Unlocking Pre-trained Weights: Parameter Inheritance for Zero-Shot Initialization

Supplementary Material

1. Computational Graph Construction

As described in the experimental setup (Section 2 background), the Graph HyperNetwork (GHN) takes a computational graph f^G as input to predict the parameters of the target network f. Here we provide a detailed explanation of how to represent a model architecture as a computational graph, using the Vision Transformer (ViT) as an example.

1.1. Graph Representation

The computational graph $f^G = (V, E)$ consists of three main components:

- **Node features (node_feat)**: Encoded operation types
- **Node information (node_info)**: Detailed metadata including shapes
- **Edge embeddings (edges_embedding)**: Connections between nodes

1.2. Example: Vision Transformer (ViT) Block

Below is a simplified example showing how a ViT block is represented as a computational graph:

```
node_feat:
tensor([
    [9],   # 'input'
    [13],  # 'pos_enc'
    (positional encoding)
    [5],   # 'msa'
    (multi-head self-attention)
    [12],  # 'ln'
    (layer normalization)
    [3],   # 'linear'
    (MLP layer 1)
    [7],   # 'sum'
    (residual connection)
    [3],   # 'linear'
    (MLP layer 2)
])

node_info:
[
    [[0, 'input', 'input',
    None, False, False]],
    [[1, 'pos_enc', 'pos_enc',
    (1, 768, 14, 14), False, False]],
    [[2, 'msa', 'msa',
    (1, 768, 14, 14), False, False]],
    [[3, 'ln', 'ln',
    (1, 768), False, False]],
    [[4, 'linear', 'linear',
    (768, 3072), False, False]],
    [[5, 'sum', 'sum',
    None, False, False]],
    [[6, 'linear', 'linear',
    (3072, 768),
    False, False]],
]

edges_embedding:
tensor([
    [0, 1, 1],  # input -> pos_enc
    [1, 2, 1],  # pos_enc -> msa
    [2, 3, 1],  # msa -> ln
    [3, 4, 1],  # ln -> linear
    [4, 5, 1],  # linear -> sum
    [5, 6, 1],  # sum -> linear
])
```

In this example, each node corresponds to an operation in the ViT architecture, and edges represent the forward pass flow of inputs through the network. The shape information in `node_info` (e.g., `(1, 768, 14, 14)`) indicates the tensor dimensions at each operation, which helps the GHN generate parameters of appropriate sizes.

2. Visual Domain Decathlon

The Visual Domain Decathlon Challenge [27] tests the ability of visual recognition algorithms to handle images from different visual domains. It includes 10 diverse datasets spanning various recognition tasks:

1. **ImageNet-1K** (IN-1K) is the largest dataset in the Decathlon Challenge, containing 1,000 categories and 1.2 million images.
2. **CIFAR-100** (C100) contains 60,000 32 $\times$ 32 color images for 100 object categories.
3. **Aircraft** (Airc.) contains 100 images for each of 100 different aircraft model variants, such as the Boeing 737-400 and the Airbus A310.
4. **Daimler Pedestrian Classification** (DPed) consists of 50,000 grayscale pedestrian and non-pedestrian images, cropped and resized to 18 $\times$ 36 pixels.
5. **Describable Textures** (DTD) is a texture database consisting of 5,640 images, organized into 47 categories such as bubbly, cracked andmarbled.

6. **German Traffic Signs** (GTSRB) contains cropped images for 43 common traffic sign categories in different image resolutions.
7. **Omniglot** (OGlt) consists of 1,623 different handwritten characters from 50 unique alphabets.
8. **SVHN** is a real-world digit recognition dataset with around 70,000 32×32 images.
9. **UCF101 Dynamic Images** (UCF) is an action recognition dataset of realistic human action videos, collected from YouTube. It contains 13,320 videos across 101 action categories. In the Decathlon Challenge, the videos are converted into images using Dynamic Image encoding, which summarizes each video into an image based on a ranking principle.
10. **Flowers102** (Flwr) is a fine-grained classification task with 102 flower categories from the UK, each consisting of 40 to 258 images.

These datasets cover diverse visual domains including natural images, fine-grained recognition, texture analysis, traffic scenes, and character recognition, providing a comprehensive evaluation of transfer learning and model adaptation capabilities. The detailed statistics of the datasets can be found at http://www.robots.ox.ac.uk/~vgg/decathlon/.

3. Detailed experimental results on ImageNet-1k

As described in Section 4 Experiments of the main paper, we evaluate different hypernetwork methods on their ability to initialize neural networks without any training. Table 1 presents the complete results on ImageNet-1K for different model configurations.

Table 1 shows the performance of different hypernetwork methods (GHN-3, LoGAH, TAL, and PITH) on ImageNet-1K. We test on Vision Transformer models with 6, 9, and 12 layers across three model sizes (Tiny, Small, Base). All models are directly initialized by the hypernetworks without any training.

4. Detailed Experimental Results of Analysis and Ablation

Experimental Setup. For ablation studies and analysis, we adopt the more compact architecture library ViTs-1K dataset [45], comprising 1000 ViT-style computational graphs with depths ranging from 3 to 9 layers and hidden dimensions ranging from 128 to 512, enabling more efficient experimentation. All hypernetwork methods are trained for 200 epochs, and we evaluate their performance across four representative configurations: 6-layer ViT-Tiny, 9-layer ViT-Tiny, 6-layer ViT-Small, and 9-layer ViT-Small architectures.

Table 1. Accuracy on ImageNet-1K initialized with different hypernetwork methods without any further training.

Model	GHN-3	LoGAH	TAL	PITH
6-Tiny	35.00	26.26	37.63	**46.38**
9-Tiny	35.02	26.33	37.69	**46.36**
12-Tiny	34.95	26.31	37.63	**46.35**
6-Small	35.36	44.78	46.71	**53.26**
9-Small	35.41	44.87	46.67	**53.34**
12-Small	35.33	44.82	46.74	**53.27**
6-Base	35.00	44.80	46.74	**53.50**
9-Base	34.88	44.80	46.86	**53.46**
12-Base	33.74	44.85	46.81	**53.35**

Detailed Ablation Results. Table 2 presents the comprehensive results of our ablation study on PITH's key components. The results demonstrate that both the pre-trained parameters projection (Proj.) and progressive dimension expansion (P.D.E.) contribute significantly to overall performance. Specifically, removing P.D.E. results in consistent performance degradation across all configurations, with the average accuracy dropping from 49.02% to 46.43% (2.59% decrease). Similarly, removing the projection mechanism leads to an average accuracy of 46.90%, representing a 2.12% performance loss. The full PITH model achieves superior performance across all tested configurations, demonstrating the necessity of both components.

Table 2. Detailed ablation study results on different ViT configurations. All methods are trained for 200 epochs on ViTs-1K dataset.

Model	**w/o P.D.E.**	**w/o Proj.**	**PITH**
6-Tiny	45.91	47.66	**49.70**
9-Tiny	46.08	47.63	**49.73**
6-Small	47.04	46.59	**48.18**
9-Small	46.71	45.75	**48.45**

Detailed Cost-Effectiveness Analysis. Table 3 provides a detailed comparison of different projection mechanism implementations. The baseline LoGAH [45] with 21.41M parameters serves as our reference point. Our dual-pathway implementation achieves higher accuracy but significantly increases the parameter count to 41.59M. To address this limitation, we propose the SHARED-PATHWAY variant (illustrated in Fig. 1), which strategically shares most MLP components between the weight generation pathway and projection matrix generation pathway. This parameter-sharing strategy reduces the hypernetwork parameters to 24.40M (only 13.9% increase over LoGAH) while maintaining competitive performance with 0.35% improvement over the baseline. This demonstrates that parameter pro-

jection from pre-trained models remains effective even with minimal architectural overhead.

Table 3. Detailed comparison of projection mechanism implementations across different ViT configurations. #Params: LoGAH (21.41M), Dual-Pathway (41.59M), Shared-Pathway (24.40M).

Model	LoGAH	Dual-Path	Shared-Path
6-Tiny	45.22	47.66	45.88
9-Tiny	45.15	47.63	45.93
6-Small	46.23	46.59	46.29
9-Small	46.34	45.75	46.22

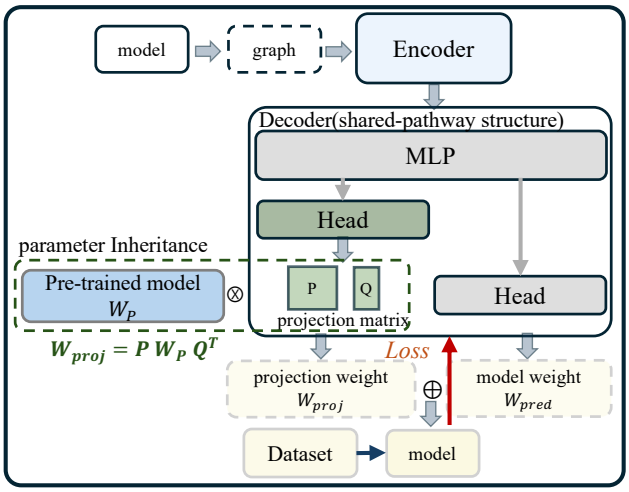

Figure 1. Architecture of the SHARED-PATHWAY variant. The weight generation pathway and projection matrix generation pathway share the majority of MLP components in the decoder, with separate output heads for generating projection matrices and model weights. This parameter-sharing design significantly reduces parameter overhead while maintaining the effectiveness of the dual-pathway mechanism.