Stratified Knowledge-Density Super-Network for Scalable Vision Transformers

Longhua Li[1,2], **Lei Qi**[1,2*], **Xin Geng**[1,2*]

[1]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
{lhli, qilei, xgeng}@seu.edu.cn

Abstract

Training and deploying multiple vision transformer (ViT) models for different resource constraints is costly and inefficient. To address this, we propose transforming a pre-trained ViT into a stratified knowledge-density super-network, where knowledge is hierarchically organized across weights. This enables flexible extraction of sub-networks that retain maximal knowledge for varying model sizes. We introduce **W**eighted **P**CA for **A**ttention **C**ontraction (WPAC), which concentrates knowledge into a compact set of critical weights. WPAC applies token-wise weighted principal component analysis to intermediate features and injects the resulting transformation and inverse matrices into adjacent layers, preserving the original network function while enhancing knowledge compactness. To further promote stratified knowledge organization, we propose **P**rogressive **I**mportance-**A**ware **D**ropout (PIAD). PIAD progressively evaluates the importance of weight groups, updates an importance-aware dropout list, and trains the super-network under this dropout regime to promote knowledge stratification. Experiments demonstrate that WPAC outperforms existing pruning criteria in knowledge concentration, and the combination with PIAD offers a strong alternative to state-of-the-art model compression and model expansion methods.

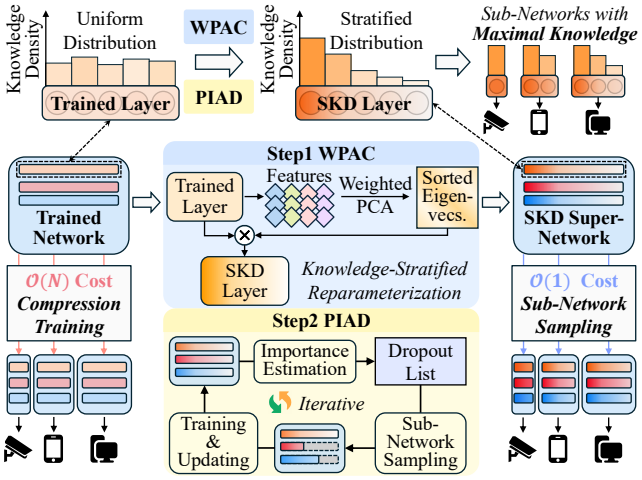


Figure 1: An overview of the proposed method. Traditional methods require separate compression for each deployment setting. In contrast, our method builds a **S**tratified **K**nowledge-**D**ensity (SKD) Super-Network in a single pass, enabling cost-free sub-network extraction for arbitrary model sizes.

Introduction

Vision Transformers (ViTs) (Dosovitskiy et al. 2020) have demonstrated remarkable performance across various computer vision tasks (Radford et al. 2021; Jose et al. 2025; Hua et al. 2024, 2025). However, deploying ViTs in real-world scenarios often requires adapting model size to diverse resource constraints, ranging from edge devices with limited memory and compute to powerful servers requiring high-throughput inference. The conventional approach of training and maintaining multiple ViT variants for each deployment setting is computationally expensive and inefficient. Moreover, many existing pruning or model compression methods either require extensive fine-tuning or suffer from performance degradation due to suboptimal knowledge retention.

Recent works (Xia et al. 2024a,b,c; Feng et al. 2025) have adopted the Learngene paradigm, which extracts or distills core parameters from a pre-trained model and expands them to construct models of varying sizes. However, these methods rely on manually designed expansion rules to reuse a limited portion of refined weights, making it difficult to introduce new or size-specific knowledge during expansion, thus limiting the capacity alignment of the resulting models.

To address these challenges, we propose transforming a pre-trained ViT into a **S**tratified **K**nowledge-**D**ensity (SKD) Super-Network, as illustrated in Figure 1. Guided by knowledge density, sub-networks of various sizes are extracted to maximize knowledge retention relative to their capacity.

Specifically, we introduce WPAC to perform initial stratification of knowledge density. WPAC leverages a small proxy dataset, randomly sampled from the training set, to compute intermediate attention features. These features are weighted token-wise using Taylor-based importance scores and then decomposed via PCA to obtain a projection matrix composed of ordered principal components. The projection and its inverse are integrated into the layers surrounding the intermediate features, preserving the original network function while concentrating knowledge into a smaller set of critical dimensions,

*Co-corresponding authors.

thus achieving a stratified distribution of knowledge density across parameter dimensions.

To further enhance this stratification, we propose PIAD, which assigns lower dropout probabilities to more important parameter groups and higher probabilities to less important ones. This progressively amplifies the distinction in knowledge density, particularly improving the performance of smaller sub-networks. Together, WPAC and PIAD enable a single ViT model to function as a scalable backbone, from which sub-networks of arbitrary sizes can be extracted at $\mathcal{O}(1)$ cost based on the learned importance ranking.

Our contributions can be summarized as follows:

- We propose WPAC, a weighted PCA mechanism that condenses knowledge into fewer dimensions via function-preserving transformations. It consistently outperforms existing pruning criteria in knowledge concentration.
- We introduce PIAD, a progressive dropout mechanism that builds a scalable super-network via importance-aware sub-network sampling and training. It achieves flexible control over compression while maintaining accuracy.
- We achieve strong results on standard benchmarks, outperforming or matching state-of-the-art model pruning and scaling methods. Our framework enables flexible and efficient ViT deployment across diverse resource constraints.

Related Work

Network Pruning. Network pruning (LeCun, Denker, and Solla 1989; Fang et al. 2023) reduces model complexity by removing redundant or less important weights. While magnitude-based pruning (Han et al. 2015; Li et al. 2018; Lee et al. 2021) is widely used, small weights can still play critical roles. FPGM (He et al. 2019) leverages geometric medians to identify redundant filters. Other methods construct importance metrics based on activation values (Hu et al. 2016; Muralidharan et al. 2024) or information entropy (Luo and Wu 2017), while many approaches (Molchanov et al. 2017, 2019) apply Taylor expansion to approximate the loss induced by pruning. However, these methods often require time-consuming pruning for each target size.

Scalable Neural Networks. Scalable neural networks are designed to support flexible model capacity adjustment for different resource constraints. Notable examples include Slimmable Networks (Yu et al. 2019) and Once-for-All (Cai et al. 2020), which enables sub-network extraction with elastic size. Recently, a paradigm called Learngene (Xia et al. 2024a,c; Feng et al. 2025; Li, Qi, and Geng 2025) was proposed, which distills core weights and expands them to generate descendant models. In contrast, we introduce a stratified knowledge-density structure to enable fast extraction of sub-networks with maximal knowledge across sizes.

Low-Rank Compression. Low-rank approximation is a widely used strategy for compressing deep models by exploiting the redundancy in weight matrices. Techniques such as SVD and other matrix factorization methods have been applied to reduce parameter count and computational cost (Noach and Goldberg 2020; Wang, Wohlwend, and Lei 2020; Mao et al. 2020; Yu and Wu 2023). Instead of splitting a layer into two low-rank components, we perform PCA on intermediate features and apply the transformation and its inverse to adjacent layers, concentrating knowledge into a few dimensions while preserving network function.

Method

Figure 1 shows the framework of our method. WPAC first uses PCA to induce a stratified distribution of knowledge in the pre-trained model, and PIAD further enhances this effect, resulting in a stratified knowledge-density super-network. This enables cost-free extraction of sub-networks that retain maximal knowledge, providing strong initialization for downstream scenarios with varying resource constraints.

Preliminaries

We consider a standard ViT consisting of a stack of transformer blocks, each with a Multi-Head Self-Attention (MHSA) module followed by a feed-forward network (MLP).

In MHSA, each attention head uses four linear projections: $W_q, W_k, W_v, W_o \in \mathbb{R}^{d\times d}$, with corresponding biases $b_q, b_k, b_v, b_o \in \mathbb{R}^{d\times 1}$. The MLP consists of two fully connected layers with a hidden dimension of d'. The weight and bias of the first and second linear layers are denoted as $W_1 \in \mathbb{R}^{d'\times d}, b_1 \in \mathbb{R}^{d'\times 1}$, and $W_2 \in \mathbb{R}^{d\times d'}, b_2 \in \mathbb{R}^{d\times 1}$.

Weighted PCA for Attention Concentration

Figure 2 illustrates the WPAC method, which uses a small proxy dataset $\mathcal{D}$ and PCA to transform the pre-trained weight matrix, concentrating knowledge into key dimensions. This transformation is applied separately to each head. For brevity, head indices are omitted unless otherwise specified.

Transformation for Linear Projections V and O. Let $X_v \in \mathbb{R}^{n\times d}$ denote the output features of the value projection layer, where n is the number of tokens. We center the data and compute its covariance matrix $\mathbf{Corr} \in \mathbb{R}^{d\times d}$. This covariance is updated incrementally across the proxy set. Applying eigen decomposition to $\mathbf{Corr}$ yields eigenvalues and eigenvectors, sorted in descending order. The eigenvectors form a transformation matrix $W^{(vo)}_{\text{Trans}} \in \mathbb{R}^{d\times d}$, where each row corresponds to a principal component. We apply $W^{(vo)}_{\text{Trans}}$ to the value and output projections as follows:

$$W_v \leftarrow W^{(vo)}_{\text{Trans}} W_v,\ b_v \leftarrow W^{(vo)}_{\text{Trans}} b_v,\ W_o \leftarrow W_o (W^{(vo)}_{\text{Trans}})^{-1}. \tag{1}$$

This transformation satisfies:

- **Function Preservation**: Let $x \in \mathbb{R}^d$ be a token representation input to the attention mechanism. Ignoring the attention map's weighting across tokens, the original forward pass is $W_o(W_v x + b_v) + b_o$. After applying the WPAC transformation, the equivalent form becomes $W_o(W^{(vo)}_{\text{Trans}})^{-1}(W^{(vo)}_{\text{Trans}} W_v x + W^{(vo)}_{\text{Trans}} b_v) + b_o$, which remains mathematically equivalent to the original.
- **Information Concentration**: Let $f_v = W_v x + b_v$ denote the output of the linear projection V. By extracting the top-k principal components, we obtain $(W^{(vo)}_{\text{Trans}})^{[:k,:]} f_v$, which retains most of the original feature's information.

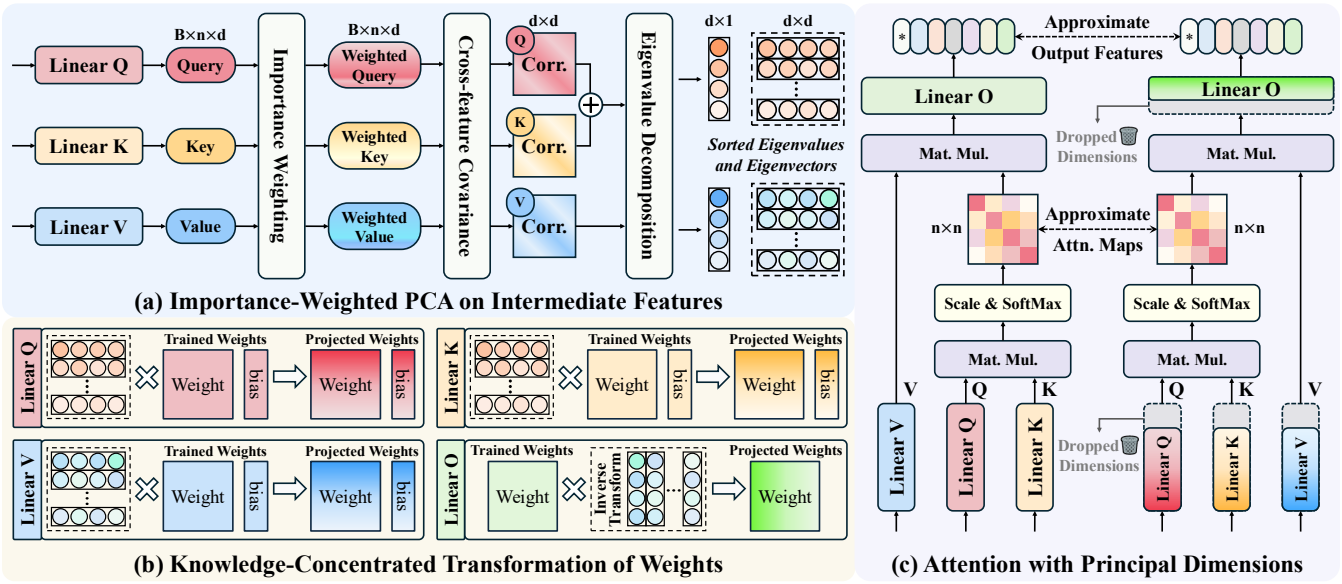


Figure 2: An overview of the proposed WPAC. (a) Principal component projection matrices are computed using importance-weighted intermediate features. (b) The resulting transformation matrices are applied to the pre-trained weights. (c) The transformed linear layers can preserve the original neural function using only a small number of principal dimensions.

This is equivalent to retaining only the top-k dimensions of the transformed linear projection parameters:

$$(W_{\text{Trans}}^{(vo)})^{[:k,:]} f_v \equiv (W_{\text{Trans}}^{(vo)} W_v)^{[:k,:]} x + (W_{\text{Trans}}^{(vo)} b_v)^{[:k]}. \tag{2}$$

Transformation for Linear Projections Q and K. For the query and key projections, we compute the covariance matrices of their output token representations and sum them before applying eigen decomposition. The resulting transformation matrix $W_{\text{Trans}}^{(qk)}$ is then applied as follows:

$$\begin{aligned} W_q &\leftarrow W_{\text{Trans}}^{(qk)} W_q, \quad b_q \leftarrow W_{\text{Trans}}^{(qk)} b_q, \\ W_k &\leftarrow W_{\text{Trans}}^{(qk)} W_k, \quad b_k \leftarrow W_{\text{Trans}}^{(qk)} b_k. \end{aligned} \tag{3}$$

This transformation ensures:

- **Function Preservation**: The original similarity between input tokens x_i and x_j is computed as follows:

$$\text{sim}_{(i,j)} = (W_q x_i + b_q)^T (W_k x_j + b_k). \tag{4}$$

Using the transformed projections, the similarity becomes $(W_{\text{Trans}}^{(qk)} W_q x_i + W_{\text{Trans}}^{(qk)} b_q)^T (W_{\text{Trans}}^{(qk)} W_k x_j + W_{\text{Trans}}^{(qk)} b_k)$. Through algebraic manipulation, this can be rewritten as $(W_q x_i + b_q)^T (W_{\text{Trans}}^{(qk)})^T W_{\text{Trans}}^{(qk)} (W_k x_j + b_k)$. Since $W_{\text{Trans}}^{(qk)}$ is orthogonal, we obtain the following identity:

$$\text{sim}_{(i,j)} \equiv (W_q x_i + b_q)^T (W_{\text{Trans}}^{(qk)})^T W_{\text{Trans}}^{(qk)} (W_k x_j + b_k). \tag{5}$$

- **Information Concentration**: Let $\boldsymbol{q_i} = W_q x_i + b_q$ and $\boldsymbol{k_j} = W_k x_j + b_k$. When retaining only the top-k dimensions of the transformed linear projections Q and K, the resulting similarity score between tokens x_i and x_j is:

$$\text{sim}_{(i,j)}^{\text{top-}k} = \boldsymbol{q_i} ((W_{\text{Trans}}^{(qk)})^{[:k,:]})^T (W_{\text{Trans}}^{(qk)})^{[:k,:]} \boldsymbol{k_j}. \tag{6}$$

Since PCA preserves principal components of the original features, the following approximation holds:

$$\boldsymbol{k_j} \approx ((W_{\text{Trans}}^{(qk)})^{[:k,:]})^T (W_{\text{Trans}}^{(qk)})^{[:k,:]} \boldsymbol{k_j}, \tag{7}$$

which leads to:

$$\text{sim}_{(i,j)}^{\text{top-}k} \approx \boldsymbol{q_i}^T \boldsymbol{k_j} = \text{sim}_{(i,j)}. \tag{8}$$

Therefore, keeping only the top-k dimensions of the transformed linear projections yields an approximate similarity matrix that closely matches the original.

Weighted PCA. PCA inherently treats all tokens and feature dimensions equally, which may not reflect their true contributions to prediction. To address this, we introduce token-wise weighting based on first-order Taylor importance. The importance of a parameter or output element h_i is estimated by the change in cost $\mathcal{C}(\mathcal{D})$, evaluated on a proxy set $\mathcal{D}$, when h_i is set to zero. The estimated importance score is:

$$\Theta_{\text{TE}}(h_i) = \left|\Delta \mathcal{C}(h_i)\right| \approx \left| \frac{\delta \mathcal{C}}{\delta h_i} \cdot h_i \right|. \tag{9}$$

Concretely, we compute the element-wise importance $\Theta_{\text{TE}} \in \mathbb{R}^{n \times d}$ for intermediate token representations $X \in \mathbb{R}^{n \times d}$in the attention module, where n is the number of tokens and d is the feature dimension. We then aggregate over the feature dimension to obtain token-wise importance scores $\Theta_{TE}^{\text{token}} \in \mathbb{R}^n$. We apply a square-root weighting to the centered token features $\tilde{X}$ before computing the covariance matrix:

$$\tilde{X} \leftarrow \text{diag}(\sqrt{\Theta_{\text{TE}}^{\text{token}}}) \cdot \tilde{X}, \quad \textbf{Corr} = \frac{1}{n-1} \tilde{X}^T \tilde{X}. \tag{10}$$

This transformation biases the resulting principal components to better preserve high-importance token directions, while maintaining stability during eigen decomposition.

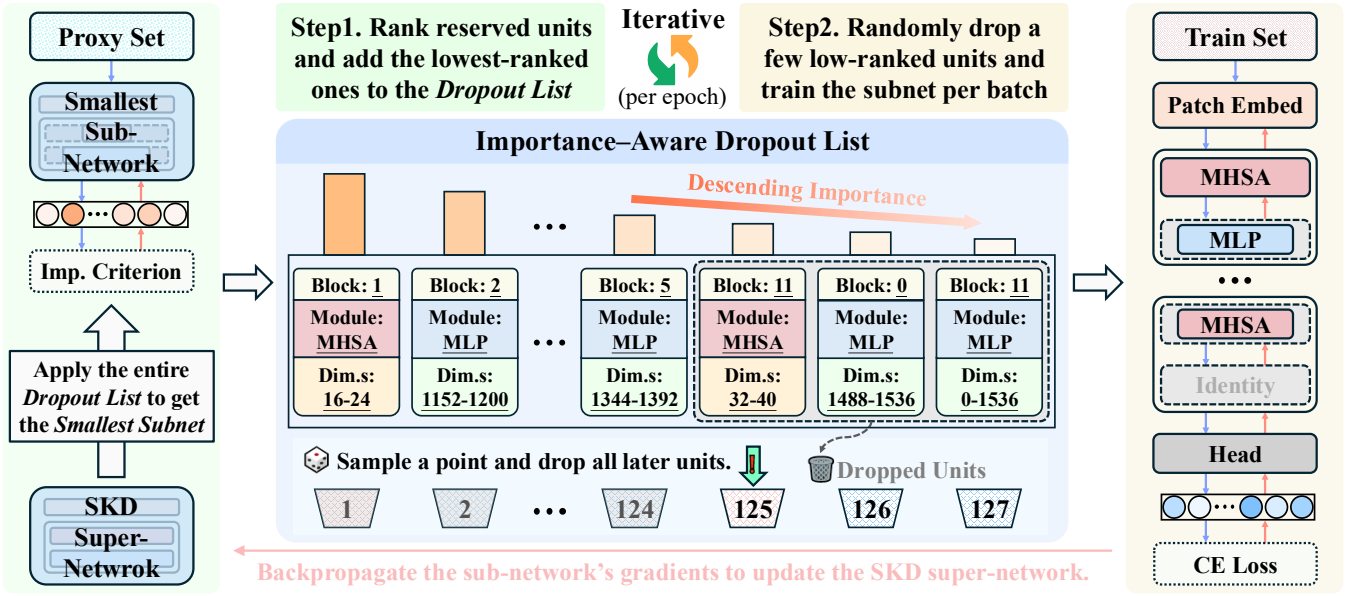


Figure 3: An overview of the proposed PIAD. **Step 1.** At the start of each epoch, evaluate the importance of parameter groups not yet in the dropout list and append the least important ones. **Step 2.** During training, sample sub-networks by randomly dropping the least important units from the dropout list and train them, propagating gradients back to the *SKD Super-Netwrok*.

Transformation for MLP. Due to the non-linear activation between MLP layers, applying PCA directly may distort the encoded information. As an alternative, we rank dimensions using Taylor importance. We first compute element-wise scores $\Theta_{\text{TE}} \in \mathbb{R}^{n \times d}$, then aggregate across tokens to obtain dimension-wise importance $\Theta_{\text{TE}}^{\text{dim}} \in \mathbb{R}^{d}$. A sorting matrix W_{sort} is then constructed to reorder dimensions accordingly:

$$W_{\text{sort}}[i, j] = \begin{cases} 1, & \text{if the } j\text{-th dimension ranks } i\text{-th,} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

We then apply the transformation and its inverse to the MLP:

$$W_1 \leftarrow W_{\text{sort}} W_1,\ b_1 \leftarrow W_{\text{sort}} b_1,\ W_2 \leftarrow W_2 W_{\text{sort}}^{-1}. \tag{12}$$

This process empirically preserves the original function while concentrating information into the top-ranked dimensions.

Progressive Importance-Aware Dropout

Figure 3 shows PIAD for enhancing knowledge stratification. After applying the above neural-equivalent transformation for knowledge concentration, the resulting network—referred to as the *SKD Network*—possesses a degree of scalability. To further enhance its adaptability across different model sizes, we introduce Progressive Importance-Aware Dropout (PIAD), a two-stage process repeated in each training epoch:

- **Before each epoch:** Rank reserved compression units and add the lowest-importance ones to the *Dropout List*.
- **During each epoch:** In each iteration, randomly drop several low-importance units in the dropout list, train the sub-network, and backpropagate to the *SKD Network*.

Building and Evaluating Droppable Units. The dropout list contains candidate droppable units, each assigned an importance score for ranking. Specifically, we divide the intermediate dimensions of each MHSA into 8 groups and those of each MLP into 32 groups, with each group treated as a droppable unit. Each group forms one droppable unit. To evaluate importance, we randomly sample a small proxy set $\mathcal{D}$ from the training data and proceed in two steps:

- **Module Sensitivity**: For each module m (e.g., MHSA or MLP), we define its sensitivity γ_m as the relative increase in the cost $\mathcal{C}(\mathcal{D})$ when the module is skipped:

$$\gamma_m = \frac{\mathcal{C}(\mathcal{D} \mid m \text{ skipped}) - \mathcal{C}(\mathcal{D})}{\mathcal{C}(\mathcal{D})}. \tag{13}$$

- **Dimension Importance**: For each dimension i in module m, we compute its importance score $\Theta_i^{(m)}$ using first-order Taylor expansion. These scores are normalized within the module to obtain contribution ratios $\alpha_i^{(m)}$, and the final global importance is defined as $I_i^{(m)}$:

$$\alpha_i^{(m)} = \frac{\Theta_i^{(m)}}{\sum_{j \in m} \Theta_j^{(m)}}, \quad I_i^{(m)} = \gamma_m \cdot \alpha_i^{(m)}. \tag{14}$$

Each droppable unit u (i.e., a group of dimensions) accumulates the scores of all its constituent dimensions:

$$I_u = \frac{\sum_{i \in u} I_i^{(m)}}{\text{MACs}\,(u)}, \tag{15}$$

where $\text{MACs}(u)$ denotes the MACs of unit u. These final scores I_u are used to rank the units for dropout selection.

Progressively Updating Dropout List. We adopt a progressive schedule to update the dropout list. Let the target

Method	Epochs	KD	DeiT-B (81.8)					DeiT-S (79.8)					DeiT-Ti (72.1)				
			4:12	6:12	8:12	10:12	12:12	4:12	6:12	8:12	10:12	12:12	4:12	6:12	8:12	10:12	12:12
Albert	10/10/10	*w/o*	71.7	75.3	76.4	77.4	77.6	65.0	69.7	71.7	72.7	73.3	55.2	59.8	62.5	64.3	65.3
LiGO	10/10/10	*w/o*	-	74.2	74.4	75.3	75.4	-	68.6	69.9	69.7	70.0	-	59.0	60.2	59.8	60.9
Heur-LG	10/10/10	*w/o*	60.5	68.7	72.2	73.6	74.0	52.3	57.3	61.7	64.4	65.9	41.5	47.4	50.5	53.5	55.5
Auto-LG	10/10/10	*w/o*	60.9	70.0	72.4	73.5	73.8	63.2	70.5	72.2	73.3	73.8	52.4	61.8	64.6	65.9	66.8
TLEG	40/35/50	*w*	71.6	76.2	78.1	79.1	79.9	63.7	69.5	72.3	73.9	75.1	-	58.2	-	-	65.4
SWS	10/10/10	*w*	-	76.9	79.0	79.7	80.1	-	70.2	73.4	75.1	75.8	-	-	-	-	-
WAVE	10/10/10	*w*	74.5	77.5	78.2	78.9	79.2	68.9	72.7	74.1	74.9	75.3	58.6	63.2	65.4	66.6	67.3
SKD (Ours)	0/0/0	*w/o*	**77.0**	**80.4**	**80.9**	**81.5**	**81.5**	**70.6**	**76.2**	**78.2**	**79.0**	**79.0**	**61.4**	**65.8**	**68.6**	**70.0**	**70.8**

Table 1: Performance comparison with network expansion methods on ImageNet-1k. "x:12" denotes the ratio of the scaled model's size to the original model. "KD" indicates whether additional knowledge distillation is required during the construction of the scalable network. "Epochs" refers to the number of fine-tuning epochs used for the scaled DeiT-B/S/Ti models.

maximum compression ratio be r, and define the progressive phase to span P_e epochs. At the beginning of epoch t, we select additional droppable units from the remaining candidates such that their total MACs approximate $\frac{r}{P_e} \times \text{MACs}_O$, where MACs_O denotes the MACs of the *SKD Network*. Specifically, we first form the *Smallest Sub-Network* by dropping all units currently in the list, then evaluate the importance of the remaining units using the proxy set. While traversing units from high to low dimensions within a module, we merge any unit with lower importance than the highest previously seen into that unit, and average their importance scores. We continue appending the lowest-ranked units to the dropout list until its accumulated MACs reach $\frac{t \times r}{P_e} \times \text{MACs}_O$.

Sampling and Training Sub-networks. The updated dropout list is applied throughout the subsequent training epoch. Specifically, for each batch, we uniformly sample a truncation index s from 1 to the current length of the dropout list. All low-importance units ranked beyond the s-th position are dropped, forming a sampled sub-network. We then perform forward and backward passes on this sub-network and propagate the gradients back to the *SKD Network*.

This cycle lasts for P_e epochs, after which the dropout list is no longer updated. In the subsequent epochs, sub-networks continue to be sampled and trained, and the *SKD Network* is updated accordingly. Eventually, the *SKD Network* evolves into a scalable super-network. At deployment, a model of the desired size can be instantiated by discarding the low-ranked dropout units until the target MACs is reached.

Experiments

Experimental Setup

We conduct experiments primarily on the DeiT (Touvron et al. 2021) models, as well as Swin Transformers (Liu et al. 2021) pretrained on ImageNet-1k (Russakovsky et al. 2015). During the WPAC and PIAD stages, we randomly sample tiny training sets of size 1024 from ImageNet-1k to serve as proxy sets. We report the top-1 accuracy using a fixed random seed of 0. When reporting the standard deviation, we repeat each experiment 5 times using random seeds from 0 to 4. All experiments across datasets are conducted at a resolution of 224×224. Unless otherwise specified, our training settings follow those of DeiT. The *SKD Network* is obtained by applying WPAC to the DeiT pretrained model. During the subsequent PIAD stage, DeiT-B is trained for 150 epochs, and DeiT-S and DeiT-Ti are trained for 300 epochs to construct the *SKD Network*. We set $P_e = 50$ as the number of epochs for progressively constructing the dropout list.

Backbone	Method	MACs	Params	Epochs	Top-1
DeiT-S	Original	4.26 G	22.05 M	-	79.83
	IA-RED^2	3.60 G	-	90	79.30
	IA-RED^2	3.30 G	-	90	78.40
	SPViT	3.30 G	15.90 M	300	78.30
	RePaViT	3.20 G	16.70 M	300	78.90
	WDPruning	3.10 G	15.00 M	100	78.55
	SKD (Ours)	3.07 G	16.03 M	30	**79.42**
	IA-RED^2	2.90 G	-	90	78.60
	WDPruning	2.60 G	13.30 M	100	78.38
	RePaViT	2.50 G	13.20 M	300	77.00
	CP-ViT	2.46 G	-	30	**79.08**
	SKD (Ours)	2.43 G	12.78 M	30	78.71
DeiT-Ti	Original	1.08 G	5.72 M	-	72.14
	SPViT	1.00 G	4.80 M	300	70.70
	WDPruning	0.90 G	3.80 M	100	71.10
	SKD (Ours)	0.89 G	4.77 M	30	**71.40**
	RePaViT	0.80 G	4.40 M	300	69.40
	SKD (Ours)	0.79 G	4.25 M	30	**70.46**

Table 2: Comparison with network compression methods on ImageNet-1k based on DeiT-S and DeiT-Ti backbones.

Main Results

Comparison with Network Expansion Methods. To efficiently obtain models of varying sizes, methods like Albert (Lan et al. 2020), LiGO (Wang et al. 2023a), and Heur-LG (Wang et al. 2022) use weight decomposition and reuse to initialize different-sized models from pretrained weights. Auto-LG (Wang et al. 2023b) automates layer selection for inheritance. TLEG (Xia et al. 2024a) applies cross-layer weight fu-

Method	DeiT-B (81.8)			DeiT-S (79.8)			DeiT-Ti (72.1)			Swin-B (83.4)			Swin-S (83.2)			Swin-Ti (81.2)		
	1:4	2:4	3:4	1:4	2:4	3:4	1:4	2:4	3:4	1:4	2:4	3:4	1:4	2:4	3:4	1:4	2:4	3:4
Random	0.9	24.4	74.0	0.7	8.9	59.7	0.8	6.9	43.3	1.4	20.7	74.4	1.5	36.7	76.2	0.9	14.8	67.3
Magnitude	1.7	29.2	71.9	1.2	13.3	55.6	0.8	4.8	32.5	4.1	45.2	75.6	3.9	47.8	76.8	1.2	20.3	65.2
Taylor FO	6.0	52.4	78.1	3.9	39.4	74.8	3.8	32.7	63.5	15.4	65.6	80.9	11.7	64.8	80.1	13.6	60.0	77.0
Taylor SO	6.0	52.5	78.0	4.0	39.3	74.8	3.8	32.7	63.5	15.4	65.6	80.9	11.6	64.7	80.1	13.6	60.0	77.0
Hessian	5.1	52.0	78.2	3.8	48.2	74.8	2.8	32.2	63.1	15.5	64.9	80.7	11.4	63.8	80.1	14.3	60.7	77.2
WPAC (Ours)	**41.8**	**76.9**	**81.2**	**31.2**	**72.9**	**78.6**	**22.1**	**61.3**	**69.1**	**39.9**	**76.9**	**82.4**	**40.7**	**76.3**	**81.9**	**36.0**	**72.3**	**79.4**

Table 3: Comparison of direct evaluation results after pruning attention modules at different sparsity levels using various importance criteria. "Random" denotes random pruning; "FO" and "SO" refer to first- and second-order approximations.

Method	MACs	Params	Samples	Epochs	Top-1
DeiT-B	16.88 G	86.57 M	-	-	81.80
Network Compression					
RePaViT	12.70 G	65.30 M	1.2 M	300	81.40
IA-RED^2	11.80 G	-	1.2 M	90	80.30
CP-ViT	11.70 G	-	1.2 M	30	80.31
WDPruning	11.00 G	60.60 M	1.2 M	100	81.09
WDPruning	10.80 G	59.40 M	1.2 M	100	80.85
RePaViT	10.60 G	51.10 M	1.2 M	300	81.30
SKD (Ours)	10.57 G	54.55 M	1.2 M	30	**81.45**
WDPruning	9.90 G	55.30 M	1.2 M	100	80.76
LPViT	8.80 G	-	1.2 M	300	80.81
X-Pruner	8.50 G	-	1.2 M	130	81.02
SKD (Ours)	8.47 G	43.92 M	1.2 M	30	**81.24**
UVC	8.00 G	-	1.2 M	300	80.57
LPViT	7.92 G	-	1.2 M	300	80.55
SKD (Ours)	7.89 G	40.96 M	1.2 M	30	**80.89**
Few-shot Compression					
PRACTISE	14.43 G	-	500	2000	79.30
DC-DeiT	14.09 G	-	500	4000	81.26
SKD (Ours)	13.42 G	69.01 M	0	0	**81.42**

Table 4: Performance comparison with network compression methods on ImageNet-1k. All methods adopt DeiT-B as the base backbone. "Samples" and "Epochs" indicate the number of samples and epochs used for fine-tuning, respectively.

sion, SWS (Xia et al. 2024c) employs stage-wise weight sharing, and WAVE (Feng et al. 2025) uses weight templates to build models of varying sizes. Table 1 compares our method with these approaches. Our method requires no additional teacher model for distillation, and the extracted sub-networks achieve better performance without any fine-tuning.

Comparison with Network Compression Methods. The construction of scalable models remains an underexplored area. Therefore, we compare our method with a range of state-of-the-art model compression approaches, including RePaViT (Xu et al. 2025), IA-RED^2 (Pan et al. 2021), CP-ViT (Song et al. 2022), WDPruning (Yu et al. 2022a), LPViT (Xu et al. 2024), X-Pruner (Yu and Xiang 2023), UVC (Yu et al. 2022b), SPViT (He et al. 2024), as well as few-shot

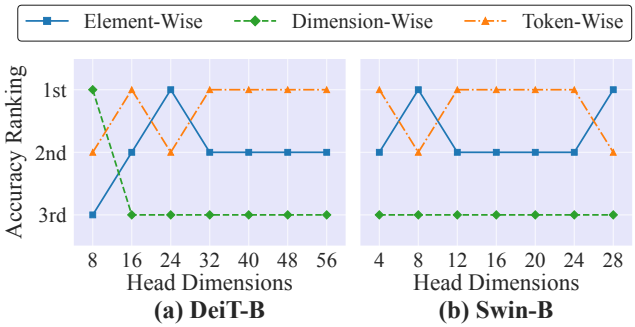
Method	Flowers-102	CUB-200-2011	Stanford Cars	CIFAR-10	CIFAR-100	Food-101	*Average*
Albert	94.1	72.4	87.2	96.5	78.5	83.0	*85.3*
LiGO	95.9	74.8	87.9	96.9	81.3	84.0	*86.8*
Heur-LG	69.1	48.0	51.2	95.1	72.8	76.8	*68.8*
Auto-LG	96.4	75.1	88.2	97.3	81.0	84.6	*87.1*
TLEG	93.7	72.6	87.2	97.2	80.2	84.9	*86.0*
WAVE	**96.9**	78.1	89.4	97.4	83.2	85.5	*88.4*
SKD (Ours)	94.1	**81.5**	**90.5**	**98.0**	**86.6**	**89.2**	***90.0***

Table 5: Comparison with network expansion methods on downstream datasets, using a DeiT-S backbone with half the original size. "*Average*" denotes the average accuracy.

compression methods such as PRACTISE (Wang and Wu 2023) and DC-ViT (Zhang, Zhou, and Wang 2024). Table 2 and Table 4 present the comparison results. Experimental findings demonstrate that our method can produce more compact models with comparable or even superior performance, while requiring significantly fewer fine-tuning resources.

Comparison with Importance Criteria. Many existing pruning methods rely on well-designed importance criteria to evaluate the significance of each weight. However, existing importance criteria for structured pruning evaluate each dimension independently, without the ability to jointly consider or integrate information across multiple dimensions. Table 3 compares our proposed WPAC with these criteria. The results demonstrate that WPAC effectively integrates information across dimensions, enabling more reliable pruning and benefiting subsequent model compression.

Comparison of Transferability. Table 5 presents the transferability of the sub-networks produced by our method on downstream tasks. We evaluate the transferability of the models on the following datasets: Flowers-102 (Nilsback and Zisserman 2008), CUB-200-2011 (Wah et al. 2011), Stanford Cars (Krause et al. 2013), CIFAR-10 (Krizhevsky, Hinton et al. 2009), CIFAR-100 (Krizhevsky, Hinton et al. 2009), and Food-101 (Bossard, Guillaumin, and Van Gool 2014).



(a) DeiT-B (b) Swin-B

Figure 4: Comparison of weighting strategies with different granularities across varying numbers of retained dimensions.

Tokens	Weight	DeiT-B	DeiT-S	DeiT-Ti
Class	1.0	65.1±0.15	46.2±0.20	30.8±0.28
Rand(1)	1.0	73.3±0.19	64.5±0.54	47.4±1.38
Rand(2)	1.0	75.7±0.17	71.0±0.11	58.7±0.10
Rand(5)	1.0	Ill-Cond.	71.4±0.07	59.4±0.28
All	1.0	Ill-Cond.	Ill-Cond.	Ill-Cond.
All	0.01	76.1±0.06	72.2±0.07	60.0±0.05
All	Imp.	**76.9**±0.05	**72.9**±0.07	**61.2**±0.07

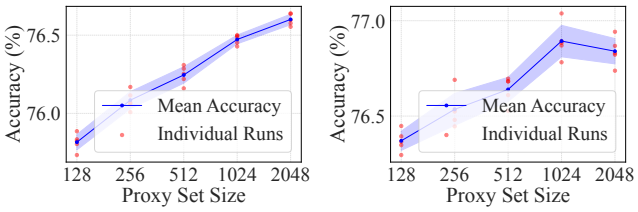
Table 6: WPAC results for different token selection and weighting strategies when retaining 50% dimensions. "Class" uses the class token, "Rand(x)" randomly selects x tokens, and "All" uses all tokens per sample. "Imp.": token-wise importance weighting. "Ill-Cond.": eigen decomposition failure.

Compared to other efficient approaches for obtaining models of different sizes, the models generated by our method exhibit superior transfer performance across various datasets.

Ablation and Analysis

Analysis of Weighting Strategies with Varying Granularities in WPAC. Given the element-wise score $\Theta_{\text{TE}} \in \mathbb{R}^{n \times d}$ obtained via Taylor expansion, aggregating along different axes yields token-wise and dimension-wise importance scores. Figure 4 presents the results and rankings of the three weighting strategies. Experimental results show that token-wise weighting is more effective at preserving information and provides greater robustness compared to the others.

Analysis of Per-Sample Token Sampling Methods and Weighting Effectiveness in WPAC. Selecting appropriate tokens for PCA is crucial for preserving essential features. Table 6 presents results for different token sampling strategies on each sample in the proxy set: using only the class token, randomly selecting a subset of tokens, using all tokens, and applying different weighting schemes. Relying solely on the class token limits the ability to recover information from other tokens, leading to the poorest performance. Randomly sampling several tokens introduces more sample information into PCA, but using too many tokens can result in an ill-conditioned covariance matrix, making eigen decomposition infeasible. In contrast, our WPAC method utilizes all tokens with importance-based weighting, achieving the best results.



(a) DeiT-B (b) Swin-B

Figure 5: Impact of proxy set size on WPAC performance. Each size is evaluated using 5 random seeds, and the results reflect performance with head dimensions reduced by half.

Method	DeiT-S (79.8)				DeiT-Ti (72.1)			
	4:12	6:12	8:12	10:12	4:12	6:12	8:12	10:12
Baseline	1.2	8.9	39.1	71.2	1.4	6.9	25.8	56.2
B+CD	7.3	20.0	34.4	46.5	2.0	8.8	23.1	36.3
B+WCD	34.2	54.4	64.7	73.8	29.4	42.2	49.1	63.8
B+LD	39.7	60.2	68.6	74.1	34.5	45.4	57.1	62.5
B+PIAD	**70.6**	**76.2**	**78.2**	**79.0**	**61.4**	**65.8**	**68.6**	**70.0**

Table 7: Direct evaluation of sub-networks of varying sizes extracted from networks trained with different dropout strategies. "x:12": the subnetwork-to-original size ratio. "CD": channel dropout. "WCD": weighted CD. "LD": layerdrop.

Analysis of the Impact of Proxy Dataset Size in WPAC. Figure 5 illustrates the results of applying PCA with proxy sets of varying sizes. The experiments demonstrate that even a small number of samples is sufficient to obtain an accurate principal component projection. This allows the pre-trained parameters to be effectively projected, concentrating the knowledge into a reduced set of transformed dimensions.

Effectiveness of the PIAD. Table 7 shows the impact of different dropout strategies on sub-network performance. Channel Dropout (Tompson et al. 2015) and LayerDrop (Fan, Grave, and Joulin 2020) tend to favor sub-networks with sparsity close to the preset dropout rate and offer limited sub-network diversity. Weighted Channel Dropout (Hou and Wang 2019) further concentrates knowledge into important dimensions by assessing their significance. However, all these methods suffer from uneven sub-network sampling across sizes. In contrast, our PIAD method effectively condenses knowledge into fewer parameters and forms a hierarchical structure, enabling incrementally expandable networks.

Conclusion

We propose constructing a Stratified Knowledge-Density (SKD) super-network for ViTs, enabling efficient extraction of sub-networks with maximal knowledge retention to meet diverse deployment needs. WPAC condenses knowledge into key dimensions, laying a solid foundation for compression, while PIAD further enhances knowledge stratification across weight groups to form a stratified super-network.

Acknowledgments

This research was supported by the Jiangsu Science Foundation (BG2024036, BK20243012), the National Science Foundation of China (62125602, U24A20324, 92464301), CAAI-Lenovo Blue Sky Research Fund, and the Fundamental Research Funds for the Central Universities (2242025K30024).

References

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101–mining discriminative components with random forests. In *ECCV*.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2020. Once for All: Train One Network and Specialize it for Efficient Deployment. In *ICLR*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fan, A.; Grave, E.; and Joulin, A. 2020. Reducing Transformer Depth on Demand with Structured Dropout. In *ICLR*.

Fang, G.; Ma, X.; Song, M.; Mi, M. B.; and Wang, X. 2023. Depgraph: Towards any structural pruning. In *CVPR*.

Feng, F.; Xie, Y.; Wang, J.; and Geng, X. 2025. WAVE: Weight Templates for Adaptive Initialization of Variable-sized Models. In *CVPR*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *NeurIPS*.

He, H.; Cai, J.; Liu, J.; Pan, Z.; Zhang, J.; Tao, D.; and Zhuang, B. 2024. Pruning self-attentions into convolutional layers in single path. *TPAMI*.

He, Y.; Liu, P.; Wang, Z.; Hu, Z.; and Yang, Y. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*.

Hou, S.; and Wang, Z. 2019. Weighted channel dropout for regularization of deep convolutional neural network. In *AAAI*.

Hu, H.; Peng, R.; Tai, Y.-W.; and Tang, C.-K. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*.

Hua, C.; Xu, Q.; Bao, S.; Yang, Z.; and Huang, Q. 2024. ReconBoost: Boosting Can Achieve Modality Reconcilement. In *ICML*.

Hua, C.; Xu, Q.; Yang, Z.; Wang, Z.; Bao, S.; and Huang, Q. 2025. OpenworldAUC: Towards Unified Evaluation and Optimization for Open-world Prompt Tuning. In *ICML*.

Jose, C.; Moutakanni, T.; Kang, D.; Baldassarre, F.; Darcet, T.; Xu, H.; Li, D.; Szafraniec, M.; Ramamonjisoa, M.; Oquab, M.; et al. 2025. Dinov2 meets text: A unified framework for image-and pixel-level vision-language alignment. In *CVPR*.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *ICCV workshops*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Toronto, ON, Canada.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal brain damage. In *NeurIPS*.

Lee, J.; Park, S.; Mo, S.; Ahn, S.; and Shin, J. 2021. Layer-adaptive sparsity for the magnitude-based pruning. In *ICLR*.

Li, G.; Qian, C.; Jiang, C.; Lu, X.; and Tang, K. 2018. Optimization based Layer-wise Magnitude-based Pruning for DNN Compression. In *IJCAI*.

Li, L.; Qi, L.; and Geng, X. 2025. One-Shot Knowledge Transfer for Scalable Person Re-Identification. In *ICCV*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*.

Luo, J.-H.; and Wu, J. 2017. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791*.

Mao, Y.; Wang, Y.; Wu, C.; Zhang, C.; Wang, Y.; Zhang, Q.; Yang, Y.; Tong, Y.; and Bai, J. 2020. LadaBERT: Lightweight Adaptation of BERT through Hybrid Model Compression. In *COLING*.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In *CVPR*.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2017. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *ICLR*.

Muralidharan, S.; Turuvekere Sreenivas, S.; Joshi, R.; Chochowski, M.; Patwary, M.; Shoeybi, M.; Catanzaro, B.; Kautz, J.; and Molchanov, P. 2024. Compact language models via pruning and knowledge distillation. In *NeurIPS*.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *ICVGIP*.

Noach, M. B.; and Goldberg, Y. 2020. Compressing pre-trained language models by matrix decomposition. In *AACL and IJCNLP*.

Pan, B.; Panda, R.; Jiang, Y.; Wang, Z.; Feris, R.; and Oliva, A. 2021. IA-RED ^{2}: Interpretability-aware redundancy reduction for vision transformers. In *NeurIPS*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV*.

Song, Z.; Xu, Y.; He, Z.; Jiang, L.; Jing, N.; and Liang, X. 2022. Cp-vit: Cascade vision transformer pruning via progressive sparsity prediction. *arXiv preprint arXiv:2203.04570*.

Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; and Bregler, C. 2015. Efficient object localization using convolutional networks. In *CVPR*.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wang, G.-H.; and Wu, J. 2023. Practical network acceleration with tiny sets. In *CVPR*.

Wang, P.; Panda, R.; Hennigen, L. T.; Greengard, P.; Karlinsky, L.; Feris, R.; Cox, D. D.; Wang, Z.; and Kim, Y. 2023a. Learning to Grow Pretrained Models for Efficient Transformer Training. In *ICLR*.

Wang, Q.; Yang, X.; Lin, S.; Wang, J.; and Geng, X. 2023b. Learngene: Inheriting condensed knowledge from the ancestry model to descendant models. *arXiv preprint arXiv:2305.02279*.

Wang, Q.-F.; Geng, X.; Lin, S.-X.; Xia, S.-Y.; Qi, L.; and Xu, N. 2022. Learngene: From open-world to your learning task. In *AAAI*.

Wang, Z.; Wohlwend, J.; and Lei, T. 2020. Structured Pruning of Large Language Models. In *EMNLP*.

Xia, S.; Zhang, M.; Yang, X.; Chen, R.; Chen, H.; and Geng, X. 2024a. Transformer as linear expansion of learngene. In *AAAI*.

Xia, S.; Zu, Y.; Yang, X.; and Geng, X. 2024b. Initializing variable-sized vision transformers from learngene with learnable transformation. *NeurIPS*.

Xia, S.-Y.; Zhu, W.; Yang, X.; and Geng, X. 2024c. Exploring learngene via stage-wise weight sharing for initializing variable-sized models. In *IJCAI*.

Xu, K.; Wang, Z.; Chen, C.; Geng, X.; Lin, J.; Yang, X.; Wu, M.; Li, X.; and Lin, W. 2024. Lpvit: Low-power semi-structured pruning for vision transformers. In *ECCV*.

Xu, X.; Li, Y.; Chen, Y.; Liu, J.; and Wang, S. 2025. RePaViT: Scalable Vision Transformer Acceleration via Structural Reparameterization on Feedforward Network Layers. In *ICML*.

Yu, F.; Huang, K.; Wang, M.; Cheng, Y.; Chu, W.; and Cui, L. 2022a. Width & depth pruning for vision transformers. In *AAAI*.

Yu, H.; and Wu, J. 2023. Compressing transformers: features are low-rank, but weights are not! In *AAAI*.

Yu, J.; Yang, L.; Xu, N.; Yang, J.; and Huang, T. 2019. Slimmable neural networks. In *ICLR*.

Yu, L.; and Xiang, W. 2023. X-pruner: explainable pruning for vision transformers. In *CVPR*.

Yu, S.; Chen, T.; Shen, J.; Yuan, H.; Tan, J.; Yang, S.; Liu, J.; and Wang, Z. 2022b. Unified Visual Transformer Compression. In *ICLR*.

Zhang, H.; Zhou, Y.; and Wang, G.-H. 2024. Dense vision transformer compression with few samples. In *CVPR*.