This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

WAVE: Weight Templates for Adaptive Initialization of Variable-sized Models

Fu Feng[1,2*] Yucheng Xie[1,2*] Jing Wang[1,2†] Xin Geng[1,2†]
[1]School of Computer Science and Engineering, Southeast University, Nanjing, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
{fufeng, xieyc, wangjing91, xgeng}@seu.edu.cn

Abstract

The growing complexity of model parameters underscores the significance of pre-trained models. However, deployment constraints often necessitate models of varying sizes, exposing limitations in the conventional pre-training and fine-tuning paradigm, particularly when target model sizes are incompatible with pre-trained ones. To address this challenge, we propose WAVE, a novel approach that reformulates variable-sized model initialization from a multi-task perspective, where initializing each model size is treated as a distinct task. WAVE employs shared, size-agnostic weight templates alongside size-specific weight scalers to achieve consistent initialization across various model sizes. These weight templates, constructed within the Learngene framework, integrate knowledge from pre-trained models through a distillation process constrained by Kronecker-based rules. Target models are then initialized by concatenating and weighting these templates, with adaptive connection rules established by lightweight weight scalers, whose parameters are learned from minimal training data. Extensive experiments demonstrate the efficiency of WAVE, achieving state-of-the-art performance in initializing models of various depth and width. The knowledge encapsulated in weight templates is also task-agnostic, allowing for seamless transfer across diverse downstream datasets. Code will be made available at https://github.com/fu-feng/WAVE.

1. Introduction

With the exponential increase in model parameters, the demands for time, energy, and computational resources during training have escalated significantly [1, 31]. Training a model from scratch for new tasks, especially those with Transformer architectures like Vision Transformers (ViTs) [8], is inefficient. As a result, fine-tuning pre-trained models has become the preferred approach [21, 23, 39]. However, in practical applications, model deployment is often constrained by various factors, such as memory usage, processing power, and response time [46], necessitating models of *variable sizes*. In contrast, most off-the-shelf pre-trained models, such as the 12-layer ViT-B models [30], are available in only a limited set of sizes. As a result, target models that do not align with these predefined sizes require re-pretraining on large datasets before deployment.

Several methods [35, 44] have explored leveraging pre-

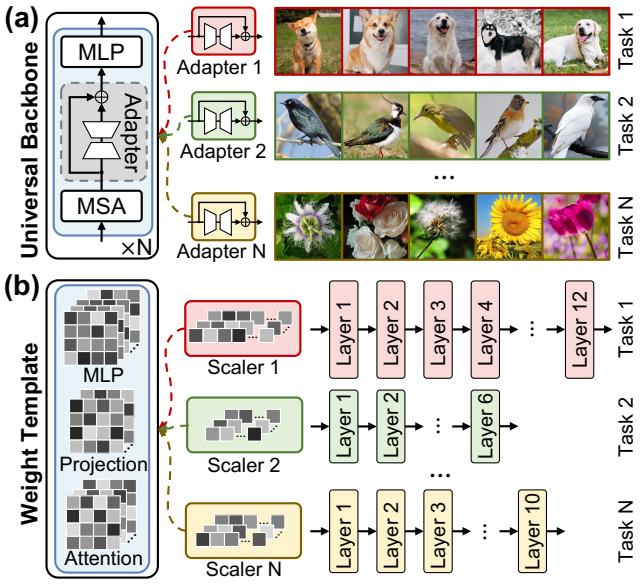


Figure 1. (a) Multi-task learning typically relies on a universal backbone with ***task-agnostic*** knowledge, complemented by a few trainable adapters for ***task-specific*** adaptation. (b) WAVE reformulates variable-sized model initialization as a multi-task problem by treating the initialization of each model size as a distinct task. By doing so, WAVE employs shared weight templates encapsulating ***size-agnostic*** knowledge, along with a few trainable weight scalers for ***size-specific*** initialization across various model sizes.

*Equal contribution
†Corresponding authors

trained models to initialize target models with mismatched sizes by selecting or transforming pre-trained weight matrices. However, these methods often disrupt the structured knowledge embedded in original models or introduce excessive random parameters, limiting their effectiveness. By abstracting the initialization of models with specific sizes as independent tasks, the challenge of initializing variable-sized models can be reformulated as a multi-task problem. Multi-task learning typically requires only a minimal number of learnable parameters for rapid adaptation to new tasks (e.g., Lora [16]), enabled by a universal backbone, encapsulating task-agnostic knowledge [32], as illustrated in Figure 1. Motivated by this, we can ask: *Can a set of universal weight templates enable the initialization of variable-sized models with only a small number of learnable parameters?*

Recently, *Learngene* [9, 37] has emerged as a novel framework for model initialization based on pre-trained models, referred to as ancestry models. Unlike existing methods that directly utilize pre-trained models for initialization [35, 44], *Learngene* introduces an additional step by integrating and encapsulating pre-trained knowledge into compact neural fragments, termed learngenes. This ***once-for-all*** process extracts size-agnostic knowledge that enables the initialization of target models with variable sizes.

Building on the *Learngene* framework, we propose WAVE, a novel and flexible approach for model initialization. WAVE employs shared *weight templates* that encapsulate size-agnostic knowledge as learngenes. These templates structurally integrate knowledge from pre-trained models through a distillation process facilitated by an auxiliary model [40]. Downstream models initialize their weight matrices by concatenating and weighting these templates using the Kronecker product, with adaptive connection rules determined by lightweight weight scalers containing only a few thousand parameters. During initializing, only weight scalers need to be tailored to the target model sizes, and their parameters can be efficiently trained with limited data.

Integrating pre-trained knowledge into weight templates is a ***once-for-all*** process requiring 150 epochs on ImageNet-1K, after which model initialization involves training only lightweight scalers while keeping the templates fixed. Extensive experiments validate WAVE's state-of-the-art performance in initializing variable-sized models across diverse datasets. Notably, with only 3.3% of pre-trained model (i.e., ancestry model) parameters transferred, models initialized by WAVE and trained for just 10 epochs outperform those directly pre-trained for 150 epochs, achieving $15n\times$ computational savings for n variable-sized models. Visualizations further reveal that the size-agnostic knowledge in weight templates is highly structured, consistent with patterns observed in pre-trained models [33, 40].

Our main contributions are as follows: 1) We propose WAVE, a novel method that approaches model initialization from a multi-task perspective. WAVE enhances flexibility and efficiency by constructing shared weight templates, termed learngenes, combined with lightweight weight scalers trained to initialize target models. 2) We introduce the first comprehensive benchmark for learngenes, systematically evaluating their initialization and transfer capabilities. 3) Extensive experiments validate the effectiveness of WAVE, which achieves state-of-the-art performance compared to other model initialization and learngene methods. Moreover, we demonstrate that the knowledge encapsulated in weight templates is highly structured, enabling efficient and robust initialization.

2. Related Work

2.1. Model Initialization

Model initialization is crucial for the convergence speed and final performance of neural networks [2, 17]. Traditional methods typically apply pre-set rules to random parameters [6, 12]. The advent of pre-trained models has advanced initialization methods, with fine-tuning emerging as the standard practice for many applications [47]. However, the fixed sizes and architectures of pre-trained models often limit their flexibility, highlighting the need for more versatile initialization methods.

Several methods have explored direct model initialization through observed parameter patterns or hypernetworks. Mimetic Initialization [33] leverages parameter patterns identified from attention heads in pre-trained models to initialize new ones, while GHN3 [18] predicts target model parameters using a graph hypernetwork. Other methods transfer parameters from pre-trained models to target models, such as Weight Selection [44], which transfers selected weights from larger models to smaller ones, and LiGO [35], which scales smaller models to initialize larger ones.

Despite these advancements, these methods often incur high transfer costs and risk inefficiency or negative transfer due to parameter mismatches. Our WAVE overcomes these limitations by constructing shared weight templates, which integrate pre-trained knowledge to improve transferability and scalability across models of varying sizes.

2.2. Learngene

Learngene represents an innovative approach to model initialization and knowledge transfer, inspired by the biological process where our brains are initialized by evolved genes rather than through random initialization [9, 37, 45]. Learngenes are compact neural network fragments that encapsulate only size-agnostic knowledge, making them lightweight and easily transferable, much like genes in nature [10]. This distinguishes learngenes from mainstream methods, such as direct fine-tuning of pre-trained mod-

els [4, 47] or transforming pre-trained models to adapt to variable-sized models [35, 41, 44].

Current learngene methods primarily focus on a layer-by-layer basis. For instance, Heur-LG [36] selects layers with minimal gradient changes as learngenes during continuous learning, while Auto-LG [37] considers that only layers with representations similar to target networks should be transferred. TLEG [40] characterizes model initialization as a linear combination of two layers. Additionally, KIND [43] and FINE [42] apply singular value decomposition (SVD) to each weight matrix, enabling the sharing of singular vectors for initializing models of varying sizes. However, existing learngenes only address depth variations.

In contrast, WAVE introduces a novel form of learngenes, overcoming layer-based limitations by treating each weight matrix as a concatenation and weighted combination of weight templates. This method enhances the flexibility of learngenes by incorporating weight scalers, thus enabling scalable initialization for variable-sized models.

3. Methods

WAVE is an innovative *Learngene* method that constructs shared weight templates, termed learngenes, to initialize models of variable sizes. This section begins by introducing how weight templates and scalers reconstruct neural network parameters using Kronecker products. It then details the integration of pre-trained knowledge into structured, size-agnostic knowledge within weight templates, followed by the procedure for initializing models of variable sizes using these weight templates.

3.1. Preliminaries

The Vision Transformer (ViT) comprises an encoder stacking L layers, each containing a multi-head self-attention (MSA) mechanism and a multi-layer perception (MLP). In each layer, a single-head self-attention A_i is composed of a query Q_i, key K_i and value $V_i \in \mathbb{R}^{N\times d}$ with parameter matrices W_q^i, W_k^i and $W_v^i \in \mathbb{R}^{D\times d}$, performing self-attention:

$$A_i = \text{softmax}(\frac{Q_i K_i^\top}{\sqrt{d}})V_i\,,\ A_i \in \mathbb{R}^{N\times d} \tag{1}$$

where N is the number of patches, D is the dimensional of patch embeddings, and d is the projected dimension for each attention head.

MSA processes information from h attention heads and uses a weight matrix W_o to project the concatenated outputs:

$$\text{MSA} = \text{concat}(A_1, A_2, ..., A_h)W_o\,,\ W_o \in \mathbb{R}^{hd\times D} \tag{2}$$

In the implementation of MSA, the parameter matrices W_q, W_k and $W_v \in \mathbb{R}^{D\times d}$ for h attention heads are typically combined into a single parameter matrix $W_{qkv} \in \mathbb{R}^{D\times 3hd}$.

The MLP consists of two linear transformations, $W_{in} \in \mathbb{R}^{D\times D'}$ and $W_{out} \in \mathbb{R}^{D'\times D}$, with a GELU [14] activation, computed as:

$$\text{MLP}(x) = \text{GELU}(xW_{in} + b_1)W_{out} + b_2 \tag{3}$$

where b_1 and b_2 are the bias for the linear transformations, and D' is the dimension of the hidden layers.

3.2. Weight Template

Foundations for the Existence of Weight Templates. Evidence suggests the existence of shared templates among diverse weight matrices. For instance, Template Kernels [22] demonstrate the ability to generate diverse adapters for various tasks, while the Universal Template [32] enhances generalization in few-shot learning scenarios. Additionally, studies on pre-trained Vision Transformers (ViTs) provide additional insights. Mimetic Initialization [33] reveals significant correlations among the weights of self-attention layers, and TLEG [40] identifies linear relationships among the weights of different layers.

Building on these insights, WAVE utilizes shared weight templates encapsulating size-agnostic knowledge as learngenes. The initialization of variable-sized models is flexibly achieved by applying size-specific weight scalers to these templates via Kronecker products, as illustrated in Figure 2.

The main weight matrices in ViTs with L layers are represented as $\theta = \{W_{qkv}^{(1\sim L)}, W_o^{(1\sim L)}, W_{in}^{(1\sim L)}, W_{out}^{(1\sim L)}\}$[1]. These matrices can be reconstructed using a unified set of shared weight templates $\mathcal{T} = \{T_{qkv}^{(1\sim N_{qkv})}, T_o^{(1\sim N_o)}, T_{in}^{(1\sim N_{in})}, T_{out}^{(1\sim N_{out})}\}$[2]. To effectively leverage the structured information in weight templates, WAVE initializes the corresponding weight matrices by concatenating and weighting these templates using the Kronecker product, facilitated by Layer-Wise Scaling Kernels [22]. Let $W_\star^{(l)}$ represent any weight matrix in layer l, and $T_\star^{(t)}$ denote the t-th corresponding weight template, where $\star \in \{qkv, o, in, out\}$. The $W_\star^{(l)}$ is initialized as follows:

$$W_\star^{(l)} = \sum_{t=1}^{N_\star} T_\star^{(t)} \otimes S_\star^{(l,t)} \tag{4}$$

$$W_\star^{(l)} \in \mathbb{R}^{m_1\times m_2}\,,\ T_\star^{(t)} \in \mathbb{R}^{w_1\times w_2}\,,\ S_\star^{(l,t)} \in \mathbb{R}^{\frac{m_1}{w_1}\times\frac{m_2}{w_2}}$$

where $\otimes$ denotes the Kronecker product. m_1 and m_2 represent the dimensions of weight matrices, while w_1 and w_2 determine the size of weight templates. $S_\star^{(l,t)}$ is the

[1] $W_{qkv}^{(1\sim L)}$ denotes the set $\{W_{qkv}^{(1)}, W_{qkv}^{(2)}, \ldots, W_{qkv}^{(L)}\}$. Similar notations throughout the paper follow this convention.

[2] Parameters in normalization, bias and image relative position encoding (irpe) account for a small part and have little impact when randomly initialized, as detailed in Table 5.

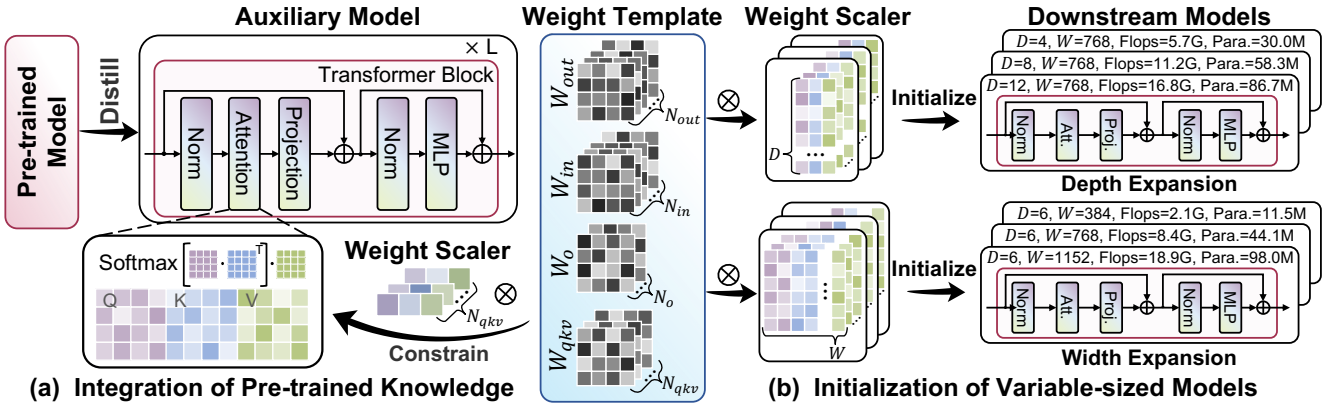


Figure 2. (a) The knowledge in pre-trained models is integrated into structured knowledge within weight templates through distillation, assisted by an auxiliary model constrained by the rules in Eq.(4), where $\otimes$ is the Kronecker product. (b) For initializing models of variable sizes, only the corresponding weight scalers are initialized according to the target model size. These scalers are trained with a small amount of data to learn connection rules of templates, while the weight templates remain frozen to retain the structured knowledge.

weight scaler performing the Kronecker product with $T_\star^{(t)}$. The set of all trainable weight scalers is denoted as $\mathcal{S} = \{S_{qkv}^{(1\sim L,1\sim N_{qkv})}, S_o^{(1\sim L,1\sim N_o)}, S_{in}^{(1\sim L,1\sim N_{in})}, S_{out}^{(1\sim L,1\sim N_{out})}\}$. To succinctly represent the Kronecker product rule in Eq. (4), we abbreviate it as:

$$\theta = \mathcal{T} \otimes \mathcal{S} \tag{5}$$

We also demonstrate that layer-based learngenes, including Heur-LG [36], Auto-LG [36], and TLEG [40], are special cases of WAVE. A detailed proof is provided in Appendix A.

3.3. Integration of Pre-trained Knowledge

The weight matrices of models can be initialized using weight templates and scalers via Kronecker product, enabling robust initialization by effectively leveraging the structured knowledge encapsulated within the weight templates. Next, we outline the process of integrating knowledge from pre-trained models into these structured weight templates, following the rules defined in Eq.(4).

To transfer knowledge from pre-trained models into weight templates, we employ knowledge distillation [15], a widely-used method in knowledge transfer. Since weight templates themselves lack the capacity to learn from data independently, we construct an auxiliary model $f_{\text{aux}}(\theta_{\text{aux}})$ [40], whose parameters are derived from weight templates and scalers under the rule defined in Eq.(4). The parameter updates of the auxiliary model are achieved by updating weight templates and scalers (see Algorithm 1 for details), following the optimization objective:

$$\arg\min_{\mathcal{T},\mathcal{S}_{\text{aux}}} \mathcal{L}(f_{\text{aux}}(\theta_{\text{aux}}, x), y), \quad \text{s.t.}\ \theta_{\text{aux}} = \mathcal{T} \otimes \mathcal{S}_{\text{aux}} \tag{6}$$

where $\mathcal{T}$ is the target set of shared weight templates for universally initializing variable-sized models. $\mathcal{S}_{\text{aux}}$ represents the set of weight scalers specific to the auxiliary model f_{aux}, which are discarded after the knowledge integration.

In this manner, the auxiliary model serves as both a medium for transferring knowledge from pre-trained models to weight templates, and a bottleneck for filtering out unstructured knowledge lacking size-agnostic properties required for initializing variable-sized downstream models.

The learning process for weight templates adopts soft distillation loss and classification loss, computed as:

$$\mathcal{L} = \text{KL}(z_{\text{pre}} || z_{\text{aux}}) + \text{CE}(z_{\text{aux}}, y) \tag{7}$$

where $\text{KL}(\cdot,\cdot)$ denotes the KL-divergence loss, with z_{pre} and z_{aux} being the logits outputs of the pre-trained model and auxiliary model. $\text{CE}(\cdot,\cdot)$ denotes the cross-entropy loss, and y is the ground-truth label of the image. Note that $\mathcal{L}$ is exclusively used to update the parameters of weight templates and weight scalers (Eq. (6)), while the parameters of the auxiliary model are indirectly updated by being reconstructed under the rule in Eq. (4) at each iteration.

3.4. Initialization of Variable-sized Models

We have successfully constructed weight templates by integrating knowledge from the pre-trained model. Next, we address the process of initializing variable-sized models using these weight templates $\mathcal{T}$. Existing learngene methods typically rely on predetermined strategies, such as linear expansion [40] or manual selection [41], which considerably limit their adaptability to models of variable sizes.

Drawing inspiration from trainable adapters in addressing multi-tasking challenges, we ***freeze*** the weights of weight templates $\mathcal{T}$ and then construct the weight scalers $\mathcal{S}$ tailored to the target model size, thereby enabling adaptation to networks of varying sizes. Further details on the initialization of weight scalers are provided in Appendix B.

Specifically, to initialize a target model f_{tar} with parameters θ_{tar}, each weight matrix $W^{(l)}_{\star,\text{tar}} \in \mathbb{R}^{m_1 \times m_2}$ in θ_{tar} requires an associated set of weight scalers. Their dimensions, determined by the size of the weight matrix being initialized and the associated weight templates, are given by $S^{(l,1\sim N_\star)}_{\star,\text{tar}} \in \mathbb{R}^{N_\star \times \frac{m_1}{w_1} \times \frac{m_2}{w_2}}$ according to Eq. (4), where w_1 and w_2 are the dimensions of the weight templates $T^{(1\sim N_\star)}_{\star} \in \mathbb{R}^{N_\star \times w_1 \times w_2}$.

A small amount of data is then used to train the weight scalers $\mathcal{S}_{\text{tar}}$ with the following objective:

$$\arg\min_{\mathcal{S}_{\text{tar}}} \mathcal{L}(f_{\text{tar}}(\theta_{\text{tar}}, x), y), \quad \text{s.t. } \theta_{\text{tar}} = \mathcal{T} \otimes \mathcal{S}_{\text{tar}} \tag{8}$$

Given the limited parameter count of the scalers (typically a few thousand), convergence is typically achieved within a few hundred iterations (around 0.16 epochs), making the computational cost negligible. The model initialization process is completed once the training of weight scalers is finalized, after which subsequent training proceeds as usual without imposing additional constraints.

4. Experiments

4.1. Benchmark for Evaluating Learngenes

Learngene offers an innovative approach to model initialization and knowledge transfer. In this study, we propose a benchmark to comprehensively evaluate the efficiency of learngenes in initializing variable-sized models and their transferability across diverse datasets.

Datasets. We use ImageNet-1K [7] as the primary dataset for knowledge integration and model evaluation. Additional datasets include Oxford Flowers, CUB-200-2011, Stanford Cars, CIFAR-10, CIFAR-100, Food-101, and iNaturalist-2019, providing a comprehensive benchmark to assess learngenes and other model initialization methods in terms of initialization flexibility, learning capacity, and knowledge transferability. Further details are provided in Appendix C.4.

Network Structures. We adopt DeiT [30] as the base architecture and explore variations in both depth and width. For depth variations, we construct models with depths ranging from D = 4, 6, 8, 10, to 12, with deeper configurations (D = 24 and 36) provided in Appendix D.2. For width variations, we adjust the number of attention heads while proportionally scaling the projection and MLP layers to target dimension (W = 384, 576, 768, 1152, 1536). These configurations cover a wide range of ViT sizes, providing a thorough evaluation of learngene initialization.

Measurements. Learngene performance is primarily evaluated through Top-1 accuracy across various model sizes and datasets. Additional considerations include extra training epochs and transferred parameters to highlight initialization efficiency.

4.2. Experimental Setup

To integrate knowledge from pre-trained models, we follow the basic settings of TLEG [40], employing the pre-trained Levit-384 [13] and performing distillation for 150 epochs. Experiments with larger pre-trained models are provided in Appendix D.1. In our experiments, we adopt DeiT-Ti, -S and -B (each with 12 layers) as auxiliary models and unify the sizes of weight templates across different components, including T_{qkv}, T_o, T_{in} and T_{out}. Specifically, we set three basic sizes of weight templates, including 192×192, 384×384, and 768×768, to accommodate basic scale variations (corresponding to DeiT-Ti, DeiT-S, and DeiT-B). Additional details are provided in Appendix C.2 and C.3.

5. Results

5.1. Performance of Initializing Variable-sized Models

In real-world scenarios, deployment constraints often necessitate models of varying sizes to meet specific needs. To demonstrate this, we construct 15 downstream models, each varying in depth and width, and compare WAVE against SOTA model initialization methods, categorized as follows: (1) Direct Initialization: Methods like He-Init [6], Mimetic Init [33], and GHN-3 [18] use prior knowledge or hypernetworks to direct initialize models. (2) Transfer Initialization: Techniques like Weight Selection [44], LiGO [35], and Share Init [20] transfer pre-trained knowledge to the target model. (3) Learngene Initialization: Methods such as Heur-LG [36], Auto-LG [37], and TLEG [40] use neural network fragments for model initialization.

5.1.1. Depth Expansion

Table 1 presents the results of initializing models with variable depths (with deeper models detailed in Appendix D.2). Remarkably, WAVE demonstrates superior performance, significantly outperforming other learngene and model initialization methods across all model scales.

Transferring pre-trained knowledge for target initialization is more effective and straightforward than direct initialization, with Share Init and LiGO significantly outperforming Mimetic and GHN-3. While LiGO enables smaller pre-trained models to initialize larger ones, it introduces numerous random parameters during weight transformation, highlighting challenges posed by size mismatches (see Table 4). In contrast, Share Init improves performance by training and reusing specific layers, emphasizing the importance of rule-based knowledge integration before initialization.

However, existing learngene-based methods impose overly strict constraints, limiting learngenes to a few layers without reserving adaptive designs for diverse model sizes. In contrast, WAVE treats each model's initialization as a distinct task, extracting a unified set of weight templates

Table 1. Performance of initializing models with variable depths on ImageNet-1K. "Para.(M)" represents the number of parameters for each model size (row) and the *average* number of parameters transferred during model initialization (column). "Epoch" refers to additional epochs required for knowledge integration or network pre-training. "—" indicates failure to initialize the corresponding model size, while "N/A" denotes cases where a metric is not applicable. All models are trained for 10 epochs after initialization.

				W_{192} (DeiT-Ti)						W_{384} (DeiT-S)						W_{768} (DeiT-B)					
				L_4	L_6	L_8	L_{10}	L_{12}		L_4	L_6	L_8	L_{10}	L_{12}		L_4	L_6	L_8	L_{10}	L_{12}
	Methods	Epoch	Para.	2.2	3.1	4.0	4.9	5.8	Para.	7.9	11.5	15.0	18.6	22.2	Para.	29.9	44.1	58.3	72.5	86.7
Direct	He-Init [6]	N/A	0	34.7	40.6	43.7	46.8	48.3	0	42.2	49.4	52.1	53.7	55.5	0	47.9	53.1	54.4	55.0	56.7
Direct	Mimetic [33]	N/A	0	35.1	40.2	43.2	46.3	48.1	0	43.3	49.1	53.0	54.1	55.6	0	50.2	54.3	56.5	58.5	58.6
Direct	GHN-3 [18]	N/A	0	40.9	45.0	46.6	49.1	48.9	0	45.4	49.0	50.2	52.3	53.2	0	49.5	52.5	53.8	54.2	54.3
Tran.	Share Init [20]	150	0.8	55.2	59.8	62.5	64.3	65.3	2.5	65.0	69.7	71.7	72.7	73.3	8.6	71.7	75.3	76.4	77.4	77.6
Tran.	LiGO [35]	N/A	2.2	—	59.0	60.2	59.8	60.9	7.9	—	68.6	69.9	69.7	70.0	29.9	—	74.2	74.4	75.3	75.4
Learngene	Heur-LG [36]	N/A	1.7	41.5	47.4	50.5	53.5	55.5	6.1	52.3	57.3	61.7	64.4	65.9	22.8	60.5	68.7	72.2	73.6	74.0
Learngene	Auto-LG [37]	50	2.2	52.4	61.8	64.6	65.9	66.8	7.9	63.2	70.5	72.2	73.3	73.8	29.9	60.9	70.0	72.4	73.5	73.8
Learngene	TLEG [40]	150	1.3	55.0	60.5	62.9	64.4	65.4	4.3	65.4	70.5	72.1	73.2	73.8	15.7	71.6	74.9	76.2	77.0	77.1
Learngene	WAVE	150	1.3	**58.6**	**63.2**	**65.4**	**66.6**	**67.3**	4.4	**68.9**	**72.7**	**74.1**	**74.9**	**75.3**	15.8	**74.5**	**77.5**	**78.2**	**78.9**	**79.2**
				↑3.4	↑1.4	↑0.8	↑0.7	↑0.5		↑3.5	↑2.2	↑1.9	↑1.6	↑1.5		↑2.8	↑2.2	↑1.8	↑1.5	↑1.6

Table 2. Performance of initializing models with variable widths on ImageNet-1K. All models are trained for 10 epochs after initialization.

				L_6				
				W_{384}	W_{576}	W_{768}	W_{1152}	W_{1536}
	Methods	Epoch	Para.	11.5	25.1	44.1	98.0	173.1
Direct	He-Init [6]	N/A	0	49.4	51.5	53.1	46.6	31.3
Direct	Mimetic [33]	N/A	0	49.1	48.0	54.3	47.7	33.0
Direct	GHN-3 [18]	N/A	0	49.0	51.8	52.5	52.7	51.0
Tran.	Wt Select [44]	N/A	26.8	48.6	50.7	55.4	—	—
Tran.	LiGO [35]	N/A	10.0	63.6	63.1	69.5	70.0	73.7
LG	WAVE	150	5.4	**64.8**	**67.0**	**72.4**	**73.5**	**78.3**
				↑1.2	↑3.9	↑2.9	↑3.5	↑4.6

and enhancing flexibility through a few trainable parameters (i.e., weight scalers) to adapt connection rules to various model sizes. This approach enables WAVE to integrate knowledge through predefined rules while allowing model-specific customization, achieving structural knowledge integration without excessive inductive bias.

5.1.2. Width Expansion

The adaptability of weight templates enables WAVE to scale the width of neural networks, achieving ***what learngenes accomplish for the first time*** and consistently outperforming other model initialization and knowledge transfer methods, as shown in Table 2.

Similar to the observations in Table 1, methods like Mimetic and GHN-3 continue to underperform compared to parameter-transfer approaches, underscoring the importance of pre-trained knowledge for model initialization. LiGO persists in introducing excessive random parameters, while Weight Selection disrupts structural knowledge and fails to initialize models larger than the pre-trained ones.

By leveraging the Kronecker product, WAVE efficiently initializes models by concatenating weight templates with minimal parameters, making it the most effective approach for initializing models of varying widths.

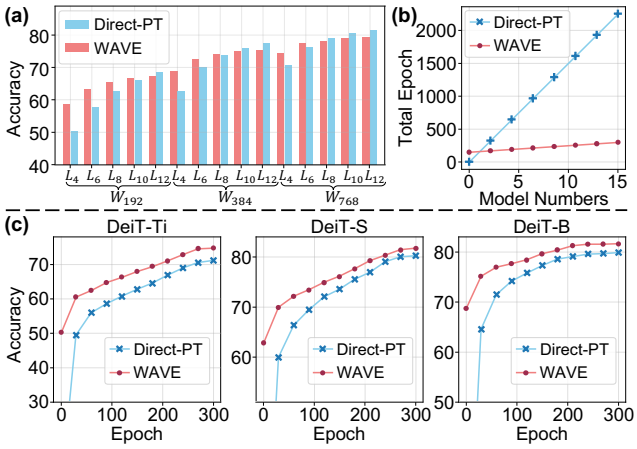


Figure 3. Compared with Direct Pre-training. (a) Comparison of models initialized by WAVE and trained for 10 epochs versus those directly pre-trained for 150 epochs across 15 downstream models of varying sizes. (b) Analysis of computational cost as the number of initialized models increases. (c) Detailed training process (300 epochs) of models initialized by WAVE and direct pre-training.

5.2. Compared with Direct Pre-Training

When target model sizes are incompatible with pre-trained ones, models initialized by WAVE achieve superior performance with significantly reduced training effort. As illustrated in Figure 3c, models initialized by WAVE demonstrate superior performance from the very start of training. Notably, these models, trained for just 10 epochs, significantly outperform those directly pre-trained for 150 epochs (Figure 3a), achieving a remarkable 15× reduction in computational costs (Figure 3b). This advantage becomes particularly pronounced in smaller networks and

Table 3. Performance of models on downstream datasets. "Para.(M)" is the average parameter transferred during model initialization.

	Methods	DeiT-Ti, 3.0M Para.	Flow.	CUB	Cars	C10	C100	Food	iNat.	*Aver.*	DeiT-S, 11.3M Para.	Flow.	CUB	Cars	C10	C100	Food	iNat.	*Aver.*
Direct	He-Init [6]	0	53.9	26.1	19.9	92.4	68.3	68.4	52.3	*54.5*	0	57.2	27.3	23.8	94.0	66.5	70.6	54.0	*56.2*
	Mimetic [33]	0	52.1	35.0	20.5	88.9	63.4	66.9	49.0	*53.7*	0	57.4	39.6	34.2	91.6	65.7	67.1	52.2	*58.3*
	GHN-3 [18]	0	50.0	41.1	23.2	92.5	70.1	76.2	51.7	*57.8*	0	52.7	45.2	30.6	93.9	72.7	76.2	55.5	*61.0*
Trans.	Wt Select [44]	2.9	55.0	34.4	21.7	92.5	67.0	67.4	51.2	*55.6*	11.0	58.3	33.1	28.1	94.1	68.1	69.0	54.2	*57.8*
	Share Init [20]	0.6	92.4	70.1	82.1	96.0	77.2	81.2	63.2	*80.3*	2.2	94.1	72.4	87.2	96.5	78.5	83.0	63.0	*82.1*
	LiGO [35]	2.0	94.2	71.8	83.9	95.6	78.5	82.1	61.6	*81.1*	7.5	95.9	74.8	87.9	96.9	81.3	84.0	66.1	*83.8*
Learngene	Heur-LG [36]	1.5	64.7	44.6	37.7	94.0	71.1	74.7	57.4	*63.5*	5.7	69.1	48.0	51.2	95.1	72.8	76.8	59.3	*67.5*
	Auto-LG [37]	2.0	93.5	71.4	83.5	96.4	77.1	81.7	62.5	*80.9*	7.5	96.4	75.1	88.2	97.3	81.0	84.6	67.0	*84.2*
	TLEG [40]	1.1	91.0	69.5	78.2	96.1	77.0	82.0	63.4	*79.6*	3.9	93.7	72.6	87.2	97.2	80.2	84.9	66.5	*83.2*
	WAVE	1.1	**94.9**	**74.8**	**84.4**	**96.6**	**80.7**	**83.8**	**65.2**	***82.9***	4.0	**96.9**	**78.1**	**89.4**	**97.4**	**83.2**	**85.5**	**67.6**	***85.4***
			↑0.7	↑3.0	↑0.5	↑0.2	↑2.2	↑1.7	↑1.8	*↑1.8*		↑0.5	↑3.0	↑1.2	↑0.1	↑1.9	↑0.6	↑0.6	*↑1.2*
PT	Direct FT	2.9	95.4	75.1	86.5	96.6	80.2	84.0	66.9	*83.5*	11.0	96.4	77.0	89.4	97.5	82.8	85.6	69.3	*85.4*

scenarios with a larger number of initialized models, underscoring WAVE's effectiveness and efficiency as an initialization approach.

5.3. Transferability of WAVE

Table 1 and Table 2 have effectively demonstrated WAVE's initialization capability on ImageNet-1K. Moreover, the knowledge encapsulated in weight templates is common enough for transfer across diverse downstream datasets. Table 3 highlights WAVE's performance on various datasets.

Evidently, WAVE achieves significant improvements across diverse downstream datasets, highlighting its effectiveness in model initialization. In contrast, Mimetic and GHN-3 may underperform compared to He-Init on certain datasets, revealing limitations in the universality of prior knowledge and constraints on parameter adaptability. Furthermore, methods like Weight Selection may experience negative transfer [26], emphasizing the importance of transferring core knowledge [10].

Furthermore, the inadequacy of small datasets (e.g., Oxford Flowers, Stanford Car, and CUB-200-2011) to support large model training without effective knowledge transfer is evident. This further underscores WAVE's ability to maximize data utilization, as only a small amount of data is required to leverage the structured knowledge in weight templates for adaptive model initialization.

5.4. Ablation and Analysis

5.4.1. Effect of Initialization Rules on Concatenating Weight Templates

For the rules when initializing models with weight templates, we compare the advantages of the Kronecker product with the *linear combination* in TLEG [40] and the *weight transformation* in LiGO [35].

As illustrated in Table 4, although linear combination avoids introducing additional trainable parameters, its predefined rules lack generality across different model sizes and do not support width expansion. Conversely, weight transformation introduces numerous random parameters to expand pre-trained weight matrices, hindering the efficient adaptation of pre-trained knowledge within a limited training period. In comparison, the Kronecker product offers a more flexible approach by concatenating weight templates with a small number of parameters, enabling efficient and adaptive initialization across various model sizes.

Table 4. Ablation study on the Kronecker product. "Init. Para." refers to the trainable parameters when initializing target models.

	W	L	FLOPs	Linear Init. Para.	Linear Acc.	Kronecker Init. Para.	Kronecker Acc.
Depth Variation	768	4	11.69	0 M	71.6	0.0005 M	**74.5**
		8	23.15	0 M	76.2	0.0011 M	**78.2**
		12	34.61	0 M	77.1	0.0016 M	**79.2**
	W	L	FLOPs	Transformation Init. Para.	Transformation Acc.	Kronecker Init. Para.	Kronecker Acc.
Width Variation	384	6	4.55	4.28 M	63.6	0.0025 M	**64.8**
	768		17.42	17.11 M	69.5	0.0025 M	**72.4**
	1536		68.14	136.84 M	73.7	0.0025 M	**78.3**

Table 5. Ablation study on weight templates initializing different components of DeiT, with models consisting of 6 layers.

Methods	Att.	Proj.	FC	Norm	Ti	S	B
He-Init					40.6	49.4	53.1
WAVE		✓			50.2	57.9	60.4
	✓				52.3	60.3	64.6
			✓		58.7	66.8	69.0
	✓	✓	✓		63.1	72.6	77.4
	✓	✓	✓	✓	**63.2**	**72.7**	**77.5**

5.4.2. Effect of WAVE on Different Components

We further analyze the impact of weight templates on the initialization of different components in DeiT, as detailed in Table 5. The results indicate that all components can be effectively initialized using the structured knowledge encapsulated in weight templates. In contrast, the weight ma-

Table 6. Analysis of the number and shape of weight templates.

	L_6 W_{192}			L_6 W_{384}			L_6 W_{768}		
	Para.	Shape	Acc.	Para.	Shape	Acc.	Para.	Shape	Acc.
↓ Num.	0.9	192^2	57.5	2.6	384^2	68.4	8.6	768^2	75.7
↓ Shape	1.3	96^2	60.3	4.4	192^2	71.6	15.8	384^2	75.8
WAVE	1.3	192^2	**63.2**	4.4	384^2	**72.7**	15.8	768^2	**77.5**

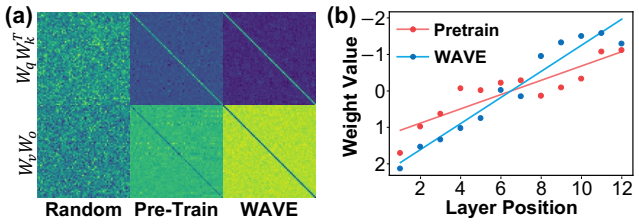


Figure 4. Visualization of structured knowledge. (a) Knowledge in self-attention layers. (b) Relationships between layer position and corresponding parameter values after PCA.

trices for normalization and bias are more data-dependent and have fewer parameters, which can be efficiently learned from the training data, making the use of weight templates for these components optional. The attention mechanism, consisting of Query, Key and Value, encodes more structured knowledge than other components, highlighting its critical role in the model's initialization [38].

5.4.3. Effect of Number and Shape of Templates

We analyze the influence of the number and size of weight templates on initializing downstream models. As shown in Table 6, reducing the number of weight templates significantly degrades model performance, as fewer parameters fail to capture the full spectrum of size-agnostic knowledge. Additionally, we experiment with reducing the size of weight templates to enhance flexibility in width expansion. However, such flexibility sacrifices the performance for the insufficient structured knowledge in weight templates.

Therefore, both the number and size of weight templates are crucial for effective initialization. WAVE achieves an optimal balance between performance and cost by utilizing approximately 16.7% of the auxiliary model's parameters. The basic size of weight templates is aligned with the *embedding dimensions* of the auxiliary model (refer to Section 4.2 for details) to facilitate the effective capture of structured knowledge, as illustrated in Figure 4a.

5.5. Visualization of Knowledge in Weight Templates

5.5.1. Visualization of Structured Knowledge

Mimetic Initialization [33] identifies diagonal properties in self-attention layers, while TLEG [40] captures linear correlations across layers. However, these observations are exclusive to pre-trained ViTs, with both Mimetic Initialization and TLEG manually preserving such structured knowledge during initialization. Remarkably, as demonstrated in Figure 4, WAVE autonomously encapsulates such structured knowledge from pre-trained models into weight templates without manual intervention. Consequently, models initialized by weight templates inherently retain these structural characteristics in their parameter matrices.

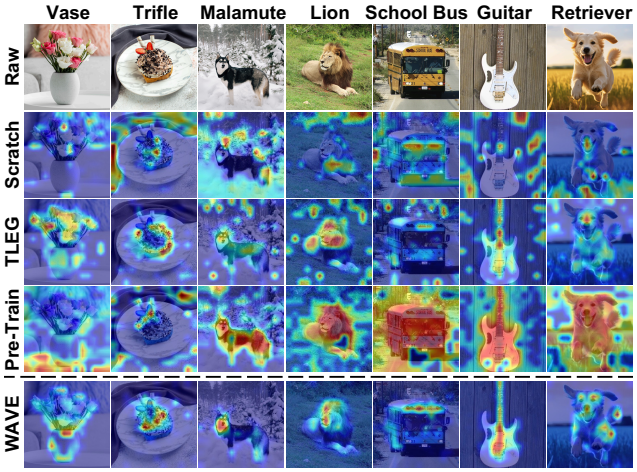


Figure 5. Visualization of knowledge encapsulated in weight templates. All networks operate directly after initialization without any additional training or fine-tuning.

5.5.2. Visualization of Common Knowledge

In addition, we demonstrate that weight templates enable neural networks to focus more on the common local features after initialization. We select sample images and employ CAM [27] to visualize the attention in pre-trained networks, as well as those initialized by learngenes (e.g., TLEG).

As shown in Figure 5, random initialization shows scattered and widespread attention, whereas pre-trained models transfer an entire set of previously learned knowledge, leading to broader attention that often includes irrelevant details, such as the image background. Both TLEG and WAVE focus more on local features (i.e., smaller red attention blocks), with WAVE demonstrating superior localization and a cleaner focus (i.e., removing attention on image background), thereby enhancing classification performance.

6. Conclusion

In this paper, we introduce WAVE, an innovative approach for initializing models of variable sizes. WAVE leverages shared weight templates that encapsulate size-agnostic knowledge, along with size-specific weight scalers, to effectively extract and transfer structured knowledge from pre-trained models for initialization across all model sizes. WAVE demonstrates superior performance in initializing models for both depth and width expansions, and the size-agnostic knowledge is also common enough to be transferred to various downstream datasets.

Acknowledgement

We sincerely thank Wenqian Li, Jianlu Shen, and Ruixiao Shi for their insightful discussions on this work. We also appreciate Freepik for contributing to the figure design. This research is supported by the Science and Technology Major Project of Jiangsu Province under Grant BG20240305, Key Program of Jiangsu Science Foundation under Grant BK20243012, National Natural Science Foundation of China under Grants U24A20324, 62125602, and 62306073, Natural Science Foundation of Jiangsu Province under Grant BK20230832, and the Xplorer Prize.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'19)*, 32, 2019. 2

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 446–461, 2014. 12

[4] Shuvam Chakraborty, Burak Uzkent, Kumar Ayush, Kumar Tanmay, Evan Sheehan, and Stefano Ermon. Efficient conditional pre-training for transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*, pages 4241–4250, 2022. 3

[5] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*, pages 1–12, 2016. 11

[6] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'21)*, pages 9640–9649, 2021. 2, 5, 6, 7, 12, 13

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'09)*, pages 248–255, 2009. 5

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*, pages 1–12, 2021. 1

[9] Fu Feng, Jing Wang, Congzhi Zhang, Wenqian Li, Xu Yang, and Xin Geng. Genes in intelligent agents. *arXiv preprint arXiv:2306.10225*, 2023. 2, 13

[10] Fu Feng, Yucheng Xie, Jing Wang, and Xin Geng. Transferring core knowledge via learngenes. *arXiv preprint arXiv:2401.08139*, 2024. 2, 7, 13

[11] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'17)*, pages 4502–4508, 2017. 12

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 2

[13] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'21)*, pages 12259–12269, 2021. 5

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 4

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR'22)*, pages 1–13, 2022. 2

[17] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *Proceedings of the International Conference on Machine Learning (ICML'20)*, pages 4475–4483, 2020. 2

[18] Boris Knyazev, Doha Hwang, and Simon Lacoste-Julien. Can we scale transformers to predict parameters of diverse imagenet models? In *Proceedings of the International Conference on Machine Learning (ICML'23)*, pages 17243–17259, 2023. 2, 5, 6, 7

[19] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 12

[20] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*, pages 1–14, 2020. 5, 6, 7

[21] Wenqian Li, Pengfei Fang, and Hui Xue. Svasp: Self-versatility adversarial style perturbation for cross-domain few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'25)*, 2025. 1

[22] Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'22)*, 35:36889–36901, 2022. 3

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer:

Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'21)*, pages 10012–10022, 2021. 1

[24] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008. 12

[25] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*, pages 10428–10436, 2020. 12

[26] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Transfer Learning*, pages 1–4, 2005. 7

[27] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of International Conference on Machine Learning (ICML'17)*, pages 618–626, 2017. 8

[28] Sebastian Seung. *Connectome: How the brain's wiring makes us who we are*. HMH, 2012. 13

[29] Kiat Chuan Tan, Yulong Liu, Barbara Ambrose, Melissa Tulig, and Serge Belongie. The herbarium challenge 2019 dataset. *arXiv preprint arXiv:1906.05372*, 2019. 12

[30] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML'21)*, pages 10347–10357, 2021. 1, 5

[31] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1

[32] Eleni Triantafillou, Hugo Larochelle, Richard Zemel, and Vincent Dumoulin. Learning a universal template for few-shot dataset generalization. In *Proceedings of the International Conference on Machine Learning (ICML'21)*, pages 10424–10433, 2021. 2, 3

[33] Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. In *Proceedings of the International Conference on Machine Learning (ICML'23)*, pages 34456–34468, 2023. 2, 3, 5, 6, 7, 8

[34] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 12

[35] Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. In *Proceedings of the International Conference on Learning Representations (ICLR'23)*, pages 1–13, 2023. 1, 2, 3, 5, 6, 7

[36] QiuFeng Wang, Xin Geng, ShuXia Lin, Shi-Yu Xia, Lei Qi, and Ning Xu. Learngene: From open-world to your learning task. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'22)*, pages 8557–8565, 2022. 3, 4, 5, 6, 7, 11

[37] Qiufeng Wang, Xu Yang, Shuxia Lin, and Xin Geng. Learngene: Inheriting condensed knowledge from the ancestry model to descendant models. *arXiv preprint arXiv:2305.02279*, 2023. 2, 3, 5, 6, 7, 11

[38] Qiufeng Wang, Xu Yang, Fu Feng, Jing Wang, and Xin Geng. Cluster-learngene: Inheriting adaptive clusters for vision transformers. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'24)*, 2024. 8

[39] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'21)*, pages 22–31, 2021. 1

[40] Shiyu Xia, Miaosen Zhang, Xu Yang, Ruiming Chen, Haokun Chen, and Xin Geng. Transformer as linear expansion of learngene. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'24)*, pages 16014–16022, 2024. 2, 3, 4, 5, 6, 7, 8, 11, 12, 13

[41] Shiyu Xia, Yuankun Zu, Xu Yang, and Xin Geng. Initializing variable-sized vision transformers from learngene with learnable transformation. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'24)*, 2024. 3, 4

[42] Yucheng Xie, Fu Feng, Ruixiao Shi, Jing Wang, and Xin Geng. Fine: Factorizing knowledge for initialization of variable-sized diffusion models. *arXiv preprint arXiv:2409.19289*, 2024. 3

[43] Yucheng Xie, Fu Feng, Jing Wang, Xin Geng, and Yong Rui. Kind: Knowledge integration and diversion in diffusion models. *arXiv preprint arXiv:2408.07337*, 2024. 3

[44] Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. Initializing models with larger ones. In *Proceedings of the International Conference on Learning Representations (ICLR'23)*, pages 1–13, 2024. 1, 2, 3, 5, 6, 7

[45] Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10:3770, 2019. 2

[46] Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Minivit: Compressing vision transformers with weight multiplexing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22)*, pages 12145–12154, 2022. 1

[47] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'20)*, pages 3833–3845, 2020. 2, 3