LoGAH: Initialize Large Transformers via Small Graph HyperNetworks

Xinyu Zhou [1] **Boris Knyazev** [2] **Alexia Jolicoeur-Martineau** [2] **Jie Fu** [3]

Abstract

A good initialization of deep learning models is essential since it can help them converge better and faster. One recent and underexplored approach to a good initialization is to use Graph HyperNetworks (GHNs) to predict good model parameters given its computational graph. One key limitation of GHNs is that for very wide networks a GHN copies small predicted chunks of parameters multiple times and requires an extremely large number of parameters to support full prediction, which greatly hinders its adoption in practice. To address this limitation, we propose LoGAH (Low-rank GrAph Hypernetworks), a GHN with a low-rank parameter decoder that expands to significantly wider networks without requiring as excessive increase of parameters as in previous attempts. LoGAH allows us to predict the parameters of large neural networks in a memory-efficient manner. We show that vision models (i.e., ViT) initialized with LoGAH achieve better performance than those initialized randomly or using existing GHNs.

1. Introduction

A good initialization has always been essential to achieve optimal model performance (Glorot and Bengio, 2010; He et al., 2015; Mishkin and Matas, 2015; Huang et al., 2020). However, to train recent large vision and language models practitioners favor simple random-based initialization and focus on other aspects to increase performance, such as scale of data and models (Radford et al., 2018; Touvron et al., 2023; AI@Meta, 2024; Dosovitskiy et al., 2021; Dehghani et al., 2023). In general, the aspects like network architectures and datasets remain similar, e.g. Transformer-based architectures (Vaswani et al., 2023) and ImageNet (Russakovsky et al., 2015) (for vision) or The Pile (Gao et al., 2020) (for language) datasets.

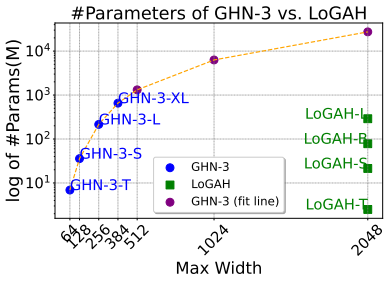

Figure 1: Comparison of parameter counts between GHN-3 and LoGAH. GHN-3 requires a larger hidden dimension to support wider networks (x axis), which increases the size of GHN-3 exponentially (y axis).

Leveraging this prior knowledge of the architecture and dataset may help to initialize models in a much stronger way. One potential approach to do so is Graph HyperNetworks (GHNs) (Zhang et al., 2018; Knyazev et al., 2021; 2023). This approach allows one to predict initial parameters of neural networks to converge faster and/or achieve better performance. Using a set of neural network architectures $\{f^G\}$ as training data, GHN $H_{\mathcal{D}}$, parameterized by θ, is trained to predict the parameters of these neural networks ($\mathbf{w}_{\text{pred}} = H_{\mathcal{D}}(f^G, \theta)$) to minimize the loss function on the dataset $\mathcal{D}$. The predicted $\mathbf{w}_{\text{pred}}$ can serve as a stronger initialization compared to random-based initialization methods. The key strength of GHNs is that a trained GHN $H_{\mathcal{D}}$ can predict parameters well even for unseen (e.g. wider and deeper) networks.

However, to predict parameters for very wide networks (often with a large number of parameters), previous GHNs (Knyazev et al., 2021; 2023) had to copy small chunks of parameters multiple times instead of fully predicting them due to the sheer amount of parameters required to predict all parameters, thus significantly limiting the performance of the resulting networks. Furthermore, to unlock the capability of predicting parameters of a larger size, GHNs need larger hidden sizes d, leading to an exponential increase in the number of parameters growing as $\mathcal{O}(d^3)$ (Figure 1).

To overcome this limitation, we propose LoGAH, a GHN with a low-rank parameter decoder. This novel approach not only supports significantly wider networks but also does so

[*]Equal contribution [1]EPFL, Switzerland [2]Samsung - SAIT AI Lab, Canada [3]Shanghai AI Lab, China. Correspondence to: Jie Fu <fujieATpjlab.org.cn>.

ES-FoMo-III Workshop, International Conference on Machine Learning (ICML), Vancouver, Canada.

without requiring an excessive number of parameters growing as $\mathcal{O}(d^2)$ instead of $\mathcal{O}(d^3)$. For instance, our smallest LoGAH-Tiny has only 2.5M parameters, yet it can predict parameters with up to 2048 channels, including ViT-Large (in 307M Parameters), without copying parameters.

In this work, we make the following contributions:

- We propose LoGAH with an improved low-rank decoder that is more scalable and can predict parameters of large networks without copying while having fewer trainable parameters and a lower training cost (Section 3).
- We create a new dataset of small ViT architectures, allowing GHNs to be trained on Transformers for vision tasks (Section 4). LoGAH shows excellent generalized capability on larger models.
- We outperform GHN-3 (Knyazev et al., 2023) as an initialization approach in multiple vision tasks by predicting more diverse and performant parameters (Section 5).

2. Preliminaries

2.1. Graph HyperNetworks

Graph HyperNetworks (GHNs) (Zhang et al., 2020; Knyazev et al., 2021) are widely used for neural networks' parameter prediction. The input fed to GHN $H_{\mathcal{D}}(\theta)$ is a computational graph f^G of a neural network f; GHN predicts its parameters $\mathbf{w}_{\text{pred}} = H_{\mathcal{D}}(f^G; \theta)$, where $\mathcal{D}$ is the training dataset. In our paper, f can be a ViT model (Dosovitskiy et al., 2021), and $\mathcal{D}$ can be the image classification task (i.e., CIFAR and ImageNet.).

Knyazev et al. (2021; 2023) trained GHN $H_{\mathcal{D}}$ by Adam over M training architectures $\{f_a^G\}_{a=1}^M$ and N training data samples $\{\mathbf{x}_j, y_j\}_{j=1}^N$ on the following optimization problem:

$$\arg\min_{\theta} \frac{1}{NM} \sum_{j=1}^{N} \sum_{a=1}^{M} \mathcal{L}(f_a(\mathbf{x}_j; H_{\mathcal{D}}(f_a^G; \theta)), y_j). \quad (1)$$

A meta-batch of m training architectures is sampled in the training stage where $H_{\mathcal{D}}$ predicts parameters. Meanwhile, a mini-batch of n training samples $\mathbf{x}$ is sampled and fed into the parameter-predicted m architectures to get $m \times n$ predictions. The cross-entropy loss $\mathcal{L}$ is computed for classification task. Afterwards, the loss is back-propagated to update the parameters θ of $H_{\mathcal{D}}$ by gradient descent. In our work, we created ViTs-1K datasets, consisting of 1K small training architectures, for predicting parameters for larger ViT models. We describe the details in Section 4.

The computational graph $f^G = (V, E)$ for input is a Directed Acyclic Graph (DAG), where V denotes the operations (e.g., pooling, self-attention, etc.), and E corresponds to the forward pass flow of inputs through f. The d-dimensional node features $\mathbf{H}^{(1)} \in \mathbb{R}^{|V| \times d}$ are obtained by an embedding layer (i-th node: $\mathbf{h}_i^{(1)} = \text{Embed}(\mathbf{h}_i^{(0)})$, where $\mathbf{h}_i^{(0)}$ is a one-hot vector representing an operation) and fed as the input for GHN. In GHN-3 (Knyazev et al., 2023), after L Graphormer layers (Ying et al., 2021), the node features $\mathbf{H}^{(L)} \in \mathbb{R}^{|V| \times d}$ are fed to the decoder described below.

2.2. GHN Decoder

Knyazev et al. (2021; 2023) have the decoder based on a simple MLP predicting a tensor of shape $d \times d \times 16 \times 16$, where d is relatively small ($d = 384$ even in the largest GHN-3). The decoder takes the output node features of the last Graphormer layer to predict parameters $\mathbf{w}_{\text{pred}}$. This tensor is copied when the target weight has a larger d or sliced when the target is smaller. The parameter count of the decoder in GHN-3 (Knyazev et al., 2021; 2023)[1] $\mathbf{P}_{\text{GHN}}$ is:

$$8d^3 + 4d^2 \times 16 \times 16 + 32d^2 + d \times \text{num\_class}. \quad (2)$$

3. Scalable Graph HyperNetworks: LoGAH

Our LoGAH model improves on the following aspects: (1) designing a novel low-rank decoder not only with fewer amounts of parameters, but also avoiding inefficient parameter repetitions on prediction, (2) supporting larger models (often wider) prediction without involving extremely larger amounts of parameters as in previous works, e.g. LoGAH-Tiny with only 2.5M parameters can in principle support ViT-Large or larger, while existing methods (Knyazev et al., 2023) would require $\sim 10^4$M parameters.

Low-Rank Decoder. In Knyazev et al. (2023), the final output dimensionality of the decoder is $d \times d \times h \times w$, where d can be 64 or 128, and typically $h = w = 16$. The key problem is that for large networks, the tensor needs to be repeated to fill all channels because d is small.

Considering a convolutional weight W with size: $(C_{out} \times C_{in} \times h \times w)$, we can reshape it into a matrix W of $(C_{out} \cdot h) \times (C_{in} \cdot w)$ where h, w are much smaller than C_{out} and C_{in}. Inspired by (Hu et al., 2021), we can now introduce the low-rank decomposition: $W = AB \in \mathbb{R}^{(C_{out} \cdot h) \times (C_{in} \cdot w)}$, where $A \in \mathbb{R}^{(C_{out} \cdot h) \times r}, B \in \mathbb{R}^{r \times (C_{in} \cdot w)}$, r denotes the low-rank. In this way, we reduce the amounts of parameters from $C_{out} \cdot C_{in} \cdot h \cdot w$ to $r \cdot ((C_{out} \cdot h) + (C_{in} \cdot w))$.

Therefore, the whole process is as follows: after the MLPs (multilayer perceptron) the input $\mathbf{H}^{(L)} \in \mathbb{R}^{|V| \times d}$ is transformed into $\tilde{W} \in \mathbb{R}^{|V| \times 2K \times r}$:

$$\tilde{W} = \text{MLP}(\mathbf{H}^{(L)}) \in \mathbb{R}^{|V| \times 2K \times r}, \quad (3)$$

where $K := \max(C_{out} \cdot h, C_{in} \cdot w)$, so that we can avoid repetition operations in GHN-3. Then we split $\tilde{W}$ into two matrices $A, B^T \in \mathbb{R}^{|V| \times K \times r}$ and only take the needed bits

[1] Please refer to Appendix C and https://github.com/SamsungSAILMontreal/ghn3/blob/main/ghn3/nn.py for more details

to construct $W = AB$. The architecture of the MLPs is shown in Appendix E, which involves the low-rank transformation inside. In this way, the number of parameters in the decoder of LoGAH is:

$$\mathbf{P}_{\text{LoGAH}} = 4d^2 + 32d^2 + 8d \times 2r^2 + r \times K. \quad (4)$$

Theoretically, we can fix r as a much smaller constant hyperparameter than d, then Equation 4 would be in $\mathcal{O}(d^2)$, less than the complexity of original GHN's decoder $\mathcal{O}(d^3)$. In practice, considering a small rank r would hinder the model's performance, so we set it to $r \approx \frac{d}{2}$ as an increase of d. Under this setting, we compare the amounts of two decoder's parameters in detail as follows.

#Parameters Reduction. Without loss of generality, we assume $K = C_{out} \cdot h$, and in our following settings for low-rank r (details in Table 3 in Appendix D)[2]: $r \approx \frac{d}{2}$. Then $\mathbf{P}_{\text{GHN}} - \mathbf{P}_{\text{LoGAH}}$ we obtain:

$$\begin{aligned}\Delta\mathbf{P} &= \mathbf{P}_{\text{GHN}} - \mathbf{P}_{\text{LoGAH}} && (5)\\ &= 4d^2 \times (16^2 - 1) - r \times C_{out} \cdot h && (6)\\ &\quad + 8d \times (d^2 - 2r^2) + d \times \text{num\_class}. && (7)\end{aligned}$$

Since $r \approx d/2$, $16^2 - 1 \approx 16^2$, and in our experiments we set $K = \max(C_{out} \cdot h, C_{in} \cdot w) = 2048 \cdot 16$, we can just compare the first term and second term in Eqn . (7):

$$\begin{aligned}\Delta_1 &= 4d^2 \times (16^2 - 1) - r \times C_{out} \cdot h && (8)\\ &\approx 4d^2 \times 16^2 - d \times 1024 \cdot 16 && (9)\\ &= 16d \cdot (64d - 1024). && (10)\end{aligned}$$

Therefore, $\Delta_1 > 0$ since in our settings $d = 64, 128, 256$, etc, which means that LoGAH's decoder requires fewer parameters ($\Delta\mathbf{P} > 0$), even if we let r increase with d.

Due to the low-rank mechanism, LoGAH can support predicting the parameter tensors with a larger shape but with fewer parameters. The parameters comparison between different versions of GHN-3 and LoGAH is shown in Figure 1. Since GHN-3 can only support the predicted parameters as the same width as the hidden dimension d, we fit the curve of GHN-3 and obtain the potential number of parameters needed to fully predict parameters with larger shapes. Compared to GHN-3, our LoGAH can support wider tensor shapes with much fewer parameters, which can support larger and wider models in practice (referring to Table 4).

4. ViTs-1K Datasets

For sampling training architectures in previous GHNs, Knyazev et al. (2021) built DeepNets-1M, a dataset of 1 million diverse computational graphs. While DeepNets-1M contains architectures with transformer layers, in most cases their transformer layers are mixed with other layers due to the random-based computational graph generation, so DeepNets-1M is not optimal to train a GHN for predicting ViT parameters. Therefore we introduce ViTs-1K, containing 1K different ViT-style computational graphs, particularly for training GHNs to predict ViT parameters.

ViTs-1K. We produce diverse ViT models by varying the number of layers L, heads H and hidden dimension D. Since ViT models have different scale versions (as illustrated in Table 4 of Appendix F), we also need to ensure that our training architectures will be diverse enough and uniformly distributed in terms of parameter count. Therefore, when generating these architectures, for deeper networks (with more layers) we control them to be narrower (with a smaller hidden dimension) and vice versa. Figure 5 shows the distribution of the amounts of parameters in ViTs-1K, which is almost uniformly distributed and the maximum parameters of these architectures are restricted to 10M (only around of half of ViT-Small's parameters). The details of ViTs-1K dataset's generation can be found in Appendix H.

5. Experiments

We evaluate if neural networks initialized with the parameters $\mathbf{w}_{\text{pred}}$ predicted by LoGAH can perform better than those by GHN-3 and random initialization after training.

LoGAH Variants. We design four different scales of LoGAH from Tiny to Large, by gradually increasing the number of layers L, hidden dimension d, heads H, as well as the low-rank r. We also vary the meta-batch size m for training GHN-3 and LoGAH, indicated at the end of the model name (e.g. /M1). We compare the number of parameters and estimate the training time difference between LoGAH with GHN-3, shown in Table 3. We highlight that GHN-3 and LoGAH are trained only once on each dataset, so that the same model can predict parameters for many architectures making the training cost of GHN-3 and LoGAH amortized. Both GHN-3 and LoGAH are trained on the ViTs-1K architectures for fair comparison. The details of GHN training setup is illustrated in Appendix I.

We test ViT-small and ViT-base on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and ILSVRC-2012 ImageNet (Russakovsky et al., 2015) with different initialization methods: (1) random initialization (RandInit) implemented by default in PyTorch, (2) orthogonal initialization (OrthInit) (Saxe et al., 2014), (3) parameters predicted by GHN-3, and (4) parameters predicted by LoGAH. We acknowledge that there are many other strong initialization methods (Dauphin and Schoenholz, 2019; Zhang et al., 2019; Trockman and Kolter, 2023) that we do not compare to. In this short work, we mainly aim to outperform GHN-3 which already outperformed such strong methods as (Zhu et al., 2021; Yang et al., 2022).

[2] Although in LoGAH-Large setting: $d = r = 256$, Eqn. (10) will obtain $16d \cdot (64d - 2048) > 0$ since d is very large.

5.1. Experiments on CIFAR-100

In the CIFAR-100 task, we train LoGAH-Tiny-m1 and LoGAH-Small-m1. The results are shown Table 1.

Low-rank decoder is more effective. Although LoGAH-Small has only 21.4M parameters, it achieves the best performance in ViT-Small and ViT-Base, much better than GHN-3-Large, which is almost $10\times$ larger. In detail, we gain $+0.53$ and $+5.39$ in accuracy on ViT-Small and ViT-Base respectively *vs* the best baseline. Additionally, in ViT-Small, LoGAH-Tiny-m1 is worse than GHN-3-Small and GHN-3-Large, which may imply that there is no significant difference when initializing smaller models. However, when the model size turns larger, the improvement becomes more obvious, from 53.95 by RandInit to 56.42 by LoGAH-Tiny-m1, while GHN-3-Large only achieves 52.80.

Table 1: CIFAR-100 top-1 accuracy (%) on ViT-Small and ViT-Base in different initialization settings. ViT models are trained for 100 epochs in each initialization setting.

Initialization	CIFAR-100	
	ViT-Small	ViT-Base
RandInit	53.97	53.95
OrthInit	49.76	48.38
GHN-3-T/m1	54.20	51.83
GHN-3-S/m1	55.57	52.71
GHN-3-L/m1	55.65	52.80
LoGAH-T/m1	54.47	56.42
LoGAH-S/m1	**56.18**	**59.34**
LoGAH-T/m8	57.48	58.52
LoGAH-S/m8	**59.67**	**60.11**

Increasing meta-batch can boost performances further. When setting meta-batch $m = 1$, we have already observed a huge improvement in both ViT-Small and ViT-Base. Now we investigate whether increasing the meta batch size can further boost the performance. Specifically, we train LoGAH-Tiny and LoGAH-Small with $m = 4, 8$ on the CIFAR-100 task, and then evaluate them.

The results are shown in Figure 6 and Figure 7 (in Appendix J). Increasing m can steadily stimulate the potential of LoGAH. For example, LoGAH-Tiny-m8 with $2.5M$ parameters can achieve 57.48 in ViT-Small and 58.52 in ViT-Base, compared with 55.65 and 52.80 via GHN-3-Large in $214.7M$ parameters.

5.2. Experiments on ImageNet

Based from experiment results on CIFAR-100, on ImageNet we train LoGAH with meta-batch size $m = 8$ directly. The evaluation results are shown in Table 2. With the increase of the LoGAH's scale, we can observe a steady improvement on the top-1 accuracy. LoGAH-Large/m8 achieves $+1.95$ and $+1.01$ enhancement over OrthInit on ViT-Small and ViT-Base, respectively.

The training loss, training top-1 accuracy and validation top-1 accuracy of ViT-Small on ImageNet initialized by RandInit and LoGAH-L/m8 are presented in Figure 2.

Table 2: ImageNet top-1 accuracy (%) on ViT-Small, ViT-Base, and ViT-Large in different initialization settings.

Initialization	ImageNet	
	ViT-Small	ViT-Base
RandInit (1 Epoch)	8.93	5.95
OrthInit (1 Epoch)	6.04	9.84
LoGAH-S/m8 (1 Epoch)	32.65	11.00
LoGAH-B/m8 (1 Epoch)	37.68	9.37
LoGAH-L/m8 (1 Epoch)	31.74	11.08
RandInit (50 Epochs)	62.04	62.53
OrthInit (50 Epochs)	62.08	62.96
LoGAH-S/m8 (50 Epochs)	62.65	63.74
LoGAH-B/m8 (50 Epochs)	63.01	63.80
LoGAH-L/m8 (50 Epochs)	**64.03**	**63.97**

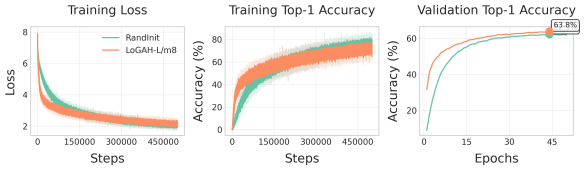

Figure 2: Loss and accuracy curves of ViT-Small comparisons between RandInit and LoGAH-L/m8 on ImageNet.

LoGAH-initialization can speed up the convergence and accuracy improvement at the early steps.

5.3. Transfer Learning Experiments

In this section, we explore the setting when LoGAH is trained on one dataset, but the predicted initialization is transferred to another dataset. We conduct the transfer learning experiments from CIFAR-100 to CIFAR-10, and from ImageNet to CIFAR-100.

CIFAR-100 to CIAFR-10. For transferring to CIFAR-10, we re-initialize the classification layer of ViT-Small or ViT-Base using a Kaiming normal distribution (He et al., 2015) with 10 outputs. Then we train the entire network for 100 epochs. The results are presented in Figure 8 in Appendix K. LoGAH trained on CIFAR-100 predicts initialization that is useful for CIFAR-10 improving on RandInit and OrthInit, which implies that LoGAH has transfer learning ability across different tasks in similar data distributions.

ImageNet to CIFAR-100. We keep the same setting as above. The results are shown in Figure 9 and Figure 10 in Appendix L. In this case LoGAH initialization does not transfer as well. This may be due to a larger distribution shift compared to our CIFAR-100 $\rightarrow$ CIFAR-10 experiment, which requires more investigation in future work.

6. Conclusion

In this work, we propose LoGAH, a low-rank Graph HyperNetwork (GHN) that provides a strong initialization for ViTs. We believe that data-driven initialization methods have a lot of potential to reduce training costs, which is especially important for large costly models. Our approach could be potentially further improved by training LoGAH on larger and more diverse ViT architectures. We also believe our approach is promising for the language transformers.

7. Acknowldgements

Jie Fu is supported by Shanghai Artificial Intelligence Laboratory.

References

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1, 4

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015. 1

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR, 2020. 1

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 1

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 1, 7

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md. 1

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1, 2, 7

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023. 1

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1, 7

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. 1, 3, 8

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. 1

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749*, 2018. 1, 7

Boris Knyazev, Michal Drozdzal, Graham W. Taylor, and Adriana Romero-Soriano. Parameter prediction for unseen deep architectures, 2021. 1, 2, 3, 7

Boris Knyazev, Doha Hwang, and Simon Lacoste-Julien. Can we scale transformers to predict parameters of diverse imagenet models?, 2023. 1, 2, 7, 8

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search, 2020. 2

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021. 2

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. *arXiv preprint arXiv:2106.09685*, 2021. 2

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3, 8

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014. 3

Yann Dauphin and Samuel S Schoenholz. Metainit: Initializing learning by learning to initialize. 2019. 3, 7

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019. 3

Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. In *International Conference on Machine Learning*, pages 34456–34468. PMLR, 2023. 3

Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W Ronny Huang, and Tom Goldstein. Gradinit: Learning to initialize neural networks for stable and efficient training. *arXiv preprint arXiv:2102.08098*, 2021. 3

Yibo Yang, Hong Wang, Haobo Yuan, and Zhouchen Lin. Towards theoretically inspired neural initialization optimization. *arXiv preprint arXiv:2210.05956*, 2022. 3, 7

Da Yin, Li Dong, Hao Cheng, Xiaodong Liu, Kai-Wei Chang, Furu Wei, and Jianfeng Gao. A survey of knowledge-intensive nlp with pre-trained language models, 2022. 7

Shangwei Guo, Chunlong Xie, Jiwei Li, Lingjuan Lyu, and Tianwei Zhang. Threats to pre-trained language models: Survey and taxonomy, 2022. 7

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423. 7

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. 7

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. 7

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 7

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20s.html. 7

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016. 7

Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4061–4070, June 2021. 7

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E. Turner. Fast and flexible multi-task classification using conditional neural adaptive processes, 2020. 7

Xixun Lin, Jia Wu, Chuan Zhou, Shirui Pan, Yanan Cao, and Bin Wang. Task-adaptive neural process for user cold-start recommendation. In *Proceedings of the Web Conference 2021*, WWW '21, page 1306–1316, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449908. URL https://doi.org/10.1145/3442381.3449908. 7

Andrey Zhmoginov, Mark Sandler, and Max Vladymyrov. Hypertransformer: Model generation for supervised and semi-supervised few-shot learning, 2022. 7

Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers, 2024. 7

Isaac Liao, Ziming Liu, and Max Tegmark. Generating interpretable networks using hypernetworks. *arXiv preprint arXiv:2312.03051*, 2023. 7

Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Max Vladymyrov, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information. *arXiv preprint arXiv:2201.05125*, 2022. 7

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. *arXiv preprint arXiv:2303.00980*, 2023. 7

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 7

Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4): 121–136, 1975. 7

A. Related Work

Large Models Pretraining. The large-scale pretrained models first appeared in the NLP field (Yin et al., 2022; Guo et al., 2022). The improvement and success are mainly attributed to self-supervised learning and Transformer (Vaswani et al., 2023). More and more large language models are developed based on it, extending to larger sizes for better performance under pretraining with massive data (Devlin et al., 2019; Brown et al., 2020; Touvron et al., 2023). Inspired by the advancement of Transformer, many Transformer-based vision models are also proposed, and some pretraining methods have been explored (Dosovitskiy et al., 2021; Carion et al., 2020; He et al., 2021; Chen et al., 2020). Our work focuses on predicting parameters for two Transformer-based models (ViT and GPT-2) to reduce pretraining costs.

Parameter Prediction. Hypernetworks (Ha et al., 2016) are often leveraged for predicting model's parameter. Many research works have extended the hypernetwork's capability to generalize on unseen architectures (Zhang et al., 2018; Nirkin et al., 2021; Knyazev et al., 2021), datasets (Requeima et al., 2020; Lin et al., 2021; Zhmoginov et al., 2022; Kirsch et al., 2024), or to generate interpretable networks (Liao et al., 2023). Our paper is also based on Graph HyperNetworks (GHNs), but overcomes the extreme increase of parameters needed in previous GHNs. LoGAH can support larger models with just 1% parameters, showing a better ability to predict parameters for larger networks.

Initialization and Learning to Grow Models. Several methods have improved on random initialization by learning from data (Dauphin and Schoenholz, 2019; Yang et al., 2022). However, GHN-3 (Knyazev et al., 2023) showed better performance making it a favourable approach to build on. Other methods learn to initialize a bigger model from a smaller pretrained model (Evci et al., 2022; Wang et al., 2023). These methods reduce training time, however, a smaller pretrained model of exactly the same architecture as the target model is not always available, which limits the approach.

B. Limitations

Although our model LoGAH shows outstanding performances compared to GHN-3 and other random initialization methods across the extensive experiments, there are still limitations. We also conduct the experiments on language tasks, however, we find it is difficult to observe the similar improvement on the language modelling, which indicate that the architecture of LoGAH may be required to be adapted to language models. Furthermore, to predict parameters for drastically novel architectures (e.g. (Gu and Dao, 2023)), the GHN might be needed to be trained to avoid a big distribution shift. In future work, it would be intriguing to show LoGAH's ability on modern LLMs (Touvron et al., 2023).

C. Details of the amounts of parameters of decoders in GHN-3

The theory amount of parameters of decoders in GHN-3 is shown below:

$$4 \times \text{in\_feature} \times d \times h \times w + \text{MLP\_d}_1 \times \text{MLP\_d}_2 \quad (11)$$
$$+ \text{MLP\_d}_2 \times d^2 + d \times \text{num\_class} \quad (12)$$

where in_feature is the input feature's dimension of the decoder (set as d in GHN-3), and MLP_d_1, MLP_d_2 denote the dimension of 1_{st} and 2_{nd} layers of MLP (set as $4d$ and $8d$ in experiments respectively), h, w are the last two dimensions of the predicted tensor's shape (set as 16) and num_class is the number of classes of the dataset. Thereby, we can simplify Equation (11) to (2).

D. Details of LoGAH variants and GHN-3 variants.

We include the details of LoGAH and GHN-3 in different scales, in Table 3.

E. Details of MLPs in the decoder of LoGAH

The MLPs has 4 layers and the activation function $\sigma(\cdot)$ is ReLU (Fukushima, 1975):

$$\mathbf{x} = M_3\Big(\sigma\Big(M_2\big(\sigma\big(M_1(\mathbf{H})\big)\big)\Big)\Big) \quad (13)$$
$$\mathbf{x} = \text{reshape}(\mathbf{x}) \in \mathbb{R}^{|V|\times 2r\times r} \quad (14)$$
$$\mathbf{x} = \text{reshape}\big(M_4(\sigma(\mathbf{x}))\big) \in \mathbb{R}^{|V|\times 2K\times r} \quad (15)$$

Table 3: Details of LoGAH variants and GHN-3 variants. All LoGAH variants are set with $K = 2048 \cdot 16$. We estimate the train time of each model based on meta-batch $m = 1$ and the CIFAR-100 dataset for 300 epochs.

Model	*r*	*L*	*d*	*H*	Max Width	P	Train Time
LoGAH-Tiny	32	3	64	8	2048	2.5M	7.05h
LoGAH-Small	90	5	128	16	2048	21.4M	7.25h
LoGAH-Base	128	5	256	16	2048	78.2M	10.30h
LoGAH-Large	256	12	256	16	2048	289.4M	21.0h
GHN-3-Tiny	-	3	64	8	64	6.9M	7.20h
GHN-3-Small	-	5	128	16	128	35.8M	7.75h
GHN-3-Large	-	12	256	16	256	214.7M	12.40h
GHN-3-XLarge	-	24	384	16	384	654.4M	24.0h

where $M_i, i \in \{1, 2, 3, 4\}$ are learnable matrices:

$$M_1 \in \mathbb{R}^{d \times 4d}, M_2 \in \mathbb{R}^{4d \times 8d}$$
$$M_3 \in \mathbb{R}^{8d \times 2r^2}, M_4 \in \mathbb{R}^{r \times K}$$

We also provide the code implementation of it as shown in Figure 3.

F. Details of variants of ViT models

We provide the details of ViT in different sizes. L, D, H, P denotes the numbers of layers, heads, hidden dimension and parameters, respectively.

Model	*L*	*D*	MLP size	*H*	P
ViT-S	12	384	1536	6	22M
ViT-B	12	768	3072	12	86M
ViT-L	24	1024	4096	16	307M

Table 4: Details of ViT variants

G. Distribution of ViTs-1K datasets

The distributions of ViTs-1K is shown in Figure 5.

H. Details of generating ViTs-1K dataset

As mentioned above, we change the values in layers L, heads H, and hidden dimension D of ViT, as well as restricting these models size. The details are shown in Figure 4.

I. Details of GHN Training Setup.

The GHN models, including GHN-3 and our LoGAH, are trained for 300 epochs on ViTs-1K. In detail, we conduct experiments on the following datasets: CIFAR-100 (Krizhevsky et al., 2009) (with batch size $b = 64$) and ILSVRC-2012 ImageNet (Russakovsky et al., 2015) (with batch size $b = 128$). When setting meta-batch $m = 1$, we train the models using automatic mixed precision in PyTorch with a cosine annealing learning rate schedule starting at $lr = 1e^{-3}$, weight decay $\lambda = 1e^{-2}$, and predicted parameter regularization $\gamma = 3e^{-5}$ (Knyazev et al., 2023). All GHN models, including GHN-3 and LoGAH, are trained separately on each task dataset.

J. Performances of LoGAH-Tiny/Small trained by different meta-batch m in and ViT-Small and ViT-Base on CIFAR-100.

The performances of LoGAH-Tiny/Small with different meta-batch in ViT-Small and ViT-Base on CIFAR-100 are presented in Figure 7.

K. Transfer experiments from CIFAR-100 to CIFAR-10

The transfer learning experiments from CIFAR-100 to CIFAR-10 on ViT-Small is presented in Figure 8.

L. Transfer experiments from ImageNet to CIFAR-100

The transfer learning experiments from ImageNet to CIFAR-100 on ViT-Small and ViT-Base are shown in Figure 9 and Figure 10.

```
class ConvDecoder3LoRA(nn.Module):
   def __init__(self,
             in_features,
             ck=32,
             r=32,
             hid=(64,),
             num_classes=None):
      super(ConvDecoder3LoRA, self).__init__()

      assert len(hid) > 0, hid
      self.r = r
      self.ck = ck
      self.num_classes = num_classes
      self.mlp = MLP(in_features=in_features,
                  hid=(*hid, r*2*r),
                  activation='relu',
                  last_activation=None)

      self.l2 = nn.Linear(int(r), ck)
      self.relu = nn.ReLU(inplace=True)

      self.seq = nn.Sequential(
         self.relu,
         self.l2
      )


   def forward(self, x, max_shape=(1,1,1,1), class_pred=False, n_dim = 4):
      if class_pred:
         n_dim = 2
      x = self.mlp(x).view(-1, 2*self.r, self.r) # [b, 2*r, r]
      x = self.seq(x).view(-1, 2*self.ck, self.r) # [b, 2*ck, r]
      A, B_t = torch.split(x, self.ck, dim=1) # A=[b, ck, r] and B=[b, ck, r]
      B = B_t.transpose(1,2) # A=[b, ck, r] and B=[b, r, ck]
      # fix shape of A and B before matmul through indexing
      c_out, c_in, k_out, k_in = max_shape
      A = A[:, :(c_out*k_out), :] # [b, c_out*k_out, r]
      B = B[:, :, :(c_in*k_in)] # [b, r, c_in*k_in]
      W = torch.bmm(A, B) # [b, c_out*k_out, c_in*k_in]
      if n_dim == 1: # We want [c_out]
         assert c_in == 1 and k_out == 1 and k_in == 1
         W = W.reshape(-1, c_out)
      elif n_dim == 2: # we already have a 2D matrix
         pass
      elif n_dim == 4:
         W = W.reshape(-1, c_out, k_out, c_in, k_in).transpose(2, 3) # [b, c_out, c_in,
            k_out, k_in]
      else:
         raise NotImplementedError("n_dim must be 1 or 2 or 3")
      #print(W.shape)
      return W
```

Figure 3: Code for Low-rank decoder in LoGAH.

```
layers = np.random.randint(3, 10)
if layers > 5:
   dim_min = 128
   dim_max = 256
elif layers > 3:
   dim_min = 256
   dim_max = 384
else:
   dim_min = 384
   dim_max = 512

hidden_dim = np.random.choice(np.arange(dim_min, dim_max+1, 32))
mlp_dim = hidden_dim * 4

if hidden_dim % 12 == 0:
   heads = np.random.choice([3, 6, 12])
elif hidden_dim % 6 == 0:
   heads = np.random.choice([3, 6])
elif hidden_dim % 3 == 0:
   heads = 3
else:
   heads = np.random.choice([4, 8])



net = _vision_transformer(
   patch_size = 2,
   num_layers = layers,
   num_heads = heads,
   hidden_dim = hidden_dim,
   mlp_dim = mlp_dim,
   num_classes = 100,
   image_size = 32,
   weights = None,
   progress = False,
)
```

Figure 4: Code for generating ViT-style models used for VITS-1K dataset.

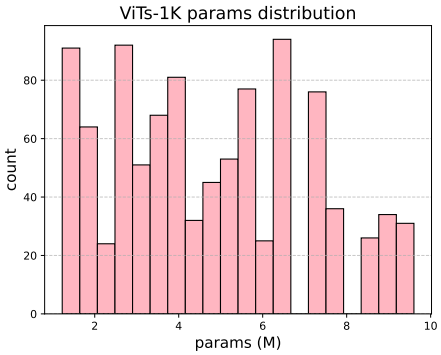

Figure 5: The parameters distribution in ViTs-1K.

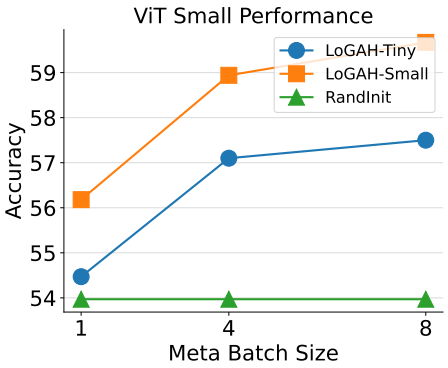

Figure 6: Performances of LoGAH-Tiny/Small trained by different meta-batch m in ViT-Small on CIFAR-100.

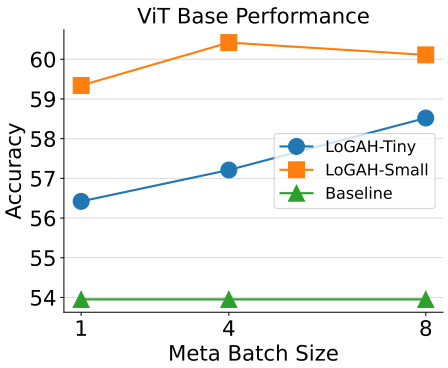

Figure 7: Performances of LoGAH-Tiny/Small trained by different meta-batch m in ViT-Base on CIFAR-100.

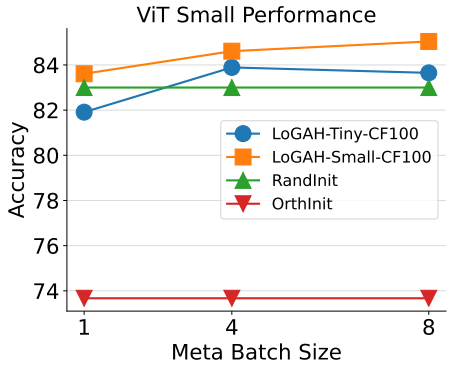

Figure 8: Performances of LoGAH-Tiny/Small trained on CIFAR-100 by different meta-batch m in ViT-Small on CIFAR-10.

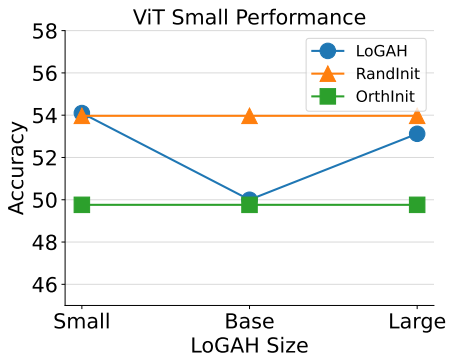

Figure 9: Performances of LoGAH-Small/Base/Large-M8 trained on ImageNet with meta-batch $m = 8$ in ViT-Small on CIFAR-100.

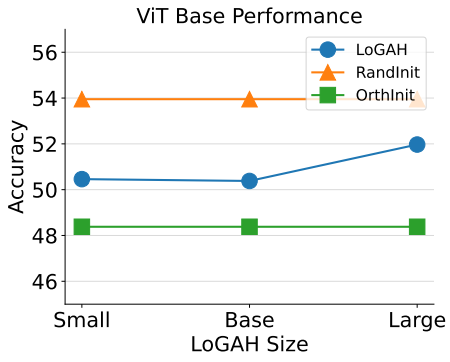

Figure 10: Performances of LoGAH-Small/Base/Large-M8 trained on ImageNet with meta-batch $m = 8$ in ViT-Base on CIFAR-100.