SAHYPER: BREAKING THE RANK BOTTLENECK AND RETHINKING PRETRAINED WEIGHT INHERITANCE IN GRAPH HYPERNETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph HyperNetworks (GHNs) generate parameters from computational graphs, enabling a single generator to serve models across scales and tasks. Recent methods incorporate pretrained knowledge to improve the quality of generated parameters. However, existing decoders limit the rank of generated weights, and increasing their rank is costly. Existing weight inheritance also treats related pretrained attention matrices independently. We introduce SAHyper, a Structure-Aware Hypernetwork that improves both direct generation and pretrained weight inheritance. Its Multi Component Decoder sums low-rank components to enable higher-rank weight generation with a compact hypernetwork. Structure-Aware Spectral Inheritance combines pretrained singular values with generated orthogonal bases and coordinates related bases within attention blocks. On ImageNet-1K, SAHyper generates a ViT-Base model that achieves 63.86% Top-1 accuracy without further training, surpassing the SOTA method by 10.5 points. It also outperforms the SOTA method on Visual Domain Decathlon and sets of 14 and 20 image classification tasks, with gains maintained after further training and on unseen tasks. SAHyper also outperforms the SOTA method in ResNet and LLaMA experiments.

1 INTRODUCTION

Training deep neural networks remains computationally expensive, especially when separate models are required across scales and tasks (Dosovitskiy, 2020; Touvron et al., 2023). Graph HyperNetworks (GHNs) (Zhang et al., 2018; Knyazev et al., 2021; Xu et al., 2025) address this by learning one parameter generator across model scales and tasks. A GHN generates target model parameters from its computational graph. During training, task losses from generated models jointly update the hypernetwork across target architectures. Figure 1(a) illustrates this process.

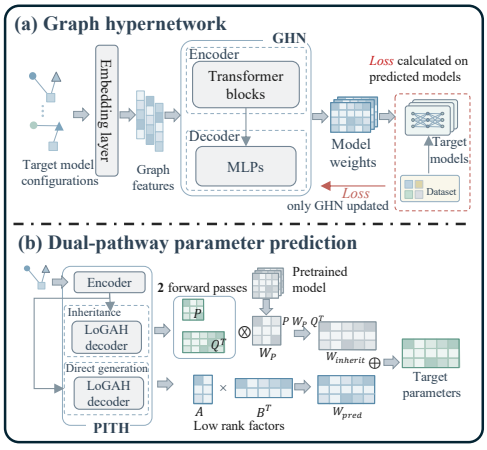

Figure 1: (a) A GHN generates parameters from target computational graphs and learns from generated models' task losses. (b) PITH fuses direct generation with pretrained parameter inheritance.

LoGAH (Zhou et al., 2024) reduces the cost of parameter generation by constructing each weight matrix as the product of two smaller matrices. TAL (Xu et al., 2025) and PITH (Xu et al., 2026) build on this decoder and introduce pretrained knowledge into GHNs. TAL uses soft labels from a pretrained model to supervise hypernetwork training. PITH retains the direct generation path and adds a second path for weight inheritance. This path maps pretrained weights into the target parameter space using projection matrices P and Q. Each projection matrix is generated separately by the LoGAH decoder. The inherited and directly gen-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

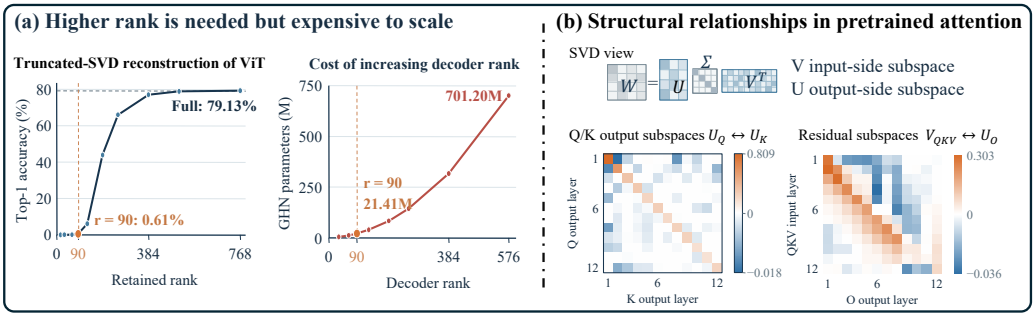

Figure 2: (a) Truncating the major weight matrices of a well trained ViT to rank 90 reduces accuracy from 79.13% to 0.61%. Increasing the LoGAH decoder rank rapidly increases GHN size. (b) SVD exposes the input and output subspaces of pretrained parameters. The heatmaps compare Q/K output subspaces (left) and QKV input/O output subspaces (right) across layers. Diagonal entries compare projections within the same block. The heatmaps show subspace affinity corrected for random overlap, using the leading 90 singular vectors of a pretrained ViT-Base model.

erated weights are then fused to produce the target parameters, as illustrated in Figure 1(b). On ImageNet-1K (Russakovsky et al., 2015), PITH (Xu et al., 2026) already provides useful initializations for 12-layer Vision Transformer (ViT) (Dosovitskiy, 2020) models at the Small and Base scales, achieving more than 50% accuracy in direct evaluation. In this paper, we aim to go beyond such useful initializations and generate higher-quality model parameters.

The LoGAH decoder used in TAL (Xu et al., 2025) and PITH (Xu et al., 2026) forms each large weight matrix from two smaller matrices, limiting its rank to at most 90 by default. However, we observe that the weights of a well trained ViT are not well represented at low rank. Reconstructing its major weight matrices with truncated singular value decomposition (SVD) (Eckart & Young, 1936) severely reduces accuracy at low ranks, with accuracy recovering as more rank is retained. Figure 2(a) shows that simply increasing the decoder rank is expensive. In the evaluated LoGAH configuration, rank-dependent parameters account for more than 90% of the whole GHN and grow approximately quadratically with rank. A more parameter-efficient way to increase rank is therefore needed. PITH (Xu et al., 2026) maps pretrained matrices independently to target dimensions, ignoring relationships among weights within a Transformer block (Vaswani et al., 2017). To capture these relationships in weight inheritance, we adopt an SVD-based representation, which naturally separates pretrained weights into singular values and input- and output-side subspaces. In pretrained Transformer blocks, we further find (1) strong alignment between the dominant output subspaces of the query (Q) and key (K) matrices; (2) clear relationships between the input subspace of the combined QKV matrix and the output subspace of the O matrix in the residual space. The relationships within blocks in Figure 2(b) motivate structure-aware weight inheritance. Experimental settings are provided in Appendix A.1.

Motivated by the above observations, we introduce the Structure-Aware Hypernetwork (SAHyper), which redesigns both parameter-generation paths in PITH. In the direct path, the Multi Component Decoder (MC decoder) sums multiple compact low-rank components to construct each weight matrix, enabling higher-rank weight generation with a compact hypernetwork. In the inheritance path, Structure-Aware Spectral Inheritance (SA Inherit) retains singular values from pretrained weights, generates orthogonal parameter bases with the hypernetwork, and coordinates related bases according to the observed attention structure. The two paths are fused as in PITH to produce the final target parameters. On ImageNet-1K, SAHyper achieves 63.9% Top-1 accuracy with ViT-Base in direct evaluation, surpassing PITH by 10.5 points. On Decathlon (Rebuffi et al., 2017), it achieves an average accuracy of 57.3%, exceeding PITH by 2.0 points. Across 14 and 20 tasks, its average accuracy reaches 69.7% and 72.4%, improving over PITH by 9.7 and 12.4 points, respectively. LLaMA experiments also reduce perplexity by 9–20% across four target sizes compared with PITH under the same training budget. Our main contributions are as follows.

1. **Multi Component Decoder.** We alleviate the rank bottleneck of existing GHN decoders with an MC decoder that combines multiple compact low-rank components to efficiently generate higher rank weights.

2. **Structure-Aware Spectral Inheritance.** We address the lack of structural modeling in existing weight inheritance methods with SA Inherit, which preserves pretrained spectra and coordinates related attention bases.
3. **Strong empirical gains across settings.** SAHyper improves ImageNet-1K accuracy over PITH by more than 10 points in direct evaluation and consistently improves downstream-task performance with gains of up to 12.4 points.

2 BACKGROUND

2.1 PARAMETER PREDICTION VIA GRAPH HYPERNETWORKS

A Graph HyperNetwork (GHN) (Zhang et al., 2018; Knyazev et al., 2021) H_D predicts the parameters of a target network f from its computational graph f^G as $\mathbf{w}_{\text{pred}} = H_D(f^G; \theta)$, where θ denotes the hypernetwork parameters.

The computational graph $f^G = (V, E)$ is a directed acyclic graph describing the architecture of the target network. Each node in V corresponds to a specific operation, such as convolution, pooling, linear projection, or self-attention, while the edges in E describe the forward connections between these operations. The operation type of each node is represented by a one-hot vector and mapped to a d-dimensional node feature through an embedding layer.

GHN-2 (Knyazev et al., 2021) and GHN-3 (Knyazev et al., 2023) encode graphs into contextualized node representations and decode them into target parameters.

Following GHN-2 (Knyazev et al., 2021) and LoGAH (Zhou et al., 2024), given M training architectures $\{f_a^G\}_{a=1}^M$ and N samples $\{(x_j, y_j)\}_{j=1}^N$, the training objective is

$$\min_{\theta} \frac{1}{NM} \sum_{j=1}^{N} \sum_{a=1}^{M} \mathcal{L}\left(f_a\left(x_j; H_D(f_a^G; \theta)\right), y_j\right). \tag{1}$$

Each iteration updates the hypernetwork by back-propagating task losses through models generated for sampled architectures.

LoGAH (Zhou et al., 2024) reduces generation cost through low-rank decoding. For $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, its decoder produces a compact $2r \times r$ matrix, then expands, reshapes, and crops it into factors $A \in \mathbb{R}^{d_{\text{out}} \times r}$ and $B \in \mathbb{R}^{d_{\text{in}} \times r}$. The generated weight is $W_{\text{pred}} = AB^\top$, with rank at most r.

2.2 USING PRETRAINED MODELS IN HYPERNETWORKS

TAL (Xu et al., 2025) uses soft labels (Hinton et al., 2015) from a pretrained model to supervise hypernetwork training, while PITH (Xu et al., 2026) directly incorporates pretrained weights during parameter generation. Given $W_P \in \mathbb{R}^{d_{\text{out}}^P \times d_{\text{in}}^P}$, PITH uses two separate LoGAH decoder calls to generate $P \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}^P}$ and $Q \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}^P}$. It maps the pretrained weight to the target dimensions as $W_{\text{inherit}} = PW_PQ^\top$ and combines the two paths as

$$W_{\text{target}} = \alpha W_{\text{inherit}} + (1 - \alpha) W_{\text{pred}}, \tag{2}$$

where α balances direct generation and inheritance.

3 METHOD

The Structure-Aware Hypernetwork (SAHyper) redesigns both parameter generation paths in PITH (Xu et al., 2026). The direct path uses the Multi Component Decoder (MC decoder) to construct weights from multiple compact low-rank components. The inheritance path uses a separate MC decoder to generate bases for Structure-Aware Spectral Inheritance (SA Inherit). SA Inherit orthogonalizes these bases, coordinates related bases within attention blocks, and combines them with pretrained singular values. SAHyper fuses the two paths using Eq. (2) with a learned coefficient α. Figure 3 illustrates the MC decoder and SA Inherit; Figure 7 in the appendix shows the overall workflow.

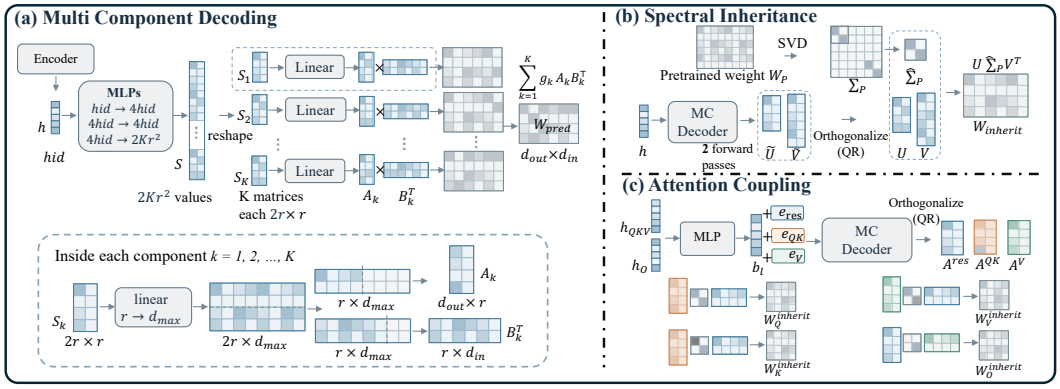

Figure 3: Overview of SAHyper. (a) The MC decoder reshapes the MLP output into K compact matrices. Each matrix is expanded and cropped to form two factors, whose products are combined using learnable gains. Here, $d_{\max}$ denotes the expansion width. (b) Spectral inheritance generates two bases in separate forward passes and orthogonalizes them using reduced QR decomposition. These bases are combined with retained pretrained singular values. (c) Attention coupling generates residual, Q/K, and value bases from a block representation and role embeddings. Related attention weights share these bases while retaining their own pretrained spectra.

3.1 Multi Component Decoder

Given a node representation, an MLP produces $2Kr^2$ values, which are reshaped into K compact matrices $S_k \in \mathbb{R}^{2r\times r}$, as shown in Figure 3(a). Each component uses a linear layer to expand its matrix to width $d_{\max}$, then reshapes and crops the result to obtain $A_k \in \mathbb{R}^{d_{\text{out}}\times r}$ and $B_k \in \mathbb{R}^{d_{\text{in}}\times r}$. The MC decoder combines their products to construct the weight matrix

$$W_{\text{pred}} = \sum_{k=1}^{K} g_k A_k B_k^{\top}, \tag{3}$$

where K is the number of components, r is the local rank of each component, and g_k is a learnable component gain.

The aggregate rank budget is $R = Kr$. For a fixed MLP width, the dominant parameter cost of generating the compact matrices decreases from $\mathcal{O}(R^2)$ in a single rank-R LoGAH decoder to $\mathcal{O}(Kr^2) = \mathcal{O}(Rr)$ in the MC decoder. At fixed r, this cost grows linearly with R.

We initialize the learnable component gains with $g_k = 1/\sqrt{K}$.

3.2 Structure-Aware Spectral Inheritance

Spectral Inheritance. For a pretrained matrix $W_P = U_P\Sigma_P V_P^{\top}$, let q denote the retained rank. As shown in Figure 3(b), the inheritance decoder generates raw bases $\widetilde{U} \in \mathbb{R}^{d_{\text{out}}\times q}$ and $\widetilde{V} \in \mathbb{R}^{d_{\text{in}}\times q}$ in two separate forward passes. Reduced QR decomposition gives U and V with orthonormal columns. The inherited weight is then constructed as

$$W_{\text{inherit}} = U\widehat{\Sigma}_P V^{\top}. \tag{4}$$

Here, $\widehat{\Sigma}_P \in \mathbb{R}^{q\times q}$ contains the retained pretrained singular values after normalization and rescaling.

For weights outside attention, including feed-forward weights, the left and right bases are generated separately.

Attention Coupling. For block ℓ, let $h_{QKV,\ell}$ and $h_{O,\ell}$ denote the encoded graph features of the QKV and O nodes. We concatenate these features and apply an MLP to obtain $b_\ell = \text{MLP}([h_{QKV,\ell}; h_{O,\ell}])$. Learned role embeddings are then added to form the inputs for basis generation, $z_{\text{res}} = b_\ell + e_{\text{res}}$, $z_{QK} = b_\ell + e_{QK}$, and $z_V = b_\ell + e_V$. The decoder maps them to raw matrices and applies QR orthogonalization to obtain a residual basis A^{res}, a shared Q/K output basis A^{QK}, and a value basis A^V, as shown in Figure 3(c).

Using the pretrained spectra $\widehat{\Sigma}_Q$, $\widehat{\Sigma}_K$, $\widehat{\Sigma}_V$, and $\widehat{\Sigma}_O$, processed as described above, SA Inherit constructs the weights in the inheritance path as

$$\begin{aligned} W_Q^{\text{inherit}} &= A^{QK}\widehat{\Sigma}_Q(A^{\text{res}})^\top, \qquad & W_K^{\text{inherit}} &= A^{QK}\widehat{\Sigma}_K(A^{\text{res}})^\top, \\ W_V^{\text{inherit}} &= A^{V}\widehat{\Sigma}_V(A^{\text{res}})^\top, \qquad & W_O^{\text{inherit}} &= A^{\text{res}}\widehat{\Sigma}_O(A^{V})^\top. \end{aligned} \tag{5}$$

Each basis uses the leading columns required by the corresponding retained rank. Q, K, and V share their input basis with the output basis of O in the residual space. Q and K also share their output basis. V maps residual features into the value space, and O maps the attended values back (Vaswani et al., 2017), motivating their shared value basis. Each weight retains its own pretrained spectrum, so shared bases do not make the inherited weights identical.

4 Experiments

ImageNet-1K experiments assess ViT parameter generation and initialization quality. Multitask experiments cover Decathlon and 14- and 20-task sets. Unseen-task transfer tests adaptation. ResNet and LLaMA experiments assess applicability beyond ViTs.

4.1 Experimental Setup

Datasets and architectures. We use ImageNet-1K (Russakovsky et al., 2015) for the ViT (Dosovitskiy, 2020) and ResNet (He et al., 2016) experiments. Multitask evaluation covers the nine Decathlon tasks other than ImageNet (Rebuffi et al., 2017) and two sets of 14 and 20 image classification tasks. We use WikiText-103 (Merity et al., 2016) for language modeling. The main ViT experiments use the ViTs$^+$-1K (Xu et al., 2025) library of 1,000 architectures and evaluate 12-layer Tiny, Small, and Base models.

Compared methods. We compare with GHN-3 (Knyazev et al., 2023), LoGAH (Zhou et al., 2024), TAL (Xu et al., 2025), and PITH (Xu et al., 2026). Comparisons on 14 and 20 tasks focus on TAL and PITH, while ResNet and LLaMA comparisons use PITH. RandInit is included in evaluation after further training and transfer to unseen tasks. Baseline results for ImageNet-1K, Decathlon, and unseen-task transfer are taken from PITH (Xu et al., 2026). We use $K = 18$ and $r = 32$ for the MC decoder.

Training and evaluation. All ViT methods that incorporate pretrained knowledge use the same pretrained ViT-Large (Dosovitskiy, 2020) model. Detailed training and evaluation settings are provided in Appendix A.

4.2 ViTs on ImageNet-1K

We train the hypernetworks on ImageNet-1K for 200 epochs using the ViTs$^+$-1K architecture library. We directly evaluate Top-1 accuracy of generated 12-layer Tiny, Small, and Base ViTs without further training. We also train Small and Base models for 75 epochs from generated or random initializations. SAHyper reports the best accuracy within this budget; baseline evaluation follows PITH (Xu et al., 2026).

ViTs generated by SAHyper achieve higher accuracy on ImageNet-1K in both direct evaluation and evaluation after further training. In direct evaluation, SAHyper reaches 51.75%, 63.79%, and 63.86% on Tiny, Small, and Base, exceeding PITH by 5.40, 10.52, and 10.51 points, respectively (Table 1). After further training, Small and Base reach 70.10% and 69.65%, improving over PITH by 2.91 and 2.36 points.

4.3 Multitask Parameter Generation

We evaluate multitask parameter generation on Decathlon and two sets of 14 and 20 image classification tasks. Starting from their respective ImageNet-trained hypernetworks, TAL, PITH, and SAHyper undergo 100 epochs of joint training on each task set. Each hypernetwork generates parameters for all tasks in that set. We directly evaluate 12-layer Tiny, Small, and Base ViTs using

Table 1: Top-1 accuracy (%) on ImageNet-1K for ViTs with 12 layers. Direct evaluation follows 200 epochs of hypernetwork training. Further training evaluates Small and Base models over 75 epochs from generated or random initializations.

Method	Direct eval. (Top-1 acc., %)			Further train. eval. (Top-1 acc., %)	
	ViT-Tiny	ViT-Small	ViT-Base	ViT-Small	ViT-Base
RandInit	-	-	-	64.38	61.31
GHN-3	34.95	35.33	33.74	49.73	45.08
LoGAH	26.31	44.82	44.85	64.53	62.45
TAL	37.63	46.74	46.81	65.48	64.28
PITH	46.35	53.27	53.35	67.19	67.29
SAHyper	**51.75**	**63.79**	**63.86**	**70.10**	**69.65**

Table 2: Average Top-1 accuracy (%) on Decathlon for ViTs with 12 layers. Direct evaluation uses generated models without further training. Further training evaluates Tiny and Small with 100 epochs from generated initializations or 200 epochs from RandInit.

Method	Direct eval. (Top-1 acc., %)			Further train. eval. (Top-1 acc., %)	
	ViT-Tiny	ViT-Small	ViT-Base	ViT-Tiny	ViT-Small
RandInit	-	-	-	50.83	50.02
GHN-3	27.28	27.40	27.33	46.36	46.15
LoGAH	27.86	31.28	30.99	51.30	50.81
TAL	50.96	54.13	54.44	58.98	58.39
PITH	53.11	55.32	55.29	61.65	60.73
SAHyper	**53.72**	**57.51**	**57.32**	**62.25**	**62.78**

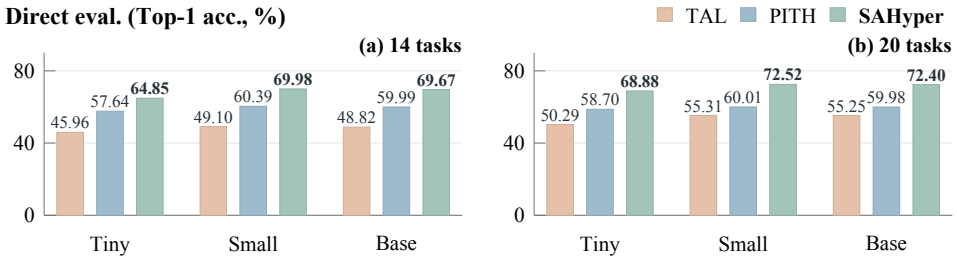

Figure 4: Average Top-1 accuracy (%) on (a) 14 and (b) 20 image classification tasks for 12-layer ViT-Tiny, ViT-Small, and ViT-Base models. Direct evaluation follows 100 epochs of joint hypernetwork training, without further training.

average Top-1 accuracy across tasks. On Decathlon, we additionally train Tiny and Small for 100 epochs from generated initializations or 200 epochs from random initialization. Baseline training protocols and model selection details are provided in Appendix A.3.

ViTs generated by SAHyper achieve higher average accuracy on Decathlon in both direct evaluation and evaluation after further training. Direct evaluation yields 53.72%, 57.51%, and 57.32% on Tiny, Small, and Base, improving over PITH by 0.61, 2.19, and 2.03 points, respectively (Table 2). After further training, Tiny and Small reach average accuracies of 62.25% and 62.78%, exceeding PITH by 0.60 and 2.05 points. Aircraft remains below PITH. Results for individual tasks are provided in Tables 8 and 9 in the appendix.

ViTs generated by SAHyper also achieve higher average accuracy on both the 14-task and 20-task sets in direct evaluation, outperforming TAL and PITH at every target size. On Base, SAHyper reaches 69.67% and 72.40%, exceeding PITH by 9.68 and 12.42 points, respectively, as shown in Figure 4. Results for individual tasks are provided in Tables 10 and 11 in the appendix.

4.4 Transfer to Unseen Tasks

We use the SAHyper hypernetwork trained for 200 epochs on ImageNet-1K and then jointly trained for 100 epochs on Decathlon. This hypernetwork generates initializations for Fashion-

Table 3: Test Top-1 accuracy (%) on three unseen tasks after 100 epochs of further training. Targets are 12-layer Tiny and Small ViTs with generated or random initializations.

Transfer to Unseen Tasks, Eval. after Further Training (Top-1 acc., %)						
Method	Fashion-MNIST		FER2013		HAM10000	
	ViT-Tiny	ViT-Small	ViT-Tiny	ViT-Small	ViT-Tiny	ViT-Small
RandInit	89.79	89.31	61.35	58.90	97.34	97.58
LoGAH	89.54	90.87	60.07	62.69	96.50	97.83
TAL	89.41	90.55	61.55	62.89	97.58	97.58
PITH	90.81	91.55	62.52	62.97	97.71	97.95
SAHyper	**91.92**	**92.98**	**63.22**	**64.28**	**97.83**	**98.31**

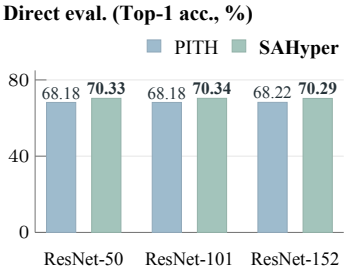

Figure 5: Direct evaluation of generated ResNets on ImageNet-1K after 100 epochs of hypernetwork training.

MNIST (Xiao et al., 2017), FER2013 (Goodfellow et al., 2013), and HAM10000 (Tschandl et al., 2018), none of which are included in its training. The hypernetwork remains fixed, while the generated 12-layer ViT-Tiny and ViT-Small models are trained on each unseen task for 100 epochs. We compare final test Top-1 accuracy against other generated and random initializations. Results after 20 epochs are provided in Appendix A.4.

ViTs initialized by SAHyper achieve higher accuracy on all three unseen tasks after further training, outperforming the compared methods at both model sizes. On Small, SAHyper reaches 92.98%, 64.28%, and 98.31% on Fashion-MNIST, FER2013, and HAM10000, exceeding PITH by 1.43, 1.31, and 0.36 points, respectively (Table 3).

4.5 ResNets on ImageNet-1K

We train PITH and SAHyper on ImageNet-1K for 100 epochs using the same library of 155 ResNet architectures and the same pretrained ResNet-152 (He et al., 2016) model. For ResNets, SA Inherit is applied without attention coupling. We directly evaluate generated ResNet-50, ResNet-101, and ResNet-152 models after the same batch normalization calibration procedure, without updating target model parameters. Details of convolutional weight inheritance, training, and calibration are provided in Appendix A.5.

ResNets generated by SAHyper achieve higher accuracy on ImageNet-1K than those generated by PITH in direct evaluation across all three target depths. SAHyper achieves Top-1 accuracies of 70.33%, 70.34%, and 70.29% on ResNet-50, ResNet-101, and ResNet-152, compared with 68.18%, 68.18%, and 68.22% for PITH, as shown in Figure 5. The average improvement is 2.12 points.

4.6 LLaMA on WikiText-103

We train PITH and SAHyper on WikiText-103 for three epochs using the same library of 91 LLaMA (Touvron et al., 2023) architectures and the same pretrained Llama-3.2-1B (Grattafiori et al., 2024) model. We directly evaluate four target sizes on the validation set without target model training, reporting negative log-likelihood (NLL), perplexity (PPL), and next token accuracy. Detailed settings are provided in Appendix A.6.

LLaMA models generated by SAHyper achieve lower perplexity and higher next token accuracy on WikiText-103 than those generated by PITH at all four target sizes. For the 16-layer model with a hidden dimension of 2048, SAHyper reduces NLL from 5.3175 to 5.2209 and PPL from 203.88 to 185.10, while increasing next token accuracy from 22.55% to 23.46% (Table 4). Across the four target sizes, NLL decreases by 0.0927 to 0.2203.

Table 4: Direct evaluation of generated LLaMA models on the WikiText-103 validation set after three epochs of hypernetwork training. NLL, PPL, and Token acc. denote negative log-likelihood, perplexity, and next token accuracy (%), respectively.

Depth	Hidden dim.	PITH			**SAHyper**		
		NLL ↓	PPL ↓	Token acc. (%) ↑	NLL ↓	PPL ↓	Token acc. (%) ↑
8	768	5.5599	259.80	21.84	**5.3877**	**218.69**	**23.32**
12	1024	5.5294	251.99	21.58	**5.3091**	**202.17**	**23.37**
16	1536	5.3923	219.71	22.31	**5.2996**	**200.25**	**23.05**
16	2048	5.3175	203.88	22.55	**5.2209**	**185.10**	**23.46**

Table 5: Component comparisons on ImageNet-1K. Direct Top-1 accuracy (%) of 12-layer ViTs after 200 epochs of hypernetwork training. Average is computed across model sizes.

Configuration	ViT-Tiny	ViT-Small	ViT-Base	Average
LoGAH	26.31	44.82	44.85	38.66
MC decoder	46.35	57.50	57.56	53.81
PITH	46.35	53.27	53.35	50.99
SA Inherit	48.92	59.33	58.99	55.75
SAHyper	**51.75**	**63.79**	**63.86**	**59.80**

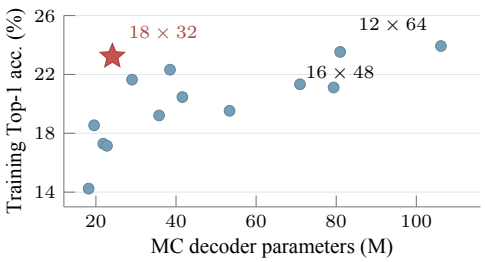

Figure 6: Training Top-1 accuracy at epoch 25 versus MC decoder parameters on ImageNet-1K. Labels show $K \times r$; the red star marks the selected configuration.

5 Analysis

We first examine the contributions of the MC decoder and SA Inherit. We then study MC decoder configurations and the ranks of generated weights. Finally, we compare layer matching strategies and evaluate execution optimizations.

5.1 Component Contributions

We compare LoGAH with the MC decoder for direct parameter generation without inheritance, and PITH with SA Inherit for parameter inheritance. The SA Inherit configuration uses a single low-rank component in each decoder. SAHyper combines both improvements. We train the hypernetworks on ImageNet-1K for 200 epochs and directly evaluate generated 12-layer ViT-Tiny, ViT-Small, and ViT-Base models. Configuration details are provided in Appendix A.7.

The MC decoder and SA Inherit improve the accuracy of generated ViTs in their respective comparisons, and SAHyper achieves the best results when they are combined. The MC decoder reaches 53.81% average accuracy, exceeding LoGAH by 15.15 points. SA Inherit reaches 55.75%, improving over PITH by 4.76 points. SAHyper achieves 59.80% and performs best at every target size (Table 5).

5.2 Analysis of the MC Decoder

We compare 14 MC decoder configurations varying component count K and rank r. Each configuration is trained on ImageNet-1K for 25 epochs without inheritance. Figure 6 plots epoch-25 training accuracy on sampled architectures against decoder parameter count. Further experimental details are provided in Appendix A.8.

The $K = 18, r = 32$ configuration provides a good balance between training accuracy and decoder size. It achieves 23.23% accuracy with 24.11M parameters. By comparison, $K = 16, r = 48$ reaches 23.53% with 80.98M parameters, while $K = 12, r = 64$ reaches 23.93% with 106.15M parameters.

We also examine the complete weights generated by the $K = 18, r = 32$ MC decoder after 200 epochs of training without inheritance. We measure 216 attention and feed-forward matrices from

Table 6: Layer matching: direct ImageNet-1K Top-1 accuracy (%) of 12-layer ViTs after 200 epochs of SAHyper training.

Alignment	Direct eval. (Top-1 acc., %)			
	ViT-Tiny	ViT-Small	ViT-Base	Average
First-N	**51.75**	**63.79**	**63.86**	**59.80**
Relative-depth	51.04	63.00	62.47	58.84

Table 7: Spectral inheritance pipeline costs before and after optimization: approximate library storage, CPU memory usage, and batch time.

Execution	Library (GB)	CPU memory (GB)	Batch time (s)
Original	96.4	200–294+	0.69
Optimized	0.222	58–63	0.46–0.47

12-layer ViT-Tiny, ViT-Small, and ViT-Base models. Numerical rank counts singular values exceeding 10^{-4} times the largest singular value.

The generated weights reach numerical ranks close to their attainable limits. Median numerical ranks are 192, 380, and 568.5 for Tiny, Small, and Base, respectively, compared with limits of 192, 384, and 576. Median ranks retaining 95% of spectral energy are 93, 115, and 120, respectively. These exceed the single-component rank of 32.

5.3 Layer Matching for Weight Inheritance

First-N matches each target layer to the pretrained layer with the same index. Relative-depth matches layers by their relative positions and interpolates spectra between adjacent pretrained layers. Both variants undergo 200 epochs of hypernetwork training on ImageNet-1K, with all other settings fixed. We directly evaluate generated 12-layer ViTs. Further details are provided in Appendix A.9.

First-N matching yields more accurate ViTs than relative-depth matching in SAHyper. It achieves 59.80% average accuracy, compared with 58.84% for relative-depth matching, and performs better at all three target sizes (Table 6).

5.4 Training Efficiency

We cache graph metadata, configurations, and model structures, then release generated weights after each iteration instead of retaining them or copying them back to the CPU. These optimizations preserve parameter generation and the training objective. We assess their effects using measurements from an earlier spectral inheritance implementation, with measurement details in Appendix A.10.

The optimized pipeline reduces storage, CPU memory usage, and training time per batch. Library storage decreases from about 96.4 GB to 222 MB, while CPU memory usage falls to 58–63 GB. Batch time falls from 0.69 s to 0.46–0.47 s, a reduction of approximately 32–33% (Table 7).

6 Related Work

Graph Hypernetworks. Graph Hypernetworks (GHNs) (Zhang et al., 2018) predict neural network parameters from computational graphs. GHN-2 (Knyazev et al., 2021) and GHN-3 (Knyazev et al., 2023) improve parameter prediction through better graph representations. LoGAH (Zhou et al., 2024) reduces the cost of generating large weight matrices through low-rank decoding. TAL (Xu et al., 2025) incorporates pretrained knowledge into GHNs through soft-label supervision, while PITH (Xu et al., 2026) directly incorporates pretrained weights through learned projections and fuses inherited and directly generated parameters. SAHyper builds on this line by addressing the limited rank capacity of existing decoders and the independent inheritance of structurally related attention weights, using multi-component decoding and structure-aware spectral inheritance.

7 Conclusion

We presented SAHyper, a graph hypernetwork that improves both direct parameter generation and pretrained weight inheritance. The MC decoder generates higher-rank weights with a compact hypernetwork. SA Inherit incorporates pretrained spectra and relationships among attention matrices into weight inheritance. Experiments across visual tasks and language modeling show consistent gains over the SOTA method.

AI USE STATEMENT

We used generative AI tools to assist with drafting portions of the manuscript and to improve the clarity, readability, and presentation of the writing. The tools were used only for writing assistance and did not contribute to the scientific methodology, experimental design, or interpretation of the results. All AI-assisted content was carefully reviewed, edited, and verified by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

All datasets used in this work are publicly available and are used in accordance with their respective licenses and terms of use. This work does not involve the collection of new human-subject data or personally identifiable information.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the experimental setup, model configurations, training procedures, hyperparameters, and evaluation protocols in the main paper and appendix. Code and instructions for reproducing the main experimental results will be made publicly available upon acceptance.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3034–3042, 2016.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *Neural information processing: 20th international conference, ICONIP 2013, daegu, korea, november 3-7, 2013. Proceedings, Part III 20*, pp. 117–124. Springer, 2013.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

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

Boris Knyazev, Michal Drozdzal, Graham W Taylor, and Adriana Romero Soriano. Parameter prediction for unseen deep architectures. *Advances in Neural Information Processing Systems*, 34:29433–29448, 2021.

Boris Knyazev, Doha Hwang, and Simon Lacoste-Julien. Can we scale transformers to predict parameters of diverse imagenet models? In *International Conference on Machine Learning*, pp. 17243–17259. PMLR, 2023.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Stefan Munder and Dariu M Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1863–1868, 2006.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

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

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11):1–7, 2012.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. scientific data. 2018; 5: 180161. *Search in*, 2, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 210–218. Springer, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Jiaze Xu, Shiyu Xia, Xu Yang, Jiaqi Lv, and Xin Geng. Learngene tells you how to customize: Task-aware parameter initialization at flexible scales. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=IRQ0n961nn.

Jiaze Xu, Shiyu Xia, Jiaqi Lv, and Xin Geng. Unlocking pre-trained weights: Parameter inheritance for zero-shot initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 34502–34511, 2026.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749*, 2018.

Xinyu Zhou, Boris Knyazev, Alexia Jolicoeur-Martineau, and Jie Fu. Logah: Predicting 774-million-parameter transformers using graph hypernetworks with 1/100 parameters. *arXiv preprint arXiv:2405.16287*, 2024.

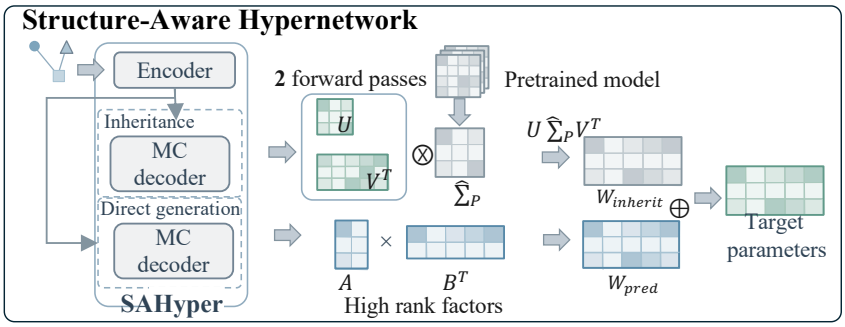

Figure 7: Overall architecture of SAHyper. The architecture encoder conditions two parameter generation paths. The direct path uses the MC decoder to construct W_{pred}, while SA Inherit combines generated orthonormal bases with pretrained singular values to form W_{inherit}. The two weights are fused with a learned coefficient to obtain the target parameters.

A Additional Experimental Details

A.1 Motivational Experiment Settings

Rank truncation. For Figure 2(a), we use the 12-layer ViT-B/16 checkpoint with the `augreg_in1k` weights from timm. We apply exact SVD to the joint QKV, attention output, and two feed-forward weight matrices in each block, giving 48 matrices. All are reconstructed at the same retained rank, using $r \in \{16, 32, 64, 90, 128, 192, 256, 384, 512, 768\}$, while other parameters remain unchanged. We evaluate without further training on all 50,000 ImageNet-1K validation images, using 224×224 center crops, crop fraction 0.9, bicubic interpolation, channel means and standard deviations of 0.5, batch size 256, and mixed precision.

Parameter counts. We instantiate the original single-component LoGAH decoder with input dimension 128, hidden widths 512 and 1024, and $d_{\max} = 32768$, varying the rank over $\{32, 64, 90, 128, 192, 256, 384, 576\}$. Whole-GHN counts add the fixed 1,226,356 non-decoder parameters. At rank 90, rank-dependent decoder parameters account for 19,595,078 of 21,412,794 total parameters (91.51%).

Attention subspaces. Figure 2(b) uses a separate 12-layer ViT-B/16 checkpoint with `augreg_in21k_ft_in1k` weights. For every layer pair, we compare the leading 90 output singular vectors of Q and K, and the leading 90 input singular vectors of joint QKV with the output singular vectors of O. Bases are computed with `torch.svd_lowrank`, using 128 components, four iterations, and fixed seeds based on 1111. For bases $X, Y \in \mathbb{R}^{d \times r}$, the displayed affinity is $(\|X^\top Y\|_F^2/r - r/d)/(1 - r/d)$, with $r = 90$ and $d = 768$. The correction removes expected random overlap; diagonal entries compare the same block. This analysis uses checkpoint weights only, without training or image evaluation.

A.2 ViT Training and Evaluation

The ViTs$^+$-1K library (Xu et al., 2025) contains 1,000 architectures with depths from 3 to 12 and hidden dimensions from 192 to 768. All evaluated ViTs have 12 layers.

We train SAHyper on ImageNet-1K for 200 epochs using AdamW, batch size 512, one sampled architecture per update, mixed precision, a peak learning rate of 3×10^{-4}, and weight decay 10^{-2}. The learning rate warms up for five epochs, is initially held constant, and is then reduced when loss on a fixed set of architectures and training batches stops improving. Newly trained vision hypernetworks use random seed 1111. Unless varied in Section 5.3, inheritance matches pretrained and target parameters by role and layer index using First-N.

On ImageNet-1K, direct evaluation uses generated weights, and further training evaluates Small and Base over 75 epochs. SAHyper reports the best accuracy within this budget. Baseline results and evaluation criteria follow PITH (Xu et al., 2026).

A.3 MULTITASK TRAINING AND RESULTS

Task sets. We use nine Decathlon tasks (Rebuffi et al., 2017) for multitask adaptation after ImageNet-1K training. These are Aircraft (Airc.) Maji et al. (2013), CIFAR-100 (C100) Krizhevsky et al. (2009), Daimler pedestrians (DPed) Munder & Gavrila (2006), DTD Cimpoi et al. (2014), German traffic signs (GTSRB) Stallkamp et al. (2012), Omniglot (OGlt) Lake et al. (2015), SVHN Netzer et al. (2011), UCF101 dynamic images (UCF) Soomro et al. (2012); Bilen et al. (2016), and Flowers102 (Flwr) Nilsback & Zisserman (2008).

The 14-task set consists of Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2012), MNIST (LeCun et al., 1998), RESISC45 (Cheng et al., 2017), SVHN (Netzer et al., 2011), SUN397 (Xiao et al., 2010), STL10 (Coates et al., 2011), Oxford-IIIT Pet (Parkhi et al., 2012), Flowers102 (Nilsback & Zisserman, 2008), CIFAR-100 (Krizhevsky et al., 2009), PCAM (Veeling et al., 2018), and FER2013 (Goodfellow et al., 2013). The 20-task set adds CIFAR-10 (Krizhevsky et al., 2009), Food101 (Bossard et al., 2014), Fashion-MNIST (Xiao et al., 2017), Rendered SST-2 (Radford et al., 2021), EMNIST (Cohen et al., 2017), and KMNIST (Clanuwat et al., 2018). All tasks in each set are included in joint hypernetwork training.

Joint training. TAL, PITH, and SAHyper each start from their own ImageNet-trained hypernetwork and undergo 100 epochs of joint training on each task set. SAHyper starts from the model used in the main ImageNet experiment. TAL keeps its loaded graph encoder frozen, and task vectors remain fixed. For task τ with N_τ training samples, the sampling probability is proportional to $p_\tau^{1/T}$, where $p_\tau = N_\tau / \sum_i N_i$ and the sampling temperature is $T = 2$ (Raffel et al., 2020).

The 14-task and 20-task experiments use batch size 512 and the full training split specified for each task. Each epoch contains 1,139 and 2,101 updates, respectively. PITH and SAHyper use learning rate warmup, a constant phase, and subsequent reductions when training loss stops improving. TAL uses warmup followed by cosine decay for 14 tasks and the same type of adaptive schedule as the other methods for 20 tasks. Comparisons between the two task sets therefore do not isolate the effect of task count.

Evaluation. We directly evaluate generated 12-layer ViT-Tiny, ViT-Small, and ViT-Base models. Decathlon results are reported in Table 8, and results for the 14-task and 20-task sets are reported in Tables 10 and 11. For the latter two sets, average accuracy is computed from unrounded task accuracies with equal weight for each task.

On Decathlon, we also train Tiny and Small models for 100 epochs from generated initializations or 200 epochs from random initialization (Table 9). SAHyper reports the best validation accuracy within 100 epochs, including a search over learning rates and regularization settings for individual Tiny tasks. GHN-3 and LoGAH retain their task-specific hypernetwork training protocols. Baseline task accuracies and evaluation criteria follow PITH (Xu et al., 2026). For direct evaluation, we recompute baseline averages from its nine reported task accuracies, giving equal weight to each task.

A.4 TRANSFER TO UNSEEN TASKS

The transfer experiment uses Fashion-MNIST Xiao et al. (2017), FER2013 Goodfellow et al. (2013), and HAM10000 Tschandl et al. (2018). None is included in the ImageNet-1K and Decathlon training of the hypernetwork used here. The 14-task and 20-task experiments use separate hypernetworks.

We keep the hypernetwork fixed and train the generated 12-layer ViT-Tiny and ViT-Small models on each unseen task. Table 12 reports test Top-1 accuracy after 20 and 100 epochs. The 100-epoch results are also summarized in Table 3. SAHyper is evaluated at these specified epochs without selecting models by test accuracy.

A.5 RESNET TRAINING AND EVALUATION

Models and training. We use a library of 155 bottleneck ResNet architectures and the torchvision ResNet-152 (He et al., 2016) model with IMAGENET1K_V2 pretrained weights. PITH and SAHyper use ImageNet-1K images at 224×224, batch size 512, mixed precision, AdamW with initial learning rate 3×10^{-4} and weight decay 0.01, and gradient clipping at 5. The learning rate

Table 8: Direct evaluation on Decathlon for 12-layer ViT-Tiny, ViT-Small, and ViT-Base models. We report Top-1 accuracy (%) for each task and the average across all nine tasks. TAL, PITH, and SAHyper undergo 100 epochs of joint hypernetwork training. Bold marks the best result for each task and model size.

Direct eval. (Top-1 acc., %)											
Model	Method	Airc.	C100	DPed	DTD	GTSRB	OGlt	SVHN	UCF	Flwr	Avg.
ViT-Tiny	GHN-3	0.99	36.31	83.95	6.76	95.87	0.06	10.00	1.23	10.39	27.28
	LoGAH	2.37	34.97	80.34	8.30	96.44	**0.14**	12.00	4.00	12.16	27.86
	TAL	0.99	52.70	93.11	30.27	99.23	0.06	90.75	40.83	50.69	50.96
	PITH	**7.35**	56.43	90.56	32.82	99.54	0.06	90.77	**46.62**	53.82	53.11
	SAHyper	4.44	**58.59**	**94.73**	**33.99**	**99.75**	0.06	**92.00**	40.06	**59.90**	**53.72**
ViT-Small	GHN-3	0.99	35.76	85.31	6.38	96.02	0.06	10.00	1.23	10.88	27.40
	LoGAH	2.10	40.78	80.63	10.11	97.96	**0.14**	30.25	5.02	14.51	31.28
	TAL	1.02	56.98	91.31	34.26	99.62	0.06	92.14	53.64	58.14	54.13
	PITH	**8.94**	58.60	92.16	34.20	99.69	0.06	91.85	54.82	57.55	55.32
	SAHyper	5.22	**64.38**	**95.49**	**37.07**	**99.82**	0.06	**93.11**	**55.84**	**66.57**	**57.51**
ViT-Base	GHN-3	0.99	35.37	84.56	6.33	95.27	0.06	10.00	1.23	12.16	27.33
	LoGAH	2.37	40.20	80.70	8.88	98.14	**0.18**	30.56	5.07	12.84	30.99
	TAL	0.99	57.36	91.38	34.47	99.64	0.06	92.10	**56.05**	57.94	54.44
	PITH	**8.22**	58.52	91.46	33.94	99.73	0.06	91.76	55.02	58.92	55.29
	SAHyper	4.74	**64.04**	**95.94**	**37.55**	**99.77**	0.06	**93.04**	55.02	**65.69**	**57.32**

Table 9: Further training evaluation on Decathlon for 12-layer ViT-Tiny and ViT-Small models. We report Top-1 accuracy (%) after 100 epochs from generated initializations or 200 epochs from RandInit. Bold marks the best result for each task and model size.

Further train. eval. (Top-1 acc., %)											
Model	Method	Airc.	C100	DPed	DTD	GTSRB	OGlt	SVHN	UCF	Flwr	Avg.
ViT-Tiny	RandInit	7.80	58.64	98.74	23.09	99.58	24.12	86.38	24.03	35.10	50.83
	GHN-3	3.57	54.31	97.98	19.36	98.28	10.54	86.08	17.06	30.10	46.36
	LoGAH	4.71	60.80	98.11	20.21	98.92	24.65	84.68	31.05	38.53	51.30
	TAL	3.84	59.63	99.52	35.16	99.68	21.33	92.10	57.53	62.06	58.98
	PITH	**14.04**	64.01	98.71	35.90	99.89	**25.51**	93.05	60.45	63.33	61.65
	SAHyper	6.03	**67.00**	**99.59**	**39.10**	**99.90**	23.37	**94.01**	**64.65**	**66.57**	**62.25**
ViT-Small	RandInit	8.01	60.17	98.44	25.05	98.89	15.70	85.57	23.31	35.00	50.02
	GHN-3	5.49	57.54	98.35	18.24	97.92	11.40	85.71	13.42	27.25	46.15
	LoGAH	4.50	60.06	97.82	21.68	98.67	**29.65**	85.45	23.00	36.47	50.81
	TAL	4.62	58.06	99.40	33.88	99.82	19.42	92.38	59.02	58.92	58.39
	PITH	**11.40**	64.07	98.95	36.65	**99.87**	20.69	92.25	60.86	61.86	60.73
	SAHyper	6.21	**69.43**	**99.54**	**39.04**	**99.87**	26.51	**93.37**	**61.12**	**69.90**	**62.78**

is reduced when training loss stops improving. The scheduler is configured for 1,500 epochs, while the reported comparison evaluates both methods after 100 epochs.

Weight inheritance. PITH uses its low-rank decoder with rank 90 and $d_{\max} = 32768$. SAHyper uses $K = 18$, $r = 32$, and $d_{\max} = 8192$. The first and last convolutions in each bottleneck use spectral inheritance. The middle convolution uses a rank-32 Tucker core from the pretrained weights with independently generated input and output bases. Pretrained and target parameters are matched by stage, role, and layer index.

Evaluation. Before direct evaluation, both methods calibrate batch normalization statistics using 200 batches totaling 102,400 images, without optimizer updates. Generated ResNet-50, ResNet-101, and ResNet-152 models are then evaluated on all 50,000 ImageNet-1K validation images.

A.6 LLaMA Training and Evaluation

Models and data. The pretrained model is Llama-3.2-1B (Grattafiori et al., 2024). The architecture library combines depths from 4 to 16 with hidden dimensions from $\{512, 768, 1024, 1280, 1536, 1792, 2048\}$, yielding 91 architectures. The feed-forward width is four times the hidden dimension, the attention head dimension is 64, and there are four query heads for

Table 10: Direct evaluation on 14 tasks for 12-layer ViT-Tiny, ViT-Small, and ViT-Base models after 100 epochs of joint hypernetwork training. We report Top-1 accuracy (%). Average gives equal weight to each task. Bold marks the best result for each task and model size.

Direct eval. (Top-1 acc., %)									
	ViT-Tiny			ViT-Small			ViT-Base		
Task	TAL	PITH	**SAHyper**	TAL	PITH	**SAHyper**	TAL	PITH	**SAHyper**
Cars	5.51	16.34	**24.19**	7.39	17.70	**31.44**	6.79	17.36	**31.28**
DTD	24.41	43.19	**46.12**	29.15	47.45	**51.65**	28.03	47.55	**52.71**
EuroSAT	76.37	85.37	**88.70**	75.15	88.19	**90.78**	74.56	85.22	**90.26**
GTSRB	84.66	79.41	**95.36**	86.22	85.29	**96.61**	85.57	83.73	**95.76**
MNIST	**98.42**	92.75	97.92	98.60	92.12	**98.85**	98.51	92.54	**98.88**
RESISC45	61.90	76.40	**76.87**	68.60	79.30	**84.27**	68.59	79.30	**84.10**
SVHN	19.59	46.52	**90.48**	19.59	55.08	**93.82**	19.59	52.24	**93.63**
SUN397	12.82	23.28	**27.63**	16.26	24.72	**32.26**	16.01	25.11	**31.87**
STL10	60.48	73.78	**76.69**	64.01	75.69	**80.22**	63.44	76.30	**79.66**
Oxford-IIIT Pet	20.52	38.68	**42.76**	24.23	41.46	**51.10**	23.93	41.54	**50.59**
Flowers102	24.72	56.16	**57.13**	33.96	59.07	**68.17**	33.96	58.79	**66.86**
CIFAR-100	32.77	51.54	**56.39**	38.96	54.20	**61.91**	38.09	54.53	**61.81**
PCAM	78.51	82.25	**82.38**	79.17	82.07	**82.23**	80.25	**82.92**	81.71
FER2013	42.71	41.35	**45.26**	46.09	43.09	**56.37**	46.18	42.73	**56.20**
Average	45.96	57.64	**64.85**	49.10	60.39	**69.98**	48.82	59.99	**69.67**

Table 11: Direct evaluation on 20 tasks for 12-layer ViT-Tiny, ViT-Small, and ViT-Base models after 100 epochs of joint hypernetwork training. We report Top-1 accuracy (%). Average gives equal weight to each task. Bold marks the best result for each task and model size.

Direct eval. (Top-1 acc., %)									
	ViT-Tiny			ViT-Small			ViT-Base		
Task	TAL	PITH	**SAHyper**	TAL	PITH	**SAHyper**	TAL	PITH	**SAHyper**
Cars	4.35	5.63	**10.97**	5.48	6.04	**13.58**	5.58	6.47	**13.77**
DTD	23.14	2.13	**45.27**	25.96	2.13	**50.90**	25.21	2.13	**50.53**
EuroSAT	68.93	77.26	**79.30**	75.96	74.37	**85.74**	77.15	74.81	**84.22**
GTSRB	58.76	88.01	**95.96**	84.90	88.95	**96.10**	80.16	87.28	**96.08**
MNIST	98.79	98.86	**99.34**	99.05	98.93	**99.42**	99.09	99.20	**99.38**
RESISC45	45.54	64.54	**75.87**	54.05	72.08	**82.10**	56.79	72.27	**82.41**
SVHN	19.59	83.73	**93.15**	19.59	87.69	**94.93**	19.59	87.13	**94.80**
SUN397	11.44	24.20	**27.11**	13.17	26.02	**31.10**	12.99	26.43	**31.20**
STL10	65.04	78.30	**81.19**	70.85	79.97	**83.01**	70.31	79.92	**83.47**
Oxford-IIIT Pet	15.64	2.73	**49.80**	20.96	2.73	**55.19**	19.71	2.73	**55.22**
Flowers102	19.47	56.72	**59.72**	27.35	59.16	**68.06**	27.39	58.82	**68.58**
CIFAR-100	42.35	54.61	**58.04**	50.23	58.83	**62.96**	49.90	59.38	**63.33**
PCAM	74.48	**79.73**	75.88	75.91	78.18	**84.30**	78.94	78.09	**83.19**
FER2013	43.81	46.50	**55.98**	48.12	47.95	**59.29**	47.27	47.99	**58.30**
CIFAR-10	70.28	83.54	**84.94**	80.95	85.52	**87.49**	80.14	85.62	**87.56**
Food101	17.41	0.99	**47.54**	21.83	0.99	**56.41**	22.10	0.99	**56.67**
Fashion-MNIST	86.33	87.26	**91.95**	88.56	89.55	**92.85**	88.53	89.24	**92.43**
Rendered SST-2	**50.08**	49.92	49.92	**49.92**	**49.92**	**49.92**	**51.95**	49.92	49.92
EMNIST	98.57	99.22	**99.54**	99.39	99.34	**99.63**	99.25	99.39	**99.65**
KMNIST	91.69	90.14	**96.18**	93.94	91.85	**97.33**	92.93	91.83	**97.28**
Average	50.29	58.70	**68.88**	55.31	60.01	**72.52**	55.25	59.98	**72.40**

each key/value head (Ainslie et al., 2023). The four evaluation architectures in Table 4 are included in this library. Training and evaluation use WikiText-103 (Merity et al., 2016).

Parameter generation. Both methods retain the pretrained vocabulary of 128,256 entries. Smaller targets use fixed nested subsets of the pretrained hidden channels. For each hidden dimension, input

Table 12: Test Top-1 accuracy (%) on three unseen tasks after 20 and 100 epochs of further training. The targets are 12-layer ViT-Tiny and ViT-Small models trained from generated or random initializations. Bold marks the best result for each task, model size, and training duration.

Transfer to Unseen Tasks, Eval. after Further Training (Top-1 acc., %)							
Epochs	Method	Fashion-MNIST		FER2013		HAM10000	
		ViT-Tiny	ViT-Small	ViT-Tiny	ViT-Small	ViT-Tiny	ViT-Small
20	RandInit	89.52	89.27	40.96	35.30	95.53	96.01
	LoGAH	89.51	90.59	47.03	48.12	92.51	96.14
	TAL	89.30	89.51	50.96	53.50	96.86	97.34
	PITH	**90.10**	90.17	52.97	52.55	**97.34**	**97.46**
	SAHyper	90.09	**91.35**	**59.74**	**61.72**	96.14	95.17
100	RandInit	89.79	89.31	61.35	58.90	97.34	97.58
	LoGAH	89.54	90.87	60.07	62.69	96.50	97.83
	TAL	89.41	90.55	61.55	62.89	97.58	97.58
	PITH	90.81	91.55	62.52	62.97	97.71	97.95
	SAHyper	**91.92**	**92.98**	**63.22**	**64.28**	**97.83**	**98.31**

Table 13: Validation NLL of generated LLaMA models on WikiText-103 during hypernetwork training. Column groups give target depth and hidden dimension. Epoch 0 precedes training. Bold marks the lower NLL for each model and epoch.

Epoch	$(8, 768)$		$(12, 1024)$		$(16, 1536)$		$(16, 2048)$	
	PITH	**SAHyper**	PITH	**SAHyper**	PITH	**SAHyper**	PITH	**SAHyper**
0	**11.2471**	11.3518	**11.4472**	11.6097	**11.7891**	11.8645	11.9042	**11.6512**
1	5.8737	**5.7726**	5.8298	**5.7586**	**5.7219**	5.7228	**5.6570**	5.6689
2	5.5745	**5.4323**	5.5478	**5.3585**	5.4133	**5.3584**	5.3330	**5.2728**
3	5.5599	**5.3877**	5.5294	**5.3091**	5.3923	**5.2996**	5.3175	**5.2209**

embeddings and output vocabulary projections share a frozen table that is not generated by the hypernetwork. Before fusion, inherited matrices are scaled according to the target layer's number of input units.

Training. Each update samples one architecture and uses an effective batch of 64 sequences of length 256, with microbatches of 16. Generated weights are constructed once per update, and target weight gradients are accumulated over the microbatches before being propagated to the hypernetwork. Only hypernetwork parameters are updated. Both methods use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01, gradient clipping at 1, and seed 1111.

The learning rate warms up from 10^{-6} to 10^{-4} over 91 updates, remains constant for five architecture cycles, and is then reduced based on training NLL. A cycle covers all 91 architectures. We monitor an exponential moving average of cycle NLL with coefficient 0.8. Eight monitored cycles without a 0.5% relative improvement halve the learning rate, with four cooldown cycles and a minimum of 10^{-6}. Both methods use this rule, with reductions determined by their own training loss. One epoch contains 7,401 updates, and both methods train for three epochs.

Evaluation. Direct evaluation uses the full WikiText-103 validation set, comprising 996 packed sequences and 253,980 valid next token prediction positions. NLL is averaged over these positions, PPL is its exponential, and next token accuracy measures the fraction of positions where the highest-scoring token matches the target. Table 13 reports validation NLL during hypernetwork training. These results use one random seed.

A.7 Component Settings

The configurations in Table 5 use 200 epochs of ImageNet-1K hypernetwork training. The MC decoder configuration has no inheritance and uses $K = 18$, $r = 32$, and $d_{\max} = 8192$. Its MLP

has widths $h \to 4h \to 4h \to 2Kr^2$, where h is the node feature dimension. Component expansion weights are stored in one matrix and applied independently to each component.

The SA Inherit configuration uses one rank-90 component in each decoder, at most 90 pretrained singular values, $d_{\max} = 32768$, and a learned scalar fusion coefficient. SAHyper uses two MC decoders with $K = 18$ and $r = 32$, at most 576 pretrained singular values, and a fusion coefficient conditioned on the target architecture. It also applies a residual transformation to the concatenated component features before expansion. This transformation is absent from the standalone MC decoder configuration. Fusion and component transformations also differ between these configurations.

Each layer retains its own pretrained spectrum, including when shapes match across layers. We retain leading singular values that cover 95% of the pretrained spectral energy, subject to decoder capacity and target dimensions. In SAHyper, retained singular values beyond rank 90 are multiplied by a learned scaling factor conditioned on the target architecture.

A.8 MC Decoder Parameters and Generated Rank

Parameter study. The configurations in Figure 6 use fixed component expansion without inheritance or residual transformations across components. Both MLP width and $d_{\max}$ vary across configurations. Accuracy is accumulated over sampled training architectures during epoch 25.

Rank measurements. We use the MC decoder trained for 200 epochs with $K = 18$, $r = 32$, and $d_{\max} = 8192$, without inheritance. Measurements use the complete generated weights written into 12-layer ViT-Tiny, ViT-Small, and ViT-Base models. Each layer contributes Q, K, V, O, and two feed-forward matrices, giving 72 matrices per model size and 216 in total.

Singular values are computed on the CPU in double precision. Numerical rank counts singular values greater than $10^{-4}\sigma_1$, where σ_1 is the largest singular value of that matrix. We also measure r_{95}, the smallest number of leading singular values whose squared sum reaches 95% of the total squared singular values. Results in Section 5.2 are medians across the 72 matrices at each model size.

A.9 Layer Matching Details

For target depth L and pretrained depth L_P, First-N assigns pretrained layer ℓ to target layer ℓ. Relative-depth instead uses

$$x_\ell = \frac{\ell + 1/2}{L} L_P - \frac{1}{2}. \tag{6}$$

Layer indices start at zero. Relative-depth interpolates positive, normalized singular-value spectra between adjacent pretrained layers in log space, then normalizes the result. A 12-layer target paired with a 24-layer pretrained ViT uses coordinates $0.5, 2.5, \ldots, 22.5$. Both methods preserve the parameter role, and all other settings in Table 6 are fixed.

A.10 Execution Efficiency Measurements

Table 7 measures the optimizations in Section 5.4 using an earlier spectral inheritance implementation. Library storage is the on-disk architecture library size, CPU memory is the reported range during extended training, and batch time is measured during steady execution. These measurements compare execution pipelines rather than the runtime of the final SAHyper model and PITH.