (i.e., $W_{\tilde{\alpha}} \subset W_{\hat{\alpha}}$ for $\tilde{\alpha} \in \mathcal{Q}_i$, $\hat{\alpha} \in \mathcal{Q}_{i+1}$). As shown in Fig. A, due to the MLP ratio, a subnet in $\mathcal{Q}_i$ can occupy a larger weight region than one in $\mathcal{Q}_{i+1}$, even when $d_i < d_{i+1}$. This violates the expected progressive structure and leads to training instability.

To mitigate this issue, we expand each training stage to include both the current group and its immediate neighbor. For example, instead of training on $\mathcal{Q}_1$ alone, we sample from $\mathcal{Q}_1 \cup \mathcal{Q}_2$, and similarly for later stages. This modification ensures that potentially overlapping regions are jointly optimized and reduces weight under-training caused by isolated updates. As a result, the number of training stages reduces from $K = 3$ (each based on a single group) to $K = 2$, each encompassing two adjacent subgroups.

Empirically, we find that this setting with $K = 2$ yields the most stable and efficient training outcome, as verified in our ablation study. Note that the case of $K = 1$ corresponds to the baseline AutoFormer training setup that samples uniformly from the entire search space. For example, in the AutoFormer-T search space, the available embedding dimensions are 192, 216, and 240. Based on our grouping strategy, we define subnet groups using the notation $\{\alpha \mid d(\alpha) = d_i\}$, where $d(\alpha)$ denotes the embedding dimension of subnet α. The two training stages are then constructed as follows:

$$\mathcal{A}_1 = \bigcup_{d \in \{192,216\}} \{\alpha \mid d(\alpha) = d\},$$
$$\mathcal{A}_2 = \bigcup_{d \in \{192,216,240\}} \{\alpha \mid d(\alpha) = d\}.$$

This corresponds to a neighborhood-aware progressive schedule with $K = 2$, where each stage covers overlapping subnet groups to ensure sufficient joint optimization of shared weights.

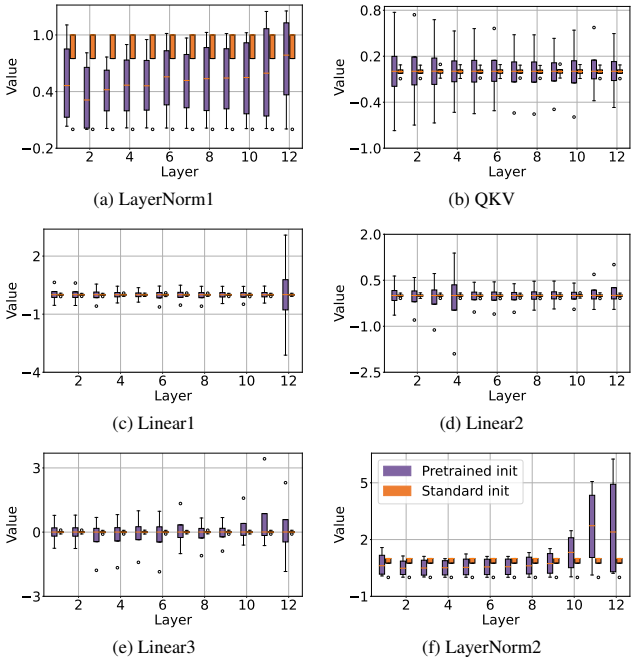


Figure B. Weight statistics (mean, variance, min, max) per layer for each module. (a)–(f): Six different components of ViT.

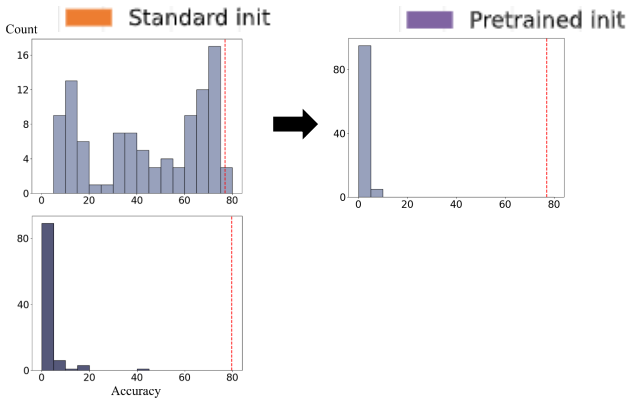
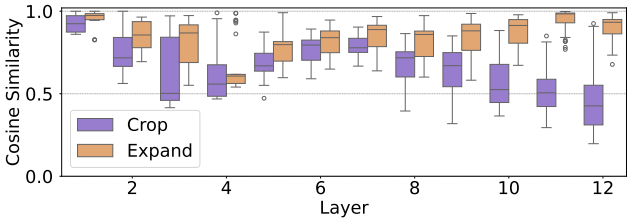


Figure C. The left histograms replicate the right plot in Fig.1, showing that many extended subnets maintain high accuracy under standard initialization. The right histogram demonstrates that when the untrained weight regions are initialized with values scaled to match the pretrained distribution (i.e., higher variance and magnitude), the performance of the same extended subnets drops significantly.

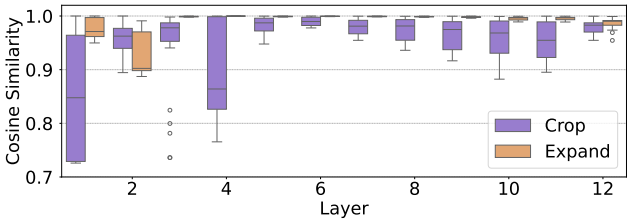
Why Weight Extension Preserves Performance We further investigate why subnets extended from smaller ones tend to retain high performance, focusing on how the initialization of newly introduced weights affects accuracy. Specifically, we analyze two initialization schemes for the untrained weight regions that are activated only in larger subnets: (1) *standard initialization* such as truncated normal or constant values close to zero, and (2) *pretrained initialization* where values are randomly sampled from the distribution of already-trained weights.

As illustrated in Fig. B, pretrained initialization produces weight values with much larger variance, especially in modules such as QKV, Linear, and LayerNorm layers. In contrast, standard initialization yields values that are tightly concentrated near zero. This difference has a significant impact on network behavior. When weights are initialized with near-zero values, they remain effectively inactive during early training iterations, allowing the trained regions from smaller subnets to dominate the representation. However, weights initialized with large random values inject noise into the forward and backward passes, disrupting the stable features learned in the small subnets. This effect is quantitatively verified in Fig. C. Under standard initialization, the accuracy of most extended subnets is preserved, with the majority achieving over 70% accuracy. In contrast, pretrained initialization causes a drastic collapse in performance, shifting the distribution heavily toward low accuracy. This suggests that initializing the extended weights with large, untrained values impairs the model's ability to reuse previously learned features. These findings support our progressive training strategy that begins with small subnets and gradually expands to larger ones. To prevent interference from the newly activated (and previously untrained) weights in larger subnets, we initialize those regions with near-zero values. This ensures that the learned representations in the already-trained regions remain intact, allowing the larger subnets to be optimized more stably in later stages. These findings highlight why progressive expansion is effective: small subnets learn reliable representations first, and when larger subnets are built on top of them with careful initialization, the added parameters do not disrupt existing knowledge. This allows larger subnets to smoothly extend and refine what smaller ones have already learned.

Additional Analysis: Extend vs. Crop in CLS Token Representations To further support our hypothesis on the asymmetric interference between expansion and cropping, we extend the analysis in the main paper by separately examining the representational similarity of the CLS token. While the main paper focused on cosine similarity across all tokens, Fig. D provides a complementary view by comparing both (a) the average similarity across all tokens and (b) the similarity specific to the CLS token for each transformer block. We observe that expanded subnets consistently exhibit higher cosine similarity to the base subnet compared to cropped subnets, not only when averaging over all tokens but also when considering the CLS token alone. The CLS token, which aggregates global information for classification, serves as a strong indicator of semantic consistency. A



(a) ALL tokens



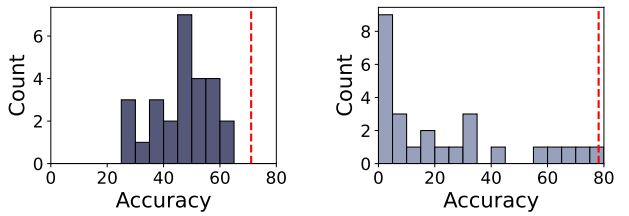
(b) CLS token

Figure D. Layer-wise cosine similarity between expanded/cropped subnets and the base subnet. (a) ALL tokens. (b) CLS token.

lower similarity in this token, as seen in the cropped subnet case, implies a significant deviation from the learned representation of the base model, leading to unstable generalization. These findings support the conclusions presented in the main paper. Subnets grown from smaller and well-trained models inherit more consistent and transferable features, while cropped subnets tend to suffer from representational drift because they remove previously trained regions. The additional CLS-based analysis provides further empirical evidence that our progressive expansion strategy is more effective.

Search Space Generalization: Experiment on OFA In the main paper (Fig. 1), we showed that in AutoFormer the performance of subnets behaves asymmetrically: extending a subnet (to a larger one) preserves accuracy, while cropping (to a smaller one) causes severe degradation. To confirm that this phenomenon is not specific to AutoFormer, we repeated the same motivation experiment on OFA [1], a CNN-based one-shot NAS search space where subnets vary by kernel size k, expand ratio e, and depth d. Figure E shows two cases: (a) training the supernet with only the smallest subnet ($k = 3, e = 3, d = 2$) and (b) training with only the largest subnet ($k = 7, e = 6, d = 4$). We again observe the same asymmetric behavior as in AutoFormer, which clearly indicates that our method is not tied to a particular search space but is broadly applicable to size-based one-shot NAS.

Training Cost Comparison. Table A reports the training cost of GrowTAS compared with AutoFormer and PreNAS under the same backbones. We see that GrowTAS



(a) Smallest-subnet ($k = 3, e = 3, d = 2$)

(b) Largest-subnet ($k = 7, e = 6, d = 4$)

Figure E. Motivation experiment on OFA. (a) Supernet trained with only the smallest subnet. (b) Supernet trained with only the largest subnet. The same asymmetric behavior as in AutoFormer is observed.

Method	Top-1 (%)	Top-5 (%)	#Params (M)	Time (days)
AutoFormer-T	74.7	92.6	5.7	2.56
PreNAS-T	77.1	93.4	5.9	2.56
GrowTAS-T	**75.2**	**92.7**	**5.9**	**2.56**
GrowTAS-T†	**77.1**	**93.4**	**5.9**	**2.56**
AutoFormer-B	82.4	95.7	54	6.62
PreNAS-B	82.6	96.0	54	6.62
GrowTAS-B	**82.6**	**96.0**	**53.2**	**6.62**
GrowTAS-B+	**82.7**	**96.1**	**53.3**	**6.69**

Table A. Comparison of training cost among baselines.

achieves consistently higher accuracy with no additional cost, and GrowTAS+ requires only a very small overhead (about 1% in the Base model) due to selective fine-tuning. This demonstrates that our framework is both effective and computationally efficient.

Dynamic Transition Scheduling In the main paper, the transition step T_k was treated as a free hyperparameter chosen manually. Here, we provide a more systematic way to define T_k. A simple baseline is the midpoint rule, which evenly splits the remaining training epochs across stages:

$$T_1 = \frac{T}{2}, \quad T_k = T_{k-1} + \frac{T - T_{k-1}}{2}. \tag{1}$$

This ensures each expansion stage receives balanced training time. However, in practice the best step depends on model size. With $T = 500$, we observed optimal first steps around 250 for AutoFormer-T, 225 for AutoFormer-S, and 200 for AutoFormer-B. Larger models converge to lower loss more quickly [3], so they can build a sufficient foundation earlier in smaller search spaces and thus move to the next transition sooner. To capture this effect, we propose a simple size-aware rule:

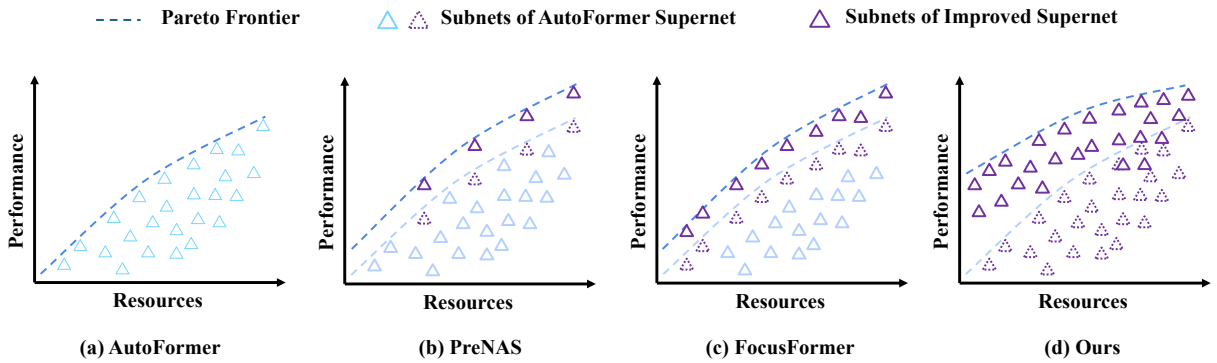


Figure F. Comparison of different sampling strategies in one-shot TAS [2, 4, 5]. AutoFormer [2] uniformly trains all subnets but fails to optimize them effectively. PreNAS [5] selects a small set of candidates under a given constraint and improves them individually. FocusFormer [4] dynamically samples high-quality subnets during training to enhance their accuracy. In contrast, our method improves the overall quality of all subnets, especially those with small resource budgets.

$$
\begin{aligned}
T_1 &= \frac{T}{2} - \beta \cdot \frac{C}{C_{\max}}, \\
T_k &= T_{k-1} + \frac{T - T_{k-1}}{2} - \beta \cdot \frac{C}{C_{\max}}.
\end{aligned} \tag{2}
$$

where C is the model size and $C_{\max}$ is the maximum size in the search space. For example, in AutoFormer, as shown in Table 1 of the main paper, the parameter counts of supernets are approximately 9M for T, 34M for S, and 75M for B. Thus, $C_{\max}$ corresponds to 75M in this case. With $\beta = 50$, this predicts 244 for T, 227 for S, and 200 for B, which are close to the observed optima (see Table 4). When we trained the models using the predicted T for each case, the resulting performance turned out to be equivalent to the optimal results reported in Table 4. These results suggest that T_k can be guided by interpretable formulas rather than arbitrary manual settings, and could be further extended to dynamic schedules that adapt to model size or convergence speed.

Practical Application of GrowTAS in Training Figure G illustrates how GrowTAS operates in practice, using AutoFormer-T as an example. We divide the subnet space into three disjoint groups, $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$, each consisting of subnets with different embedding dimensions (192, 216, and 240, respectively). At each training stage, GrowTAS progressively expands the sampling space by including the next group. Within each active group, subnets are sampled uniformly, and the sampling probabilities are visualized in the bar chart. This figure concretely demonstrates how GrowTAS schedules the training process in a size-aware and stage-wise manner.

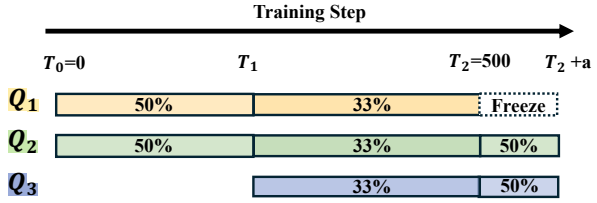


Figure G. Stage-wise sampling schedule of GrowTAS. Subnet groups $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$ correspond to subnets with embedding dimensions of 192, 216, and 240, respectively. Each bar shows the sampling probability at a given stage, indicating uniform sampling within each active group.

Comparison with Existing Sampling Strategies. Figure F provides a conceptual comparison between our method and previous sampling strategies proposed in one-shot TAS. AutoFormer [2] samples all subnets uniformly throughout training, which fails to optimize them effectively due to weight interference. PreNAS [5] narrows the training scope to a small set of subnet candidates selected at initialization, thus improving performance for only a few constraints. FocusFormer [4] dynamically learns to favor high-performing subnets during training but increases training complexity and neglects the rest. In contrast, our method improves the quality of all subnets by progressively increasing the subnet size during training. This strategy helps preserve the performance of small subnets, which are more sensitive to weight interference, while still ensuring that all subnets in the search space are sufficiently trained without adding extra parameters or requiring retraining.

References

[1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020.

[2] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12270–12280, 2021.

[3] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 5958–5968. PMLR, 2020.

[4] Jing Liu, Jianfei Cai, and Bohan Zhuang. Focusformer: Focusing on what we need via architecture sampler. *arXiv preprint arXiv:2208.10861*, 2022.

[5] Haibin Wang, Ce Ge, Hesen Chen, and Xiuyu Sun. Prenas: Preferred one-shot learning towards efficient neural architecture search. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.