Efficient Domain Adaptation of Robotic Foundation Models via Hypernetwork-Generated LoRA

Siddhant Sharma[*]
Department of Computer Science
University of Oxford

Zheng Xiong[*]
Department of Computer Science
University of Oxford

Kang Li
Department of Statistics
University of Oxford

Risto Vuorio
Department of Computer Science
University of Oxford

Shimon Whiteson
Department of Computer Science
University of Oxford

Abstract

This paper investigates how to efficiently adapt a pre-trained robotic foundation model to a new domain with many different tasks to solve. We introduce Hyper-LoRA, a method built upon LoRA and Hypernetworks (HNs), to make this domain adaptation process both parameter-efficient via low-rank adaptation, and data-efficient by sharing knowledge across tasks in the target domain via the HN. By training Hyper-LoRA on a moderate number of multi-task demonstrations from the target domain, we achieve not only significantly better performance on the training tasks, but also promising zero-shot generalization to unseen tasks.

1 Introduction

Foundation models have recently achieved great success in different domains like NLP [1, 21, 8] and CV [7]. In the domain of robotics, robotic foundation models (RFMs) [9] can help a robot make better decisions from high-level task planning to low-level action control, providing a promising approach towards learning a truly generalist robot. In this paper, we focus on a specific type of RFM that directly outputs action commands based on image observations and language instructions, which is also known as Vision-Language-Action (VLA) model in previous work [4, 5, 18, 24, 14]. By training on large-scale real-world robotic data collected from diverse scenes and different robots, these RFMs have achieved much stronger performance and better generalization than conventional robotic controllers trained on a narrow distribution of tasks.

However, the performance of these RFMs is still far from perfect, and usually require further fine-tuning to achieve good performance in downstream tasks. While some recent work has investigated how to efficiently fine-tune an RFM on a single task [24, 14], how to adapt an existing RFM to a target domain with *multiple different tasks* in a parameter- and data-efficient way remains under-explored. In this paper, we define a domain as an environment in which a robot needs to accomplish a set of different tasks, e.g., imagine training industrial robots that can work in factories. After training an RFM on data collected from source domains, we want to deploy it in factories that may differ in the environments, tasks to complete, etc. Directly deploying the RFM in a zero-shot fashion may not

[*]Equal contribution. Correspond to `zheng.xiong@cs.ox.ac.uk`.

perform well due to the domain gap between the source and target domains, and further adaptation is required before deployment. We want the domain adaptation process to be both data-efficient to minimize the efforts for collecting expert demonstrations in each new factory, and parameter-efficient to reduce computational cost and further improve data efficiency.

To achieve this goal, we introduce Hyper-LoRA, which builds upon two key components: (1) Low-Rank Adaptation (LoRA) [13], a popular parameter-efficient fine-tuning method for foundation models; and (2) Hypernetworks (HNs) [11], networks that generate the parameters of a base network. While LoRA is commonly used for single-task fine-tuning, we extend it to our domain adaptation setting by generating task-specific LoRA parameters for different tasks in the target domain with a task-conditioned HN. Our method is both parameter-efficient with LoRA adaptation, and data-efficient by sharing knowledge across different tasks in the target domain via the HN. Preliminary experimental results show that Hyper-LoRA not only significantly improves the model's performance on the training tasks from the target domain, but also shows promising zero-shot generalization to unseen tasks without any demonstrations.

2 Background

2.1 Problem formulation

An RFM, or more specifically a VLA model, is usually trained via behavior cloning on real-world robotic demonstrations collected from many different scenarios.

Given such an RFM pre-trained on some source domains, we investigate how to efficiently adapt it to an unseen target domain with a set of different tasks $\mathcal{T}_{\text{target}}$ to accomplish. The new domain may be different from the pre-training domains in scenarios, task set, visual appearance, robot setup, etc. Each task from the target domain is specified by a sentence of language instruction.

To enable efficient domain adaptation, we assume access to expert demonstrations on a task subset $\mathcal{T}_{\text{train}} \subset \mathcal{T}_{\text{target}}$ in the target domain. Each demonstration is a trajectory of observation-action pairs $(o_0, a_0, \cdots, o_H, a_H)$ with episode length H. We want to fine-tune the RFM with these demonstrations from the target domain to achieve not only better performance on the training tasks $\mathcal{T}_{\text{train}}$, but also better generalization performance on unseen test tasks $\mathcal{T}_{\text{test}} = \mathcal{T}_{\text{target}} \setminus \mathcal{T}_{\text{train}}$ in the same domain.

2.2 Low-rank adaptation

Low-rank adaptation (LoRA) was first proposed for parameter-efficient fine-tuning of large language models [13], and has recently been successfully applied to RFMs as well [14]. Instead of fine-tuning all the parameters of a large model, LoRA freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the model, which significantly reduces the number of trainable parameters during fine-tuning with little performance degradation. For a linear layer $h = \mathbf{W}x$ with weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$, LoRA introduces two low-rank matrices $\mathbf{W}_{\text{down}} \in \mathbb{R}^{m\times r}$ and $\mathbf{W}_{\text{up}} \in \mathbb{R}^{r\times n}$, where $r \ll \min(m, n)$ is the rank of LoRA matrices, and updates the layer output as

$$h = \mathbf{W}x + \alpha \mathbf{W}_{\text{up}} \mathbf{W}_{\text{down}} x,$$

where α is a hyperparameter determining how much the additional term influences the layer output. During fine-tuning, $\mathbf{W}$ is frozen and only $\mathbf{W}_{\text{up}}$ and $\mathbf{W}_{\text{down}}$ are updated, which reduces the number of trainable parameters from $m \times n$ to $(m + n) \times r$.

2.3 Hypernetworks

A hypernetwork (HN) [11] is a network that generates the parameters of a base network. In the context of multi-task learning, HNs are usually used by taking some task context c as input to generate different parameters for each task in a task-conditioned way.

Specifically, we can decompose an HN into a context encoder f and several output heads g (parameterized as linear layers) that generate different parameter blocks in the base network. To generate the parameters of a linear layer $h = \mathbf{W}x$ (the bias term is omitted for simplicity, and can be generated in a similar way), the HN first encodes the task context as $e = f(c)$, then passes the context embedding into a linear output head g_W to generate $\mathbf{W} = g_W(e)$.

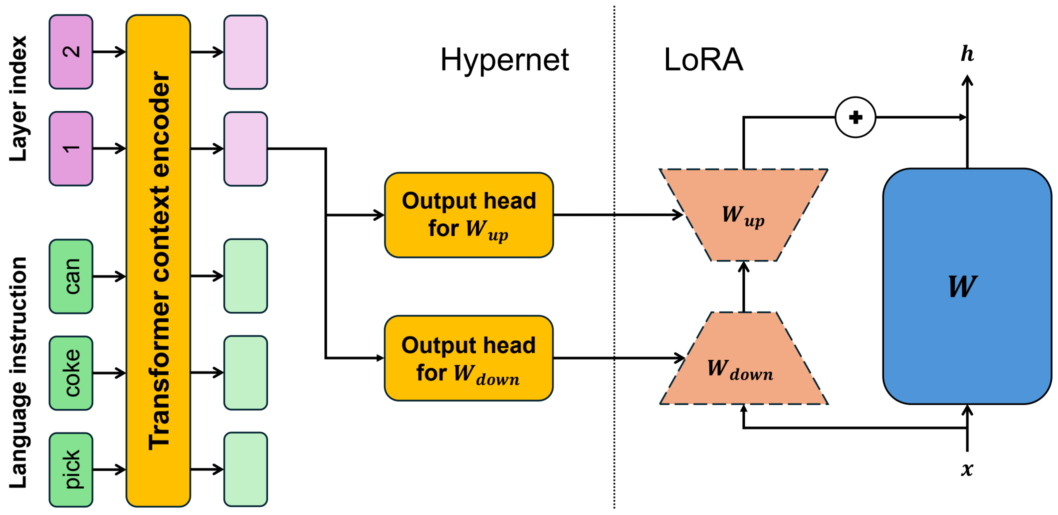

Figure 1: The framework of Hyper-LoRA. For ease of illustration, we only show how to generate LoRA parameters for a weight matrix in a single layer of the RFM. The HN takes in a language instruction and layer indexes as token inputs, and encodes them with a Transformer. The context embeddings of the layer index tokens are passed into the output heads to generate the LoRA parameters for different layers in the RFM. Trainable parameters are marked in orange, and HN-generated parameters are boxed by dashed lines.

3 Hyper-LoRA

3.1 Motivation

LoRA is a commonly used parameter-efficient fine-tuning technique in foundation models. However, it is usually used for fine-tuning on a single task in previous work. To apply LoRA in our problem setting, a straightforward approach is to learn a separate set of LoRA parameters for each training task in the target domain. However, doing so forfeits the chance to share knowledge across different tasks to facilitate the data efficiency of domain adaptation. Furthermore, it is not clear which learnt LoRA module should we use for an unseen task from the target domain. Finally, the memory costs of saving the LoRA parameters grow linearly with the number of tasks seen in the new domain.

To tackle these limitations, one solution is to learn a multi-task LoRA (MT-LoRA) module on all the training tasks from the target domain, and apply the same LoRA module to any new tasks. However, learning a single LoRA module that is agnostic to task context may not be expressive enough to model the diverse skills required to solve different tasks, and thus may harm domain adaptation performance.

Therefore, we propose to learn LoRA modules that explicitly condition on task context. Such task conditioning can be modeled in different ways, such as feeding task context as an additional input to LoRA ($\mathbf{W}_{\text{down}}$ specifically), feature-wise linear modulation [20], and HN. In this paper we adopt the HN approach, as it has strong model capacity to encode the complicated dependency between task context and control parameters, supported by both theoretical analysis [10, 23] and empirical evidence [29, 19, 2, 22, 3, 26] in previous work.

3.2 Model architecture

As shown in Figure 1, Hyper-LoRA generates task-specific LoRA parameters via a context-conditioned HN. We highlight some key designs in Hyper-LoRA as follows:

Reducing HN size Most parameters of an HN are in its output heads, as the total output dimension of these heads equals to the number of parameters to generate in the base network. Although each LoRA block contains just two low-rank matrices, generating LoRA blocks for all the matrices in a foundation model still introduces a considerable memory cost. To reduce the size of the HN, we utilize the fact that most foundation models are built by stacking multiple layers of Transformer (TF) blocks [25] with the same architecture. We thus feed the layer index of the RFM as an additional input to the HN, so that we can reuse the HN's output heads to generate different LoRA parameters for different layers in the RFM, instead of learning separate output heads for each layer.

Improving generalization of HN As an HN generates parameters based on task context, it may be prone to overfitting when the task context is not diverse enough in the training data, e.g., if the HN encounters a new object not seen during training, the context encoder will likely not encode the token of the new object correctly, and may generate LoRA parameters that will lead to unexpected control policies. To tackle overfitting and improve generalization of the HN, we apply dropout regularization to the context embedding before the output heads [27], and find that this simple strategy improves generalization of Hyper-LoRA to unseen tasks in the new domain.

Due to space limitation, see Appendix 5.1 for related work and how Hyper-LoRA differs from them.

4 Experiments

We conduct experiments to validate if Hyper-LoRA can: (1) Improve the RFM's performance on the training tasks in the target domain. (2) Zero-shot generalize to unseen tasks from the target domain.

4.1 Experimental setup

Pre-trained RFM We use Octo [24] as the pre-trained RFM, as it achieves a good trade-off between performance and computational cost, and is easy to fine-tune based on its open-sourced code. Octo is pre-trained on the Open X-Embodiment (OXE) dataset [18] that contains real-world robotic manipulation demonstrations collected from diverse robots in different scenarios.

Target domain We use SIMPLER [15] as the target domain. SIMPLER reproduces a subset of OXE tasks in simulation to facilitate reproducible and easier evaluation of RFMs in simulation instead of on real robots. Although SIMPLER is calibrated to better align with the real robots, there still exists a real-to-sim gap between OXE and SIMPLER, and Octo is shown to be sensitive to this domain gap [15], which makes SIMPLER a good choice for the target domain. We also include more target domains for a more thorough evaluation in the future.

Demonstration collection We split the tasks in SIMPLER into two disjoint sets for training and test respectively (see Appendix 5.2 for more details), and only collect demonstrations on the training tasks. As SIMPLER does not provide any oracle policy for each task and it is time consuming to train an expert policy for each task from scratch, we adopt a much cheaper approach by running an RFM on each task and collecting successful episodes as demonstrations. We consider both Octo and RT-1-X [18] as the RFM for demo generation. We collect about 300 demonstrations in total for domain adaptation.

Baselines We compare Hyper-LoRA with (1) The pre-trained RFM, to show the importance of domain adaptation to achieve good performance in a target domain; and (2) Multi-task LoRA (MT-LoRA) as discussed in Section 3.1, to show the advantage of introducing HN for task conditioning.

Setup of Hyper-LoRA We compare three variants of Hyper-LoRA: (1) Use an MLP as the context encoder; (2) Use a Transformer (TF) as the context encoder; (3) Add dropout regularization to the TF context encoder to improve task generalization. See Appendix 5.3 for the architecture configurations of each variant.

Training and evaluation We train each method with behavior cloning for 100k steps with a batch size of 32. During the fine-tuning process, we freeze the TF parameters in Octo, add LoRA to all the MLP layers in each TF layer, and further update the parameters of the diffusion head. In total about 6M parameters are updated during domain adaptation, taking only about 3% of the parameters in Octo. Other fine-tuning hyperparameters follow the same setup as in Octo [24].

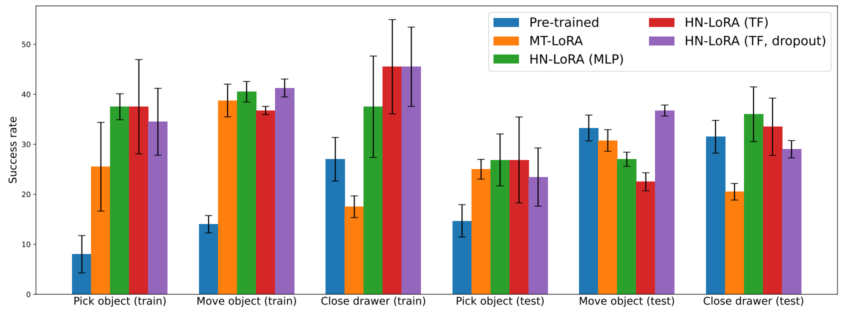

Figure 2: Mean success rate of different methods on the training and test tasks after fine-tuning on expert demonstrations generated by RT-1-X, with standard deviation over 4 random seeds.

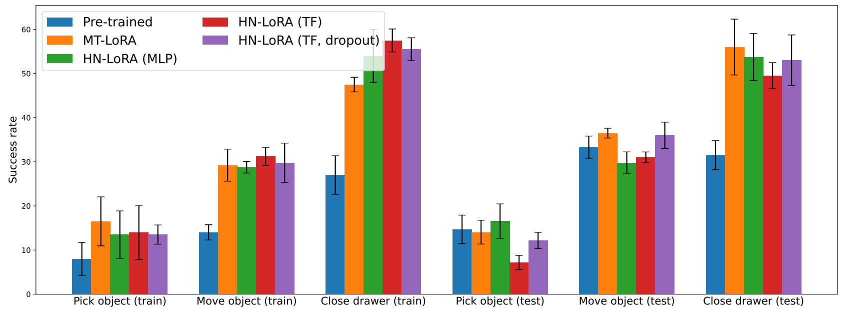

Figure 3: Mean success rate of different methods on the training and test tasks after fine-tuning on expert demonstrations generated by Octo itself, with standard deviation over 4 random seeds.

After fine-tuning, we evaluate the adapted model on both the training tasks with parametric variations and unseen test tasks with new instructions. As Octo's diffusion head [6] generates actions in a nondeterministic way, we evaluate each model on 4 random seeds and report the mean and standard deviation of success rate for each method.

4.2 Results

The success rate on the training and test tasks after fine-tuning with demonstrations generated by RT-1-X or Octo is shown in Figure 2 and 3 respectively. We find that: (1) Both MT-LoRA and Hyper-LoRA significantly outperform the pre-trained model on most training tasks from the new domain, validating the importance of domain adaptation. (2) Hyper-LoRA outperforms MT-LoRA on most training tasks, validating the importance of introducing HN. (3) Using either an MLP or TF context encoder does not have a significant influence on the performance of Hyper-LoRA, while dropout helps on the pick and move tasks that require generalization across different objects.

However, the advantage of Hyper-LoRA to both the pre-trained model and MT-LoRA is less significant on the test tasks, indicating that Hyper-LoRA is more prone to overfitting. We hypothesize this is because that Hyper-LoRA only sees a small set of training tasks with different instructions, thus can not generalize well to out-of-distribution task instructions with unseen objects or tokens. Moreover, some results are inconsistent between the two different ways of generating demonstrations, e.g., although RT-1-X outperforms Octo on the close drawer task, the models fine-tuned with RT-1-X's demonstrations perform much worse on this task than their counterparts fine-tuned with Octo's demonstrations. We plan to tackle the task generalization and inconsistency issues in the future.

5 Conclusion and future work

This paper introduces Hyper-LoRA for efficient adaptation of RFMs to new domains with many different tasks by learning to generate task-specific LoRA parameters with an HN. Hyper-LoRA not only significantly boosts model performance on the training tasks in the new domain, but also shows some extent of zero-shot generalization to unseen test tasks. The experimental results in this paper are still in a preliminary stage, and future work will focus on:

1. Evaluate Hyper-LoRA on more target domains with different extent of domain gaps to the source domains, such as Meta-World [30] and CALVIN [17].
2. Further improve the zero-shot generalization ability of Hyper-LoRA to unseen tasks in the target domain.
3. Ablation study on some key design choices that may have a large influence on Hyper-LoRA performance, such as the amount of demonstrations, the number of training tasks, etc.

Acknowledgments

We would like to thank the authors of SIMPLER for their helpful discussion on the implementation details of SIMPLER. We would also like to thank the workshop reviewers for their constructive feedback on the paper. Zheng Xiong is supported by UK EPSRC CDT in Autonomous Intelligent Machines and Systems (grant number EP/S024050/1) and AWS. Risto Vuorio is supported by EPSRC Doctoral Training Partnership Scholarship and Department of Computer Science Scholarship. The experiments were made possible by a generous equipment grant from NVIDIA.

References

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

[2] Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *6th Annual Conference on Robot Learning*, 2022.

[3] Jacob Beck, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Recurrent hypernetworks are surprisingly strong in meta-rl. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[7] 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. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

[8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[9] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *arXiv preprint arXiv:2312.07843*, 2023.

[10] Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33:10409–10419, 2020.

[11] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[13] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[14] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[15] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[16] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021.

[17] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[18] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[19] Matt Peng, Banghua Zhu, and Jiantao Jiao. Linear representation meta-reinforcement learning for instant adaptation. *arXiv preprint arXiv:2101.04750*, 2021.

[20] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[21] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[22] Sahand Rezaei-Shoshtari, Charlotte Morissette, Francois Robert Hogan, Gregory Dudek, and David Meger. Hypernetworks for zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:2211.15457*, 2022.

[23] Elad Sarafian, Shai Keynan, and Sarit Kraus. Recomposing the reinforcement learning building blocks with hypernetworks. In *International Conference on Machine Learning*, pages 9301–9312. PMLR, 2021.

[24] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[25] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[26] Zheng Xiong, Jacob Beck, and Shimon Whiteson. Universal morphology control via contextual modulation. In *International Conference on Machine Learning*, pages 38286–38300. PMLR, 2023.

[27] Zheng Xiong, Risto Vuorio, Jacob Beck, Matthieu Zimmer, Kun Shao, and Shimon Whiteson. Distilling morphology-conditioned hypernetworks for efficient universal morphology control. *arXiv preprint arXiv:2402.06570*, 2024.

[28] Mengdi Xu, Yuchen Lu, Yikang Shen, Shun Zhang, Ding Zhao, and Chuang Gan. Hyper-decision transformer for efficient online policy adaptation. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=AatUEvC-Wjv`.

[29] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Multi-task reinforcement learning without interference. In *Proc. Optim. Found. Reinforcement Learn. Workshop NeurIPS*, 2019.

[30] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

Supplementary Material

5.1 Related work

Inspired by the great success of foundation models in domains like NLP and CV, robotic learning has recently shifted from tailoring models for specific downstream tasks to a new paradigm of learning a generalist model on a broad distribution of large-scale robotic data [9]. Vision-Language-Action (VLA) models are one type of RFMs that directly predict control actions based on visual inputs and language instructions. RT-1 [4] trains a Transformer-based model from scratch on real-world manipulation data with behavior cloning. RT-2 [5] trains on the same robotic data as RT-1, but utilizes existing vision-language models with strong semantic reasoning ability to improve generalization of the robot. RT-X [18] further trains a cross-embodiment RFM on data collected from many different robots, and shows positive transfer across embodiments. Octo [24] and OpenVLA [14] introduce further algorithm and architecture improvements, such as action prediction with diffusion policy [6], and how to improve visual encoding. Octo also investigates data-efficient fine-tuning on unseen tasks, while OpenVLA shows that the fine-tuning process can also be parameter-efficient by using LoRA [13]. However, neither of them considers efficient adaptation to a new domain with multiple tasks.

Similar to our work, Mahabadi et al. [16] utilize HN for multi-task fine-tuning in NLP, and show positive knowledge transfer across tasks. Most similar to our work is Hyper-Decision Transformer (HDT) [28], which also utilizes HN to generate task-specific adapter [12] parameters for efficient adaptation. However, HDT only considers state-based robotic control in a relatively narrow task distribution, while we focus on a more challenging setting of image-based control on a much broader task distribution based on the recent progress in RFMs. Moreover, HDT uses HN to generate a good initialization for the adapter but still requires demonstrations on unseen tasks for further fine-tuning, while we aim to achieve zero-shot generalization to unseen tasks.

5.2 Task split in SIMPLER

We select three types of Google robot tasks from the SIMPLER benchmarks as the tasks in the target domain: pick object, move object, and close drawer. We discard the open drawer task as the pre-trained Octo can hardly generate any successful episode for this task.

Each type of task contains multiple different tasks with different instructions. Task difference can be different objects to manipulate for the pick and move tasks, and the drawer location (top, middle or bottom) for the close drawer task. Each task also has parametric variations such as the initial positions of the robots and objects, but this parametric information is not reflected in the language instruction and is not available to the model.

The pick task provides 14 different objects in total. We use 10 for training and hold out 4 for test. For the move task, three objects are given for each episode, and the goal is to move object A near object B, while the distractor object C will not appear in the language instruction. SIMPLER provides 5 different triplets for the move task, and 6 different tasks can be defined over each triplet. We hold out 1 triplet (containing 6 tasks) to test generalization to unseen object combination. For the remaining 4 triplets, we hold out 2 pairs of source and target objects for each triplet to test generalization to unseen source and target pairs. The remaining 16 tasks are used for training. For the close drawer task, we use close top and bottom drawer for training, and close middle drawer for test. See Table 1 for more details on the task split.

Task type	# Training	# Test	# demonstrations per training task	# Evaluation per test task
Pick	10	4	10	20
Move	16	14	10	10
Close drawer	2	1	50	50

Table 1: Task split on SIMPLER.

5.3 Further experimental details

We generate LoRA modules with a rank of 4. For MT-LoRA, the rank is increased to 48 so that it has a similar number of trainable parameters as Hyper-LoRA.

For Hyper-LoRA, the context embedding dimension is set to 128. For the MLP context encoder, we use a two-layer MLP with a hidden size of 128. For the TF context encoder, we use a two-layer TF with 4 attention heads, and the hidden size of the MLP in each TF block is set to 256. When dropout is applied to the context embedding, we use a dropout rate of 0.1.