Multi-Task Generalization for Robotics

Zheng Xiong
Mansfield College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity 2025

Acknowledgements

First of all, my great thanks go to my DPhil supervisor Shimon Whiteson. Thank you for supporting me to freely explore different directions, giving insightful feedback all the time from high-level research ideas to polishing my papers word by word, and all the help you have generously provided during my DPhil. I have learned many valuable lessons from you, especially reflecting on research with rigorous logic and always trying to diagnose the problem first. You will always be a role model inspiring me of how to be a good supervisor in my future academic career.

I also want to thank Alessandro Abate and Huazhe Xu for their time and expertise to assess my thesis, and Jakob Foerster, Yarin Gal, and Alessandro again for providing valuable feedback on my transfer and confirmation.

During my DPhil, I was fortunate to collaborate with Luisa Zintgraf, Jacob Beck, Risto Vuorio, Even Liu, Chelsea Finn, Matthieu Zimmer, Kun Shao, Siddhant Sharma, Kang Li, Zilin Wang, Matthew Jackson and Jakob Foerster. Thank you for all your contributions! I especially want to thank Luisa for supervising on my first paper during my DPhil and introducing me to the research area of meta-RL, Jacob for making me believe in the power of Hypernetworks, Risto for your expertise in imitation learning and meta-RL, Matthew for teaching me a lot about diffusion and foundation models, Kang and Zilin for your great efforts in helping realize our first VLA paper. I also want to thank Lars Kunze and Mingfei Sun for their supervision when I explored different research directions during the early years of my PhD. My thanks also go to Agrim Gupta, Alex Goldie, Alex Rutherford, Alex Zakharov, Shiji Zhou, Tianpei Yang, Vitaly Kurin, Xuanlin Li, Yifu Tao and Zhengdao Chen for their valuable discussions on the works included in this thesis.

I was fortunate to meet and work with many great people in WhiRL: Luisa, Vitaly, Shangtong, Mingfei, Tarun, Jelena, Mattie, Matt S, Kristian, Risto, Jake, Ben, Matt J, Alex G, Alex Z, Clémence, Zilin and Bidipta. I have really enjoyed our talks and lunches in the town. I also want to thank the members of AIMS CDT (both teachers and students) and the FLAIR lab, with whom I have had many interesting discussions and activities. Thank you all for enriching my academic and social life in Oxford!

I was fortunate to join Oxford as a member of AIMS CDT, which offers me great freedom and support to explore different research directions. I especially appreciate

the generous funding from AIMS CDT and AWS that supported me during my whole DPhil. My gratitude also goes to Wendy Poole, our great CDT administrator who is always there to help with any question or need I have.

My academic career started from my Master study at Tsinghua University, and I want to thank my Master supervisor Wenwu Zhu, Wenpeng Zhang and Xin Wang for guiding me to AI research, Frank Hutter and Jian Tang for offering me intern opportunities and supervision at University of Freiburg and Mila. Time flies, now comes the end of my DPhil and the start of my working career, and I want to thank many people who have given valuable suggestions and help on my career plan at this critical point: Aijiao Zhou, Heng Chang, Kun Shao, Libo Wu, Min Xu, Tao Tong, Wei Zheng, Yi Wu, Yutian Chen, and Zhongming Wei.

Personal

To my great friends: Zhengdao Chen, Hanxiong Wang, Zuo Huang, Yimin Wei, Yige Li, Zihan Zang, Caiwu Liang, and Zongyu Yang, thank you not only for all the happy time we've had together, but also for our in-depth chats that help me realize how AI may play an important role in different domains and make this world a better place. And thanks to all my "random" teammates of the football games we've played together, which form one of my happiest hours in Oxford.

To all the members in my large extended family, thank you for all your care and support during my whole life, especially your greetings during my study abroad and the warm welcome whenever I returned back in China. To Master Dayun and Daxiong, you are not only my uncles but also my life mentors, and you will always inspire me to be a good person.

To Tris, I'm lucky to have met you, and have really enjoyed our trips around Europe which I may be too busy/lazy to go by myself, our daily chats from science to history between the "scientist" and "entrepreneur", and you "little bad". I look forward to the new experience we'll create together.

Finally, to my parents Zehu and Xiaogui, thank you for bringing me up to who I am today with your full devotion, and your unconditional love and support all the time to realize my dream. I'm always grateful and proud to be your son, and I look forward to making you more proud of raising a son who will try his best to make this world a better place with his passion, thinking and actions.

Abstract

While robots that follow hard-coded instructions have been widely applied in the real world, learning intelligent robots that can autonomously accomplish different tasks in unstructured environments with unforeseen variations remains a key challenge. Inspired by the recent success of foundation models in different domains, robotic learning has been going through a paradigm shift from learning specialist robots on narrow task distributions to learning generalist robots on large-scale multi-task data, which enables broader generalization across different dimensions like embodiments, skills and scenarios.

In this thesis, we formulate generalist robot learning as a Contextual Markov Decision Process, and investigate two problem settings of generalization across embodiments and skills under this unified framework (Chapter 2). We propose four novel methods to tackle the following three challenges under these two problem settings: model pretraining, inference efficiency and efficient adaptation.

In Part II of the thesis, we focus on the first problem setting of cross-embodiment control. In chapter 6, we propose ModuMorph, a Transformer-based universal controller that better models how the optimal policy conditions on the robot morphology via contextual modulation, to improve pretraining. In Chapter 7, we improve inference efficiency of generalist robots via *knowledge decoupling*, i.e., decoupling the knowledge required to solve different tasks and only activating a compact specialist policy to solve each specific task at test time. We realize knowledge decoupling via the hierarchical architecture of Hypernetworks (HNs), networks that generate the parameters of a base network, and investigate how to successfully train HNs via policy distillation. HyperDistill, our proposed method that combines these two key components, achieves similar performance as ModuMorph, while significantly accelerating inference by two orders of magnitude.

In Part III of the thesis, we focus on cross-skill control by following language instructions. In Chapter 8, we extend the knowledge decoupling principle to Vision-Language-Action (VLA) models, the mainstream approach for language-conditioned control that suffers from high inference cost. We propose HyperVLA to generate a compact policy via an HN that conditions on task context, and investigate several key algorithm design choices to improve the performance of HyperVLA. It achieves similar and even better performance compared to some SOTA VLAs, while

significantly improving inference efficiency by two orders of magnitude. Finally in Chapter 9, we investigate how to adapt a pretrained VLA to a new domain with many different tasks in a sample- and computation-efficient way. To achieve these goals, we propose HyperLoRA, which utilizes the strong expressive power of HNs and the parameter-efficient fine-tuning method LoRA to generate task-conditioned LoRA parameters, and significantly outperforms task-agnostic LoRA fine-tuning.

In summary, the contributions in this thesis significantly improve model pretraining, inference efficiency and adaptation performance for learning generalist robots. We hope the key ideas developed in this thesis, such as the knowledge decoupling principle and utilizing HNs as a key building block in learning generalist agents, will be utilized more frequently in the future to help build more versatile and efficient foundation models for robotics and other domains.

Contents

1 Introduction

Contents

1.1 From Specialist to Generalist Robots

Robots have significantly improved production efficiency and promoted economic growth in the past few decades. However, most robots used nowadays are not intelligent. They can only precisely and repeatedly execute a sequence of fixed actions $a_0, a_1, \cdots, a_T$ by following hard-coded programs. This open-loop control fashion significantly limits their application to highly controlled environments like factories, while failing to work in a changing world with unforeseen variations [Billard and Kragic, 2019, Kroemer et al., 2021]. Moreover, robotic programming requires extensive human expertise and trial and error, and the programming complexity of some dexterous tasks is even beyond the scope of human experts.

To tackle these challenges, robotic learning aims to automatically learn how to solve different tasks from data. Instead of outputting fix actions regardless of the environment state, robotic learning aims to learn a closed-loop control policy

$\pi(a|s)$ that can adaptively choose which action a to take conditioned on the current environment state s. The policy is usually trained with either imitation learning (IL) [Pomerleau, 1988] by imitating the behaviors of an expert demonstrator, or reinforcement learning (RL) [Sutton et al., 1998] by interacting with the environment to search for the optimal control policy via trial and error. This learning-based approach has achieved promising progress beyond the scope of robots that can only follow hard-coded instructions, such as in-hand dexterous manipulation [OpenAI et al., 2019], locomotion of quadrupedal robots in diverse terrains [Lee et al., 2020], etc. However, these robots are still *specialists* that are usually trained on a relatively narrow task distribution from scratch, which are limited in:

1. **Generalization.** A specialist robot can only solve a narrow set of tasks that it is trained on, while failing to generalize to related but unseen tasks, which significantly limits its application to controlled environments with only small variations. For example, a robot arm may fail to manipulate an object if the shape and size of the object changes, and a quadrupedal robot may fail in locomotion if one of its legs is broken.

2. **Sample efficiency.** It requires a significant amount of data collection, either by human experts for IL or by the robot itself for RL, to train a specialist robot for each task from scratch. Given the high cost of collecting robotic data, training a specialist robot for each task to solve is not scalable.

Inspired by the great success of foundation models in domains like natural language processing (NLP) [Devlin et al., 2019, Brown et al., 2020, Raffel et al., 2020, Touvron et al., 2023, Team et al., 2023, Liu et al., 2024] and computer vision (CV) [Dosovitskiy et al., 2021, Caron et al., 2021, Radford et al., 2021, Kirillov et al., 2023, Zhai et al., 2023, Oquab et al., 2024] in recent years, robotic learning has also been going through a paradigm shift from training specialist robots on a narrow task distribution to training *generalist* robots on large-scale multi-task robotic data, which provides a promising solution to the generalization and sample efficiency challenges as discussed above. First, by training on a broad and diverse

task distribution, the emergent generalization ability of foundation models [Wei et al., 2022] as observed in NLP and CV may also promote task generalization in robotics. Second, learning a generalist robot enables knowledge sharing across related tasks, thus significantly improves sample efficiency compared to training a specialist robot on each separate task from scratch.

Specifically, instead of learning a specialist policy $\pi(a|s)$ that can only generalize across different states within a specific task, we now learn a generalist policy $\pi(a|s, c)$ to solve different tasks by further conditioning the policy on task context c. After trained on a wide range of tasks, the generalist policy learns to generalize in not only the state space but also the task space, so that it can better solve a new task with unseen task context at test time either in a zero-shot fashion or via few-shot adaptation.

Dimensions of Generalization

A generalist robot is expected to generalize well along many different dimensions in the task context, such as skills (goals), scenarios and embodiments. For example, consider deploying a household robot in your home. For skill generalization, we hope the robot can accomplish many different tasks like cooking, laundry, room cleaning, etc., instead of having to buy many different robots to do each job separately. For scenario generalization, the robot is expected to work in different houses with diverse layouts and visual appearance with little tuning, so that it can be easily deployed at scale. For embodiment generalization, as there will be many different types of household robots working in the future, the controller is expected to transfer well across different robot embodiments, instead of having to collect a large amount of new data to train a separate controller for each robot design.

1.2 Challenges in Learning Generalist Robots

While learning a generalist policy can significantly improve task generalization and sample efficiency, it also introduces new challenges as the task space, data

size and model size all significantly scale up. We highlight three key challenges that motivate the works in this thesis as below:

1. **Pretraining.** How to pretrain a generalist policy to accommodate the diverse behaviors required to solve different tasks is nontrivial. For example, the optimal locomotion strategy may be very different when the embodiment of the robot slightly changes, like running with or without a tail. A slight difference in language instructions may lead a household robot to behave in totally different ways, like "clean bedroom" and "clean kitchen". How to parameterize $\pi(a|s, c)$ with the correct inductive bias to better model the complicated dependency between task context and the correspondingly optimal behaviors is thus a critical question.

2. **Inference efficiency.** Generalist robots usually have millions or billions of parameters to provide sufficient model capacity for learning from large-scale multi-task data, which introduces significantly higher inference cost compared to specialist robots that usually have moderate size. For example, many existing robotic foundation models are too large to support high-frequency control, and can only operate at a low frequency around 10 Hz even equipped with powerful GPUs [Brohan et al., 2023, Zitkovich et al., 2023, Kim et al., 2024]. How to improve inference efficiency of generalist robots is critical to enable high-frequency dexterous control, and reduce energy, memory and computational cost of robotic hardware.

3. **Adaptation to downstream tasks.** While generalist robots show promising generalization to unseen tasks, they still require further fine-tuning to be proficient in downstream tasks from a new domain. For example, after trained on data collected from different houses, a household robot may still need further fine-tuning on data collected from a new house before it can work there reliably. How to efficiently adapt a generalist robot to new domains is thus important for scalable deployment.

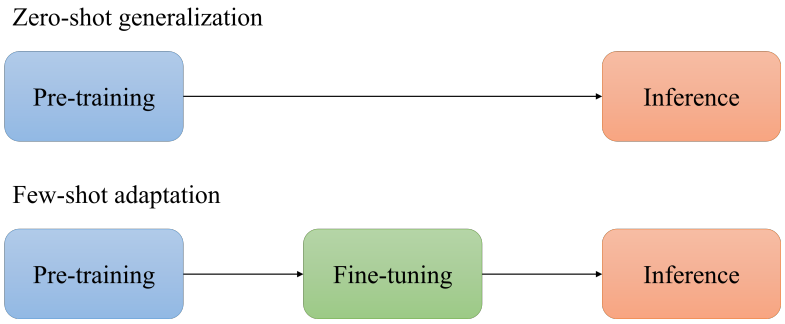

Figure 1.1: The relationship between the three stages of pretraining, fine-tuning and inference (deployment) for zero-shot generalization or few-shot adaptation.

Figure 1.1 shows the relationship between these three stages under two different settings. For zero-shot generalization, we directly deploy the pretrained generalist policy on a robot for inference. This is suitable for cases where the test tasks are drawn from the same distribution as the training tasks. For few-shot adaptation, we first pretrain a generalist policy, then fine-tune it with a small amount of data from the downstream task(s), and finally deploy the fine-tuned policy on a robot to accomplish the task(s). This further fine-tuning process is usually necessary when there is a distribution shift between the training and test tasks, such as asking the robot to master a new skill or work in a new scenario that is very different from those seen during training.

1.3 Contributions

This thesis aims to tackle the aforementioned challenges to learn generalist robots that can learn better and more efficiently across tasks. We investigate two specific problem settings, corresponding to two different dimensions of task generalization we care about as introduced before.

First, we investigate cross-embodiment generalization, where we aim to learn a universal policy to generalize across robots with different embodiments (morphologies). We will also call this setting as universal morphology control interchangeably in the thesis. As many robots with different embodiments are deployed in the real world, learning a universal morphology controller across them can not only improve data efficiency by pretraining on a much larger multi-robot dataset and sharing

knowledge across robots, but also enable better generalization or more efficient adaptation to new robots with different embodiments to come in the future.

Second, we investigate generalization across skills (goals) specified by language instructions, where we aim to learn a generalist robot that can perform different skills to achieve different goals by following human instructions. This is a very practical problem setting which reflects how generalist robots will interact with and assist human in the future. Models learned under this setting are usually called Vision-Language-Action (VLA) models [Zitkovich et al., 2023], as they take visual observations and language instructions as input, and output robotic actions for control.

Given the grid expanded by the two problem settings and the three technical challenges as introduced before, Table 1.1 summarizes the works included in the thesis, showing which challenge they aim to solve under which problem setting.

	Pretraining	Inference efficiency	Adaptation
Embodiment generalization	ModuMorph	HyperDistill	
Skill generalization		HyperVLA	HyperLoRA

Table 1.1: Summary of the contributions of the thesis. Each method aims to tackle a specific challenge under a specific problem setting. Most methods contain "Hyper" in their names, which is the abbreviation of "Hypernetworks" [Ha et al., 2017], a hierarchical neural architecture that we will frequently use with different motivations.

Thesis Structure

The thesis consists of four main parts:

Part I: Foundations This part provides relevant background on the works to present in the thesis.

- **Chapter 2** introduces how to formulate robotic control as Markov Decision Process (MDP) and multi-task robotic control as Contextual MDP (CMDP) respectively, and how to solve them with RL or IL. Then we formally formulate the two problem settings to investigate by defining the task context of CMDP in different ways.

- **Chapter 3** introduces neural architectures that will be frequently used in the following chapters, especially Hypernetworks (HNs) [Ha et al., 2017], a hierarchical neural architecture that is frequently used in our works for different purposes.
- **Chapter 4** reviews related work on the first problem setting of cross-embodiment control.
- **Chapter 5** reviews related work on VLAs for the second problem setting of language-conditioned cross-skill control.

Part II: Generalization across Embodiments This part focuses on the cross-embodiment setting of learning a universal policy that can generalize across different robot morphologies, while the task distribution is relatively narrow, like locomotion on different terrains. The two works included in this part tackle the pretraining and inference efficiency challenges respectively.

- **Chapter 6** introduces ModuMorph, which better models the complicated dependency between a robot's morphology and its optimal control policy via contextual modulation, and leads to better locomotion performance on both seen and unseen robots in different terrains. This work was published at ICML 2023 [Xiong et al., 2023].
- **Chapter 7** introduces HyperDistill, which significantly improves inference efficiency of universal morphology controllers by a novel technique called *knowledge decoupling*. It works by learning a generalist model with high capacity at training time, but only generating and activating a compact specialist model at test time for efficient inference on a specific embodiment. We parameterize the model as an HN, as its hierarchical architecture provides a natural way to decouple the knowledge required to control different embodiments. We further train this HN via policy distillation to tackle the instability issue when directly training the HN with RL. This work was published at ICML 2024 [Xiong et al., 2024b].

Part III: Generalization across Skills This parts focuses on generalization across skills (goals) by following different language instructions, while the robots to control have the same or very similar morphology, like accomplishing different manipulation tasks with robot arms. The two works included in this part aim to tackle the inference efficiency and adaptation challenges respectively.

- **Chapter 8** introduces HyperVLA, which applies the knowledge decoupling principle to VLAs to improve their inference efficiency. Unlike most existing monolithic VLAs that need to activate the whole model at inference time, HyperVLA generates a compact task-specific policy when encountering a new task, and then only activates this compact policy to solve the task, which significantly reduces inference cost. This work is currently under review [Xiong et al., 2025].

- **Chapter 9** introduces HyperLoRA, which investigates how to efficiently adapt an pretrained VLA model to multiple tasks from a new domain with HN-generated LoRA fine-tuning. This work was published at NeurIPS 2024 Adaptive Foundation Models Workshop [Xiong et al., 2024a].

Part IV: Discussion We conclude the thesis with a reflection on the works presented, and a discussion of open questions on learning generalist robots.

Part I

Foundations

2

Problem Formulation

Contents

This chapter formally formulates the two problem settings that will be investigated in the thesis. We start by formulating robotic control as a Markov Decision Process (MDP), and introducing how it can be solved with reinforcement learning (RL) or imitation learning (IL) (Section 2.1). Then we introduce how to formulate multi-task robotic control as a contextual MDP, an extended version of a standard MDP with task context (Section 2.2). By specifying the task context in different ways, we formulate cross-embodiment generalization as solving a CMDP with multi-task RL in Section 2.2.1, and cross-skill generalization as solving a CMDP with multi-task IL in Section 2.2.2.

2.1 Robotic Control as Markov Decision Process

Controlling a robot to solve a specific task can be formulated as a Markov Decision Process (MDP) [Bellman, 1957], expressed as a tuple $M = (\mathcal{S}, \mathcal{A}, R, T, \rho_0, \gamma, H)$, where

- $\mathcal{S}$ is the state space;
- $\mathcal{A}$ is the action space;
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, with $r_t = R(s_t, a_t, s_{t+1})$ for step t;
- $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition probability function, with $P(s_{t+1}|s_t, a_t)$ as the probability of transition to s_{t+1} if taking action a_t in state s_t;
- $\rho_0 : \mathcal{P}(\mathcal{S})$ is the initial state distribution;
- $\gamma \in [0, 1]$ is the discounting factor; and
- H is the maximal horizon length of the task. In principle H can be infinite, which further requires γ to be smaller than 1 to avoid infinite accumulated rewards. In practice we set H as a finite number, as robotic control needs to be finished within a limited amount of time.

The robot starts from an initial state $s_0 \sim \rho_0$. At each step t, the robot samples an action a_t from the action distribution generated by a control policy $\pi(a|s)$, executes a_t, and gets a reward r_t. The environment then transits to the next state s_{t+1} following the transition probability function. This process will iterate until a terminal state is reached or the timestep reaches the maximal horizon length H. A whole trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_H)$ is called an episode. The discounted rewards accumulated over an episode is called the return, which is defined as $R(\tau) = \sum_{t=0}^{H-1} \gamma^t r_t$.

Policy Learning The learning objective is to maximize the policy's expected return, i.e., $J(\pi) = \mathbb{E}_{\tau\sim\rho_0,P,\pi}[R(\tau)]$. As the state space is usually too large to enumerate the optimal action distribution for every possible state, the policy is parameterized as a neural network θ to generalize across the state space.

Partial Observability Sometimes the agent can only have access to an observation o_t instead of the full state s_t at each step, which is known as partial observable MDP (POMDP) [Spaan, 2012]. For example, in vision-based robotic control, the policy does not know the full state of the environment, and has to make decisions based on the image observations alone. To learn under partial observability, the policy usually conditions on a sequence of historical observations instead of just the current observation to better infer the hidden state, i.e., we learn $\pi(a_t|o_{0:t})$ instead of $\pi(a_t|o_t)$.

2.1.1 Reinforcement Learning for Robotic Control

Reinforcement learning (RL) [Sutton et al., 1998] learns to maximize expected returns by interacting with the environment via trial and error. The collected experience can be used to learn different things, which lead to different RL algorithms.

Value-Based Methods

Value-based methods learn the value functions w.r.t. a policy π, i.e., the expected return to get if you start from a specific state or state-action pair, then act by following π afterwards. The value function of a state s is defined as

$$V^\pi(s) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s]. \tag{2.1}$$

Similarly, the value function of a state-action pair (s, a) is defined as

$$Q^\pi(s,a) = \mathbb{E}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a]. \tag{2.2}$$

Based on the definition of returns, the value functions follow a recursive property called Bellman equations, i.e.,

$$V^{\pi}(s_t) = \mathbb{E}_{\pi,P}\left[r_t + \gamma V^{\pi}(s_{t+1})\right], \tag{2.3}$$

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi,P}\left[r_t + \gamma Q^{\pi}(s_{t+1}, a_{t+1})\right]. \tag{2.4}$$

A critic, parameterized as a neural network ϕ, is usually learned to estimate the value functions. For example, to learn the state value functions, the critic is usually trained by minimizing the mean square error (MSE) loss w.r.t. a target value, i.e.,

$$L^{VF}(V_{\phi}^{\pi}) = \mathbb{E}_{s_t \sim \pi}\left[\left(V_{\phi}(s_t) - V_t^{target}\right)^2\right]. \tag{2.5}$$

The target value can be specified in different ways. On one hand, we can estimate it with the reward-to-go from s_t, i.e., $V_t^{target} = \sum_{t'=t}^{H-1} \gamma^{(t'-t)} r_{t'}$, which is an unbiased estimation of the value function but has high variance due to the stochasticity in action selection and transition dynamics. On the other hand, we can estimate the target value with Bellman equation, i.e., $V_t^{target} = r_t + \gamma \bar{V}_{\phi}(s_{t+1})$, where $\bar{V}$ is a stop gradient operation that leads to semi-gradient update, and the delta value $\delta = r_t + \gamma V(s_{t+1}) - V(s_t)$ is called the temporal difference (TD) error. This has lower variance as only one step of empirical reward is used for estimation, but has higher bias in the estimation of $V(s_{t+1})$. To reach a good trade-off between variance and bias, n-step return $G_t^{(n)} = r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n \bar{V}_{\phi}(s_{t+n})$, and a weighted sum of n-step returns with n varying from 1 to the horizon length are usually used to further stabilize RL training [Sutton, 1988, Schulman et al., 2015b].

Given a learned value function, we can either directly choose the action with the maximal state-action value at each step without explicitly learning a policy if the action space is discrete like in Deep Q Network (DQN) [Mnih et al., 2015], or concurrently learn both a value function and a policy like in Deep Deterministic Policy Gradient (DDPG) [Lillicrap et al., 2019] and actor-critic methods [Mnih et al., 2016].

Policy Gradient Methods

Policy gradient methods directly optimize the policy π by maximizing the expected return $J(\pi)$ with gradient ascent. According to the policy gradient theorem [Sutton et al., 1999], the policy gradient can be computed as

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi}\left[\nabla_\theta \log \pi_\theta(\tau) R(\tau)\right] = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)\right]. \tag{2.6}$$

As the action taken at each step has no influence on the rewards collected from previous steps, we can further write it as

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{R}_t\right], \tag{2.7}$$

where $\hat{R}_t = \sum_{t'=t}^{H-1} \gamma^{(t'-t)} r_{t'}$ is the reward-to-go from state s_t.

As $\int_x p_\theta(x) \nabla_\theta \log p_\theta(x) = \int_x \nabla_\theta p_\theta(x) = \nabla_\theta \int_x p_\theta(x) = \nabla_\theta 1 = 0$ for any probability distribution p, we have $\mathbb{E}_{a_t\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta\left(a_t|s_t\right) b(s_t)\right] = 0$. Based on this nice property, we can subtract any function b that only depends on s_t from $\hat{R}_t$ to further reduce variance in gradient estimation, where $b(s_t)$ is called a baseline. The state value function $V(s_t)$ is usually used as the baseline function, and $A^\pi(s_t, a_t) = \mathbb{E}\left[\hat{R}_t - V(s_t)\right] = Q(s_t, a_t) - V(s_t)$ represents how much more return we can get by always choosing action a_t at s_t compared to sampling from $\pi(\cdot|s_t)$, which is known as the advantage function. In this way, the policy gradient can be written as

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A^\pi(s_t, a_t)\right]. \tag{2.8}$$

So intuitively, policy gradient works by increasing (decreasing) the log-probability of state-action pairs with positive (negative) advantages proportionally.

Trust-Region Methods and Proximal Policy Optimization

Policy gradient methods are sensitive to the choice of the learning rate. If the learning rate is too high, the action distribution may change dramatically which makes policy learning unstable. If the learning rate is too low, the policy will improve too slowly which harms sample efficiency.

Trust region methods [Schulman et al., 2015a] instead maximize a surrogate objective function $\mathbb{E}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}A^\pi(a_t|s_t)\right]$ subject to a constraint on the KL divergence between π_θ and $\pi_{\theta_{\text{old}}}$, i.e., the trust region. Instead of solving this complicated constrained optimization problem, Proximal Policy Optimization (PPO) [Schulman et al., 2017] optimizes a clipped surrogate objective which achieves both simplicity and strong performance empirically, i.e.,

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon))A_t\right], \tag{2.9}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the action probability ratio between the updated policy and the policy before update, ϵ is a hyperparamter that controls the clipping ratio. Intuitively, if the advantage of a state-action pair is positive, PPO will increase its probability unless the action probability ratio exceeds $1+\epsilon$. In contrast, if the advantage of a state-action pair is negative, PPO will decrease its probability unless the action probability ratio is below $1-\epsilon$. In general, this clipped objective maintains the simplicity of policy gradient update while constraining the updated policy to be not too different from the old policy.

In this thesis, we adopt PPO when learning with RL due to its simplicity, scalability and strong performance.

2.1.2 Imitation Learning for Robotic Control

Different from RL methods that learn the optimal control policy by interacting with the environment, imitation learning (IL) [Pomerleau, 1988, Ng and Russell, 2000] assumes access to a set of expert trajectories that show how to optimally solve the task, and train a policy to imitate the expert's behaviors.

In this thesis, we adopt Behavior Cloning (BC), a very simple yet effective IL method that directly predicts the expert actions by supervised learning. Given a set of expert demonstrations $\mathcal{D} = \{\tau^i | i = 1, \ldots, N\}$, where $\tau^i = \left(s_0^i, a_0^i, \ldots, s_{H-1}^i, a_{H-1}^i, s_H^i\right)$ is the i-th expert trajectory of states and the corresponding actions at each step, N is the total number of demonstrations. The

objective function of BC is to maximize the log-likelihood of the expert action under the policy's action distribution, i.e.,

$$L^{BC}(\theta) = \mathbb{E}_{(s_t^i, a_t^i)\sim\mathcal{D}}\left[-\log \pi_\theta(a_t^i|s_t^i)\right]. \tag{2.10}$$

Similarly, we can also train a deterministic policy by minimizing the MSE loss.

Compounding Error and Action Chunking

A key challenge in IL is compounding error / covariate shift, i.e., an error in action prediction of each step may lead to a next state outside the training state distribution, and the policy does not know how to act properly in this out-of-distribution state. Even small prediction errors will accumulate quadratically over time [Ross et al., 2011], which leads the policy to fail.

Action chunking [Chi et al., 2023, Zhao et al., 2023b] provides a simple yet effective solution to alleviate this issue. Instead of only predicting the action for the current step, action chunking predicts K steps of actions as $\pi_\theta(a_{t:t+K}|s_t)$, which introduces several key advantages:

1. It reduces the effective task horizon, i.e., the number of times that the policy is called, by K times. As the compounding error grows quadratically with the effective horizon length, a shorter horizon will lead to less compounding error.

2. It encourages the policy to predict action sequences with temporal consistency and smoothness, instead of minimizing myopic single-step prediction error.

However, the control process will change from closed-loop to open-loop when executing each action chunk. So action chunking is actually reducing compounding error at the cost of higher feedback error in open-loop control. We thus hypothesize that its empirical effectiveness may be highly attributed to the fact that prediction error on a long action sequence can be significantly reduced with expressive model architectures like Transformer [Zhao et al., 2023b] and learning objectives like diffusion [Chi et al., 2023], which is easier than improving generalization to out-of-distribution states.

Action chunking can be used together with action ensemble to further improve prediction accuracy and action smoothness, i.e., the policy makes different predictions of a_t from step $t-K$ to t, and the final action to execute at step t will be a weighted sum over all these predictions [Zhao et al., 2023b].

Multi-Modal Action Distribution

Another key challenge in BC is the multi-modality in action distribution, i.e., the demonstrations may illustrate multiple different optimal actions to take at some states, such as avoiding an obstacle in front by turning either left or right. However, the policy will learn to go ahead instead, as this is the action that can minimize the BC loss but is not reasonable for task solving.

Common approaches to handle multi-modal action distributions include:

1. Discretize the actions into bins, and learn a categorical action distribution where multi-modal distribution can be easily modeled [Brohan et al., 2023].

2. Use variational autoencoder (VAE) [Kingma and Welling, 2022] to model the latent variables that determine the variations in actions [Zhao et al., 2023b].

3. Use diffusion-based methods that can naturally model multi-modal distributions [Chi et al., 2023, Ze et al., 2024].

2.2 Multi-Task Robotic Control as Contextual MDP

In this section, we extend the MDP formulation of robotic control as introduced in the last section to a Contextual MDP (CMDP) [Hallak et al., 2015] formulation for multi-task robotic control. For clarity, we will use superscript k to represent the index of a task, and subscript t to represent the timestep within an episode.

A CMDP is defined as a tuple $(\mathcal{C}, \mathcal{S}, \mathcal{A}, R, T, \rho_0, \gamma, H)$, where

- $\mathcal{C}$ is the task context space, with c^k representing the context of task k;

- $\mathcal{S}$ is the state space, with s_t^k representing the state at step t in task k;

- $\mathcal{A}$ is the action space, with a_t^k representing the action at step t in task k;
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{C} \to \mathbb{R}$ is the reward function, with $r_t^k = R(s_t^k, a_t^k, s_{t+1}^k, c^k)$;
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{C} \to \mathcal{P}(\mathcal{S})$ is the transition probability function, with $P(s_{t+1}^k | s_t^k, a_t^k, c^k)$ as the probability of transition to s_{t+1}^k if taking action a_t^k in state s_t^k and task context c^k;
- $\rho_0 : \mathcal{C} \to \mathcal{P}(\mathcal{S})$ is the initial state distribution of different tasks;
- γ is the discounting factor; and
- H is the maximal horizon length of all tasks.

Compared to MDP, the key difference is the additional term of task context c, and the reward function, transition probability function and initial state distribution will all further condition on it.

Each unique task context c specifies a different task which can be formulated as an MDP. In other words, a CMDP can be seen as a set of MDPs, where the difference between the MDPs is determined by their task context. With this CMDP formulation, we can easily investigate different generalization dimensions by defining the context space $\mathcal{C}$ in different ways. As shown in Figure 2.1, for generalization across skills (goals), embodiments, and scenarios as discussed in Section 1.1, the goal context mainly influences the reward function of the corresponding MDP, while the embodiment and scenario context mainly influence the transition dynamics of the MDP. From an RL perspective, both embodiment and scenario context belong to the environment that the agent interacts with, and we split them apart mainly for practical purpose, as the embodiment context specifies which robot to control, while the scenario context specifies all the external factors that influence environment dynamics beyond the robot itself.

By defining $\mathcal{C}$ as the space of different morphologies, we get the universal morphology control setting that will be investigated in Part II. By defining $\mathcal{C}$ as the space of different language instructions, we get the generalization across skills setting that will be investigated in Part III. We do not explicitly define the scenario

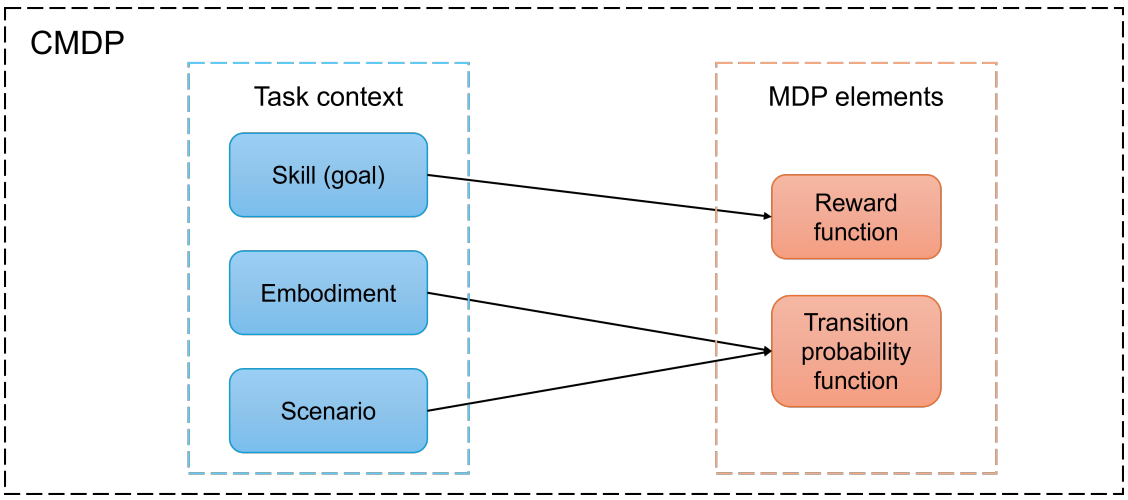

Figure 2.1: How different task context of a CMDP influences the reward function or dynamics of the corresponding MDP.

context, as it includes all remaining environmental factors that can have an influence on the environment dynamics, which are hard to fully represent in numerical format. Instead, we expect the generalist policy to learn how to infer and adapt to different scenario context from its observations and interactions with the environment, and evaluate its scenario generalization ability in both problem settings. For universal morphology control, different scenarios correspond to changing terrains that the robot needs to locomote on. For skill generalization, the policy is evaluated in scenarios that are not identical to those seen during training, such as different environment background, object layouts and appearance, etc.

Intuitively, we can also see a CMDP as an MDP with an extended state space $\mathcal{S} \times \mathcal{C}$, where the state features corresponding to $\mathcal{C}$ cannot change within an episode. Consequently, to learn a generalist policy to solve a CMDP, it is natural to further condition the policy on the task context, i.e., learning $\pi_\theta(a|s, c)$ instead of $\pi_\theta(a|s)$. A key focus of this thesis will be how to condition π_θ on c to better represent the complicated dependency between the task context and the corresponding optimal behaviors.

2.2.1 Multi-Task Reinforcement Learning for Cross-Embodiment Control

In this subsection, we formally formulate cross-embodiment control as solving a CMDP with multi-task RL.

We consider the problem setting of learning a universal policy to control a set of robots with different morphologies for locomotion, as locomotion is a basic skill required by legged robots that may appear in different shapes and hardware configurations. Learning a universal morphology controller can significantly improve learning efficiency on both existing robots and new ones to be designed in the future. As shown in Figure 2.2, different morphologies can be seen as topology trees with different structures. By specifying the task context as the robot morphology, we can formulate cross-embodiment control as a CMDP as follows:

- The task context c^k represents morphology information of robot k, such as the size and mass of each limb and its initial position relative to its parent node, and an adjacency matrix that defines the topology of the morphology tree;
- The state s_t^k represents sensor inputs from robot k at step t, such as the position and velocity of each limb (node) of the robot;
- The action a_t^k is defined as the actuation force that is applied to each joint (the red circles as shown in Figure 2.2) that connects a limb to its parent limb in robot k at step t;
- The reward r_t^k is defined as the moving distance of robot k in the desired locomotion direction from step t to $t+1$;
- The transition probability function is deterministic, i.e., conditioned on the robot's current state and action, its next state is deterministic based on rigid-body dynamics;
- γ is set to 1 to maximize the overall locomotion distance within an episode; and

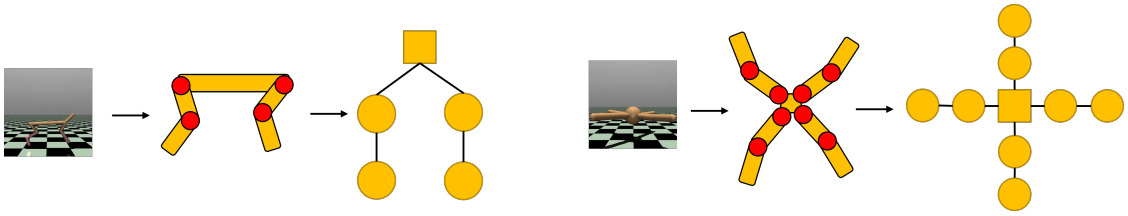

Figure 2.2: Two examples (2-leg cheetah and 4-leg ant) of how to convert a robot's morphology into a topology tree. For each example, the left image shows the robot in simulation; the middle plot shows its morphology, with orange rectangles representing limbs and red circles representing joints; the right plot shows the topology tree, with tree nodes representing limbs and edges representing joints, and the robot's torso is defined as the tree root represented by a square.

- ρ_0 and H follow their default definitions.

The Modular Design Space Assumption

As different morphologies have different number of limbs and joints, the dimensions of state and action space will be different across robots (tasks). However, we assume that all the robots are drawn from a modular design space [Gupta et al., 2021], i.e., each robot can be seen as a morphology tree over a set of nodes (limbs), and the node-level state and action space are consistent across different nodes in different robots. Based on this assumption, we have $s_t^k = \{s_t^{k,i} | i = 1, \ldots, N_k\}$ and $a_t^k = \{a_t^{k,i} | i = 1, \ldots, N_k\}$, where N_k is the number of nodes in robot k, and $\dim(s_t^{k,i})$ and $\dim(a_t^{k,i})$ are fixed for any k, i and t. The consistency of state and action dimensions at the node level makes it possible to parameterize the universal policy as a Graph Neural Network (GNN) [Wu et al., 2020] or Transformer to model node interactions, which we will discuss in Section 4.1.2.

Learning Objectives

Given a set of K different robots, we aim to learn a universal policy $\pi_\theta(a_t^k | s_t^k, c^k)$ to maximize the average return over all the training morphologies, i.e.,

$$J(\theta) = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{H-1} r_t^k. \tag{2.11}$$

In addition to good training performance, we also expect the learned policy to generalize well to unseen test morphologies in a zero-shot manner.

We adopt RL instead of IL for cross-embodiment control for the following reasons:

1. It is hard for human to provide expert demonstrations on how different morphologies locomote.

2. We can easily generate different morphologies, simulate their locomotion behaviors in different terrains in high fidelity via rigid-body simulators like Mujoco [Todorov et al., 2012].

2.2.2 Multi-Task Imitation Learning for Cross-Skill Control

In this subsection, we formally formulate cross-skill control as solving a CMDP with multi-task IL.

Consider the problem of learning a generalist policy that controls a robot arm to accomplish different manipulation tasks by following language instructions from human. By specifying the task context as the language instruction, we can formulate cross-skill control as a CMDP as follows:

- The task context c^k represents the language instruction of task k;

- The full state of the environment is not observable to the robot. Instead, the robot receives image observations o_t collected by cameras from different views. o_t can also include further inputs like joint states, tactile sensors, depth images, etc. In this thesis, we focus on image observations alone for simplicity, while other input modalities can be easily added without loss of generality;

- The action space of a robot arm can be defined in different ways, and we adopt end-effector position deltas as the action space in the thesis. Specifically, the action a_t^k is a 7D vector including delta in position (3D) and orientation (3D), and gripper open/close (1D);

- The transition probability function is complicated, including both deterministic parts like the movement of the robot arm, and nondeterministic parts like human interventions by moving the object in the scene;

- The reward function and discounting factor are not needed given that we are solving an IL problem; and
- ρ_0 and H follow their default definitions.

As scenario variations mainly influence the visual observations, we expect the generalist policy to also learn a robust visual representation for scenario generalization during pretraining.

Learning Objectives

Given a set of multi-task demonstrations $\mathcal{D} = \{(l^i, \{o_t^i, a_t^i\})|i = 1, \ldots, N\}$, where l^i is the language instruction of the i-th demonstration, and $\{o_t^i, a_t^i\}$ is the expert trajectory of observation-action pairs of the i-th demonstration, we aim to learn a generalist policy $\pi_\theta(a_t|o_{0:t}, l)$ to minimize the BC loss (Equation 2.10) on $\mathcal{D}$. The learned policy is expected to master the skills that have been seen during training, and generalize to unseen but relevant skills either in a zero-shot fashion or via fast adaptation.

We adopt IL instead of RL for cross-skill control for the following reasons:

1. Unlike locomotion tasks that can be simulated in high fidelity with rigid-body physics, simulating manipulation tasks requires modeling complicated interactions between robots and a broad range of different objects, which is much more challenging especially for deformable objects like clothes, while RL training on real robots is costly and even dangerous.

2. In recent years, collecting real-robot demonstrations for manipulation tasks has become much easier and more scalable thanks to the development of many low-cost teleoperation platforms [Mandlekar et al., 2018, Zhao et al., 2023b, Fu et al., 2024, Wu et al., 2024b], and many large-scale manipulation datasets [O'Neill et al., 2024, Khazatsky et al., 2024] have been proposed which make it possible to pretrain generalist robots for manipulation via IL.

3

Background on Neural Architectures

Contents

This chapter introduces necessary background on the neural architectures that will be frequently used in the works to present in the thesis. We first introduce Transformer (TF) [Vaswani et al., 2017] in Section 3.1, which is widely used as the building blocks of different models we propose. Then we introduce Hypernetworks (HNs) [Ha et al., 2017] in Section 3.2, a hierarchical neural architecture that we will adopt to achieve different goals in the thesis.

3.1 Transformer

Transformer has achieved great success in recent years, especially in foundation models, as it provides a very flexible and scalable way to learn from large-scale data in different domains.

To learn a TF on the data from a specific domain, the first key question is how to tokenize the data, as TF requires a sequence of token embeddings with the

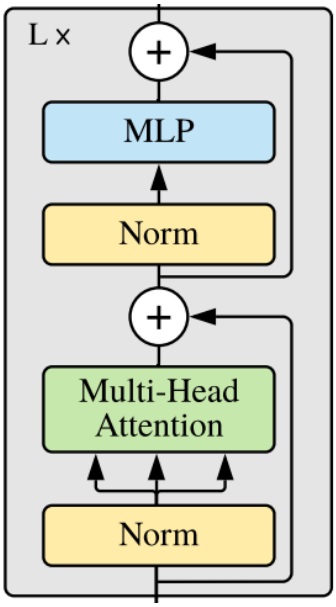

Figure 3.1: The overview of a TF layer. Image credit to Dosovitskiy et al. [2021].

same dimension as its inputs. Tokenization provides a flexible way to integrate multi-modality data as TF inputs. For NLP which TF is originally proposed for, tokenization is straightforward by treating each (sub-)word as a token. For images, a common practice is to divide an image into image patches with the same size, and treat each patch as a token [Dosovitskiy et al., 2021]. For numerical features, we can either treat each scalar feature dimension as a token or treat the whole feature vector as a token. The tokens then go through a linear projection layer to get fixed-dimension token embeddings as the inputs to TF.

Figure 3.1 shows the architecture of a TF layer. The key building block is an attention layer, which requires three input vectors from token i, i.e., a query $\boldsymbol{q}_i$, a key $\boldsymbol{k}_i$ both of dimension d_k, and a value vector $\boldsymbol{v}_i$ of dimension d_v. The three vectors of all the input tokens are stacked into matrices Q, K, V, and the attention module updates the token embeddings as

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V,$$

i.e., the dot product of $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ determines how much attention token i pays to token j to update its token embedding. Usually multiple attention heads are trained independently to learn different interactions between tokens. The attention block is

followed by a 2-layer multi-layer perceptron (MLP) with residual connections to further update the embedding of each token. Several TF layers can be stacked to further improve model capacity. Note that the TF layer is agnostic to the order of the input tokens, which is not desirable for domains like NLP where token order may encode useful information. Thus positional encoding (PE) is usually added to the token embeddings or attention matrices to make TF be aware of token order.

Output heads are usually added to the final TF layer for prediction. By defining the loss function over different tokens, TF can flexibly support different learning objectives. For example, GPT-1 defines the next-token prediction loss over the last token in the current text [Radford et al., 2018], while Vision Transformer (ViT) defines the image classification loss over a held-out class token [Dosovitskiy et al., 2021].

3.2 Hypernetworks

A hypernetwork (HN) [Ha et al., 2017] is a network that generates the parameters of a base network θ conditioned on some context c, i.e., $\theta = \text{HN}_\phi(c)$, where ϕ are the HN parameters to learn. This hierarchical architecture offers a powerful tool for multi-task robotic control, as we can generate different policies for different tasks conditioned on their task context, instead of using a single policy to solve different tasks. Please see Figure 3.2 for a comparison between the high-level architectures of a monolithic policy and an HN-based policy, and refer to Chauhan et al. [2024b] for a more detailed review of HNs.

An HN typically consists of a context encoder f and several output heads (parameterized as linear layers) that take the context embedding $e = f(c)$ as input, and output flattened parameters of different blocks in the base network. For example, to generate the parameters of a linear layer $y = Wx + b$, the HN needs two output heads to generate W and b respectively, i.e., $W = h_W(e)$, $b = h_b(e)$. If W is a $M \times N$ matrix, and the context embedding dimension is E, then h_W is a linear layer with input dimension E and output dimension $M \times N$, while h_b is a linear layer with input dimension E and output dimension N. As the whole parameter

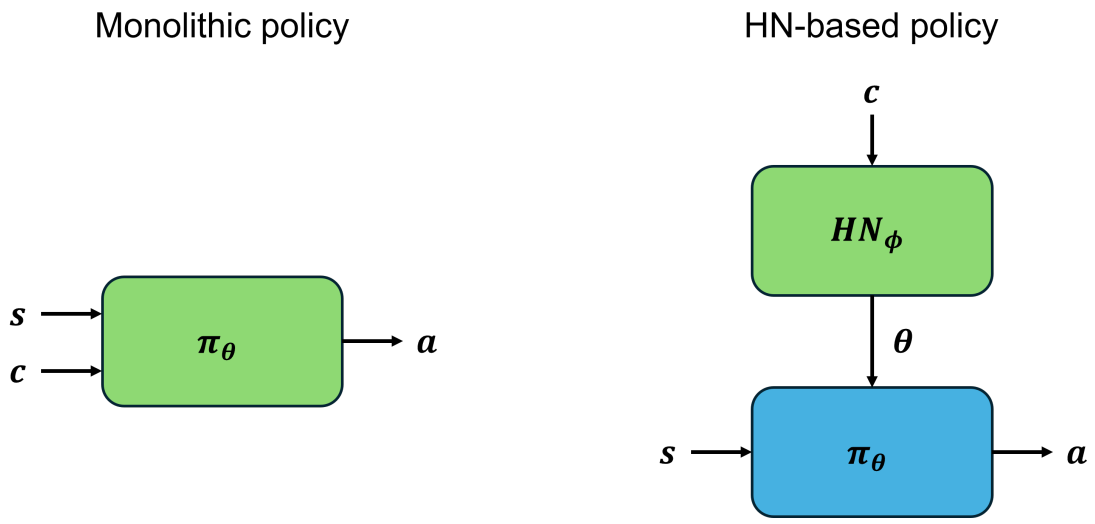

Figure 3.2: Comparison between the high-level architecture of a monolithic policy (left) and an HN-based policy (right). The learnable parameters that are shared across tasks are in green, while the HN-generated task-specific parameters are in blue.

generation process is differentiable, we can directly backpropagate from the loss function defined over the base network outputs to the HN to update ϕ.

Note that TF and HN are not mutually exclusive. HN focuses on the high-level architecture of a model, like monolithic or hierarchical, while TF is a specific neural module to build up a model. In the thesis, we will frequently use TF as the building block of both the HN and the generated base network.

Scaling up to Larger Base Networks If the base network to generated by an HN has N parameters, and the context embedding dimension of the HN is E, then the number of learnable parameters in the HN output heads will be $E \times N$. This makes it hard to scale up the size of the base network, as the model size of the HN will be further increased by E times. To tackle this challenge, a common solution is to generate the base parameters in a block-wise or chunk-wise fashion [Brock et al., 2018, Mahabadi et al., 2021, Chauhan et al., 2024a] by reusing the output heads, instead of generating all the base parameters at once. For example, if the base parameters are divided into M chunks, then the size of the HN output head will be divided by M. The same HN output head is called for M times

by conditioning on both the context embedding and the chunk ID to generate different base parameters for each chunk.

3.2.1 Advantages of Hypernetworks

In this subsection, we discuss about some key advantages of using HNs for multi-task learning, which motivate us to adopt it as a key building block in the works to present in the thesis.

In general, multi-task learning aims to learn a function $y = f(x, c)$ to generalize across different task context c. A common approach is to encode the task context c with an embedding function e, then concatenate the context embedding with the input features for prediction, i.e., $y = q\left(x, e(c)\right)$, which we name as embedding-based methods. Alternatively, we have HN-based methods that learn $y = g(x; \theta_c)$, where $\theta_c = h(c)$ is the parameters of function g generated by the HN h.

Galanti and Wolf [2020] theoretically investigate the function approximation capacity of these two approaches, and prove that under certain conditions and when letting the functions e and h be as large as we wish, g can be smaller than q by orders of magnitude to reach the same level of function approximation error. They further show that under some further common assumptions on the function to be approximated, the overall number of trainable parameters in an HN is much smaller than that of embedding-based methods.

Jayakumar et al. [2020] propose a framework of multiplicative interactions (MI) to unify different neural architectures that model the interactions between $\boldsymbol{x}$ and $\boldsymbol{c}$. Specifically, a MI layer is defined as $f(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{c}^T \mathbb{W} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{U} + \boldsymbol{V}\boldsymbol{x} + \boldsymbol{b}$, where $\mathbb{W}$ is a 3D weight tensor, $\boldsymbol{U}$, $\boldsymbol{V}$ are 2D weight matrices and $\boldsymbol{b}$ is a vector. Many commonly used architectures for modeling task conditioning, such as HNs, attention layers and gating functions, can be seen as different ways to parameterize these learnable parameters in a MI layer. By setting $\boldsymbol{W'} = \boldsymbol{c}^T \mathbb{W} + \boldsymbol{V}$ and $\boldsymbol{b'} = \boldsymbol{c}^T \boldsymbol{U} + \boldsymbol{b}$ as HN-generated parameters, we have $f(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{W'}\boldsymbol{x} + \boldsymbol{b'}$, which follows the same format as a linear layer generated by HN as discussed before. They show that multiplicative interactions enlarge the hypothesis space of vanilla MLPs, and

offer a more expressive inductive bias and neural building block to solve some problem types like multi-task learning.

While these two works mainly focus on theoretical analysis and only experiment on toy examples and relatively simple tasks, many other works have empirically validated the effectiveness of HNs on more practical and larger-scale problems in different domains, such as multi-task learning [Zhang et al., 2019, Yu et al., 2019, Mahabadi et al., 2021, Navon et al., 2021, Sarafian et al., 2021], meta learning [Rusu et al., 2019, Peng et al., 2021, Beck et al., 2023a,b, Rezaei-Shoshtari et al., 2023, Sendera et al., 2023], continual learning [von Oswald et al., 2020, Huang et al., 2021], multi-agent learning [Shamsian et al., 2021, Tessera et al., 2025], etc.

In summary, HNs provide a more expressive and flexible way to model task conditioning. Compared to embedding-based methods that force hard parameter sharing across tasks and only condition on the task context via the model input, HNs enable soft parameter sharing, i.e., each task can have a different set of parameters, while knowledge transfer is maintained via the HN shared across all the tasks.

In addition to this well-established advantage of HNs for modeling task conditioning, we further propose another key advantage of HNs called *knowledge decoupling*, i.e., the ability to decouple the optimal policies to solve different tasks in the parameter space, which can be utilized to significantly improve inference efficiency of a generalist model and will be discussed in more details in Chapter 7 and 8.

3.2.2 Challenges of Hypernetworks

Despite its advantages as discussed in the previous subsection, training an HN also introduces new challenges that need to be solved to fully realize its learning potential.

Network Optimization

Successful training of neural networks heavily depends on the proper setup of many optimization choices, such as network initialization [Glorot and Bengio, 2010, He et al., 2015], normalization layers [Ioffe and Szegedy, 2015, Ba et al., 2016] and gradient transformations [Kingma and Ba, 2014, Loshchilov and Hutter, 2019].

However, these methods are mainly tailored for training monolithic models, and may not fit the different optimization dynamics of HNs with hierarchical architectures. For example, initializing an HN with commonly used Xavier or Kaiming initializers will lead to extremely large parameter range in the base network and harm the learning performance. Chang et al. [2020] and Beck et al. [2023a] investigate how to initialize HNs properly so that the base network is initialized in the same way as a monolithic network. However, many other optimization choices in HN training remain underexplored, which may be crucial for stabilizing HN training and improving its learning performance.

Task Generalization and Overfitting

While HNs provide a more expressive neural architecture to model task conditioning, they also increase the chance of overfitting, as a small change in the task context could lead to a very different base policy if the learning process of the HN is not properly regularized. It is thus important to improve task generalization of HNs by applying techniques such as data augmentation, dropout and weight decay to the task space [Chauhan et al., 2024a,b].

We believe that more research into these key challenges will further improve the performance of HNs, and we will also propose some solutions to these challenges in the works to present in the thesis, which we find to be critical for improving the performance of HNs under our problem settings.

4

Literature Review on Cross-Embodiment Control

Contents

This chapter reviews related work for the first problem setting considered in the thesis, i.e., learning a universal controller that can generalize across different embodiments. We fist review main approaches for cross-embodiment control in Section 4.1, then review literature from other related fields in Section 4.2, including cross-domain transfer learning and morphology design. In addition to the literature reviewed in this chapter, we will discuss about some further related work in Part II that is relevant to the research topic in Chapter 6 and 7 respectively.

4.1 Main Approaches for Cross-Embodiment Control

In this section, we review literature that learns a universal policy to generalize across robot embodiments, including: (1) Generalization across robots with the same topology but different kinematics or dynamics parameters, which is an easier setting with consistent state and action space across robots, but provides an important cornerstone towards our goal of universal morphology control. (2) Morphology-aware methods that explicitly condition policy learning on the robot morphology, which focus on exactly the same problem setting as ours. (3) Morphology-agnostic methods that learn a cross-embodiment policy without explicitly conditioning on the robot morphology.

4.1.1 Robotic Control with Parametric Variations

We start from the simplest case of learning a universal policy that can generalize across robots with the same topology but parametric variations. From a CMDP perspective, this setting defines the task context as different kinematics and dynamics parameters of the robot, such as the size, mass, density of its limbs, armature and damping coefficients, etc. Generalization to parametric variations is desirable for two main reasons:

1. Due to the high sample complexity of RL algorithms, training a RL controller on real robots is too expensive or even dangerous. Instead, existing works mainly follow a sim-to-real transfer [Zhao et al., 2020b] paradigm, i.e., first train a controller in simulation, then deploy it on real-world robots. However, as we can't perfectly simulate the real-world robot and environment, we must learn a policy that can automatically handle this sim-to-real gap [Peng et al., 2018] by generalizing across parametric variations.

2. The hardware parameters may change dynamically during deployment, e.g., the motor voltage will gradually decrease as the power runs out. Thus the controller must adapt to such changes during deployment.

One solution is to learn a policy that is invariant to such variations, i.e., generalize across different context parameters without explicitly conditioning on the context. Domain randomization (DR) [Tobin et al., 2017, Peng et al., 2018, Muratore et al., 2021] is a commonly used approach which trains a policy over a distribution of tasks with random task configurations to achieve invariance to the variations.

However, DR may not be suitable in many cases where invariance to parametric variations is hard to learn, e.g., learning an policy that can lift both light and heavy objects with the same force. Instead, it may make more sense to adaptively adjust the control policy based on the parametric variations.

When the task context is known in prior, such as changing the size and mass of the robot, we can learn an adaptive policy by conditioning it not only on the state, but also on the context parameters [Chen et al., 2018].

However, when the parametric variations are unknown or hard to measure directly, such as the friction between the robot's feet and the floor, we need to first infer the task context based on historical trajectories with a context encoder, wich is usually parameterized as a recurrent neural network (RNN) or Transformer, then condition the policy on the inferred task context [Yu et al., 2017, Clavera et al., 2019, OpenAI et al., 2019, Schoettler et al., 2020, Zhao et al., 2021b, Kumar et al., 2021, Feng et al., 2023, Beck et al., 2023a,b, Ji et al., 2023, Yu et al., 2023]. This is very similar to how we train a policy to solve a POMDP as discussed in Section 2.1, as intuitively we can see the unknown task context as the unobserved part of an augmented state. More formally, meta-RL [Beck et al., 2025] and Bayesian RL [Duff, 2002, Zintgraf et al., 2020] methods are designed for learning a generalist policy across a distribution of MDPs with unknown task context. They trade-off exploration, which infers the unknown task context, and exploitation, which adapts the policy behaviors based on the agent's current belief about the task context to maximize returns. Intuitively, the model learned by these methods can be decomposed into two components, a context encoder that infers the task context from the agent's interaction history with the environment, and a multi-task policy that adapts its behaviors conditioned on the current inferred task context.

Moreover, as training the context encoder together with the context-conditioned policy end-to-end may be hard, many works propose auxiliary losses to help train the context encoder [Zintgraf et al., 2020, Kumar et al., 2021, Liu et al., 2021b], such as minimizing a reconstruction error on predicting future states, etc.

4.1.2 Morphology-Conditioned Control

Compared to generalization across parametric variations with the same robot morphology, generalization across different morphologies introduces several new challenges:

1. The state and action space dimensions are inconsistent across morphologies (tasks).

2. While each state feature dimension has the same physical meaning across different robots with the same topology, this does not hold across robots with different morphologies. For example, the first feature in the state vector may represent the x position of the torso for one robot, but represent the x position of a front leg for another robot. In other words, there is not a consistent order of limbs across different morphologies, and the policy needs to be order-invariant to tackle this issue.

3. The diversity between the optimal control policies for different morphologies may be significantly larger than that of robots only with parametric variations.

In this subsection, we review literature that aims to tackle these challenges by learning a universal policy which explicitly conditions on the morphology context.

MLP-Based Methods

We start from the simplest idea of learning an MLP-based controller, which is usually sufficient to generalize across robots with only parametric variations, but fails to tackle the above challenges for universal morphology control.

To learn an MLP policy for the state and action space defined over a morphology tree, we first need some way to order the limbs, so that we can concatenate their

node-wise states and actions into 1D vectors as the MLP's input and output. As the morphology of a robot has a tree structure, the limbs in each robot are usually ordered by some tree traversal methods like depth-first search [Gupta et al., 2022].

To handle the variable number of limbs across different robots, it is common to assume a maximal limb number $N_{\max}$, and zero-pad the state and action vectors to this maximal length so that the state and action dimensions are aligned across different robots. To condition the policy on the morphology context, the node-wise context features $c^{k,i}$ are concatenated with the state features $s_t^{k,i}$ as node-wise input [Gupta et al., 2022], where k and i are robot index and limb index respectively. So the final input to the multi-robot MLP policy is $x_t^k = \left[x_t^{k,1}, x_t^{k,2}, \ldots, x_t^{k,N_k}, \vec{0}, \ldots, \vec{0}\right]$, where $x_t^{k,i} = [s_t^{k,i}, c^{k,i}]$, and N_k is the number of limbs in robot k.

A key limitation of MLP-based methods for universal morphology control is that it is not invariant to the order of limbs across robots, i.e., the same set of MLP parameters is used to process the i-th limb's features of all the robots, though these limbs may have very different physical meanings across robots. This may explain why MLP-based methods perform poorly for universal morphology control as reported in previous work [Wang et al., 2018, Gupta et al., 2022].

GNN-Based Methods

NerveNet [Wang et al., 2018] utilizes the fact that the morphology of a robot is a graph (tree), and uses GNNs [Kipf and Welling, 2017, Gilmer et al., 2017, Hamilton et al., 2017, Veličković et al., 2018, Seo et al., 2018] as the controller to handle morphology graphs with arbitrary sizes. It achieves similar single-robot performance as MLPs, and significantly outperforms MLPs for multi-robot training and transfers better to unseen morphologies.

Huang et al. [2020] further utilize the fact that the morphology graph actually follows a tree structure. Thus instead of doing message passing across neighborhood as in standard GNNs, they propose a tree-structured network called Shared Modular Policy (SMP), which first aggregates morphology information into a root node with a bottom-up module, then propagates control signals from the root to subtrees with a

top-down module. SMP achieves much better training and zero-shot generalization performance compared to MLP-based methods.

Transformer-Based Methods

While NerveNet and SMP build their computational graph identical to the morphology graph, Kurin et al. [2021] show that the benefits introduced by GNN's structural inductive bias may be outweighed by the difficulty of message passing between distant nodes on the graph [Li et al., 2018], e.g., they show that building a computational graph with all nodes connected to a virtual root node can achieve similar or even slightly better performance compared to building the computational graph based on physical connections. Consequently, they propose Amorpheus, a Transformer-based policy to enable immediate interactions between any node pairs, which achieves better performance than SMP and NerveNet.

However, while Transformers achieve better performance than GNNs for universal morphology control, they ignore the structural information which may be helpful for learning. Consequently, Hong et al. [2022] propose Structure-aWAre Transformer (SWAT), which uses Amorpheus as the backbone, and further introduces traversal-based positional encoding (PE) and graph-based relational embedding (RE) to encode morphology information. SWAT outperforms Amorpheus in both training and zero-shot generalization performance on the multi-robot benchmark as proposed in Huang et al. [2020], which validates the importance of distinguishing between limbs that play different roles in different robots. Interestingly, they show that when trained on multiple morphology types, Amorpheus controls a humanoid robot by jumping forward, as this is how other robot types (such as hopper) move forward and such knowledge hinders humanoid to learn its optimal control policy. On the other hand, SWAT correctly controls humanoid by walking forward, thanks to the structural encoding which helps maintain structure-specific behaviors when necessary.

Similar to SWAT, MetaMorph [Gupta et al., 2022] is also a Transformer-based method. It builds upon the insight that robot morphology is just another modality

on which we can condition the output of a Transformer. In addition to PE, they add node-wise context features as additional node inputs to better learn a morphology-conditioned controller. Furthermore, instead of uniformly sampling training morphologies, they propose a dynamic replay buffer balancing scheme, which prioritizes the morphologies that have relatively lower training performance, so that the training process won't be dominated by the robots that are easier to control. They experiment on a benchmark called UNIMAL [Gupta et al., 2021] with 100 training and test robots respectively, and achieve better training and generalization performance than baselines like Amorpheus [Kurin et al., 2021].

Dong et al. [2022] investigate universal morphology control on robots with high Degree of Freedom (DoF). They propose a hierarchical architecture called Synergy-Oriented LeARning (SOLAR), which groups actuators into synergies by an unsupervised learning method, and learns a synergy action to control multiple actuators in synchrony to reduce action dimensions. SOLAR builds upon Amorpheus and achieves better training and generalization performance, especially on the morphologies with higher DoF.

More recent works further pretrain on a larger distribution of morphologies to enable broader generalization [Bohlinger et al., 2025, Liu et al., 2025b], improve representation learning of morphology context to better encode the different roles of the limbs in different robots [Hao et al., 2024, Li et al., 2024a, Luo et al., 2025], and introduce recurrence into the policy to better encode unobserved context information [Engwegen et al., 2025, Liu et al., 2025b].

Generalization across both Morphologies and Goals

All the methods reviewed so far can only generalize across different morphologies to achieve a single goal, like maximizing locomotion distance. To enable broader generalization across both morphologies and goals, Devin et al. [2017] decompose the control policy into a robot-specific module and a goal-specific module. Such decomposition enables combinatorial generalization to novel robot-goal pairs. However, it can not zero-shot generalize to robots or goals that do not exist in

the training set. Furuta et al. [2023] propose morphology-goal graph by treating observations, actions and goals in a unified graph representation. Specifically, they treat the goal conditions as extra nodes in the morphology graph, which is able to cover a wide range of common goals such as locomotion, reaching, manipulation and twisting. They propose a new benchmark called MxT-Bench, which includes more diverse morphologies and goals compared to previous ones [Huang et al., 2020, Gupta et al., 2022], and show better training and generalization performance compared to baseline methods.

4.1.3 Morphology-Agnostic Control

Unlike morphology-conditioned methods, morphology-agnostic methods learn a universal policy without explicitly conditioning on the morphology context.

One way to achieve this goal is to find invariant state and action spaces across different morphologies. For example, although different robot arms may have different DoF and size, the action space for the end-effector is usually identical, i.e., 3D position + 3D rotation + 1D gripper open/close, and the actions of the arm joints can be analytically derived by inverse kinematics. Utilizing such prior knowledge in robotics, many VLAs can generalize across different robot arms in a morphology-agnostic way [O'Neill et al., 2024, Ghosh et al., 2024, Kim et al., 2024] (see Chapter 5 for a more detailed review of VLAs). However, it is only applicable to the problem settings where an aligned action space exists, but cannot work in the more complicated setting of locomotion with different morphologies.

Another line of works does not require manual alignment of action space, but utilizes the strong sequence-to-sequence generation ability of Transformer to map from observations to actions by tokenizing everything [Reed et al., 2022, Trabucco et al., 2022, Wang et al., 2024a, Yang et al., 2024, Doshi et al., 2024]. They can flexibly generalize across manipulation, navigation and locomotion tasks by using different token blocks for different embodiments and goals. The irrelevant token blocks are masked out when solving a specific task, while the TF backbone is shared across all the tasks to facilitate knowledge transfer. These approaches utilize

tokenization as a flexible and unified interface to train on data collected from a broad range of morphologies, but at the cost of losing potentially useful structural information in the morphology context, and cannot zero-shot generalize to a new embodiment which the policy does not know how to tokenize.

4.2 Other Related Fields

This section reviews literature in two fields related to universal morphology control. First, we review cross-domain transfer learning, which investigates how to efficiently transfer skills from a source robot to a target robot, which is different from universal morphology control that aims to learn a universal policy that can generalize across many different morphologies. Second, we review literature on morphology design and control, where we need to jointly learn both the optimal morphology design and its corresponding control policy to solve a specific task.

4.2.1 Cross-Domain Transfer Learning

In addition to learning a universal controller with generalization ability, another way to accelerate learning on a new robot is to transfer the knowledge from an expert source robot to a target robot. This transfer learning [Taylor and Stone, 2009, Zhu et al., 2023] paradigm is called cross-domain transfer (imitation) learning in literature, where different domains correspond to different morphologies under our problem setting. The learning objective is to utilize expert behaviors from the source robot to accelerate the target robot's learning on a specific task.

In addition to the source robot's demonstration data on the target task we want to solve, these methods usually also require trajectories on some proxy tasks from both the source and target robots to learn the correspondence between the two robots. We categorize the literature in this field according to what kind of data on the proxy tasks is required to enable knowledge transfer.

Data on Proxy Tasks with Time Alignment This line of work uses the skills learned by both agents on some proxy tasks to learn an invariant feature space, which is then used to transfer other skills from the source robot to the target robot [Gupta et al., 2017, Hu and Montana, 2019]. Specifically, a state encoder is learned for each robot, which maps the state vector to an embedding in the shared invariant space. The encoders are trained to minimize the distance between the embeddings of time-aligned state pairs of the two robots in the shared feature space. Consequently, time-aligned data on some proxy tasks is required for encoder training. When transferring the skill to learn, the embedding distance between the two robots in the invariant feature space is used as a shaping reward to accelerate the learning process of the target robot on the target task.

Data on Proxy Tasks without Time Alignment As it is expensive or even impossible to acquire time-aligned data on proxy tasks, this line of work aims to relax the time alignment assumption by aligning the trajectory distribution instead of state pairs on the proxy tasks [Kim et al., 2020, Zhang et al., 2021, Watahiki et al., 2022]. To compensate for the loss of temporal correspondence information in the data, these methods propose additional objectives to regularize the distribution matching process, such as using cycle consistency [Zhu et al., 2017].

While the aforementioned methods optimize for distribution matching at the *state* level, Hejna et al. [2020] and Shankar et al. [2022] learn cross-morphology correspondence at a *skill* level, motivated by the assumption that robots with different morphologies use similar high-level skill sequences to solve similar tasks, while the low-level primitive actions that constitute a skill may not be well aligned between morphologies.

No Proxy Tasks While all previously mentioned methods require data on some proxy tasks to learn the state and action correspondence between morphologies, Fickinger et al. [2022] directly align and compare states between morphologies based on optimal transport [Villani et al., 2009] to enable cross-morphology transfer with a single demonstration from the source robot on the target task.

Other Approaches In addition to transferring from a fixed source robot, Liu et al. [2022] gradually evolve the morphology of the source robot to match the morphology of the target robot, which forms a curriculum with only slight morphology change at each step to accelerate the transfer learning process.

4.2.2 Morphology Design and Control

All the literature reviewed so far assumes a set of pre-given morphologies on which a universal controller is trained. However, in many real-world problems, we do not know what morphologies are suitable to solve a task in prior, and need to jointly design a good morphology and learn its control policy. This joint design and control problem can be seen as a bi-level optimization [Colson et al., 2007] problem, consisting of an outer-loop searching for the optimal design, and an inner-loop learning the optimal controller for a specific design, i.e.,

$$\begin{aligned} M^* &= \arg\max_M J(M, \pi_M), \\ \text{subject to} \quad \pi_M &= \arg\max_\pi J_M(\pi), \end{aligned}$$

where M and π_M are the morphology design and its corresponding controller respectively. A key challenge here is learning efficiency, as the inner-loop optimization itself is a time-consuming RL problem, and the outer-loop involves searching in a combinatorial design space. A lot of efforts have been put into accelerating this joint process, and we categorize the literature in this field based on whether they mainly accelerate the inner-loop or the outer-loop.

Inner-Loop Acceleration Instead of training a controller for each morphology from scratch, knowledge sharing across morphologies is the main idea to accelerate inner-loop optimization, which shares a similar motivation as universal morphology control. Schaff et al. [2019], Luck et al. [2020], Belmonte-Baeza et al. [2022] train a universal controller on a distribution of designs to quickly evaluate the performance of a new design. However, these methods can only tune the hardware parameters of a robot, while the robot morphology needs to be fixed. To enable designs of different

morphologies, Wang et al. [2019] use GNNs as the controller, which accelerates inner-loop optimization on new morphologies by reusing the GNN parameters across different morphology graphs. Pathak et al. [2019] propose to assemble a set of identical modules into a more complex morphology, so morphology design turns into how to determine the assembling action of each module in each time step, which can be learned jointly with the motor actions by RL.

Outer-Loop Acceleration Evolution algorithms [Sims, 1994, Wang et al., 2019] are commonly adopted for the outer-loop optimization, which are known to suffer from low sample efficiency. Liao et al. [2019] utilize Bayesian Optimization [Shahriari et al., 2015] to improve searching efficiency in the design space, but can only search for hardware parameters of a fixed morphology. Yuan et al. [2022] and Wang et al. [2023] formulate the joint design and control problem as a unified RL problem, which is significantly more efficient than non-differentiable evolution algorithms. Zhao et al. [2020a] propose a graph grammar to express possible robot morphologies, which turns robot design into the combination of a sequence of grammar rules. They introduce a heuristic-based method to efficiently search in the combinatorial design spaces built upon their graph grammars.

Task-Agnostic Morphology Design All the methods reviewed so far aim to find the optimal morphology design for a specific task, which requires a huge amount of reward labeling during the learning process. III et al. [2021] use evolution algorithms to search for a set of morphologies suitable for an environment in a task-agnostic way. A population of robots are controlled by randomly sampled action primitives, and an information-theoretic objective [Barber and Agakov, 2003] is used to rank the robots by their ability to reach diverse states in the environment and the causality of their actions. They show that the evolved morphologies can achieve similar performance on downstream tasks compared to those found with task-specific rewards [Wang et al., 2019].

5

Literature Review on Cross-Skill Control

Contents

This chapter reviews related work for the second problem setting considered in the thesis, i.e., learning a generalist policy to perform different skills or accomplish different goals. The idea of learning a goal-conditioned generalist policy to solve different tasks has been extensively explored long before the era of foundation models. Similar to how a generalist policy across dynamics and kinetics variations is learned as introduced in Chapter 4.1.1, a generalist policy across skills/goals can be learned by explicitly conditioning on known task context information [Schaul et al., 2015, Andrychowicz et al., 2017, Yu et al., 2020b, Yang et al., 2020, Kalashnikov et al., 2022], such as the goal position of an object to move, and inferring unknown task context from interaction history with the environment [Duan et al., 2016, Zintgraf et al., 2020, Beck et al., 2023a, Chen et al., 2022, Qi et al., 2023, Liang

et al., 2024a, Memmel et al., 2024], such as the shape and mass of an object which are unknown to the policy. But most of these works can only generalize within a limited task distribution, e.g., picking up objects with different shapes, solve simple short-horizon tasks in structured environments, such as table-top manipulation without distractor objects, or require privilege information in simulation during policy training, which limit their scalability to real-world scenarios that are more unstructured and require broader generalization. Only with the recent emergence of foundation models, it has become possible to learn a generalist robot that can interact more naturally with people by following language instructions and work in unstructured real-world environments by utilizing the strong generalization ability of LLMs and VLMs. In this thesis, we specifically focus on VLA models, as they take language instructions (L) and visual observations (V) as inputs to predict robot actions (A), which perfectly match the need of language-conditioned generalization across skills in our problem setting.

We review VLAs from the following perspectives: In Section 5.1, we first take the Robotics Transformer (RT) model series [Brohan et al., 2023, Zitkovich et al., 2023, O'Neill et al., 2024, Belkhale et al., 2024, Gu et al., 2024] as an illustrative example to show how some key ideas in VLAs are gradually developed. Then in Section 5.2, we discuss about key design choices in VLA pretraining, which lays the foundation for Part III of the thesis. In Section 5.3 and 5.4, we review literature for VLA inference acceleration and fine-tuning, which are closely related to the works to present in Chapter 8 and 9 respectively.

Note that the main goal of this section is to highlight the key ideas in the development of VLA models, instead of providing an exhaustive survey, given that this is a rapidly evolving research area with a large volume of recent publications. Please see Ma et al. [2025], Din et al. [2025], Kawaharazuka et al. [2025], Zhong et al. [2025a] for more detailed reviews on VLAs.

5.1 RT Model Series as an Illustrative Example

In this section, we take the RT model series as an example to illustrate the development of some key ideas in VLAs.

Inspired by the great success of foundation models in domains like NLP and CV, RT-1 [Brohan et al., 2023] is a pioneering VLA that validates the effectiveness of pretraining high-capacity models on large-scale robotic data to achieve strong generalization across a diverse range of robotic tasks. RT-1 uses EfficientNet [Tan and Le, 2019] as the visual encoder, adds FiLM layers [Perez et al., 2018] to it for language conditioning, and applies a Transformer on the features extracted by the EfficientNet to predict discretized action tokens. It is trained on about 130k demonstrations of more than 700 different skills.

While RT-1 is trained from scratch on robotic data alone, RT-2 [Brohan et al., 2023] investigates if we can build VLAs upon existing language and vision backbones to utilize their strong generalization ability learned from web-scale data, which is significantly larger than the scale of the robotic data used to train RT-1. RT-2 uses existing Vision Language Models (VLMs) [Chen et al., 2024, Driess et al., 2023] as the backbone, and co-fine-tunes them with both the robotic data from RT-1 and the VLMs' original training data. It performs similarly as RT-1 on seen tasks, while generalizing much better to unseen tasks, which validates the effectiveness of transferring the knowledge from existing VLMs for broader task generalization. Please see Figure 5.1 for a comparison between the high-level architectures of RT-1 and RT-2.

Due to the high cost of collecting robotic data, instead of learning with the data collected by a single type of robot, O'Neill et al. [2024] propose the Open X-Embodiment (OXE) dataset which integrate a set of robotic datasets collected by different robots from different affiliations. They train RT-1 and RT-2 on this significantly expanded dataset and observe promising performance improvement and skill transfer across embodiments, which validates the effectiveness of learning from cross-embodiment robotic datasets.

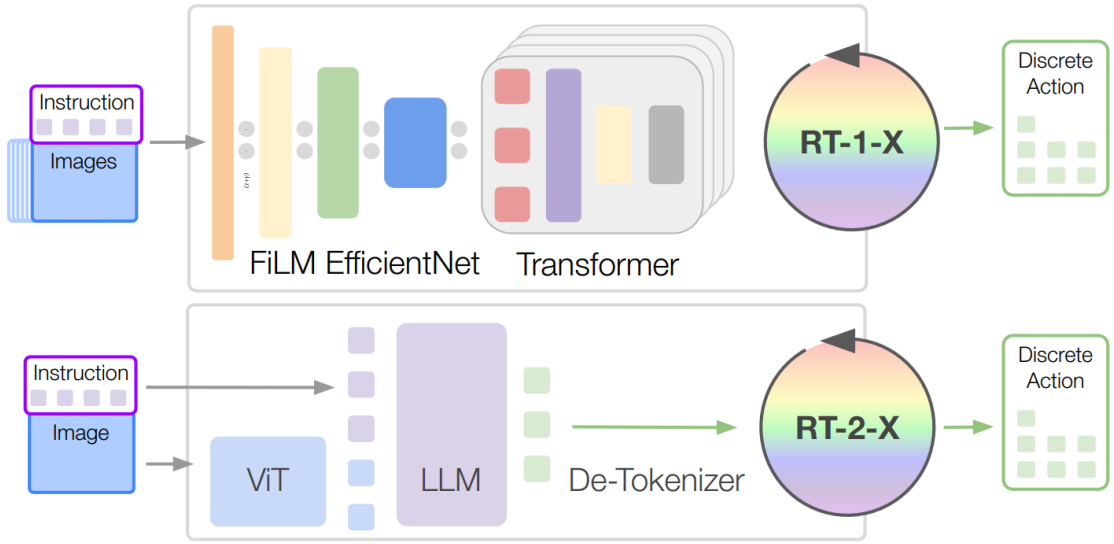

Figure 5.1: Comparison between the high-level architectures of RT-1 and RT-2. Image credit to O'Neill et al. [2024].

All the RT models introduced so far are trained to directly predict primitive actions from high-level language instructions and image observations in an end-to-end fashion, which lack intermediate representations that may facilitate generalization, especially given that the size of robotic datasets collected nowadays is still much smaller than the size of datasets used for LLM and VLM training. To tackle this challenge, RT-H [Belkhale et al., 2024] uses low-level language instructions that are more transferable across tasks as an intermediate layer between high-level command and primitive actions, and RT-Trajectory [Gu et al., 2024] uses robot trajectories as the intermediate representation on which the policy conditions, both achieving better generalization performance especially on unseen tasks.

Based on this overview of the RT model series, we learn three key lessons for training a better VLA:

1. Scale up the training data.

2. Scale up the model capacity and utilize existing language and vision foundation models as the backbone.

3. Design effective intermediate representations to facilitate generalization, at least before we can collect a huge enough robotic dataset such that the benefits

of learning from data outweigh the benefits of introducing human-designed inductive bias [Sutton, 2019].

These key points serve as a high-level guidance that motivates many of the VLAs that we will broadly review in the next section.

5.2 Pretraining

In this section, we review some key design choices in VLA pretraining.

5.2.1 Action Tokenization and Generation

Although the research community has reached a consensus on standard practices for tokenizing language and visual inputs, robotic action is a new modality raised with the recent emergence of VLAs, and different approaches have been explored for action tokenization.

As actions are multi-dimension continuous values in their original format, the most straightforward approach is to just predict their numerical values with an MLP action head attached to the output layer of the policy network. This approach has achieved some promising results [Figure, 2024, Kim et al., 2025], but may not well handle multi-modal action distributions as discussed in Section 2.1.2.

To handle multi-modal action distributions, many VLAs have adopted diffusion [Sohl-Dickstein et al., 2015, Ho et al., 2020] or flow matching [Lipman et al., 2023] for action generation [Black et al., 2025b, Ghosh et al., 2024, Liu et al., 2025c]. This is usually achieved by separately learning an action expert that conditions its action denoising process on the output from a TF policy backbone which processes language and visual inputs.

To align with the next token prediction paradigm of most existing foundation models, another popular action tokenization approach is to discretize each dimension of a continuous action vector into action bins and treat each separate bin as a token [Brohan et al., 2023]. Then action prediction can be seen as generating a sequence of action tokens in an autoregressive way trained with cross-entropy loss.

Further improvements have been proposed to compensate for the precision loss in discretization [Shafiullah et al., 2022], and compress the generated action token sequence via discrete cosine transform [Pertsch et al., 2025].

5.2.2 Intermediate Representations and Model Hierarchy

As discussed in the previous section, fully end-to-end learning that directly maps from language and visual inputs to action outputs is still very hard due to the limited amount of robotic data we have nowadays, and it could be beneficial to learn some intermediate representations that facilitate generalization. On one hand, language instructions may be too semantically different to learn shareable knowledge across tasks. For example, "pick coke can" and "close drawer" look very different, but they both contain a more fine-grained skill of moving close to some goal objects that transfers better across tasks [Belkhale et al., 2024]. On the other hand, raw actions can be very different across robot embodiments which may hinder knowledge transfer when training on a cross-embodiment dataset [O'Neill et al., 2024]. For example, a gripper and a dexterous hand have very different action spaces, but they may look more similar if we compare their behaviors from a more abstract level like the robot trajectory [Gu et al., 2024]. Thus learning a proper middle-level representation can serve as a bridge between high-level language task specification and low-level raw actions, i.e., instead of directly learning a monolithic VLA as $\pi(a|o_{0:t}, l)$, we can learn both a high-level policy $\pi_{\text{high}}(c_t^{\text{mid}}|o_{0:t}, l)$ that generates some middle-level context representation c_t^{mid} based on the language instruction and the observations, and a low-level policy $\pi_{\text{low}}(a_t|o_{0:t}, c_t^{\text{mid}})$ that predicts the raw actions based on the current middle-level context and observations. Please see Figure 5.2 for an overview of the high-level architecture of a hierarchical VLA. This hierarchical VLA architecture also provides a natural way for inference acceleration, as we will discuss later.

We categorize existing works in this direction according to what intermediate representations they learn [Zhong et al., 2025a]. A straightforward idea is to decompose the high-level instruction into a sequence of more fine-grained instructions by utilizing the strong reasoning and planning ability of existing LLMs and VLMs

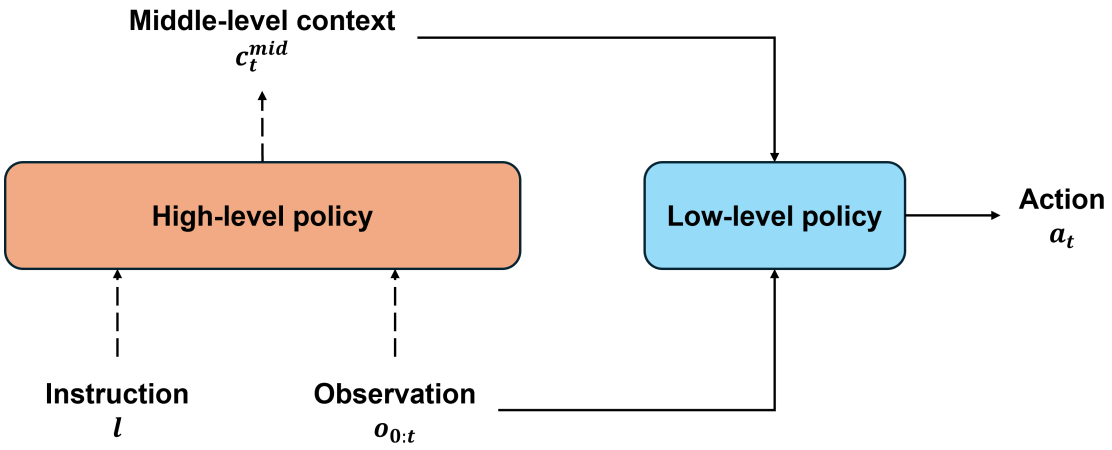

Figure 5.2: The overall architecture of a hierarchical VLA. We use dashed and solid lines to show the data flow for the high-level and low-level policy respectively to highlight that they can operate at different frequency. The box for low-level policy is smaller as it usually has a smaller size than the high-level policy.

[Ichter et al., 2023, Driess et al., 2023, Huang et al., 2023b, Mu et al., 2023, Black et al., 2025a, Shi et al., 2025b], which is especially important for solving long-horizon tasks that require task decomposition. While natural language may be ambiguous for (sub-)goal specification, formal language like linear temporal logic (LTL) provides unambiguous target representation for instruction grounding [Liu et al., 2023b, Pan et al., 2023]. Extending LTL from relatively simple benchmarks [Vaezipoor et al., 2021, Qiu et al., 2023, Jackermeier and Abate, 2025] to VLA application in real-world scenarios is a promising direction for future work, while some key challenges remain to be solved, such as how to decompose diverse high-level instructions, e.g., "clean bedroom", into a set of atomic propositions that (1) are generalizable across tasks; and (2) can be accurately and efficiently labeled based on high-dimensional image observations. Sub-goals can be generated not only in language format, but also in the pixel space. The visual sub-goal can be specified by either an image [Black et al., 2024, Ni et al., 2024, Zhao et al., 2025, Zhang et al., 2025c], or a video [Bharadhwaj et al., 2024, Hu et al., 2025b, Gao et al., 2025, Zhang et al., 2025a] upon which an inverse dynamics model is trained for action prediction. Another main body of works generate intermediate representations that can provide more guidance on where or how to interact with the environment, such

as keypoint [Huang et al., 2025b, Pan et al., 2025], bounding box [Stone et al., 2023, Deng et al., 2025, Zhong et al., 2025b], robot trajectory [Huang et al., 2023a, Gu et al., 2024, Li et al., 2025b, Xu et al., 2025b], etc. Learning latent actions as the intermediate representation has also been investigated to better mitigate the embodiment gaps between different robots and utilize the huge amount of actionless videos from the Internet [NVIDIA et al., 2025, Bu et al., 2025a,c, Ye et al., 2025].

However, learning intermediate representations also introduces new challenges compared to end-to-end learning. First, the pretraining process is usually divided into several stages, and the prediction error in each stage will accumulate and influence the performance of the subsequent stages. Second, the learning objectives between different stages may not be well aligned. Finally, some intermediate representations, such as goal images and videos, are time-consuming to generate, which introduce additional inference latency to the model. So as the size of robotic datasets increases, directly learning from data may gradually outperform introducing human-designed intermediate representations according to the "bitter lesson" [Sutton, 2019], while high-level task decomposition that has little interference with low-level control may still be necessary and helpful.

5.2.3 Scaling up Training Data

Motivated by the great success of pretraining on web-scale data in NLP and CV and the recently promising results of scaling up robotic datasets [Brohan et al., 2023, Walke et al., 2023, O'Neill et al., 2024], a greater effort has been put into collecting robotic data in even larger scale [Khazatsky et al., 2024, Black et al., 2025b, Bu et al., 2025a, NVIDIA et al., 2025]. Furthermore, it has been found that improving task diversity and data quality [Lin et al., 2025, Shi et al., 2025c, Xing et al., 2025] is more important than simply increasing the absolute number of demonstrations, which provides valuable guidance on how to scale up data collection in the future.

However, given the high cost of collecting real-world robotic demonstrations, another research focus is how to utilize other data sources for generalist policy learning, such as simulated and synthetic data, human videos and web data. As

Figure 5.3: Data pyramid of different data sources for training generalist robots. The data quantity decreases from bottom to top, while the data quality, i.e., alignment with the target robot in embodiment, task and environment, increases. Image credit to NVIDIA et al. [2025].

shown in Figure 5.3, a data pyramid is usually used to show the trade-off between data quantity and quality of these different data sources. At the top level is real-world robotic data that best aligns with the robot embodiment and environment that will be seen during deployment. At the middle level, synthetic data can be generated in a cheaper way at a larger scale by either high-fidelity simulation [James et al., 2020, Gu et al., 2023, Nasiriany et al., 2024, Wang et al., 2024c] or trajectory generation by augmenting existing real-world demonstrations [Mandlekar et al., 2023, Jiang et al., 2024, NVIDIA et al., 2025]. However, synthetic data generated in this way has sim-to-real gap, may suffer from hallucination in neural generation, and is hard to generate for complicated tasks like folding clothes. At the bottom level, human videos and web data [Goyal et al., 2017, Damen et al., 2022, Grauman et al., 2022, 2024] provide the richest source of data that can help learn visual representation and world dynamics, but they are usually actionless and have significant domain gap to robotic control in embodiment, camera view, etc.

To learn from videos without action labels, common approaches include pretraining visual representations from videos for downstream control tasks [Nair et al., 2023, Wu et al., 2024a, Cheang et al., 2024], semantic matching [Sikchi et al., 2024], and learning latent actions as pseudo action labels for videos as discussed before [NVIDIA et al., 2025, Bu et al., 2025a,c, Ye et al., 2025].

In addition to scaling up data size, another important dimension to expand is to collect and learn from robotic data with more input modalities [Zhen et al., 2024, Bi et al., 2025b, Hao et al., 2025, Huang et al., 2025a, Yu et al., 2025], such as 3D point cloud, tactile, force feedback, etc., which provide valuable information beyond the scope of visual inputs alone, especially for contact-rich tasks.

5.3 Inference Efficiency

While robotic foundation models significantly improve task generalization and performance compared to conventional specialist models, they also introduce significantly higher inference cost due to their huge model size, which is a key bottleneck for deployment on real robots with constrained computational resources and battery life, and hinders dexterous manipulation that requires high-frequency operations.

To accelerate VLA inference efficiency, we can either reduce the amount of time/computation required for each model forward pass, or reduce the frequency of model calling.

To reduce the computation involved in each model calling, a straightforward way is to reduce the model size of VLAs by using smaller backbone models [Belkhale and Sadigh, 2024, Wen et al., 2025, Shukor et al., 2025]. Another idea is to early exit from an intermediate layer of a huge VLA if the layer output is sufficient for action prediction [Yue et al., 2024]. For VLAs that predict discrete action tokens in an autoregressive way, inference can be accelerated by action sequence compression [Pertsch et al., 2025] and converting from autoregressive action generation to parallel generation [Kim et al., 2025, Song et al., 2025].

To reduce the frequency of model calling, a common approach is to adopt action chunking [Zhao et al., 2023b, Ghosh et al., 2024, Black et al., 2025b] as introduced

in Section 2.1.2 to predict K steps of actions at each inference step for execution, so that the VLA can be called only every K environment steps. Instead of training a monolithic model that infers at a fixed frequency, hierarchical (dual-system) VLAs [Bu et al., 2025b, Figure, 2024, Han et al., 2024, Shentu et al., 2024, Zhang et al., 2025b, NVIDIA et al., 2025, Cui et al., 2025, Zhou et al., 2025] learn both a high-level planner that generates a latent goal, and a low-level policy that conditions on this latent goal to generate per-step actions. The high-level planner has a larger model size to solve complicated reasoning and planning tasks and works at a low frequency, while the low-level controller has a smaller model size and works at a high frequency. This asynchronous mechanism reduces the overall inference cost, and resembles System I and II thinking in human brains [Kahneman, 2011].

In Chapter 8, we will propose an HN-based inference acceleration method that is orthogonal to the acceleration methods as reviewed above, which can be easily combined with them for further acceleration of VLA inference.

5.4 Downstream Fine-Tuning

While VLAs show promising zero-shot generalization performance, they still require further fine-tuning to achieve satisfying performance on downstream tasks for deployment in real world. To reduce the cost and difficulty of deployment, the fine-tuning process should be both sample- and computation-efficient.

Similar to LLMs and VLMs, VLAs also show strong few-shot adaptation ability to new tasks by utilizing their strong prior knowledge learned from large-scale pretraining. It has been widely shown that VLAs can fast adapt to a new task by just fine-tuning on a moderate amount of demonstrations collected from it, significantly outperforming learning from scratch on the new task with the same amount of data, which demonstrates the importance of large-scale pretraining [Ghosh et al., 2024, Black et al., 2025b, Kim et al., 2024, Liu et al., 2025c]. Instead of fine-tuning the whole VLA which is computationally expensive, parameter-efficient fine-tuning methods like LoRA [Hu et al., 2022] have been shown to achieve similar fine-tuning performance while being significantly more efficient [Kim et al., 2024].

Kim et al. [2025] systematically investigate some key design choices in VLA fine-tuning, including action decoding schemes, action representations and learning objectives, and show that fine-tuning with a proper combination of these choices can significantly outperform fine-tuning with the same recipe as pretraining.

While supervised fine-tuning (SFT) as introduced above has achieved promising progress, its performance is limited by the quality of human demonstrations. On the contrary, RL fine-tuning (RFT) provides the opportunity to let a VLA continuously improve its performance by learning from its own experience [Hu et al., 2025a, Chen et al., 2025, Guo et al., 2025, Liu et al., 2025a, Lu et al., 2025]. However, RFT also introduces new challenges. First, unlike in simulation, environment reset and reward specification are hard when running RL on real robots, which explains why most works in this direction only fine-tune on simulation tasks. Second, collecting data on real robots in large scale is expensive and even dangerous, which requires RFT to be sample-efficient and safety-constrained. Finally, reward design for complicated tasks is a tedious work, which may be even harder and more time-consuming than collecting demonstrations for SFT, thus requires further automation. Tackling these challenges is important for realizing the full potential of RFT in robotics.

Part II

Generalization across Embodiments

6

ModuMorph: Universal Morphology Control via Contextual Modulation

Contents

In this chapter and the next, we will investigate the first problem setting of cross-embodiment control, i.e., learning a universal locomotion policy that can both perform well on a set of training robots, and zero-shot generalize to unseen morphologies via multi-task RL.

This chapter proposes *ModuMorph*, a hierarchical Transformer-based model that

improves cross-embodiment pretraining via contextual modulation. We start by analyzing the limitations of existing methods in modeling morphology conditioning, and propose to tackle this challenge via contextual modulation in Section 6.1. In Section 6.2, we give a background introduction on MetaMorph [Gupta et al., 2022], the baseline method that we will improve upon in this chapter. Our methodology is introduced in Section 6.3. We present the experimental setup and results in Section 6.4 and 6.5 respectively, and end this chapter with a review of related work and discussion on future work.

6.1 Introduction

As introduced in Section 4.1.2, Transformer-based methods [Kurin et al., 2021, Gupta et al., 2022, Hong et al., 2022] have achieved state-of-the-art (SOTA) performance for universal morphology control, as they provide a natural and expressive approach to model the interactions between arbitrary number of limbs in different morphologies.

However, little attention has been paid to how to effectively utilize the morphology context in the control policy. While previous work proposes to concatenate morphology context to the node-wise state vector as an additional input to the policy network [Gupta et al., 2022], or add a morphology-aware positional encoding (PE) to the node embedding [Gupta et al., 2022, Hong et al., 2022], in effect they are equivalent to just adding a context-conditioned bias term to the node embedding layer in the network. This may lack sufficient model capacity to represent the diverse policies required to control different morphologies, as supported by both theoretical [Galanti and Wolf, 2020, Jayakumar et al., 2020] and empirical evidence [Sarafian et al., 2021, Ben-Iwhiwhu et al., 2022, Beck et al., 2023a, Rezaei-Shoshtari et al., 2023] from previous work in multi-task RL and meta-RL.

To better utilize task context for morphology control, we propose a hierarchical policy architecture consisting of a base controller, and a context modulator that regulates the control policy according to the characteristics of different morphologies. We name our method as ModuMorph to highlight its architecture novelty in contextual modulation. Specifically, ModuMorph includes two submodules. First,

we modulate network parameters in the base controller with HNs. Conditioned on the morphology context, HN can generate different policy parameters for different robots, which helps improve behavior diversity across morphologies. Second, we modulate the attention weight matrices in the TF layers of the base controller with morphology context alone, which introduces a structure-aware inductive bias on how different limbs in a robot should attend to each other to update their behaviors.

In principle, the proposed contextual modulator can be incorporated into any TF-based architectures for morphology control, while the HN module can also work with GNN-based architectures. Specifically, we use MetaMorph [Gupta et al., 2022], a SOTA TF-based method at the time when this work was conducted, as the backbone algorithm for experiments due to its superior performance and efficient implementation. Our experiments on a challenging morphology control benchmark called UNIMAL (UNIversal aniMAL) [Gupta et al., 2021], which includes hundreds of diverse morphologies, show that using contextual modulation improves not only the learning performance on training morphologies, but also the zero-shot generalization performance on unseen test morphologies, which validates the effectiveness of our method.

6.2 Background

MetaMorph [Gupta et al., 2022] is a TF-based method for universal morphology control (Figure 6.1). It concatenates time-variant proprioceptive state $s_t^{k,i}$ and time-invariant morphology context $c^{k,i}$ as the input vector of node i in robot k. The node input first goes through an embedding layer shared across all nodes to get node embedding $\boldsymbol{e}^i$, where robot index k and time index t are omitted for simplicity. Then the node embeddings are updated by a TF encoder. The key, query and value inputs to each TF layer are all determined by the node embedding, i.e., $\boldsymbol{k}^i = W_k \boldsymbol{e}^i, \boldsymbol{q}^i = W_q \boldsymbol{e}^i, \boldsymbol{v}^i = W_v \boldsymbol{e}^i$, where W_k, W_q, W_v are learnable weight matrices. After TF encoding, if there are globally exteroceptive observations, such as a height map of the agent's surroundings in a changing terrain, they are processed by an MLP and concatenated to the node embedding. Incorporating exteroceptive

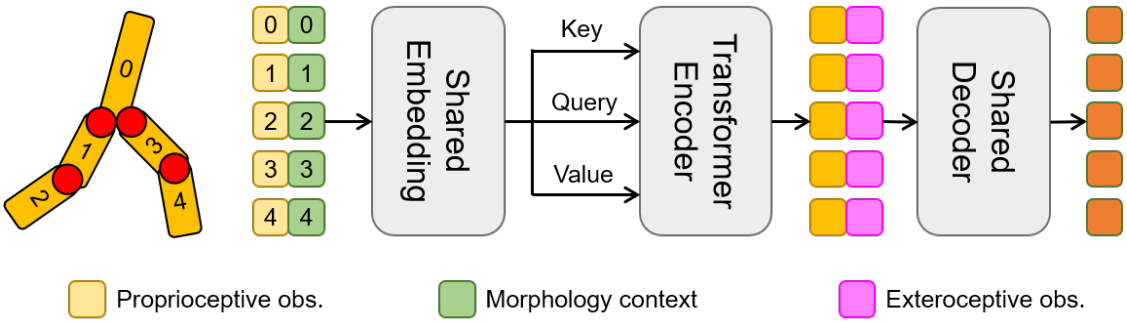

Figure 6.1: The framework of MetaMorph [Gupta et al., 2022]. On this two-leg robot for example, its nodes are ordered by depth-first tree search, with the torso node as the tree root. MetaMorph concatenates proprioceptive observations and morphology context as node inputs, processes them with a shared embedding layer, a TF encoder and a shared decoder sequentially. Exteroceptive observations are concatenated as decoder inputs if needed.

observations is essential to enable the agent to take different actions in different environmental conditions. Finally, the concatenated features go through an MLP decoder shared across all nodes to generate the actions for each node.

In MetaMorph, the morphology context c^k is only utilized as an additional node input, which is equivalent to adding a context-conditioned bias to the node embedding, as c^k remains unchanged on each robot. However, the optimal control policy can significantly vary across robots. Simply adding context-conditioned bias terms to the node embeddings while sharing all the other model parameters thus may not have sufficient expressive power to represent the diverse policies required for different morphologies [Galanti and Wolf, 2020, Ben-Iwhiwhu et al., 2022, Beck et al., 2023a]. To tackle this limitation, we propose two contextual modulation approaches to learn more diverse context-conditioned policies across different morphologies in Section 6.3.

MetaMorph also adds a learned positional encoding (PE) to the node embedding, i.e., $\boldsymbol{e}_t^{k,i} = \text{Encoder}(s_t^{k,i}, c^{k,i}) + \text{PE}^i$, where PE^i is a learnable vector that is shared across all the nodes with index i across different morphologies. Although PE is a common way to inject positional information back into TF as introduced before, we find that it actually provides little help in universal morphology control and thus omit it in Figure 6.1 for simplicity. We analyze why it does not work in Appendix A.3.

6.3 Universal Morphology Control via Contextual Modulation

In this section, we introduce two novel approaches to modulate the controller with morphology context. The framework of our proposed method is shown in Figure 6.2, which includes a base controller that generally follows the same architecture as MetaMorph, and a context network that modulates the base controller in two ways: (1) Instead of using shared embedding layer and decoder across all nodes, we generate node-wise embedding and decoder parameters with an HN conditioned on the morphology context. (2) The node embedding in the base controller is only used to generate the value input to the TF encoder, while the key and query are conditioned on the morphology context to generate a fixed attention matrix. We call these two approaches hypernetworks (HN) and fixed attention (FA) respectively.

6.3.1 Learning Morphology-Conditioned Policy Parameters

While GNN and TF-based controllers enable generalization across different morphologies, they also introduce a structural constraint that all nodes, both within a single robot and across different morphologies, have to share the same modular control policy. This hard parameter sharing mechanism [Ruder, 2017] may limit the behavior diversity across different nodes and the controller's model capacity to learn the optimal policy, as we usually expect different limbs to follow different control strategies based on their roles in the morphology. There is even evidence from neuroscience that the muscles in human body are controlled by different types of motor neurons based on their identities [Stifani, 2014].

Consequently, we hypothesize that instead of learning parameters shared across all nodes, learning morphology-conditioned node-wise parameters may improve behavior diversity and learning performance. We first conduct a proof-of-concept experiment to validate our hypothesis, then show how to generate context-conditioned parameters for each node in a scalable way via HN modulation.

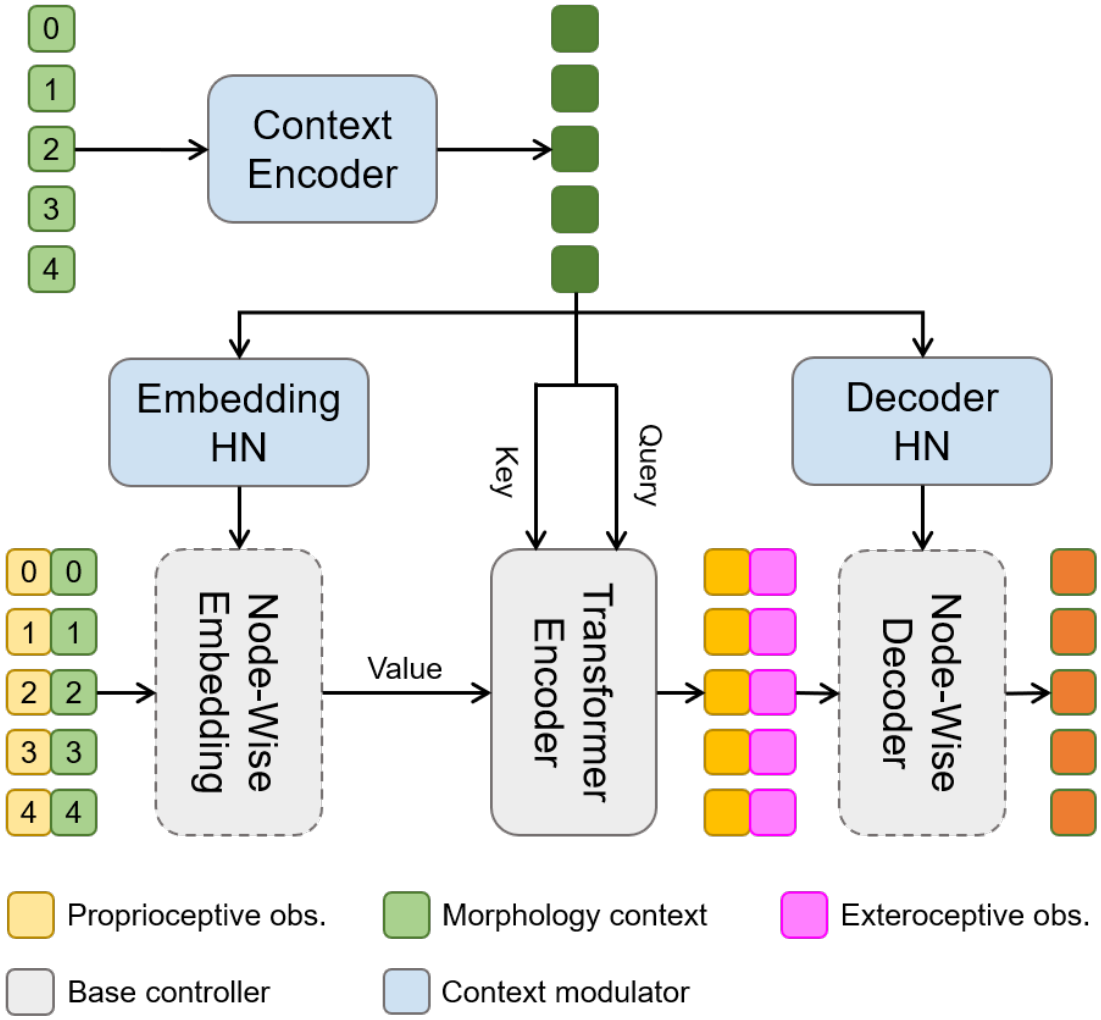

Figure 6.2: The hierarchical framework of ModuMorph. The morphology context modulates the base controller in two ways: (1) Generating context-conditioned embedding and decoder parameters via HNs. We use dotted edges to highlight that these two modules are not shared across different nodes and morphologies as in MetaMorph. (2) Generating morphology-conditioned attention matrices by using context embeddings as the key and query inputs to the TF encoder. Note that we use two separate context encoders for these two submodules, but show only a single shared context encoder in this figure for ease of illustration.

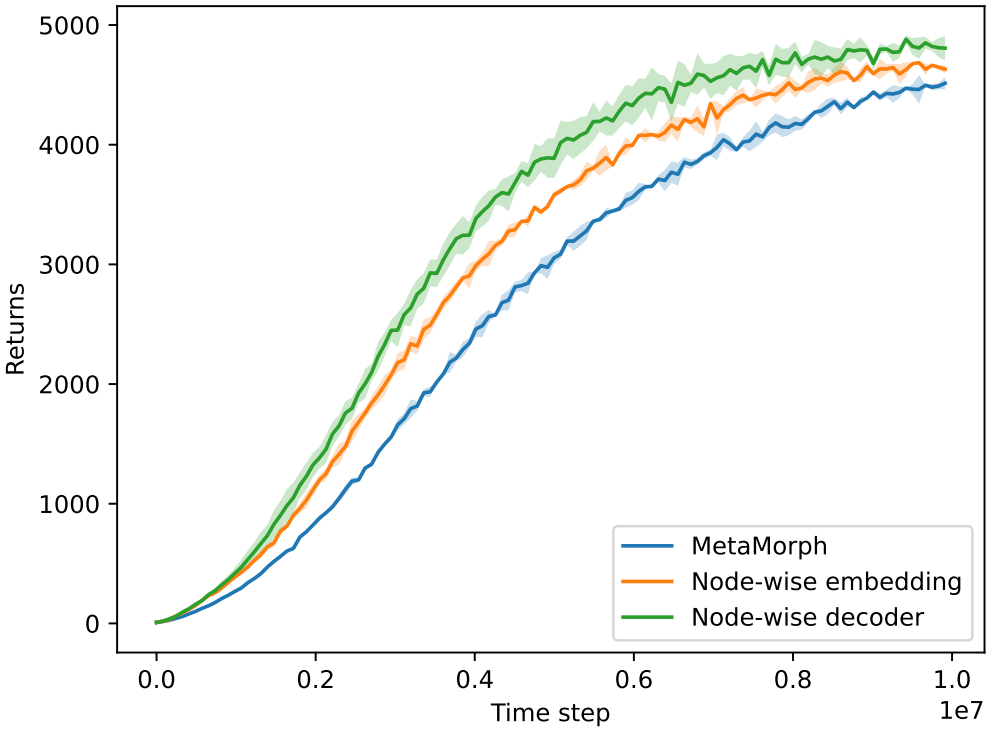

Figure 6.3: Single-robot learning curves averaged over 20 robots. We train with three different models: (1) MetaMorph with no modification; (2) MetaMorph with node-wise input embedding layer; and (3) MetaMorph with node-wise decoder layers. The shaded area of each curve represents the standard deviation of the return across the 20 robots.

A Proof-of-Concept Experiment

We design a motivating experiment to show that enabling behavior diversity across nodes via learning node-wise parameters can improve training performance.

We train a MetaMorph model on a single robot. However, instead of learning a single embedding layer shared across all the nodes, we learn a separate embedding layer for each node. Similarly, we train another MetaMorph variant with node-wise decoder. We randomly sample 20 morphologies from the UNIMAL benchmark and run single-robot training on each of them for 10M steps. As shown in Figure 6.3, the node-wise embedding and node-wise decoder variants both outperform MetaMorph, which validates that enabling behavior diversity across different nodes in a robot is helpful.

However, the approach used in our proof-of-concept experiment is impractical for two reasons: (1) It cannot generalize to new morphologies unseen during training; (2) Learning separate parameters for each node is not scalable, as the number of parameters to learn grows linearly with the number of morphologies. Consequently, we next introduce HN modulation to enable behavior diversity while maintaining generalization and scalability of the learned model.

Morphology-Conditioned Parameter Generation via Hypernetworks

Learning separate parameters for each node has generalization and scalability issues, as it does not utilize the contextual similarity between nodes. Intuitively, if two nodes play similar roles in two different morphologies, e.g., they are both the left thigh in their robots, then we may expect them to also have similar node-wise parameters. As the morphology context of each node can provide rich information about the similarities between nodes, we propose to generate node-wise parameters via a morphology-conditioned HN, i.e., $\theta^{k,i} = \text{HN}_\phi(c^{k,i})$, where $\theta^{k,i}$ is the node-wise parameters of node i in robot k, and HN_ϕ is the morphology-conditioned HN shared across all nodes. Generating node-wise parameters via HN is scalable, as we only need to additionally learn one set of HN parameters ϕ regardless of the number of morphologies, and generalizable, as we can directly feed new node context on unseen morphologies into the HN to generate the corresponding control parameters.

To better illustrate how HN-generated parameters work, we take the node embedding layer as an example. In MetaMorph, the embedding layer consists of a single set of weights W and bias b shared across all the nodes, i.e., $e_t^{k,i} = Wx_t^{k,i} + b$, where $x_t^{k,i}$ and $e_t^{k,i}$ are the node input and node embedding of node i in robot k at timestep t. For our HN approach, however, the embedding layer's parameters are different across nodes, i.e., $e_t^{k,i} = W^{k,i}x_t^{k,i} + b^{k,i}$, where $W^{k,i} = \text{HN}_W(c^{k,i})$ and $b^{k,i} = \text{HN}_b(c^{k,i})$ are node-wise weights and bias generated by HN conditioned on the node context.

In practice, we only generate the parameters of the embedding layer and the decoder in the base network with HN. The TF encoder is still shared across all

morphologies, as there are too many weight matrices in a TF layer to efficiently generate them via HN.

6.3.2 Morphology-Conditioned Fixed Attention

The attention matrix plays an important role in TF as it determines how different nodes attend to each other to update their node embeddings.

Existing methods use the node embedding in the base controller as the key and query inputs to generate the attention weights, which change dynamically at every timestep due to the time-variant proprioceptive state. However, determining attention weights in such a dynamic way may not well reflect how different nodes interact. Instead, it may be the case that the attention of one node to the others should depend *solely* on the robot morphology. For example, when you want to grasp an object within your reach, you pay more attention to the state of your arm than your leg to determine the movement of your hand. And whether you are standing or sitting, which changes the proprioceptive observations of body parts, has little influence on this attention strategy for grasping.

Consequently, we hypothesize that it may be beneficial to incorporate such intuition as an inductive bias into the controller architecture, i.e., different nodes should attend to each other in a static way, and the attention weights should be learned from the morphology context alone. To realize this inductive bias, we pass the node context through a context encoder, and use the context embedding as the key and query to modulate the TF in the base controller, while the node embedding in the base network is only used as the value input (Figure 6.2). For each robot, as the morphology context remains unchanged, the attention matrix will be fixed to reflect the structural relationships between nodes.

6.3.3 Computational Cost

HN learning will increase computational cost during training. However, there is no additional cost during deployment, as we can generate node-wise parameters with HN on each robot in advance. On the other hand, except for context encoding, FA

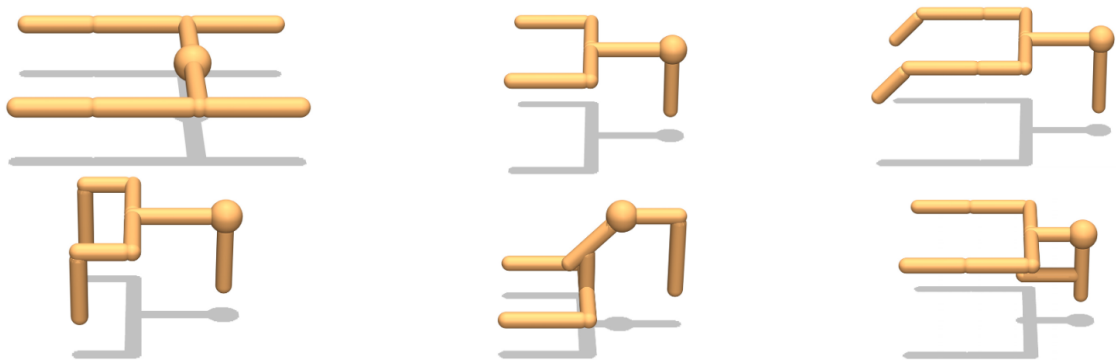

Figure 6.4: Example morphologies from the UNIMAL benchmark. Image credit to Gupta et al. [2021].

will introduce no additional computation during training, as it just changes the query and key inputs to the TF. Furthermore, FA could even reduce the computation during evaluation, as we need to compute the FA weights only once for each robot and then can reuse it afterwards. See Appendix A.1 for more implementation details of our contextual modulation method.

6.4 Experimental Setup

Environments We experiment on the UNIMAL benchmark as used in MetaMorph [Gupta et al., 2022] for a fair comparison, which includes 100 training robots and 100 test robots with diverse morphologies (Figure 6.4) [Gupta et al., 2021].

We consider five different environments from Gupta et al. [2021] for our experiments (Figure 6.5): (1) Flat terrain (FT): maximize locomotion distance on a flat floor; (2) Incline: maximize locomotion distance on an incline of 10 degrees; (3) Exploration: maximize the number of distinct grids visited on a flat arena discretized into grids; (4) Variable terrain (VT): maximize locomotion distance on a variable terrain with three different terrain types. For each episode, a new terrain is generated by randomly sampling a sequence of terrain types and interleaving them with flat terrain. (5) Obstacles: maximize locomotion distance on a flat terrain with randomly positioned obstacles. The first three environments only require proprioceptive observations and morphology context as model input, while the last two require height map information surrounding the robot as additional

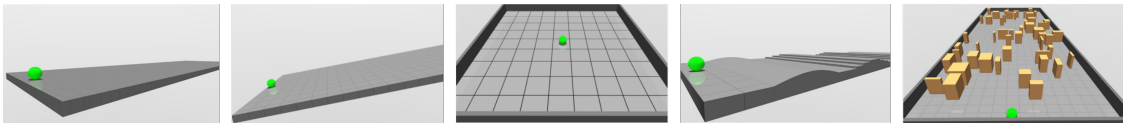

Figure 6.5: The five environments used for experiments. From left to right: Flat terrain (FT), Incline, Exploration, Variable terrain (VT), Obstacles. Image credit to Gupta et al. [2021, 2022].

exteroceptive observation input to the controller, so that the agent can perceive and react to different terrains or obstacles in its way.

Baselines We consider both multi-robot (MR) and single-robot (SR) baselines.

For MR training, we use MetaMorph [Gupta et al., 2022] as the baseline. However, we notice two issues in the MetaMorph code and thus implement a modified version to eliminate these issues. We name the modified version as MetaMorph*, and build our modulation modules upon it. In general, MetaMorph* achieves similar or even better performance compared to MetaMorph in most environments, and we report the results of both for a fair comparison. See Appendix A.3 for more details on the difference between MetaMorph and MetaMorph*.

For SR training, we train an MLP policy on each robot, and consider two different training budgets for different purposes. First, we do SR training with the same per-robot budget as in MR training, which is named as *SR-fair* and used to compare the sample efficiency of MR and SR learning. Second, We do SR training for 10M steps on each robot till convergence, which is named as *SR-10M* and used as a performance upper bound. We choose to use an MLP of 3 hidden layers, each with 256 hidden units by performing a grid search over the layer number and hidden size.

Ablations As we propose two different approaches for contextual modulation, we ablate by adding only HN or FA to the MetaMorph* baseline, and compare them with the full version of adding both to MetaMorph*.

Training Setup We train for 100M steps in FT, Incline, and Exploration, and 200M steps in VT and Obstacles, as they are more challenging to solve due to variable terrains. We run three random seeds for each method in each environment, and

report the average performance and standard deviation. Following the same setup as in MetaMorph, we use PPO [Schulman et al., 2017] as the optimization algorithm. Similar to previous work [Dossa et al., 2021, Sun et al., 2022], we notice that the early stopping threshold has a significant influence on PPO performance (see Appendix A.2). We thus tune this hyperparameter over the candidate set of $\{0.03, 0.05\}$ for each method in each environment. All the remaining hyperparameters follow the same setup as in MetaMorph for a fair comparison.

Evaluation Setup We evaluate zero-shot generalization to unseen robots under two settings with increasing difficulties. First, we test on new robots that have the same topology as the training ones but differ in kinematics or dynamics parameters. For each parameter to test, we create 4 variants of each training robot by randomly changing the value of the corresponding parameter on all the limbs. Second, we evaluate zero-shot generalization to new morphologies which have different topology graphs compared to those seen during training. The robots used for both settings are adopted from Gupta et al. [2022] for a fair comparison. For each robot, we collect 64 episodes with randomly sampled initial states. We use the average episodic return over the test morphologies to measure the policy's zero-shot generalization performance.

6.5 Results

6.5.1 Training Results

As shown in Figure 6.6, all MR methods significantly outperform SR-fair, illustrating the advantage of Multi-task RL in sample efficiency. However, there is still a clear performance gap between the two MR baselines and SR-10M. Our method significantly reduces this gap (even outperforms SR-10M in Exploration), and consistently outperforms the two MR baselines in all the five environments w.r.t. both learning efficiency and final performance. Compared to MetaMorph*, which our method builds upon, contextual modulation improves the final performance by 19%, 53%, 48%, 31% and 29% in each environment respectively. Although

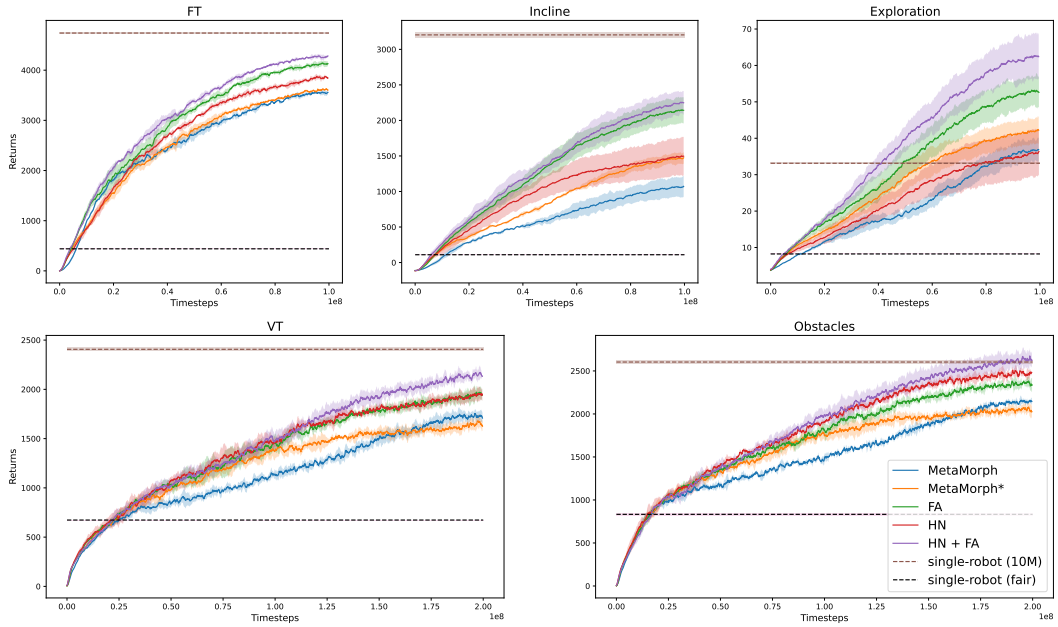

Figure 6.6: The training curves of different methods in each environment. Each curve represents the average return over the 100 training robots attained by each method during training. The shaded area of each curve represents the standard deviation across three random seeds.

MetaMorph outperforms MetaMorph* in VT and Obstacles, our method still consistently outperforms MetaMorph in these two environments.

Ablation results show that both FA and HN contribute to the effectiveness of our method. FA consistently improves upon MetaMorph* in all the environments, and seems to be more effective in the three environments with unchanged terrain (VT, Incline and Exploration). On the other hand, HN helps in three of the five environments, and contributes more in the two environments with changing terrains (VT and Obstacles). Next we give some more detailed analysis on the two submodules of our method.

Fixed Attention FA introduces a strong inductive bias that how each node attends to the others should be solely determined by the morphology, and is proved to be effective in all the five environments. However, in environments with changing terrains, the robot may need to adopt different gaits in different terrains. While in principle this can be realized by taking terrain information as additional decoder

input, an alternative idea is to further condition the attention weights on the terrain information, so that the nodes can attend to each other with dynamic terrain-conditioned weights to realize different gaits. We thus tried adding the height map as an additional input to compute attention weights, but got results even worse than the MetaMorph* baseline. The reason might be that the robot can already adapt to different terrains by taking the height map as decoder input, so the attention module only needs to model node interactions, while using terrain info as attention inputs may introduce further optimization challenges. Nevertheless, it remains an interesting open question whether learning performance can be further improved by properly incorporating terrain information into attention computation.

Hypernetworks HN provides more significant improvement in the more challenging environments of VT and Obstacles with variable terrains. This may imply that behavior diversity across nodes is more important in environments that require complicated locomotion skills, while in easier terrains, the benefits of HN may be outweighed by its optimization challenges. Moreover, adding HN harms learning performance in the Exploration environment. The training statistics show that the HN variant has a much higher error in value prediction compared to the other methods in Exploration, which may be the reason for its worse performance. Value prediction is particularly hard in Exploration, as the value depends on not only the robot's current state, but also the robot's visitation history in the arena, which is not accessible to the robot. We hypothesize that this problem is more severe when using the more complex HN architecture.

6.5.2 Zero-Shot Generalization to Kinematics and Dynamics Variations

As shown in Figure 6.7, our method consistently outperforms the baselines, with an average improvement ratio of 26%, 50%, 43%, 28%, 30% in each environment compared to MetaMorph*, which validates that our method not only enables better multi-robot training, but also generalizes better to unseen robots with parametric variations. However, we also notice that zero-shot generalization to kinematics

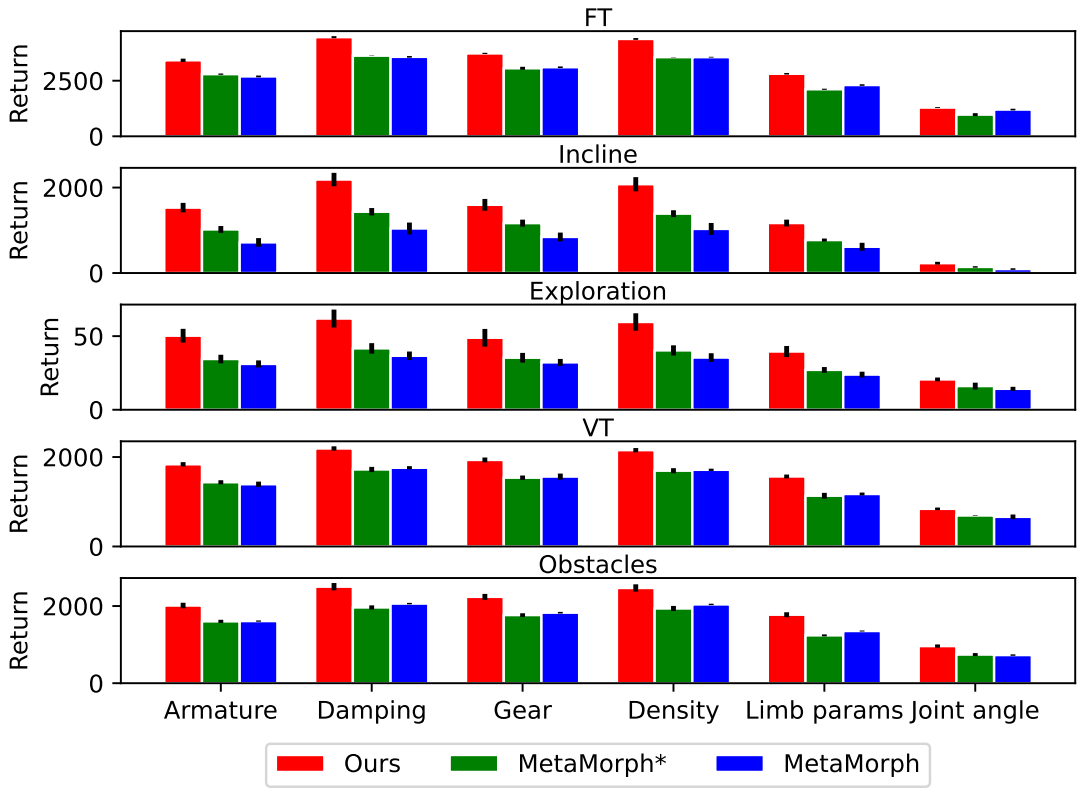

Figure 6.7: Zero-shot generalization performance of different methods to kinematics and dynamics variations. The rows correspond to the 5 environments, and the columns correspond to parametric variations in 6 different morphology context parameters. Each bar represents the average return over the 400 test robots attained by each method. The error bar of each method represents the standard deviation across three random seeds.

variation (especially joint angles) is much harder, as is also reported in Gupta et al. [2022]. This is mainly because that changes in joint angles may significantly influence the feasible actions and the gait for locomotion, and how to tackle this challenge is an interesting direction for future work.

6.5.3 Zero-Shot Generalization to Unseen Morphologies

Table 6.1 shows the zero-shot generalization performance to unseen morphologies of different methods in each environment, which generally follows the same trend as during training, i.e., the model with higher training scores also performs better on unseen morphologies. Specifically, our method outperforms MetaMorph* by 18%, 29%, 24%, 27% and 37% in each environment respectively. This implies that

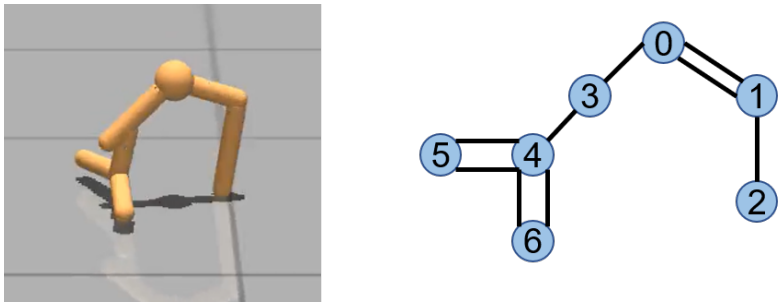

Figure 6.8: The example morphology and its morphology tree. Some limbs are connected to their parents via two joints, represented by the two edges between nodes. The sphere node in the left figure is the torso of the robot and also the root of the morphology tree.

our contextual modulation method can indeed better model the dependence of the control policy on the robot morphology, instead of simply overfitting to the training morphologies via its more complicated architecture designs. However, the large variance in the return across different seeds does imply that improving zero-shot generalization on unseen morphologies is still an open problem for future work.

Environment	MetaMorph	MetaMorph*	FA	HN	FA+HN
FT	$\underline{1384 \pm 62}$	1266 ± 105	$\underline{1439 \pm 27}$	1259 ± 112	$\mathbf{1490 \pm 59}$
Incline	27 ± 32	$\underline{312 \pm 136}$	$\mathbf{468 \pm 58}$	$\underline{312 \pm 97}$	$\underline{403 \pm 66}$
Exploration	$\underline{19 \pm 1}$	$\underline{19 \pm 1}$	$\underline{22 \pm 2}$	16 ± 3	$\mathbf{23 \pm 3}$
VT	$\underline{752 \pm 62}$	$\underline{767 \pm 23}$	$\underline{860 \pm 112}$	$\underline{900 \pm 24}$	$\mathbf{971 \pm 122}$
Obstacles	866 ± 30	829 ± 50	937 ± 46	969 ± 47	$\mathbf{1133 \pm 12}$

Table 6.1: Zero-shot generalization performance of different methods to test morphologies with unseen topology graphs. Each score represents the mean and standard deviation of the average return over the 100 test robots attained by each method across three random seeds. The best method in each environment is marked in **Bold**. The methods that are not statistically significantly different from the best method are marked by underline based on Welch's t-test with a significance level of 0.05 [Colas et al., 2019].

6.5.4 Qualitative Analysis

In general, both our method and the baselines can control different robots for locomotion. However, ModuMorph generates smoother and more stable locomotion behaviors, thus performs better overall. We conduct a detailed qualitative analysis in the FT environment to illustrate the difference in the locomotion behaviors learned by different methods.

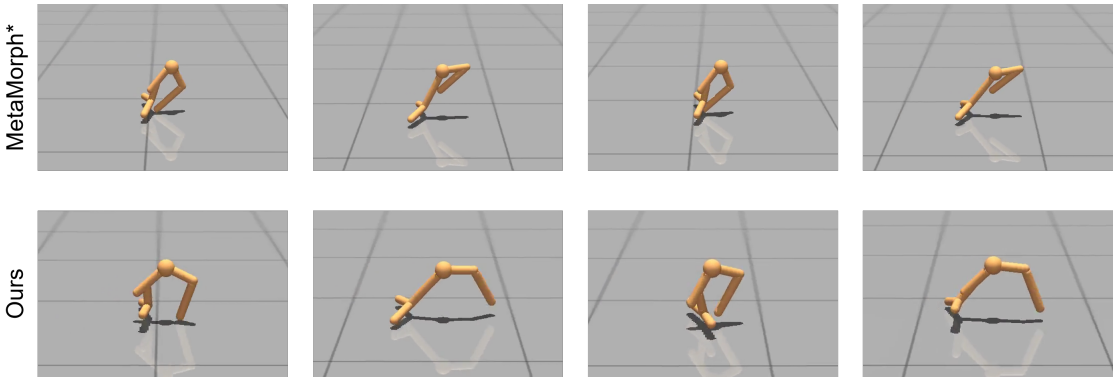

Figure 6.9: Visualization of the locomotion trajectories learned by MetaMorph* and our method on the example robot.

We experiment on an example morphology as shown in Figure 6.8, and compare the locomotion learned by MetaMorph* and our method in Figure 6.9. For MetaMorph*, the robot moves forward by kicking the ground with its front limb. However, the front limb does not fully stretch out, thus provides limited forward force and makes the body unstable. In 1000 timesteps, the robot falls twice and only achieves a return of 1375 in its best trial. By contrast, our method learns a policy that fully stretches out the front limb to provide stronger forward force, and better coordinates the movement of the front and back limbs. The robot runs more stably without any failure during evaluation, and achieves a much higher return of 4612.

In addition to behavior visualization, we further analyze the correlation between the action sequences taken by different limbs as an indicator of behavior synergies. Intuitively, if the behavior of two limbs are better coordinated, we expect their action sequences to have a higher correlation coefficient in absolute value. Figure 6.10 shows the correlation matrix between different action dimensions on the example morphology. For our method, joint 2 (which controls the front limb 2) is much better synchronized with joints 3 and 4 (which control limbs 3 and 4 in the back). This reflects how the periodic gait of our method in Figure 6.10 emerges.

In the VT environment, MetaMorph* has an average action correlation of 0.24 across all training morphologies, while our method has 0.29. This higher correlation indicates a better synergy between limbs, which may help explain why our method shows more fluent and stable locomotion skills.

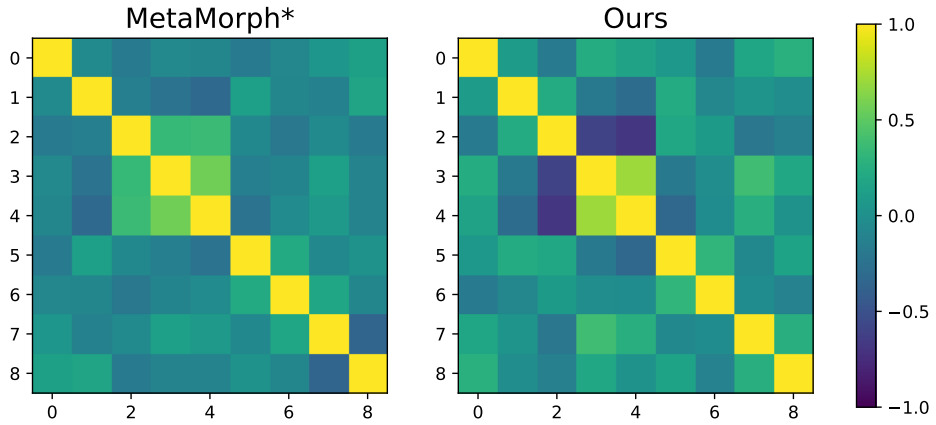

Figure 6.10: The correlation matrix of different action dimensions on the example morphology.

6.6 Related Work

In addition to the related work on morphology-aware methods for universal morphology control that we have introduced in Section 4.1.2, we further review related work that applies contextual modulation to RL in other domains.

The optimal policy for a task usually critically depends on the task context that defines the task's characteristics. Consequently, conditioning the policy on the task context can significantly improve its training performance and generalization ability over a distribution of tasks compared to context-agnostic learning [Benjamins et al., 2023]. To learn a context-conditioned policy, an importance design choice is the architecture used to incorporate the task context into the policy, which reflects our inductive bias on the task structure. Instead of simply concatenating the context features to the state features, which is limited in model capacity [Galanti and Wolf, 2020], different architectures have been proposed to modulate the policy via task context, such as feature-wise multiplication [Ben-Iwhiwhu et al., 2022, Benjamins et al., 2023], a routing network that determines how to combine different skill modules for a specific task [Yang et al., 2020, Sodhani et al., 2021, Ponti et al., 2022], and HNs [Yu et al., 2019, Peng et al., 2021, Sarafian et al., 2021, Beck et al., 2023a, Rezaei-Shoshtari et al., 2023]. Our work shares a similar motivation as these

methods, but focuses on a more challenging domain of universal morphology control where different tasks do not share the same state and action space.

6.7 Conclusion and Future Work

In this chapter, we investigated how to better pretrain a universal control policy for different robot morphologies. To better model the dependency of the control policy on the robot morphology, we proposed a hierarchical architecture to modulate the base controller with morphology context, which includes an HN module that generates morphology-conditioned policy parameters, and a morphology-dependent fixed attention module to modulate the TF layers in the base controller. Experimental results validated the effectiveness of our method on both multiple training robots and unseen test morphologies.

For future work, an interesting direction is how to learn better context representation for modulation. In this work, we directly used the original node context features provided in the benchmark as the modulator input, and a simple MLP as the context encoder. How to design better context features, such as utilizing node connectivity information, and how to design better context encoding architectures are both interesting topics to investigate. Another potential direction is how to improve zero-shot generalization performance on unseen robots, as there is still a large gap between the current generalization results and the optimal performance we can achieve by directly training on the test robots. Finally, while we focus on the problem setting of learning a universal controller over a set of pre-given robot morphologies, an interesting direction for future work is to apply our method to the closely related problem setting of jointly optimizing the morphology design and its control policy [Schaff et al., 2019, Wang et al., 2019, Yuan et al., 2022] as introduced in Section 4.2.2.

Before ending this chapter, we want to highlight a very interesting observation from the results reported in Figure 6.6: Although MLP-based methods cannot generalize well across morphologies, they achieve the best performance on each single robot if given a sufficiently large training budget (see the results of SR-10M),

and have much smaller model size and higher inference efficiency compared to TF-based methods. So an interesting follow-up question is that if we can combine the advantages of MLP- and TF-based methods into a single model, which motivates the idea that we will explore in the next chapter.

7

HyperDistill: Distilling Morphology-Conditioned Hypernetworks for Efficient Universal Morphology Control

Contents

This chapter proposes *HyperDistill*, an HN-generated MLP policy for universal morphology control that enjoys both good generalization performance as ModuMorph and high inference efficiency. The key novelty of this chapter is the proposal of a general principle called *knowledge decoupling* for inference acceleration, which we will utilize not only for universal morphology control in this chapter, but also for cross-skill generalization in the next chapter.

7.1 Introduction

Based on the evidence from the previous chapter and related work [Wang et al., 2018, Gupta et al., 2022], MLP- and TF-based methods have different advantages for universal morphology control. On one hand, a compact MLP policy is usually sufficient to achieve good performance on a single robot, but generalizes poorly to other robots. On the other hand, a TF policy achieves SOTA multi-robot training and generalization performance, but has significantly higher memory, computation, and energy costs than MLP, all of which are key considerations when deploying the policy on real-world robots with constrained hardware [Hutter et al., 2016, Zhao et al., 2021a, Brohan et al., 2023, Leal et al., 2024]. For example, ModuMorph requires about 40M FLOPs for a single inference step on a robot with just 10 limbs, more than 100 times that of a single-robot MLP policy with similar performance, and this efficiency gap increases proportionally with the number of limbs. So a natural question arises: *Can we learn a universal policy with the strong performance of TF but the inference efficiency of MLP?*

In this chapter, we get the best of both worlds by utilizing HNs. Our key intuition is that, compared to monolithic TF that needs to activate the whole model during inference, the hierarchical architecture of HN provides a natural way to decouple inter-task knowledge about how to generalize across tasks which is encoded by the context-conditioned HN, and intra-task knowledge about how to solve a specific task which is encoded by the base network generated for the corresponding task context, which we call the *knowledge decoupling hypothesis*. In other words, *HNs can learn a generalist policy at training time, but only activate a compact specialist policy that is sufficient to solve a specific task at test time for efficient inference.*

Under the problem setting of universal morphology control, a morphology-conditioned HN can provide sufficient model capacity to accommodate inter-robot knowledge, while a compact generated MLP policy is sufficient to control a single robot with good performance. By contrast, TF encodes both kinds of knowledge with a single large model, which introduces high inference redundancy when deployed on a specific robot.

Specifically, suppose we have trained a set of compact single-robot MLP policies $\{\pi_k\}$, each with good performance on a different robot k. Motivated by the intuition that the optimal control policy of a robot critically depends on its morphology [Gupta et al., 2022, Xiong et al., 2023], we train an HN that takes the morphology context c^k of robot k as input to predict the corresponding MLP policy parameters π_k. For each robot, as the morphology context c^k is constant, the HN-generated parameters are also fixed. Consequently, we only need to call it once before deployment to generate the base MLP, while the HN itself is not needed for policy execution. This yields a universal policy that works like an MLP at inference time but still has the potential to achieve good performance on both training and unseen test robots as long as the HN can generalize well across morphologies.

While training such an HN policy via RL is a straightforward option, empirically we find that it is unstable and significantly underperforms a universal TF policy. Instead, we adopt a policy distillation (PD) approach by distilling a universal TF teacher policy into an HN student policy via behavior cloning (BC). While it is not hard to distill a student policy to match the teacher's performance on the training task(s) [Parisotto et al., 2015, Rusu et al., 2015, Czarnecki et al., 2019], a key challenge in our problem setting is to maintain the teacher policy's zero-shot generalization performance after distillation. We identify several critical algorithmic choices in PD that influence the generalization performance of the student policy to unseen tasks: (1) The choice of the teacher(s); (2) Architecture alignment between the teacher and the student; (3) The number of tasks on which to collect training data for PD; and (4) Regularization in task space. We believe that these algorithmic choices could serve as general guidelines for improving task generalization of PD in other domains as well.

We name our approach as HyperDistill to highlight its two key components: (1) The HN architecture to achieve both good performance and efficient inference via knowledge decoupling; and (2) Training via policy distillation to successfully learn such an HN policy. We experiment on the same UNIMAL benchmark [Gupta et al., 2021] as in the previous chapter. HyperDistill achieves similar performance as

ModuMorph on both training and unseen test robots, while significantly reducing the model size by 6-14 times, and computational cost by 67-160 times in different environments at inference time. The experimental results strongly support our knowledge decoupling hypothesis, which could serve as a general principle to improve inference efficiency in other domains.

7.2 Limitations of Existing Architectures

In this section, we analyze the limitations of two representative architectures for universal morphology control: TF, which achieves SOTA performance but is expensive to run at inference time, and MLP, which runs efficiently and achieves good performance on each single robot but performs poorly in the multi-robot setting. We first highlight a knowledge decoupling issue that exists in both TF and MLP, then discuss the lack of order-invariance in MLP.

Knowledge Decoupling Intuitively, knowledge learned by a universal policy can be decomposed into two parts: inter-task knowledge for generalization across robots, and intra-task knowledge about how to control a specific robot. Since both TF and MLP use a single set of parameters to encode both kinds of knowledge, they cannot decouple them from each other. Consequently, at inference time, there may be a lot of redundant knowledge in the policy that is irrelevant to solving the current task, which harms efficiency. It also explains why MLP can perform well on a single robot but not under the multi-robot setting, as the model capacity of a compact MLP is sufficient to accommodate task-specific knowledge of a single robot, but not enough to encode both inter-task and intra-task knowledge under a multi-robot setting. The same conclusion also holds for TF when compressing a large TF into a smaller one, as we confirm experimentally in Section 7.4. In summary, due to lack of knowledge decoupling, TF and MLP have to trade off between performance and efficiency at inference time.

Order-Invariance Issue with MLP A further issue with MLP is that it is not invariant to the order of limbs across robots. TF is order-invariant, as all the limbs across different robots share the same set of parameters. By contrast, in a multi-robot MLP, only the limbs with the same index across different robots share the same subset of parameters in the input and output layers, so how we order the limbs influences the policy output. As we do not have a consistent way to order the limbs across different morphologies, multi-robot MLP can overfit to spurious patterns in the manually chosen limb indexing method, harming generalization.

7.3 HyperDistill

This section introduces HyperDistill, which achieves both good performance and high efficiency at inference time. In Section 7.3.1, we introduce an HN-generated MLP policy to tackle the limitations of existing methods as discussed in the previous section. In Section 7.3.2, we discuss why and how we train this HN via policy distillation, and analyze some of its critical algorithmic designs.

7.3.1 HyperDistill Architecture

To tackle the limitations of TF and MLP, we introduce an HN that takes morphology context as input to generate an MLP base policy for each robot. First, it supports knowledge decoupling, as the morphology-conditioned HN encodes inter-task knowledge, while the base MLP encodes intra-task knowledge. The expensive HN is called only once to generate a task-specific base policy for each new robot, while the much smaller base network is sufficient to efficiently control the robot. Second, the base MLP generated by the HN is order-invariant, as the parameters associated with each limb no longer depend on the limb index, but only condition on the limb's context representation.

The overall architecture of HyperDistill is shown in Figure 7.1. To handle variable number of limbs across robots, we generate limb-wise parameters for the input and output layers of the base MLP, as the subset of parameters corresponding to each limb still has fixed dimension based on the modular space assumption in

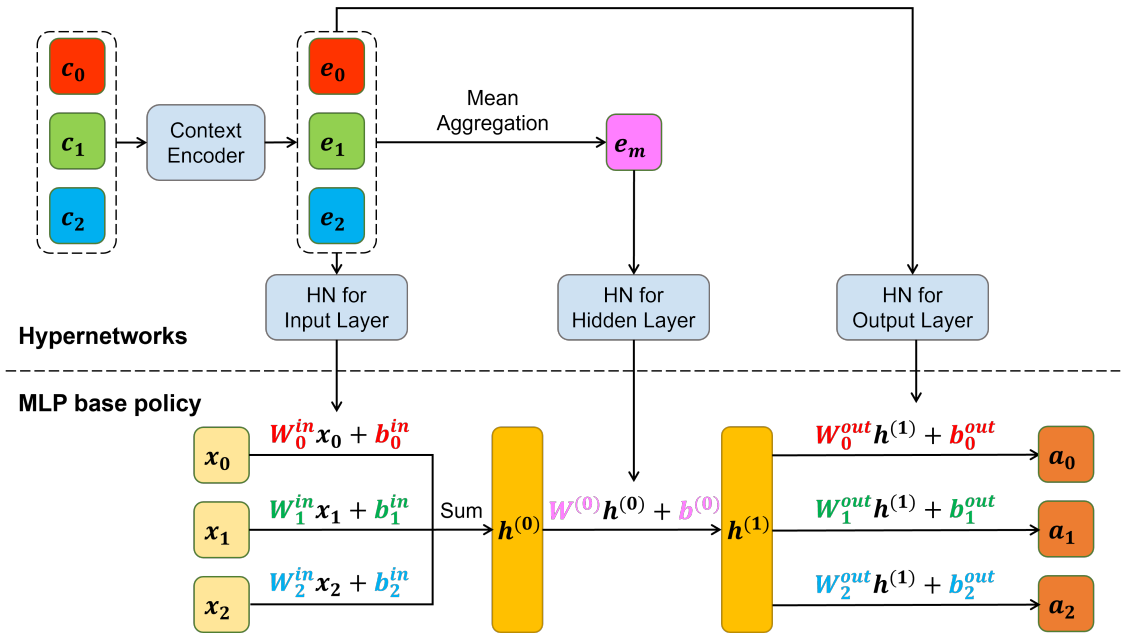

Figure 7.1: The architecture of HyperDistill. Different colors highlight the correspondence between the parameters in the base network and the context embedding they condition on via HN. We only show one hidden layer in the base MLP for ease of illustration. More hidden layers can be easily added in a similar way.

Section 2.2.1. For simplicity, we only use the limb index i for subscript, and omit the robot index k and timestep t. For the input layer of the base MLP, we have $h^{(0)} = \sigma\left(\sum_i W_i^{\text{in}} x_i + b_i^{\text{in}}\right)$, where $W_i^{\text{in}} = \text{HN}_W^{\text{in}}(e_i)$, $b_i^{\text{in}} = \text{HN}_b^{\text{in}}(e_i)$, σ is the activation function, x_i is the input features of limb i to the base MLP, and $e_i = f(c_i)$ is the context embedding of limb i generated by the context encoder f. As the whole policy conditions on the morphology context through HN, we no longer need to concatenate context features c_i to the network input, so we have $x_i = s_i$. Similarly, for the output layer, we have $a_i = W_i^{\text{out}} h^{(L)} + b_i^{\text{out}}$, where $W_i^{\text{out}} = \text{HN}_W^{\text{out}}(e_i)$, and $b_i^{\text{out}} = \text{HN}_b^{\text{out}}(e_i)$. For the hidden layers with fixed dimensions, we first aggregate the context embedding of different limbs using the mean to get a context embedding for the whole morphology as $e_m = \frac{1}{N}\sum_i e_i$, then pass e_m through HN output heads to generate the hidden layer parameters, i.e., $h^{(l+1)} = \sigma\left(W^{(l)} h^{(l)} + b^{(l)}\right)$, where $W^{(l)} = \text{HN}_W^{(l)}(e_m)$, and $b^{(l)} = \text{HN}_b^{(l)}(e_m)$.

Context Representation Learning

The quality of the HN-generated policy critically depends on the quality of the learned context embedding, e.g., if e_i of different limbs are too similar, the policy will generate similar actions across different limbs, which is unlikely to be a good policy.

The context representation should be both discriminative to encode the diverse behaviors of different limbs, and generalizable to unseen robots. We adopt two approaches to achieve this goal. First, we apply some simple transformations to the context features c_i to make them more discriminative across limbs (see Appendix B.1). Second, we use a TF as the context encoder to enrich each limb's context representation by interacting with other limbs in the robot, e.g., if two limbs in two different morphologies have the same hardware configurations, then we cannot tell them apart based on their own context features alone, but the TF context encoder can learn a distinguishable representation by further encoding global morphology information. Since the TF context encoder is also a part of the HN, it is not needed at inference time, unlike a TF policy that uses TF as the controller.

7.3.2 Training HyperDistill via Policy Distillation

In principle, we can train a universal HN policy from scratch via RL. However, empirically we find that the RL training process is unstable and the learned policy significantly underperforms a TF policy, possibly because both HN and RL are known to be unstable during learning, and combining them together further exacerbates the optimization challenges.

Instead, we adopt policy distillation (PD) [Parisotto et al., 2015, Rusu et al., 2015] by first training a universal TF policy, then distilling it into an HN policy via BC, which replaces RL training with more stable supervised learning to alleviate the optimization challenge. Moreover, as BC empirically requires much less time to train than RL, we can reuse the same pretrained teacher policy for more efficient evaluation of different algorithmic choices in PD.

Given a set of training robots and a universal TF policy trained on them as the teacher π^T, we first collect expert trajectories on each training robot with π^T to

generate a training dataset $\mathcal{B}^T$ for PD. Then we train an HN student policy π by minimizing the KL-divergence between the action distributions generated by the teacher and student on transitions randomly sampled from the buffer:

$$L_\pi^{\text{PD}} = \mathbb{E}_{s,c\sim\mathcal{B}^T}\left[KL(\pi^T(a|s,c)||\pi(a|s,c)\right]. \tag{7.1}$$

While more sophisticated loss functions [Czarnecki et al., 2019] have been proposed to facilitate PD by collecting online data with the student, empirically we find that this simple BC loss is sufficient to learn a student policy that matches the teacher's performance on the training robots, without having to collect further samples with the student.

However, we also want the student to zero-shot generalize as well as the teacher on unseen test robots. While several prior works evaluate zero-shot generalization of a student policy to unseen tasks in different problem settings [Chen et al., 2022, Furuta et al., 2023, Wan et al., 2023], none of them systematically investigate how different algorithmic choices in PD influence the student's generalization performance. In this work, we highlight four key factors that may influence the generalization gap between the teacher and student policies in PD.

The Choice of PD Teacher(s) Theoretically, we can either train a universal TF teacher on multiple robots, or train multiple single-robot MLP policies on different morphologies as the teachers [Furuta et al., 2023]. While the average performance of the single-robot MLPs is similar to or even better than that of a universal TF on the training robots as shown in the previous chapter, we hypothesize that the student policy distilled from a universal teacher policy generalizes better to unseen robots. Intuitively, as the single-robot teacher policies are trained via RL separately, they can be dramatically different from each other, which may exacerbate the discontinuity of the distilled policy in the parameter space and harm generalization.

Student-Teacher Architecture Alignment We hypothesize that the generalization gap between teacher and student is smaller when their neural architectures are more aligned [Hao et al., 2023]. Intuitively, the inductive bias introduced by a more aligned architecture helps the student extrapolate to unseen task context and state in a more consistent way with the teacher, which helps reduce the generalization gap.

However, as we are trying to distill from a TF to an HN, there is an unavoidable mismatch between the teacher and student architectures. The two algorithmic choices discussed next may help compensate for the generalization gap caused by this architecture misalignment.

Number of Tasks for Distillation Training To better align the teacher and student policies' behaviors on unseen tasks, a natural idea is to train the student policy on the teacher's demonstrations collected from more tasks, so that the PD training data better covers the task space. This is different from training the teacher policy on more tasks, which is an effective but orthogonal way to improve task generalization. Our approach does not require modifying the teacher's training process. Instead, it simply requires collecting data from more tasks with the pretrained universal teacher, which introduces little computational overhead. For clarity, we use "training robots" to denote the robots on which we train the teacher policy, and "PD robots" to denote the robots on which we collect expert trajectories for PD. Advanced methods like curriculum learning [Dennis et al., 2020, Narvekar et al., 2020, Wan et al., 2023] could be utilized to get a more robust task distribution to further mitigate generalization gap, which is an interesting direction for future work.

Regularization in Task Space Regularization is a common approach to reduce overfitting and improve generalization [Cobbe et al., 2019]. We consider it to be especially important for HyperDistill, as HN may be more prone to overfitting than other architectures. In prior work, morphology context is usually just used as an additional policy input [Gupta et al., 2022], while in HyperDistill, it is used as HN input to generate the whole control policy. As we only have a few hundred

different robots for PD, which is much less than the number of parameters in the HN, there is a high chance of overfitting. To reduce overfitting, we apply dropout to the context embedding e_i and e_m. This can be seen as regularization in task space, which encourages the HN to learn an ensemble of different MLP policies that can all work well on the same robot. It can also be seen as domain randomization [Tobin et al., 2017] by making the generated policy more robust to changes in morphology context. We leave it for future work to investigate other regularization methods like weight decay.

7.4 Experiments

Our experiments aim to answer the following questions:

1. Can HyperDistill achieve good performance on both training and unseen test robots? How does it compare to other methods w.r.t. performance and efficiency at inference time? (Section 7.4.2)

2. How do different algorithmic and architecture choices in HyperDistill influence its training and generalization performance? (Section 7.4.3)

7.4.1 Experimental Setup

We experiment on the UNIMAL benchmark [Gupta et al., 2021] built upon the Mujoco simulator [Todorov et al., 2012], which includes 100 training robots and 100 test robots with diverse morphologies, while new morphologies can be easily generated via mutation operations supported by the UNIMAL design space. Following the setup of Gupta et al. [2022], we consider three different environments with increasing difficulties: (1) Flat terrain (FT): maximize locomotion distance on a flat floor; (2) Variable terrain (VT): maximize locomotion distance on a variable terrain randomly reset for each episode; and (3) Obstacle: maximize locomotion distance on a flat terrain while avoiding randomly positioned obstacles.

Teacher Policy For each environment, we train a universal TF policy on the 100 training robots as the teacher policy. We adopt ModuMorph as the TF teacher, as it achieves SOTA performance on the UNIMAL benchmark as introduced in the previous chapter. Note that although ModuMorph also adopts HNs, it uses HNs to better model the diverse behaviors across limbs, but is still a large TF-based model with high inference cost. By contrast, HyperDistill utilizes HNs to enable knowledge decoupling for efficient inference, which provides a novel perspective on the role of HNs. Furthermore, as we show in Section 7.4.2, ModuMorph can be compressed to a much smaller size without much performance loss, but only when equipped with these HN-generated layers, while a standard TF cannot.

Data Collection for Distillation To collect data for policy distillation, we first generate an augmented robot set of 1,000 PD robots by mutating the 100 training robots. See Appendix B.2.1 for how we do the mutation. Then we collect 8,000 transitions from each PD robot, forming a multi-robot dataset with 8M transitions for policy distillation.

Baselines We compare HyperDistill with the following architectures as the student policy:

1. **ModuMorph (oracle):** A ModuMorph student with the same architecture as the teacher. It serves as a performance upper bound on how well the student can perform without architecture misalignment.

2. **TF (compressed):** A standard TF, i.e., MetaMorph [Gupta et al., 2022], with a similar number of parameters as the base MLP in HyperDistill. It is used to validate that monolithic TF cannot achieve both good performance and high efficiency like HyperDistill due to lack of knowledge decoupling.

3. **ModuMorph (compressed):** A compact ModuMorph student with a similar number of parameters as the base MLP in HyperDistill. As it adopts HNs to generate some layers, we expect it to have better knowledge decoupling

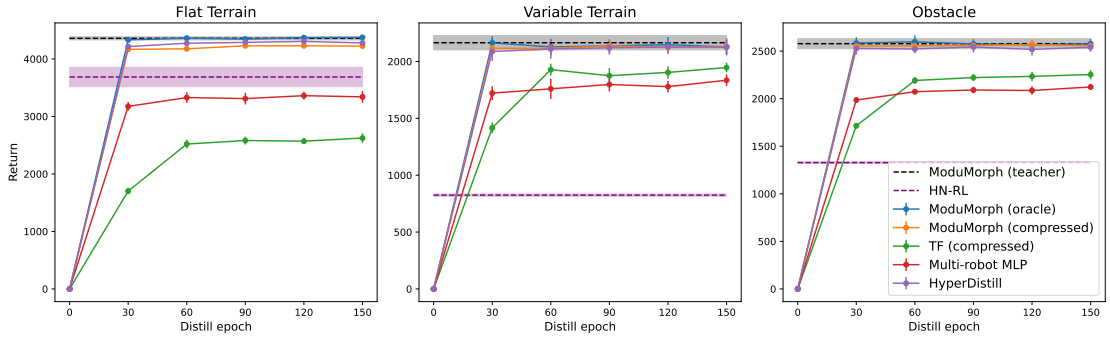

Figure 7.2: The performance of different methods on the *training* robots in each environment. Each curve represents the average return over the 100 training robots attained by each method w.r.t. to the step number of policy distillation. The error bar at each evaluation checkpoint represents the standard error across three random seeds.

ability than monolithic TF, but may still be less efficient than HyperDistill due to the attention blocks.

4. **Multi-robot MLP:** A multi-robot MLP with the same architecture as the base MLP in HyperDistill, which is used to validate the advantages of HyperDistill over vanilla MLP as discussed in Section 7.3.1.

Finally, we also compare with training an HN policy via RL (**HN-RL**) to show the importance of policy distillation.

Distillation Setup The distillation process runs for 150 epochs, with a batch size of 5120. We use the Adam [Kingma and Ba, 2014] optimizer with a learning rate of 0.0003, and clip the gradient norm to 0.5. We run three random seeds for each method in each environment, and report the average performance with standard error. For HyperDistill, we apply dropout to context embedding with $p = 0.1$. See Appendix B.2.3 for the size of each student model in each environment.

7.4.2 Main Results

Figures 7.2 and 7.3 illustrate how different methods perform on the training and test robots during the distillation process. Table 7.1 compares the model size and FLOPs of different methods at inference time.

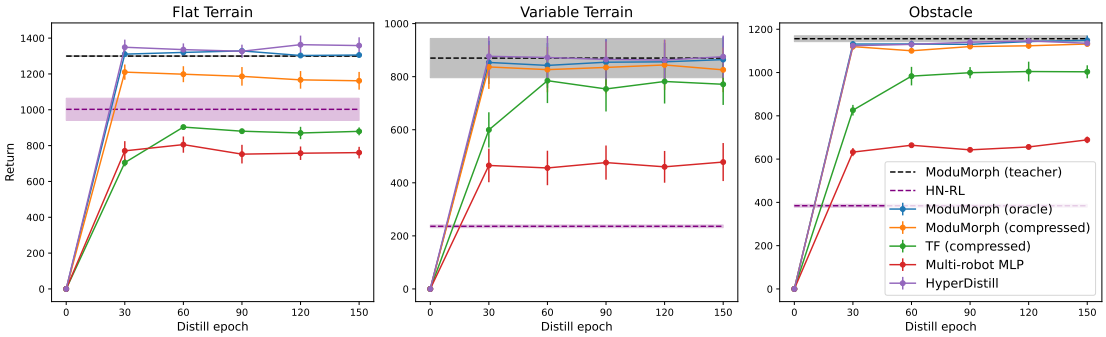

Figure 7.3: The performance of different methods on the *test* robots in each environment. Each curve represents the average return over the 100 test robots attained by each method w.r.t. to the step number of policy distillation. The error bar at each evaluation checkpoint represents the standard error across three random seeds.

Task	Method	Model size		FLOPs	
		Abs.	Rel.	Abs.	Rel.
FT	ModuMorph (oracle)	1.73 M	14.0	39.86 M	160.8
	TF (compressed)	0.14 M	1.1	3.39 M	13.7
	ModuMorph (compressed)	0.15 M	1.2	1.78 M	7.2
	Multi-robot MLP	0.23 M	1.9	0.46 M	1.9
	HyperDistill	**0.12 M**	**1**	**0.25 M**	**1**
VT	ModuMorph (oracle)	1.97 M	6.6	40.33 M	67.2
	TF (compressed)	0.31 M	1.0	5.51 M	9.2
	ModuMorph (compressed)	0.39 M	1.3	2.26 M	3.8
	Multi-robot MLP	0.41 M	1.4	0.82 M	1.4
	HyperDistill	**0.30 M**	**1**	**0.60 M**	**1**
Obstacle	ModuMorph (oracle)	2.02 M	6.7	40.43 M	67.4
	TF (compressed)	0.32 M	1.0	5.61 M	9.3
	ModuMorph (compressed)	0.44 M	1.5	2.35 M	3.9
	Multi-robot MLP	0.41 M	1.4	0.82 M	1.4
	HyperDistill	**0.30 M**	**1**	**0.60 M**	**1**

Table 7.1: Model size and FLOPs of different methods *at inference time*. Abs. denotes the absolute value, and Rel. denotes the relative value w.r.t. HyperDistill. See Appendix B.2.2 for how we compute the FLOPs, and B.3 for more analysis on the results in this table.

HyperDistill achieves performance similar to ModuMorph (oracle), matching the teacher's performance on both the training and test robots in all the three environments, while reducing model size by 6-14 times, FLOPs by 67-160 times in different environments. The inference cost of calling the HN in HyperDistill is

similar to that of calling ModuMorph, thus HN calling only introduces marginally computational overhead when deployed on a new robot.

TF (compressed) cannot match the performance of HyperDistill in all the three environments, as standard TF needs to trade off between performance and efficiency due to a lack of knowledge decoupling.

As expected, ModuMorph (compressed) consistently outperforms TF (compressed), as it generates some layers of the TF via HNs, which enables better knowledge decoupling. However, it still lags behind HyperDistill w.r.t. both generalization performance and efficiency, as the knowledge encoded in the attention blocks still cannot be decoupled. This result further indicates that, in contrast to previous work [Kurin et al., 2021, Gupta et al., 2022, Xiong et al., 2023], we may not need complicated attention modules to achieve good performance for universal morphology control. In addition, the performance gap between ModuMorph (compressed) and HyperDistill is larger in FT than in VT and Obstacle, possibly because for the latter two environments, ModuMorph has more layers in the HN-generated decoder, which makes it more similar to HyperDistill.

HyperDistill also significantly outperforms multi-robot MLP, which validates the importance of the HN. Moreover, the performance gap between multi-robot MLP and other methods is much larger on the test robots than on the training ones, which may be overfitting due to the order-invariance issue of MLP discussed in Section 7.2.

HN-RL performs poorly on both training and test robots, which reflects the optimization difficulty of combining HN and RL, and validates the importance of training via PD.

7.4.3 Ablation Studies

In this subsection, we investigate how generalization performance of the student policy is influenced by (1) The algorithmic choices in PD; and (2) Context representation learning in the HN. To save computation, we run PD for only 50 epochs as most methods can already converge within this budget, and experiment only in the FT experiment unless otherwise stated.

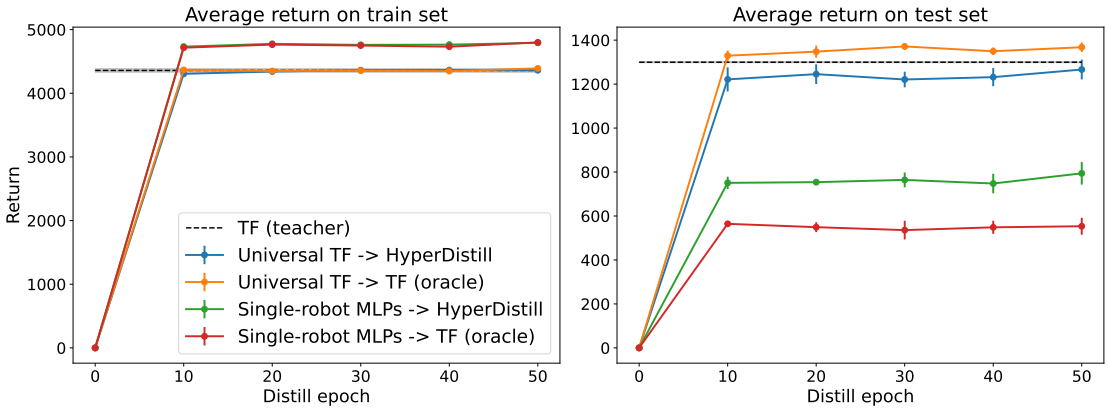

Figure 7.4: The student's learning curves under different teacher choices. In the figure legend, "X → Y" means that we distill from teacher X into student Y. Each curve represents the average return over the training (left) or test (right) robots attained by each method w.r.t. to the step number of policy distillation. The error bars represent the standard error across three random seeds.

Algorithmic Choices in Policy Distillation

The Choice of PD Teacher(s) We train (1) A universal TF teacher on all the training robots; and (2) A set of single-robot MLP teachers on each training robot. Then we collect 80,000 transitions from each training robot with the teacher(s) to get $\mathcal{B}^T$. We experiment with both HyperDistill and TF (oracle) to ensure that the PD choice applies to different student architectures. Figure 7.4 shows how the student policies perform on the training and test robots with different teacher choices. As expected, when using single-robot MLP teachers, although the student policies perform well on the training robots, they generalize much worse than the students distilled from a universal TF teacher, which validates our hypothesis in Section 7.3.2.

Student-Teacher Architecture Alignment Now we re-examine the results in Figure 7.4 from a different perspective. When using a universal TF teacher, the TF student has a more aligned architecture and achieves better generalization performance than HyperDistill. When using single-robot MLP teachers, while both students generalize much worse, HyperDistill has a more aligned architecture (as its base MLP has the same architecture as the teachers), and generalizes better than the TF student. Moreover, for the same teacher, both students achieve similar

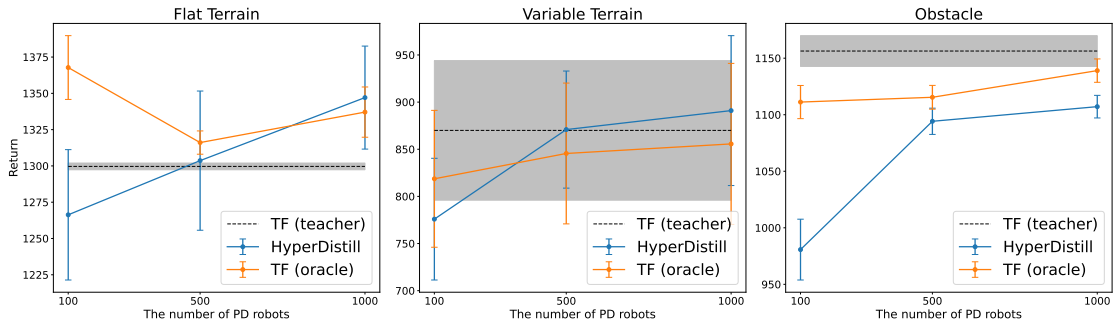

Figure 7.5: Final generalization performance of HyperDistill and TF (oracle) w.r.t. the number of PD robots in different environments. Each curve represents how the average return over the test robots attained by each method after policy distillation changes w.r.t. to the number of PD robots. The error bars represent the standard error across three random seeds.

performance on the training robots regardless of their architectures, indicating that the generalization gap is not caused by the difference in model capacity of different student models, but is more likely the consequence of architecture misalignment.

Number of Distillation Training Tasks To validate our hypothesis that collecting distillation data from more robots can improve the student's task generalization, we experiment with 100, 500, and 1,000 PD robots. For a fair comparison, the number of transitions collected from each robot decreases proportionally so that the total data size remains unchanged. As shown in Figure 7.5, HyperDistill's generalization performance increases by 6%, 15% and 13% in the three environments as the number of PD robots increases from 100 to 1,000. There is no significant improvement in the TF student's generalization performance due to a ceiling effect and its better aligned architecture with the TF teacher.

Regularization in Task Space As shown in Figure 7.6, compared to not using dropout, applying dropout to the context embedding improves generalization performance by 8.5%, while applying dropout to the base MLP does not provide significant improvement, which agrees with our intuition that regularization may be more important in task space than in state space to enable better task generalization.

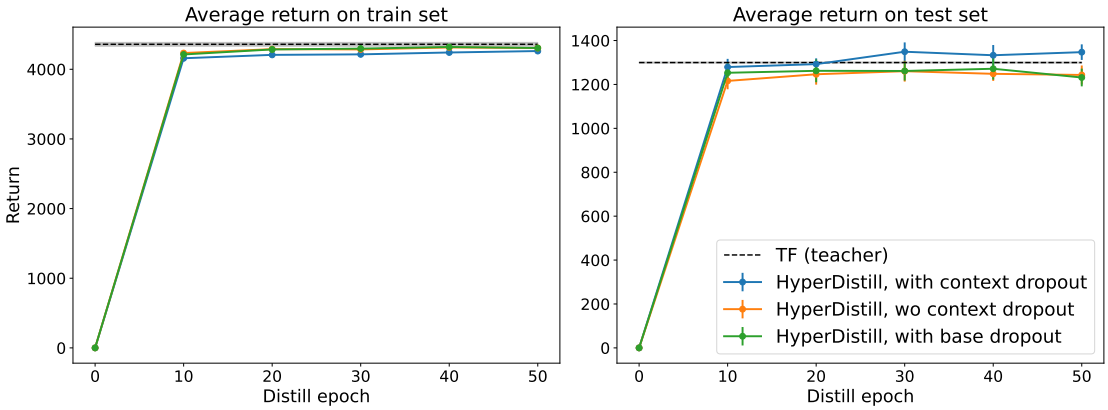

Figure 7.6: Learning curves of HyperDistill in FT with different ways of incorporating dropout regularization. The error bars represent the standard error across three random seeds.

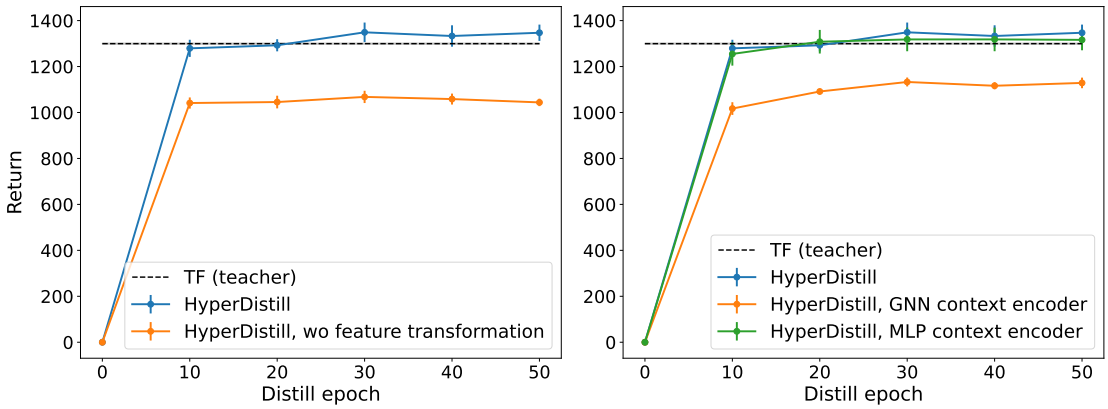

Figure 7.7: How context representation learning in HyperDistill influences generalization performance in FT. Left: context feature transformation. Right: architecture of the context encoder. The error bars represent the standard error across three random seeds.

Context Representation Learning

Context Feature Transformation As shown in Figure 7.7 (left), conducting feature transformations (see Appendix B.1) to improve feature discrimination plays an important role in improving performance.

Architecture of Context Encoder As shown in Figure 7.7 (right), for different choices of the context encoder in the HN, GNN performs the worst, possibly due to the over-smoothing issue of GNN [Chen et al., 2020] which harms discrimination of the context embedding. While the MLP context encoder does not consider node interaction, it still achieves quite good performance in FT. We thus further compare

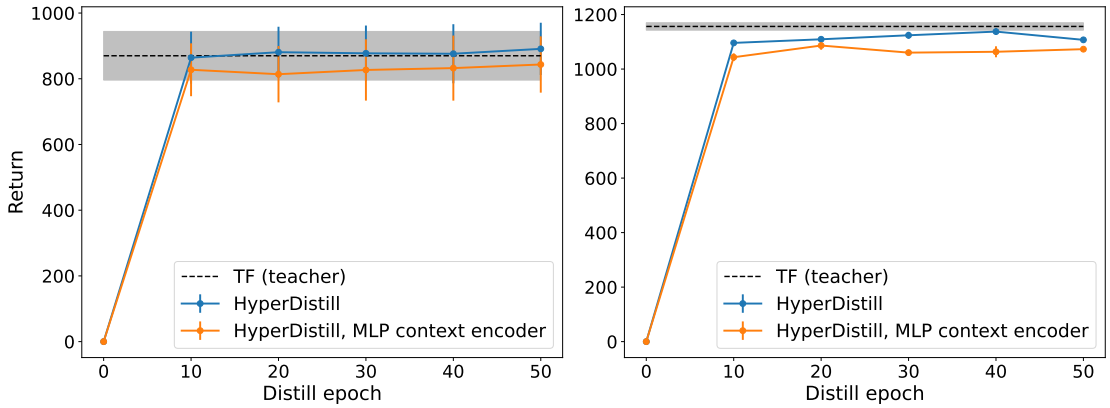

Figure 7.8: How TF and MLP context encoder influence HyperDistill's generalization performance in the VT (left) and Obstacle (right) environments. The error bars represent the standard error across three random seeds.

MLP and TF context encoder in the other two more challenging environments (Figure 7.8), and find that the TF context encoder performs better, which validates the benefits of modeling node interaction for context representation learning.

7.5 Discussion

In this section, we re-examine our knowledge decoupling hypothesis to see if it is supported by experimental results. First, HyperDistill achieves similar performance as TF but runs much more efficiently at inference time, which validates the efficiency benefits provided by knowledge decoupling. Second, as expected, due to the lack of knowledge decoupling, when we compress a universal TF policy, the compressed TF does not have sufficient model capacity to accommodate both inter-task and intra-task knowledge, as evidenced by its worse performance in our experiments. The knowledge decoupling hypothesis further suggests that if we compress a universal TF into a single-robot TF, it should outperform a compressed universal TF on that specific robot, as we relieve it of the burden of distilling inter-task knowledge. To validate this idea, we conduct a proof-of-concept experiment in the Obstacle environment. We randomly sample 10 training robots and distill the universal TF teacher into a single-robot TF with the same architecture as TF (compressed) for each robot. As expected, the compressed single-robot TFs achieve an average return

of 2345, outperforming the compressed universal TF with an average return of 2115, but at the cost of losing the generalization ability to other robots.

In summary, vanilla TF needs to trade off between performance and efficiency at inference time, while HyperDistill can enjoy both thanks to knowledge decoupling. It resembles related ideas explored in other fields, such as mixtures-of-experts (MoE) [Riquelme et al., 2021, Shen et al., 2023] that learn a large model with high capacity while sparsely activating sub-modules for different tasks to improve inference efficiency. There is also evidence from neuroscience that although the human brain contains billions of neurons, it is still power-efficient because it activates only a small fraction of neurons simultaneously to solve a given task [Barth and Poulet, 2012]. Consequently, we believe that our knowledge decoupling hypothesis could serve as a general principle to improve inference efficiency in other domains as well.

7.6 Related Work

In addition to related work on universal morphology control that has been reviewed in Section 4.1.2, in this section we further review related work on (1) HNs for inference acceleration; and (2) Policy distillation.

Hypernetworks for Efficient Inference In the context of multi-task RL, HNs provide a powerful way to model the complex dependency between task context and the optimal control policy for the task [Galanti and Wolf, 2020, Sarafian et al., 2021], and has been widely adopted in multi-task RL and meta-RL [Yu et al., 2019, Peng et al., 2021, Beck et al., 2023a,b, Rezaei-Shoshtari et al., 2023]. While these works mainly utilize the expressive power of HNs to improve task performance, we focus more on utilizing HN's knowledge decoupling ability to improve inference efficiency. This idea has been explored to generate compact base networks for few-shot image classification [Zhmoginov et al., 2022], tabular data prediction [Bonet et al., 2024, Mueller et al., 2025] and federated learning [Shamsian et al., 2021]. In general, utilizing HN for inference acceleration still does not receive much attention from the

research community yet, and we are the first to make this idea work for multi-task robotic control to the best of our knowledge.

Policy Distillation Distillation [Buciluundefined et al., 2006, Hinton et al., 2015] transfers knowledge from a teacher model or multiple teacher models to a student model. In the context of RL, it has been used to compress the policy network [Rusu et al., 2015], accelerate learning on a new task [Parisotto et al., 2015], and facilitate policy learning [Lee et al., 2020, Chen et al., 2022, Wan et al., 2023, Xu et al., 2025a]. However, less attention is paid to measuring and minimizing the generalization gap between the teacher and the student. Igl et al. [2021] distill a teacher policy into a student with identical architecture, which is used as an intermediate step to improve task generalization of a policy trained via RL. Chen et al. [2022] and Wan et al. [2023] investigate generalization of a distilled policy to unseen objects in robotic manipulation. However, as the teacher has access to privileged information while the student does not under their setting, there is always a performance gap between the teacher and the student, and this gap is larger on the test tasks than that on the training tasks, which indicates a further generalization gap. Most similar to our work is Furuta et al. [2023], which distills multiple single-robot MLP teachers into a TF student for universal morphology control to avoid the difficulty in directly training a universal TF policy via RL. But they only show the advantage of using TF as the student architecture, but do not compare with a universal TF teacher to measure generalization gap. Furthermore, the TF student policy they learn still has efficiency issue during deployment. To the best of our knowledge, we are the first to systematically investigate how to mitigate generalization gap in policy distillation via proper algorithmic choices.

7.7 Conclusion and Future Work

In this chapter, we achieved both good performance and high efficiency at inference time for universal morphology control via knowledge decoupling, which is realized by utilizing the hierarchical architecture of HNs. While training an HN via RL is

hard, we took a policy distillation approach, and systematically investigated how some key algorithmic choices influence task generalization of the HN student. Our proposed method, HyperDistill, matches the performance of the SOTA universal TF teacher on both training and test robots, while significantly reducing memory and computational cost, which supports our hypothesis that decoupling inter-task and intra-task knowledge can improve inference efficiency.

Given the wide application of TF in foundation models and their extremely high inference cost [OpenAI et al., 2024, Brohan et al., 2023, Zitkovich et al., 2023, O'Neill et al., 2024], knowledge decoupling could serve as a general principle to reduce the inference cost of TF in other domains as well. However, a potential limitation when extending our method to other domains is the additional cost introduced by the HN. This is not an issue for universal morphology control, as the morphology context remains unchanged for each robot. So we only need to call the HN once before deployment while the whole HN can be discarded afterwards. However, if task context changes over time, such as controlling a robot to solve different tasks by following language instructions, the HN can not be discarded and needs to be called whenever a new instruction is given. This requires additional space to save the HN parameters, and the HN will be called more frequently. In the next chapter, we will investigate how to extend the knowledge decoupling principle to the more challenging setting of language-conditioned cross-skill control with visual inputs.

Part III

Generalization across Skills

8

HyperVLA: Efficient Inference in Vision-Language-Action Models via Hypernetworks

Contents

In this chapter and the next, we will investigate the second problem setting of cross-skill control, i.e., learning a language-conditioned manipulation policy via imitation learning to achieve good performance on both tasks that have been seen during pretraining, and unseen tasks in a zero-shot fashion or via efficient fine-tuning.

In this chapter, we extend the knowledge decoupling principle as proposed in the previous chapter to the setting of learning VLAs for cross-skill control, as inference efficiency is a key bottleneck of existing VLAs due to their extremely huge size (Section 8.1). Unlike existing monolithic VLAs that activate the whole model during both training and inference (Section 8.2.1), we propose a novel HN-based VLA that activates only a small task-specific policy during inference, while retaining the high model capacity needed to accommodate diverse multi-task behaviors during pretraining (Section 8.2.2). Successfully training such an HN-based VLA is nontrivial, so we further propose several key algorithm design features that improve its performance, including properly utilizing the prior knowledge from existing vision foundation models, HN normalization and action generation strategy (Section 8.2.3). Compared to existing monolithic VLAs, HyperVLA achieves a similar or even higher success rate during evaluation, while significantly reducing inference costs (Section 8.3).

8.1 Introduction

As introduced in Chapter 5, Vision-Language-Action (VLA) models have achieved promising progress towards learning a generalist robot by following this general recipe: (1) Use existing VLM models with strong generalization performance as the backbone; (2) Post-train it on large-scale robotic datasets to achieve strong zero-shot generalization performance on robotic tasks; and (3) Further fine-tune on downstream tasks with a small number of task-specific data if needed to achieve better performance.

However, one key drawback of VLAs is their extremely high inference cost, e.g., OpenVLA [Kim et al., 2024], a SOTA VLA model, has more than 7B parameters and can only infer at 6 Hz even when equipped with an NVIDIA 4090 GPU. Such a high inference cost not only consumes significant memory, computation, and battery, but also makes it hard to solve dexterous tasks that require high-frequency manipulation.

By contrast, conventional methods for robotic learning from before the era of foundation models typically learn compact models that are much smaller than

VLAs. Although such models cannot generalize across a diverse set of tasks, they can perform well on the specific task they are trained on given sufficient training data [Yu et al., 2020b, Kim et al., 2024]. In other words, the minimal model required to solve a specific task can be much smaller than a VLA with millions or billions of parameters. So a natural question arises: *Can we learn a generalist policy that combines the best of both worlds: the strong generalization ability of VLAs, and the efficient inference of single-task policies?* To achieve this, we need to learn a generalist policy with high model capacity to accommodate the diverse behaviors in multi-task data at training time, but only activate a small part of it at test time to keep inference efficient.

In this chapter, we realize this goal via HNs by extending the knowledge decoupling principle as proposed in the previous chapter to the more challenging setting of VLA for skill generalization. The hierarchical architecture of HN provides a natural way to decouple the skills required to solve different tasks, so that we can learn an HN with high model capacity at training time, but only activate a compact base network generated by the HN to solve a specific task efficiently at test time.

Specifically, we learn an HN-based VLA that generates policy parameters conditioned on task context, which includes both the language instruction and the initial image of the episode (Figure 8.1). We find it helpful to include the initial image in the task context, as it contains context information about the environment that the robot is working in, which helps the policy generalize better across different scenarios. At training time, we train an HN with high model capacity to capture the complex mapping from the task context to the corresponding policy parameters π_θ. At inference time, the large HN is called at a low frequency only at the beginning of each episode, while the compact generated policy is called at every timestep to process image observations and output action predictions, which significantly reduces inference cost compared to existing monolithic VLAs that activate the entire model at every timestep during inference.

However, HNs are known to be hard to optimize [Chang et al., 2020, Beck et al., 2023a, Xiong et al., 2024b], and training an HN with millions of parameters on

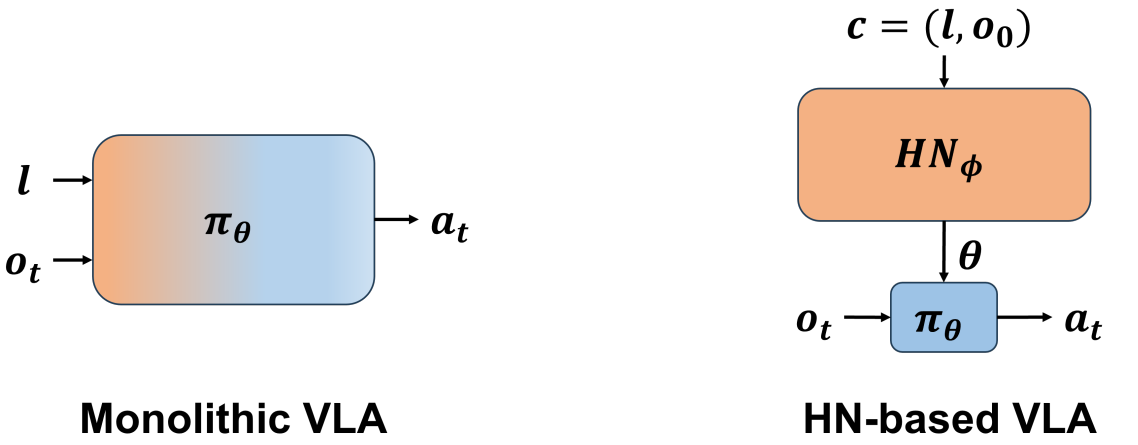

Figure 8.1: Comparison between the high-level framework of monolithic VLA (left) and HN-based VLA (right). We use orange to represent parameters activated during training, and blue to represent parameters activated at every timestep during inference. The monolithic VLA activates the whole model during both training and inference and is thus colored both orange and blue. By contrast, an HN-based VLA calls the HN at a low frequency only when the instruction changes at test time, and calls a compact base network at every timestep for action prediction.

large-scale robotic data further aggravates this issue. We thus introduce several key algorithm design features that help stabilize and improve HN learning:

1. **Vision backbone**. While in principle we can generate the whole base policy with HN, empirically we find it important to use existing vision foundation models as the backbone to improve generalization, as training an HN from scratch on robotic data alone is prone to overfitting due to the relatively small data size of existing robotic datasets.

2. **HN normalization**. Successful training of neural networks heavily depends on the proper setup of many optimization choices, which are mainly tailored for training monolithic models and may not work well with the different optimization dynamics of HNs [Chang et al., 2020, Beck et al., 2023a, Auddy et al., 2024]. We thus investigate how parameter updates in HN training differ from standard training, and propose a simple yet effective solution by normalizing the context embedding in HNs, such that the base network parameters can be updated with similar dynamics as directly training the base network.

3. **Action generation strategy**. Unlike most existing VLAs that predict actions via autoregression [Brohan et al., 2023, Zitkovich et al., 2023] or diffusion [Chi et al., 2023, Ghosh et al., 2024], we find that simply learning a deterministic linear action head with MSE loss performs better when training an HN-based VLA and further accelerates inference.

We name our method as *HyperVLA*, an HN-based VLA learned with the above algorithm design features. We train HyperVLA on the Open X-Embodiment (OXE) dataset [O'Neill et al., 2024], and evaluate on the SIMPLER [Li et al., 2025a] and LIBERO [Liu et al., 2023a] benchmark. Compared to existing monolithic VLAs, HyperVLA achieves a similar or higher success rate during evaluation and fine-tuning, while significantly improving inference efficiency by only activating a compact HN-generated policy at every timestep during inference, which validates the effectiveness of utilizing HNs for inference acceleration of VLAs. Notably, compared to OpenVLA [Kim et al., 2024], a SOTA VLA with the best performance among the baselines, HyperVLA reduces the number of activated parameters at test time by 90×, and accelerates inference by 120×.

8.2 HyperVLA

This section introduces the motivation, architecture, and algorithm design of HyperVLA. In Section 8.2.1, we analyze why existing monolithic VLAs have high inference costs and how an HN-based VLA can tackle this challenge via knowledge decoupling. Then in Section 8.2.2 we introduce the architecture of HyperVLA. Finally, in Section 8.2.3, we propose several key algorithm design features to stabilize HyperVLA training and improve its performance.

8.2.1 From Monolithic to HN-Based VLA

Existing VLAs usually have millions or billions of parameters. While such a high model capacity is necessary to accommodate diverse behaviors in multi-task data at training time, it introduces significant computational redundancy at test time

as the minimal model size required to solve a specific task is often much smaller than the size of a large VLA [Yu et al., 2020b, Kim et al., 2024].

Consequently, there is a significant space for inference acceleration if we can only activate a small part of a huge VLA that is sufficient to solve the task at hand. However, existing VLAs with monolithic architectures need to activate the whole model during both training and inference, thus cannot decouple the different skills required for solving different tasks from each other in the parameter space [Xiong et al., 2024b].

In this work, we tackle this challenge by learning an HN-based VLA, as the hierarchical architecture of HN provide a natural way to decouple inter-task and intra-task knowledge as introduced in the previous chapter. Intuitively, the HN part encodes inter-task knowledge about how to map from different task context to the corresponding policy parameters, while the generated base network encodes intra-task knowledge about how to solve a specific task. At training time, the whole HN is activated to ensure sufficient model capacity to accommodate the diverse behaviors in large-scale multi-task data. However, at test time, we only need to call the HN once at the beginning of each episode to generate a compact task-specific policy that is used for the remainder of the episode to solve the task.

8.2.2 The Architecture of HyperVLA

The Base Policy

We formulate the base policy as a Vision Transformer (ViT) [Dosovitskiy et al., 2021], which takes the image observation o_t as input to predict the robot's action a_t. Unlike existing VLAs, we do not feed the language instruction l into the base policy as it is already indirectly conditioned on the instruction via the HN that generated its parameters. The base policy consists of the following blocks in sequence (we omit the time index t for simplicity):

1. An image encoder, formulated as a ViT, encodes the image observation o into a sequence of token embeddings $\{e_i^{\text{image}}\}$ for the image patches;

2. A linear projection layer maps $\{e_i^{\text{image}}\}$ into a lower dimension for more efficient inference, represented as $\{e_i^{\text{proj}}\}$;

3. A policy head θ^{policy}, formulated as a small TF, takes $\{e_i^{\text{proj}}\}$ and an action token e^{act} as inputs, and updates their token embeddings;

4. An action head takes the updated action token embedding e^{act} as input to predict the robot's action $\hat{a}$.

The Hypernetwork

The HN consists of a context encoder parameterized as a TF with high model capacity, and linear output heads to generate base policy parameters. The context encoder takes in three inputs:

1. Pretrained token embeddings of the language instruction generated by a frozen T5 encoder [Raffel et al., 2020].

2. The class token generated by feeding the initial image of the episode into a frozen DINOv2 encoder [Oquab et al., 2024]. We find it helpful to condition the HN on the initial image, as the robot may see the same instruction in different scenarios, and a compact base policy may not have enough model capacity to generalize across scenarios with diverse visual appearance. By conditioning policy generation further on the initial image, the HN with high model capacity takes the responsibility of generalizing across both instructions and scenarios, while the generated base policy only needs to solve a specific task in a specific scenario. Moreover, we only use the class token outputted by DINOv2 as HN input and discard the image patch tokens to avoid overfitting in the HN.

3. A learnable task context token that integrates task context information.

The TF context encoder updates the embeddings of these tokens via self-attention, and the updated embedding of the context token is fed into the HN output heads to generate base policy parameters.

8.2.3 Algorithm Design Features

HNs are known to be unstable and hard to optimize [Chang et al., 2020, Beck et al., 2023a, Xiong et al., 2024b], and scaling them up to millions of parameters further aggravates this issue. In this subsection, we introduce several key algorithm design features that help stabilize HyperVLA training and improve its performance.

Vision Backbone

While in principle we can generate the whole base policy by HN, empirically we find that training a large HN from scratch on robotic data alone is prone to overfitting due to the relatively small data size of existing robotic datasets. Instead, we use existing vision foundation models as the image encoder in the base network to improve generalization. Our method is agnostic to the choice of the vision encoder, and empirically we find that DINOv2 [Oquab et al., 2024] achieves the best performance and thus adopt it in HyperVLA.

Similar to previous work [Kim et al., 2024], we find it helpful to fine-tune this vision backbone instead of keeping it frozen when training on robotic data. Furthermore, we find it important to use a smaller learning rate for fine-tuning the vision backbone than for HN training, because DINOv2 is already well pretrained and only needs to be fine-tuned at a conservative rate to better align with robotic data.

Context Embedding Normalization

Successful training of neural networks heavily depends on the proper setup of many optimization choices, such as network initialization, normalization layers and gradient transformations. However, these methods are mainly tailored for training monolithic models, and may need to be re-designed to fit the different optimization dynamics of HNs. For example, Chang et al. [2020] and Beck et al. [2023a] investigate how to initialize HNs properly so that the base network is initialized in the same way as commonly used initializers, an approach we adopt as well.

However, a proper initialization can only provide a good starting point for HN training, but has no direct effect on the parameter update process during training.

So in this work, we further investigate how parameter updates in HN training differ from standard neural network training, and propose a simple yet effective solution by normalizing the context embedding in HNs, such that the base network parameters can be updated with similar dynamics as directly training the base network.

For simplicity, we use an SGD optimizer in our derivation below. In standard neural network training, each parameter θ_i is updated by $\Delta\theta_i = -\alpha \cdot \frac{\partial L}{\partial \theta_i}$, where L is the loss function and α is the learning rate.

Now we generate θ with an HN. Let us denote the output head of the HN as ϕ, and the context embedding input to the output head as e, then we have $\theta_i = \sum_j e_j \phi_{ij}$. We omit the bias term as it has the same gradient as the base parameter. According to the chain rule, we have $\frac{\partial L}{\partial \phi_{ij}} = \frac{\partial L}{\partial \theta_i}\frac{\partial \theta_i}{\partial \phi_{ij}} = \frac{\partial L}{\partial \theta_i} e_j$, and the parameter update in the HN is $\Delta\phi_{ij} = -\alpha \cdot \frac{\partial L}{\partial \theta_i} e_j$. Then the parameter update in the base network is:

$$\begin{aligned}
\Delta\theta_i &= \sum_j (e_j + \Delta e_j)(\phi_{ij} + \Delta\phi_{ij}) - \sum_j e_j \phi_{ij}, && (8.1)\\
&= \sum_j e_j \Delta\phi_{ij} + \Delta e_j \phi_{ij} + \Delta e_j \Delta\phi_{ij}, && (8.2)\\
&\approx \sum_j e_j \Delta\phi_{ij} + \Delta e_j \phi_{ij}, \quad \text{(Omit the multiplication of two delta terms)} && (8.3)\\
&= -\alpha \cdot \frac{\partial L}{\partial \theta_i} \sum_j e_j^2 + \sum_j \Delta e_j \phi_{ij}. && (8.4)
\end{aligned}$$

If we assume that both ϕ_{ij} and Δe_j are i.i.d. and follow a Gaussian distribution with zero mean, then $\mathbb{E}\left[\sum_j \Delta e_j \phi_{ij}\right] = 0$. Accordingly, we have $\Delta\theta_i \approx -\alpha \cdot \left(\sum_j e_j^2\right) \cdot \frac{\partial L}{\partial \theta_i}$, which indicates that when learning with HNs, the update on the base network parameters is scaled by a factor of $\sum_j e_j^2$ compared to directly optimizing the base network.

In HyperVLA, as the context embedding e is the output of a TF context encoder with layer normalization as its final layer, we have $\mathbb{E}\left[e_j^2\right] = 1$ and $\mathbb{E}\left[\sum_j e_j^2\right] = d_e$, where d_e is the dimension of e. Consequently, to keep the scale of parameter update in the base network unchanged, we can simply divide the context embedding by $\sqrt{d_e}$ before feeding it into the output head so that $\mathbb{E}\left[\sum_j e_j^2\right] = 1$ after normalization.

The derivation is different for more complex optimizers like Adam, which makes it much harder to theoretically keep the update scale unchanged like with the SGD optimizer. However, empirically we find that the same normalization operation

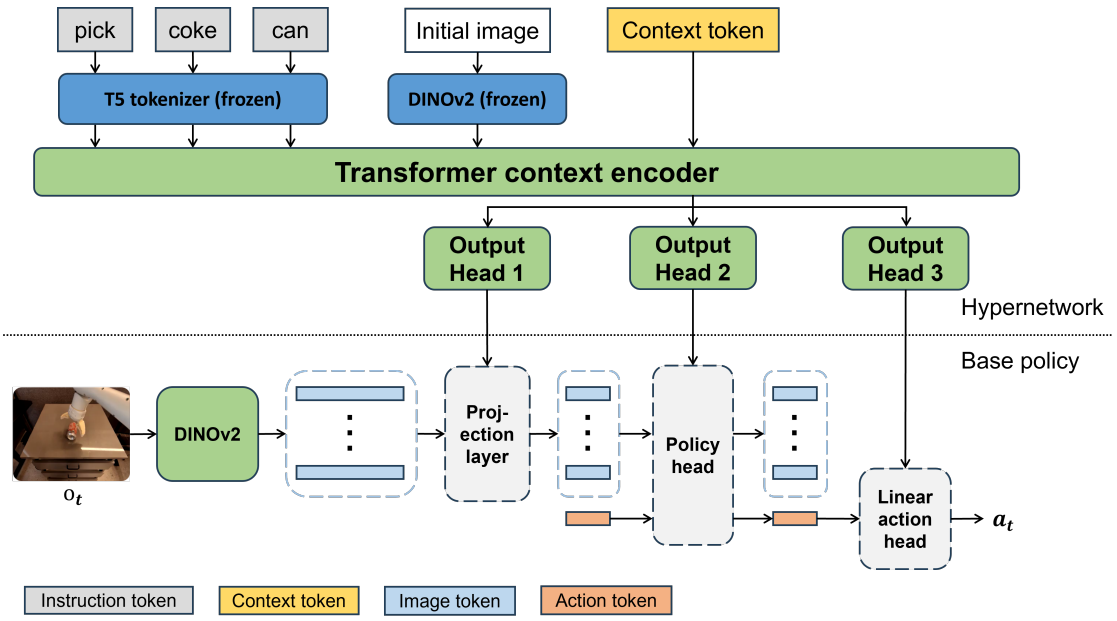

Figure 8.2: The framework of HyperVLA. The trainable parameters are marked as green blocks, while the HN-generated parameters are marked as light grey blocks with dashed edges.

of dividing the context embedding by $\sqrt{d_e}$ before feeding it into the HN output head still works well in practice.

Action Generation Strategy

As introduced in Section 5.2.1, existing VLAs usually predict discretized actions autoregressively [Brohan et al., 2023, Zitkovich et al., 2023, Kim et al., 2024], which requires multiple runs of a huge TF model to generate different action dimensions sequentially, or learn a diffusion action head [Chi et al., 2023] that must be iteratively called to denoise actions [Ghosh et al., 2024], both of which are time-consuming at training and test time. Instead, we find that training a simple linear action head with an MSE loss outperforms these more complicated action generation strategies in HyperVLA, while further reducing training and inference cost. This also agrees with the findings from some recent work [Figure, 2024, Kim et al., 2025].

The overall framework of HyperVLA, which combines all these algorithm design features, is shown in Figure 8.2.

8.3 Experiments

Our experiments aim to answer the following questions:

1. Can HyperVLA achieve similar zero-shot generalization performance on both seen and unseen tasks as existing monolithic VLAs? (Section 8.3.2)

2. Can HyperVLA adapt to new tasks by fine-tuning only on a small amount of demonstrations, especially for long-horizon tasks? (Section 8.3.3)

3. How does HyperVLA compare to monolithic VLAs w.r.t. inference efficiency? (Section 8.3.4)

4. How do different algorithm designs in HyperVLA influence its performance? (Section 8.3.5)

8.3.1 Experimental Setup

Baselines We compare HyperVLA with the following monolithic VLAs as baselines: (1) RT-1-X [O'Neill et al., 2024]: uses EfficientNet [Tan and Le, 2019] as the vision backbone, and conditions on the language instruction via FiLM [Perez et al., 2018]. Each action dimension is discretized and predicted autoregressively. It has roughly 35M parameters in total. (2) Octo [Ghosh et al., 2024]: uses T5 [Raffel et al., 2020] as the language backbone and learns the visual encoder on robotic data alone. It predicts actions via policy diffusion [Chi et al., 2023]. It has roughly 200M parameters in total. (3) OpenVLA [Kim et al., 2024]: uses SigLIP [Zhai et al., 2023] and DINOv2 [Oquab et al., 2024] as the vision backbones, and Llama 2 [Touvron et al., 2023] as the language backbone. Llama 2 is further fine-tuned on robotic data to predict action tokens autoregressively like RT-1-X. It has about 7.6B parameters in total. All the baselines are trained on the Open X-Embodiment (OXE) dataset [O'Neill et al., 2024], which contains demonstrations collected from different robot embodiments.

We choose these models as the baselines as they are all trained on the OXE dataset and have the same input and output space for the model, which makes it

easier to control variates and focus only on the influence of changing the model architecture from monolithic to HN-based. Many later VLAs are trained on larger and different datasets, which makes it hard to tell apart whether the performance difference is caused by the data or the introduction of HN. Nevertheless, future work can easily apply our HN-based architecture to other VLAs by adopting their original training recipe for a fair comparison.

Hyperparameters of HyperVLA In the base network, we use DINOv2 [Oquab et al., 2024] as the image encoder. The policy head is a TF with 4 layers, each with 4 attention heads. Its token embedding dimension and hidden layer dimension are set to 64 and 128 respectively. The base network takes only the current image observation as input, and predicts an action chunk of 4 steps. During evaluation, we further apply action ensemble to improve prediction accuracy.

For the HN, we adopt T5 [Raffel et al., 2020] as the instruction encoder and DINOv2 as the initial image encoder, and freeze them during training. We learn a TF context encoder with 6 layers, with an embedding dimension of 128, MLP hidden dimension of 512 and 4 attention heads for each layer. The output heads are linear layers that map the context embedding to the parameters of the base network.

Training For a fair comparison with the baselines, we also train HyperVLA on the OXE dataset. We train HyperVLA for 100k steps with a batch size 256. To stabilize the performance of the learned model, we apply exponential moving average to the model parameters with a smoothing factor of 0.999, and use the smoothed parameters instead of the latest parameters in the model checkpoints for evaluation. We optimize with AdamW [Loshchilov and Hutter, 2019], with a weight decay coefficient of 0.05 on HN output heads. Following the setup in Octo [Ghosh et al., 2024], we set the peak learning rate as 3e-4, and apply learning rate warmup for 2k steps, then anneal it with an inverse square root schedule. The learning rate for the DINOv2 image encoder in the base network shares the same schedule, but uses a much lower peak value of 3e-5. To enable better generalization, we augment both the language instruction by rephrasing, and the image observation

by image augmentation as done in Octo. Other training hyperparameters follow the same setup as in Octo.

8.3.2 Zero-Shot Generalization Results

Robot	Task suite	# Instruction	Seen during training	# Demonstration in OXE	# Eval
Google Robot	pick	13	6 seen, 7 unseen	~600 per seen object	150
	move	30	Yes	40 to 100 per instruction	180
	close drawer	3	Yes	> 2000 per instruction	180
WidowX	spoon on towel	1	Yes	1	60
	carrot on plate	1	Yes	332	60
	stack cube	1	No	0	60
	eggplant in basket	1	No	0	60

Table 8.1: Summary of evaluation tasks in SIMPLER.

To evaluate zero-shot generalization performance, We evaluate on the SIMPLER benchmark [Li et al., 2025a], which reproduces some tasks from the OXE dataset in simulation and is specifically design to align with real-world evaluation results, so that different VLAs can be compared in a reproducible way. SIMPLER includes two commonly used robot arms, Google Robot and WidowX, and defines a set of different tasks for each robot. For Google Robot, we evaluate on: (1) pick: pick up an object from the table; (2) move: move a source target object to a goal object, with another distractor object on the table; and (3) close drawer: close top/middle/bottom drawer of the table. For WidowX, we evaluate on: (1) spoon on towel: put the spoon on the towel; (2) carrot on plate: put the carrot in the plate; (3) stack cube: stack the green cube on the yellow cube; and (4) eggplant in basket: put the eggplant into the yellow basket. As shown in Table 8.1, SIMPLER evaluates on both tasks that have been seen during training with different demonstration number ranging from 1 to more than 2,000, and unseen tasks with new instructions. For seen tasks, generalization across parametric variations is evaluated, such as object layout, position and orientation. For unseen tasks, generalization across instructions is evaluated. For each task suite in SIMPLER, we evaluate each method on multiple episodes with the same sequence of randomly sampled instructions and initial states to fairly evaluate the generalization performance of different methods.

Table 8.2 and Table 8.3 compare the evaluation success rates of different methods on Google Robot tasks and WidowX tasks respectively. Among the baselines, OpenVLA performs the best overall, as it builds upon strong language and vision foundation models which facilitate generalization, but at the expense of a higher inference cost. Our method achieves similar performance as OpenVLA on most task sets, while significantly outperforming all baselines on the picking task set.

Task set	pick	move	close drawer	Average
ID or OOD	ID + OOD	ID	ID	
RT-1-X	29	46	**65**	47
Octo	7	26	31	21
OpenVLA	10	**72**	54	45
HyperVLA (Ours)	**58 ± 3**	**73 ± 1**	**58 ± 7**	**63 ± 3**

Table 8.2: Evaluation success rates of different methods on Google Robot tasks of SIMPLER. Each column represents a task set which contains multiple different instructions, and we report the average success rate over the whole task set. The "ID or OOD" row represents if the task set is in-distribution (ID) or out-of-distribution (OOD). For our method, we report the performance mean and standard error averaged over 5 random seeds. For the baselines, we cannot report the confidence interval, as only a single model checkpoint is publicly available for each baseline method.

Task set	spoon on towel	carrot on plate	stack cube	eggplant in basket	Average
ID or OOD	ID	ID	OOD	OOD	
RT-1-X	10	12	0	0	6
Octo	5	1	0	45	13
OpenVLA	25	**18**	**34**	**65**	**36**
HyperVLA (Ours)	**48 ± 3**	**21 ± 5**	**39 ± 8**	**52 ± 13**	**40 ± 5**

Table 8.3: Evaluation success rates of different methods on WidowX tasks of SIMPLER.

8.3.3 Few-Shot Adaptation Results

To evaluate few-shot adaptation performance of HyperVLA, we evaluate on the LIBERO benchmark [Liu et al., 2023a], which is commonly used to evaluate data-efficient fine-tuning of VLAs. Following previous work, we evaluate on four task suites from LIBERO, each containing 10 tasks (instructions) with 50 demonstrations for each task:

1. **LIBERO-Spatial** evaluates generalization to different layouts of the same set of objects.

2. **LIBERO-Object** evaluates generalization to different object types with the same scene layout.

3. **LIBERO-Goal** evaluates generalization to different goals (instructions) with the same set of objects and layout.

4. **LIBERO-Long** evaluates performance on long-horizon tasks with diverse objects, layouts, and tasks.

For a fair comparison, we preprocess the demonstrations in the same way as OpenVLA. We fine-tune HyperVLA on LIBERO-Spatial, LIBERO-Object, LIBERO-Goal for 10k steps, and LIBERO-Long for 60k steps as long-horizon tasks are harder to solve. All the other hyperparameters are set in the same way as for pretraining.

As shown in Table 8.4, our method significantly outperforms Octo and OpenVLA on all the task suites after fine-tuning, which validates the effectiveness of our method for few-shot adaptation to unseen tasks. The significant advantage of our method on LIBERO-Long further validates that our method can also solve complicated long-horizon tasks by only activating a compact base policy at inference time.

	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-Long	Average
Octo	79	86	85	51	75
OpenVLA	85	88	79	54	77
HyperVLA	**95**	**94**	**92**	**74**	**89**

Table 8.4: Evaluation success rate of different methods on LIBERO after fine-tuning. We report the average success rate over all the 10 tasks of each task suite. We evaluate on each task for 50 episodes. The results of the baselines are taken from Kim et al. [2024].

8.3.4 Inference Efficiency

Table 8.5 compares the number of parameters activated during training and inference, the inference speed and FLOPs of different methods. For the number of activated parameters at inference time, we exclude the parameters that are only activated

once at the beginning of each episode, as their computational costs are negligible compared to the overall inference cost in the whole episode, which include the instruction encoder in all the methods and the HN in our method. Moreover, the inference cost of HN calling in HyperVLA is at a similar level as the baseline models.

	# Params activated per training step	# Params activated per inference step	Time per inference step (ms)	FLOPs
RT-1-X	35M	35M	88	-
Octo	200M	100M	96	5.6×10^{10}
OpenVLA	7.6B	7.6B	482	4.0×10^{12}
HyperVLA	86M (shared) + 216M (HN)	86M (shared) + 0.1M (generated)	**4**	$\mathbf{4.7 \times 10^{10}}$

Table 8.5: Number of parameters activated per step during training and test, inference speed and FLOPs of different methods. Time per inference step is measured by running each model on an NVIDIA L4 GPU. We were unable to measure the FLOPs of RT-1-X as its model checkpoint is wrapped up.

At training time, HyperVLA activates both a shared DINOv2 image encoder with 86M parameters and an HN with 216M parameters (100M for the frozen T5 encoder + 86M for the frozen DINOv2 encoder + 30M for the learned context encoder). At test time, it only activates the shared DINOv2 backbone and a generated base network with 0.1M parameters for each inference step, which leads to a significant acceleration. Compared to OpenVLA, the baseline with the best performance, HyperVLA reduces the model size at inference time by 90 times and accelerates the inference speed by 120 times. Although RT-1-X and Octo have a similar or smaller number of activated parameters during inference than HyperVLA, their inference is still much slower, as they use either autoregression or a diffusion policy to predict the action, both of which require more iterations over the model parameters than the simple linear action head used in HyperVLA.

Based on the above results, we conclude that HyperVLA not only achieves similar or better performance compared to the baselines, but also significantly reduces inference costs. Moreover, while our primary goal is to improve the inference efficiency of VLAs, our method also significantly reduces computational costs during training. Specifically, OpenVLA is trained on 64 A100 GPUs for 14 days [Kim et al., 2024], while HyperVLA is only trained on 4 A5000 GPUs for 20 hours.

8.3.5 Ablation Studies

We run ablation studies by removing each of the algorithm design features proposed in Section 8.2.3 from HyperVLA. The ablation results in Table 8.6 and 8.7 validate that all the proposed designs contribute to the success of HyperVLA. Please see Appendix C.1 for more ablation results on other design choices in HyperVLA, especially that training a compact base policy alone cannot perform well, which validates the importance of introducing HNs to ensure sufficient model capacity.

	pick	move	close drawer	Avg
HyperVLA (Full)	58 ± 3	73 ± 1	58 ± 7	63 ± 3
- Vision backbone	24	30	38	31
- HN normalization	53 ± 4	71 ± 4	48 ± 1	57 ± 1
- Linear action head	49	56	55	53

Table 8.6: Ablation results on how different algorithm designs in HyperVLA influence its performance on Google Robot tasks. For full HyperVLA and the variant without HN normalization, we report the standard error across five random seeds, as they perform similarly and need to be compared with more random seeds. The other two variants have large performance gap to full HyperVLA, thus are trained with only one seed to reduce computational cost.

	spoon on towel	carrot on plate	stack cube	eggplant in basket	Avg
HyperVLA (Full)	48 ± 3	21 ± 5	39 ± 8	52 ± 13	40 ± 5
- Vision backbone	21	0	0	0	5
- HN normalization	48 ± 3	27 ± 6	19 ± 3	31 ± 9	31 ± 4
- Linear action head	42	23	15	39	30

Table 8.7: Ablation results on how different algorithm designs in HyperVLA influence its performance on WidowX tasks.

Vision Backbone When removing the DINOv2 backbone, we increase the number of training steps to 600k for a fair comparison, as training the whole model from scratch takes longer to converge. However, even with a larger training budget, it still significantly underperforms HyperVLA, illustrating the importance of utilizing the prior knowledge from vision foundation models. To reduce computational cost, we only train this variant for one seed, but the performance gap is significant enough to illustrate the importance of using vision foundation models as backbone.

HN Normalization To ablate the importance of HN normalization, we do not normalize the context embedding before feeding it into the HN output heads, while keeping all the other configurations unchanged. Compared to HyperVLA, it generally performs slightly worse on seen tasks, but significantly worse on the two OOD WidowX tasks, which validates the importance of stabilizing HN learning with context embedding normalization.

Action Generation Strategy We replace the linear action head in HyperVLA with a diffusion action head like in Octo [Ghosh et al., 2024], and increase the training budget to 400k steps. The diffusion-head variant underperforms HyperVLA even with a larger training budget, which illustrates that a simple linear action head trained with MSE loss is sufficient when training an HN-based VLA, and also improves training efficiency compared to more complicated action head designs like diffusion.

8.3.6 Qualitative Analysis

We qualitatively analyze the common failure patterns of different methods as follows:

The main failure reason of our method is inaccurate grasping of the object to manipulate, e.g., the policy sometimes may close the gripper when the end-effector is still slightly above the target object, which makes the robot fail to pick up the object. In general, the action error of HyperVLA is small and may be mitigated by integrating more camera views or further fine-tuning.

The OpenVLA baseline significantly underperforms our method on the picking task of Google Robot, while performs similarly on the other tasks. Its main failure reason is similar to HyperVLA due to inaccurate grasping. However, it also makes some other obvious mistakes, such as the robot arm getting stuck in the air, and not picking the target object up as expected after successfully grasping it. We also find that the robot arm movement controlled by OpenVLA is less smooth than HyperVLA, possibly due to its autoregressive way of predicting discretized action tokens.

For the other two weaker baselines RT-1-X and Octo, in addition to the grasping error, they sometimes even can not correctly locate the object to manipulate, or misunderstand the language instruction semantically, such as moving a wrong object to a wrong target object in the moving tasks, possibly because they are not built upon language and vision foundation models with strong generalization ability.

8.4 Related Work

In addition to related work on VLA pretraining and inference acceleration that has been reviewed in Chapter 5, we further review related work on context-conditioned policy generation in this section.

Faccio et al. [2023] and Di Ventura et al. [2025] adopt HNs to generate policies that can achieve different amount of expected return in a single environment, and achieve promising performance on relatively simple continuous control tasks. Generating task-conditioned policies via HNs has been investigated in multi-task and meta-RL [Yu et al., 2019, Sarafian et al., 2021, Beck et al., 2023a,b, Rezaei-Shoshtari et al., 2023], but such work mainly focuses on learning lightweight models on narrow task distributions, and do not use HNs for inference acceleration. Make-An-Agent [Liang et al., 2024b] treats parameter generation as a denoising process in the parameter space, and generates policy parameters via diffusion conditioned on demonstration trajectories, while our method generates the policy via HNs conditioned on the task context, thus can zero-shot generalize to a new task without additional demonstration trajectories from the new task. Another key difference is that Make-An-Agent treats parameter generation as a multi-stage denoising process, while we treat parameter generation as a direct mapping from the task context space to the parameter space.

8.5 Conclusion and Future Work

In this chapter, we analyzed why existing VLAs have high inference cost due to their monolithic architectures, and proposed an HN-based solution which decouples

the skills required to solve different tasks at test time for inference acceleration. To stabilize HN training and improve its performance, we further proposed several key algorithm design features, including how to properly integrate vision backbones, HN normalization, and a simple linear action head trained with MSE loss. Building upon the knowledge decoupling principle and this algorithm design, we proposed HyperVLA, which achieves performance similar to or even better than that of existing monolithic VLAs, while significant improving both training and inference efficiency.

This work paves the first step towards many interesting directions for future work on HN-based VLAs:

1. Due to hardware limitations, we only evaluate HyperVLA in simulation. Sim2real gap [Tobin et al., 2017, Peng et al., 2018] is a key challenge in robotic learning, thus a natural next step is to evaluate our method on real robots to further validate (1) If HyperVLA can maintain similar performance as monolithic VLAs on real robots, as HNs may be more prone to overfitting as discussed in Chapter 3.2.2 and transfer worse; and (2) If its significant advantage in inference efficiency can be maintained, as deployment on real robots will introduce many other latency factors other than model inference, such as data acquisition, preprocessing and communication.

2. Scale up the HN model size and train on more recent and larger robotic datasets [Khazatsky et al., 2024, NVIDIA et al., 2025] for further performance improvement.

3. Scale up to more complicated tasks, like long-horizon tasks that require high-level planning or memory of interaction history.

4. While HyperVLA significantly reduces the size of the policy head during inference, the image encoder still uses vision foundation models as the backbone, which becomes the main efficiency bottleneck now. Future work could investigate how to further reduce the size of the image encoder, such as distilling the large image encoder into an HN-generated one in an additional training phase.

While this chapter focused on utilizing HNs in the pretraining phase of VLAs to improve inference efficiency, in the next chapter we will investigate how to utilize HNs in the fine-tuning/post-training phase of VLAs to enable better few-shot adaptation to tasks from a new domain.

9

HyperLoRA: Efficient Domain Adaptation of Vision-Language-Action Models via Hypernetwork-Generated LoRA

Contents

In this chapter, we shift our attention from VLA pretraining to VLA fine-tuning, and investigate how to utilize HNs to enable sample- and computation-efficient adaptation of a pretrained VLA to a new domain with many different tasks to solve. We propose to generate task-specific LoRA parameters via HNs to better accommodate the different skills required to solve different tasks, and empirically validate its effectiveness on a challenging benchmark containing 90

tasks in different household scenarios.

9.1 Introduction

As introduced in Section 5.4, while VLAs have achieved promising zero-shot generalization performance, they still require further fine-tuning to achieve satisfying performance on downstream tasks for deployment in real world, especially on those with domain gaps to the training tasks.

To reduce the cost and difficulty of deployment, the fine-tuning process should be both sample- and computation-efficient, i.e., requiring the collection of only a small amount of additional demonstrations on the new tasks and introducing little computational overhead. As proven by many existing VLAs, the fine-tuning process can be sample-efficient thanks to the strong prior knowledge encoded in the pretrained VLAs [Ghosh et al., 2024, Black et al., 2025b, Kim et al., 2024, Liu et al., 2025c]. Computational efficiency can be improved by adopting parameter-efficient fine-tuning (PEFT) methods like Low-Rank Adaptation (LoRA) [Hu et al., 2022], instead of fine-tuning the whole VLA with millions or billions of parameters [Kim et al., 2025, 2024].

However, existing methods mainly fine-tune only on a single or a small number of downstream task(s), while in practice a generalist robot is more likely to be deployed in an unseen environment to accomplish many different tasks. Thus in this chapter, we investigate a more challenging setting of adapting a pretrained VLA to a target domain with *many different tasks* in a parameter- and data-efficient way, where the domain is defined as an environment in which a robot needs to accomplish a set of different tasks. For example, consider learning an industrial robot that can work in different factories (Figure 9.1). After pretraining a VLA 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 VLA in a zero-shot fashion may not 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

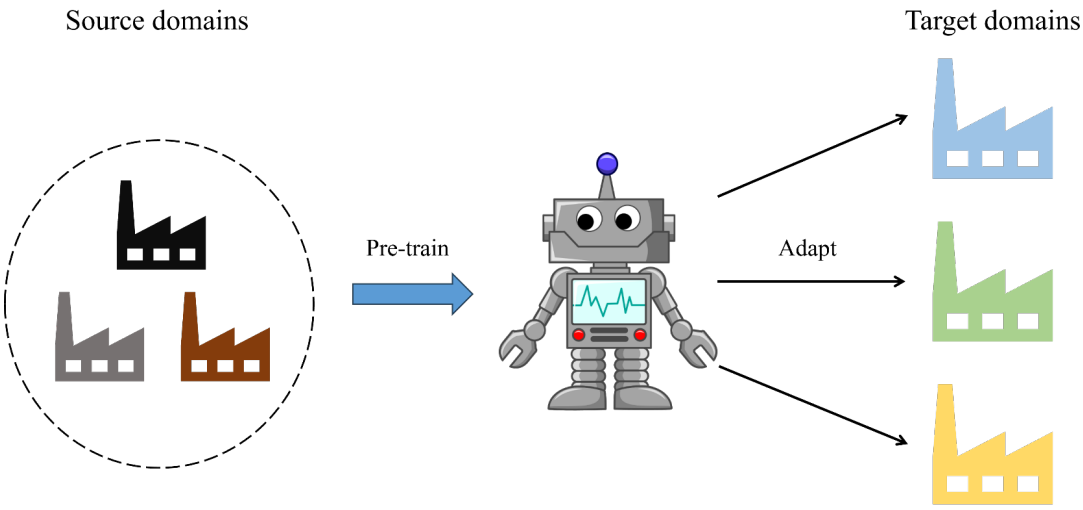

Figure 9.1: A domain adaptation example of fine-tuning a pretrained industrial robot to work in different factories.

collecting expert demonstrations in each new factory, and parameter-efficient to reduce the computational cost and further improve data efficiency.

To achieve this goal, we introduce *HyperLoRA*, which generates 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. Experimental results on the LIBERO benchmark show that HyperLoRA significantly improves the model's performance on the training tasks from the target domain. However, though HyperLoRA also shows some extent of zero-shot generalization to unseen tasks from the target domain, task generalization in the new domain remains challenging due to the limited number of tasks and demonstrations available for VLA fine-tuning, which is an interesting direction for future work.

9.2 Background

9.2.1 Problem Formulation

Given a VLA pretrained on some source domains, this chapter investigates 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 pretraining domains

in scenarios, task set, robot embodiment, etc. Each task from the target domain is specified by a sentence of language instruction.

For supervised fine-tuning, we assume access to a small amount of expert demonstrations on a task subset $\mathcal{T}_{\text{train}} \subset \mathcal{T}_{\text{target}}$ in the target domain. We want to fine-tune the VLA with these demonstrations from the target domain via BC to achieve not only better performance on the training tasks $\mathcal{T}_{\text{train}}$ compared to zero-shot generalization, 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.

9.2.2 Low-Rank Adaptation

Low-rank adaptation (LoRA) was first proposed for parameter-efficient fine-tuning of large language models [Hu et al., 2022], and has recently been successfully applied to VLAs as well [Kim et al., 2024]. Instead of fine-tuning all the parameters of a large model, LoRA freezes the pretrained 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 learned, which reduces the number of trainable parameters from $m \times n$ to $(m+n) \times r$.

9.3 HyperLoRA

9.3.1 Motivation

To fine-tune a VLA on multiple different tasks with LoRA, a straightforward approach is to learn a separate set of LoRA parameters for each training task in the target domain, which we called single-task LoRA (ST-LoRA). 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 [Perez et al., 2018], and HN. In this work, we adopt the HN approach, as it has high model capacity to encode the complicated dependency between task context and control parameters, supported by both theoretical analysis [Galanti and Wolf, 2020, Sarafian et al., 2021] and empirical evidence from previous work [Yu et al., 2019, Peng et al., 2021, Beck et al., 2023a,b, Rezaei-Shoshtari et al., 2023] and the previous chapters of the thesis.

9.3.2 Model Architecture

As shown in Figure 9.2, HyperLoRA generates task-specific LoRA parameters via a context-conditioned HN. The TF context encoder in the HN takes two inputs:

1. Token embeddings of the language instruction from a frozen T5 tokenizer [Raffel et al., 2020].

2. Multiple learned layer index tokens, each corresponding to one TF layer in the VLA model to fine-tune.

After context encoding, the embeddings of the layer index tokens go through a set of shared HN output heads to generate the LoRA parameters in each layer of the VLA.

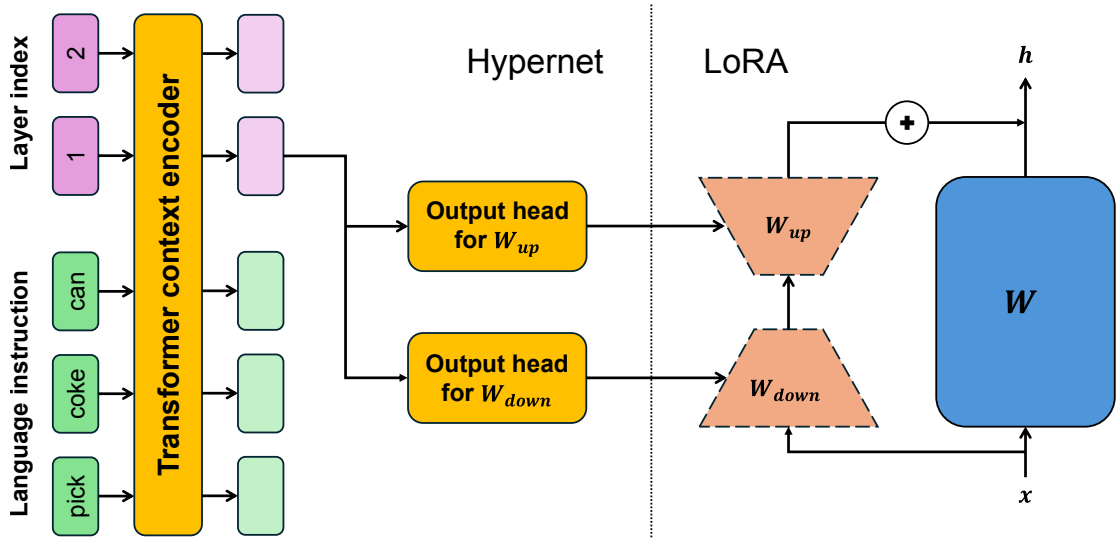

Figure 9.2: The framework of HyperLoRA. For ease of illustration, we only show how to generate LoRA parameters for a single weight matrix W in the VLA. The HN takes in language instruction tokens and layer index tokens as input, and encodes them with a TF context encoder. The context embeddings of the layer index tokens are passed into shared output heads to generate the LoRA parameters for different layers in the VLA. Trainable parameters are marked in orange, and HN-generated LoRA parameters are represented by trapeziums bounded by dashed lines.

Unlike in HyperVLA where we only use a single context token as the input to the HN context encoder, here we use multiple layer index tokens as the context tokens to reduce the number of learnable parameters in the HN and improve parameter efficiency. Specifically, although each LoRA block contains just two low-rank matrices, generating LoRA blocks for all the matrices in a VLA still introduces a considerable memory cost. To reduce the size of the HN, we utilize the fact that most VLAs are built by stacking multiple layers of TF blocks with the same architecture. By feeding the layer index of the VLA as an additional input to the HN, we can reuse the same set of HN output heads to generate different LoRA parameters for different layers in the VLA, instead of learning separate output heads for each layer, which can reduce the size of HN by approximately N times, where N is the number of TF layers in the VLA.

9.4 Experiments

We conduct experiments to validate if HyperLoRA can:

1. Improve the performance of a VLA on the training tasks in the target domain?

2. Enable zero-shot generalization to unseen tasks from the target domain?

9.4.1 Experimental Setup

Pretrained VLA We use Octo [Ghosh et al., 2024] as the pretrained VLA, as it achieves a good trade-off between performance and computational cost, and was a SOTA VLA at the time when this work was conducted.

Target Domain We use the LIBERO-90 task suite from the LIBERO benchmark [Liu et al., 2023a] as the target domain, which contains three household environments of kitchen, living room and study room. Each environment contains multiple scenes, where each scene is defined as a specific layout of a set of commonly seen objects in the corresponding environment. Multiple different tasks can be defined in each scene by specifying different goals, such as moving different objects to different goal positions. In this way, LIBERO-90 defines 90 different tasks in total. We hold out one task from each scene to make a set of 20 test tasks in the target domain. The remaining 70 tasks are used as the training tasks from the target domain. 50 demonstrations are provided for each task, leading to 3,500 demonstrations in total for VLA fine-tuning. To evaluate data efficiency of different fine-tuning methods, we fine-tune with different data budgets of 1, 5, 10, or 50 demonstrations per training task.

Baselines As discussed in Section 9.3.1, we compare HyperLoRA with (1) Multi-task LoRA (**MT-LoRA**) to validate the importance of introducing HN to generate task-conditioned LoRA parameters; and (2) Single-task LoRA (**ST-LoRA**) to see if HyperLoRA improves sample efficiency via knowledge sharing across tasks.

Training Hyperparameters During the fine-tuning process, we freeze the TF parameters in Octo, add LoRA to the two MLP layers in each TF layer, and further update all the parameters of the diffusion head.

For HyperLoRA, we fine-tune for 300k steps with a batch size of 32. We generate LoRA modules with a rank of 4. For the TF context encoder, we use a two-layer TF with 4 attention heads, and set the embedding dimension and MLP hidden dimension as 128 and 256 respectively.

For MT-LoRA, the LoRA rank is increased to 48 so that it has a similar number of trainable parameters as HyperLoRA. Other hyperparameters follow the same setup as HyperLoRA.

For ST-LoRA, we fine-tune for 100k steps on each single training task. Other hyperparameters follow the same setup as MT-LoRA. Due to the high computational cost of running ST-LoRA on all the training tasks separately, we only train ST-LoRA with the lowest data budget of 1 demo per task.

In total about 6M parameters are updated during domain adaptation for all the methods considered, taking only about 3% of the parameters in Octo.

Evaluation We evaluate the fine-tuned model on both the training tasks with parametric variations and unseen test tasks with new instructions in the target domain. Each task is evaluated for 50 episodes.

9.4.2 Results

Comparison with MT-LoRA

As shown in Figure 9.3 and Table 9.1, HyperLoRA consistently outperforms MT-LoRA on both the training and test tasks under different data budgets, validating the importance of learning context-conditioned LoRA parameters for each task to achieve better adaptation performance. Notably, HyperLoRA is about 10 times more sample-efficient than MT-LoRA on the training tasks. Specifically, HyperLoRA achieves an average success rate of 30% on the training tasks with only 1 demo per task, while MT-LoRA achieves a similar success rate of 31% when fine-tuned with 10 demos per

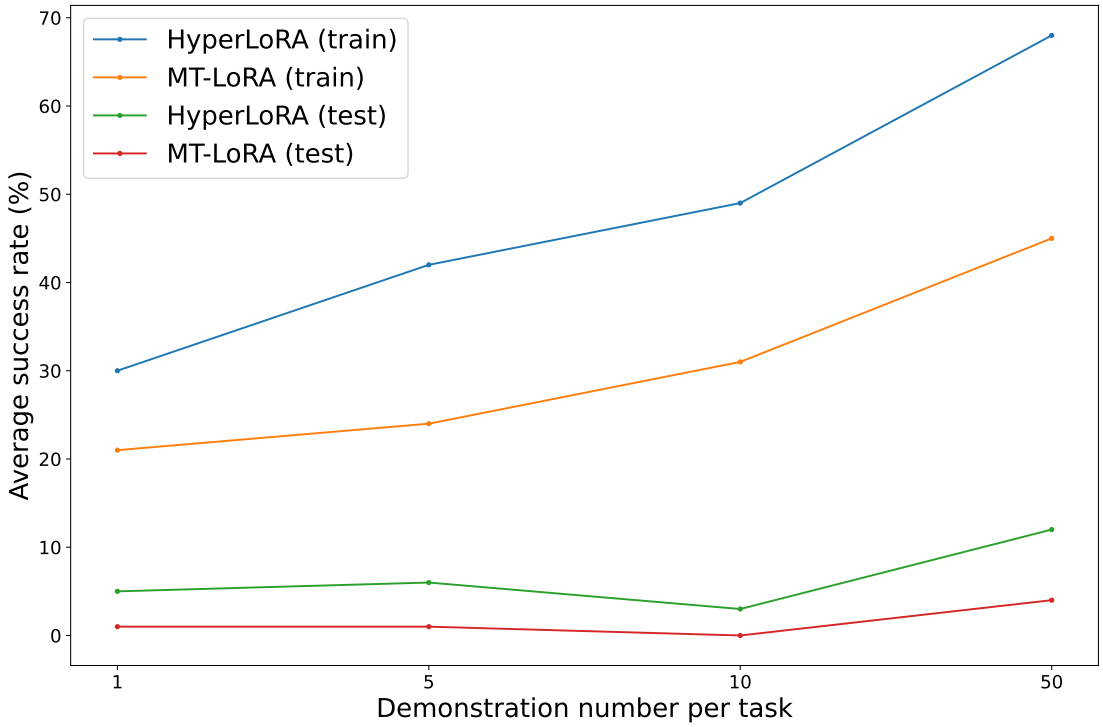

Figure 9.3: The average success rate of different methods over the training and test tasks from the target domain with different number of demonstrations per task.

Task set	Method	# Demonstration per training task			
		1	5	10	50
Train	MT-LoRA	21	24	31	45
	HyperLoRA	**30**	**42**	**49**	**68**
Test	MT-LoRA	1	1	0	4
	HyperLoRA	**5**	**6**	**3**	**12**

Table 9.1: The average success rate of different methods over the training and test tasks from the target domain with different data budgets for fine-tuning.

task. Similarly, HyperLoRA scores 42% on the training tasks with 5 demos per task, matching MT-LoRA's score of 45% when fine-tuned with all the 50 demos per task.

However, for both methods, the average success rate on the test tasks is much lower than the success rate on the training tasks, which indicates a significant generalization gap. We hypothesize this is because that only a small number of training instructions have been seen during fine-tuning, thus the fine-tuned models can not generalize well to out-of-distribution task instructions containing unseen objects or skills. In general, adaptation to unseen tasks in a new domain is still hard,

as we can only collect data from a limited number of tasks in the target domain, which is not sufficient to enable good task generalization of the fine-tuned model.

Comparison with ST-LoRA

When fine-tuned with only 1 demo from each training task, ST-LoRA achieves an average success rate of 32.5% on all the training tasks, which is similar to HyperLoRA's average success rate of 33.6% under the same data budget. We further train ST-LoRA on a subset of 8 training tasks separately with a sample budget of 50 demonstrations per task, and find that its average success rate is also similar to that of HyperLoRA under the same sample budget. These results indicates that although HyperLoRA enables knowledge sharing across tasks, it does not lead to significant performance improvement compared to ST-LoRA as hypothesized in Section 9.3.1. We list several potential reasons as below:

1. Task interference in multi-task learning [Ruder, 2017, Yu et al., 2020a, Liu et al., 2021a, Kurin et al., 2022] may hinder HyperLoRA from learning the optimal LoRA parameters for each task, which counteracts the benefits of knowledge sharing.

2. Knowledge sharing across the fine-tuning tasks may provide marginal help given that the pretrained VLA has already encoded strong prior knowledge for fast adaptation to a new task.

3. HyperLoRA and ST-LoRA are not compared under a fair computational budget. Specifically, ST-LoRA runs for 7M steps in total on all the training tasks, which is more than 20 times larger than the budget of 300k steps for HyperLoRA fine-tuning. The conclusion may be different if we fine-tune for longer until each method converges.

As discussed in Section 9.3.1, another potential benefit of HyperLoRA compared to ST-LoRA is the zero-shot generalization ability to unseen tasks from the target domain. However, given that HyperLoRA only achieves a low average success rate

on the test tasks as shown in Table 9.1, this advantage is also negligible and may only hold for unseen tasks that are very similar to the training tasks used for fine-tuning.

Nevertheless, HyperLoRA's advantage in memory cost still holds. While ST-LoRA needs to save a separate set of LoRA parameters for each task in the target domain, HyperLoRA only needs to save a single set of HN parameters regardless of the task number. In our experiments, as ST-LoRA learns a separate set of LoRA modules that has a similar number of parameters as HyperLoRA for each training task, the total memory cost of ST-LoRA is 70 times higher than that of HyperLoRA.

9.5 Related Work

In addition to related work on VLA fine-tuning that has been reviewed in Section 5.4, we further review related work on HN-generated PEFT methods in this section.

In the domain of NLP, Mahabadi et al. [2021] and Zhao et al. [2023a] utilize HNs to generate the parameters of adapter [Houlsby et al., 2019], a commonly used PEFT method, and show positive knowledge transfer across tasks when fine-tuning on multiple tasks. Similarly, HyperPrompt [He et al., 2022] adopts HNs to generate task-conditioned prompts for PEFT of language models. Kim et al. [2023] further extends this idea to federated learning to generate different adapter parameters for heterogeneous clients via HN, while Adhikari et al. [2025] applies this idea to continual learning.

Most similar to our work is Hyper-Decision Transformer (HDT) [Xu et al., 2023], which also utilizes HN to generate task-specific adapter parameters for efficient adaptation in robotic control. 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 VLAs.

9.6 Conclusion and Future Work

This chapter introduced HyperLoRA for efficient adaptation of VLAs to new domains with many different tasks by learning to generate task-specific LoRA parameters with an HN. HyperLoRA not only significantly improves model performance on the training tasks in the new domain, but also shows some extent of zero-shot generalization to unseen test tasks from the same domain.

Given the promising preliminary results of HyperLoRA, future work could focus on:

1. Evaluate HyperLoRA on more target domains both in simulation [Yu et al., 2020b, Mees et al., 2022] and on real robots.

2. Apply HyperLoRA to more recent VLAs to see if its advantage over vanilla LoRA still holds when the pretrained VLA is scaled up [Kim et al., 2025], as it is possible that the benefits introduced by HN's higher model capacity will become marginal when the expressive power of the VLA itself is strong enough.

3. Further improve zero-shot generalization of HyperLoRA to unseen tasks in the target domain, such as decomposing the high-level language instruction into more fine-grained skills that transfer better between tasks.

Part IV

Discussion

10 Reflection and Future Directions

Contents

While we have discussed about future work directions for each specific work in previous chapters, in this chapter we give a high-level reflection on the research conducted in this thesis and propose some general directions for future work.

10.1 Unifying Cross-Embodiment and Cross-Skill Control

In this thesis, we investigate generalization across embodiments and skills under two separate problem settings. For embodiment generalization, we only consider a fixed goal of maximizing locomotion distance. For skill generalization, we only consider embodiment generalization over different robot arms that share the same end-effector action space.

However, more recent work has made some initial efforts to unify these different dimensions of generalization under a single problem setting, such as extending

universal morphology control to a multi-goal setting by treating different goals as additional nodes in a unified morphology-goal graph [Furuta et al., 2023], and learning VLAs on a broader range of diverse embodiments by training a unified TF with different token blocks corresponding to each embodiment type [Yang et al., 2024, Doshi et al., 2024, Zheng et al., 2025]. We believe this is a promising direction for future work to learn a more powerful generalist policy by training on larger-scale data, while one of the key challenges here is how to design a unified interface that can flexibly support both different embodiments and goals, and integration of useful domain knowledge into the model. Moreover, whether training on a broader task distribution is beneficial or not is an empirical question that depends on many factors, such as our data size (for IL) or interaction budget (for RL), the task distribution we want to generalize to, etc. Intuitively, training on a larger task distribution enables learning from a larger data source, but needs to deal with a harder multi-task optimization problem with potentially more interference across tasks. So it boils down to a trade-off of whether the benefits of knowledge transfer across more tasks and data can outweigh the difficulty in optimization or not. For example, as shown in Chapter 6 and 7, a specialist MLP model outperforms a generalist TF model if we only care about the locomotion performance of a specific robot, but this holds only if we have sufficient computational budget to train these models with RL. On the other hand, for complicated manipulation tasks where demonstrations are expensive to collect, even if we only care about performance on a narrow distribution of downstream tasks, it has been proven by many existing VLAs [Ghosh et al., 2024, Kim et al., 2024, Black et al., 2025b,a] that it is crucial to first pretrain on a broad task distribution to learn a strong and robust prior, and then fine-tune with a small amount of demonstrations on the downstream tasks, as it is too expensive to collect a large enough number of demonstrations on these tasks alone to learn a model with good performance. As data scarcity will remain as a key bottleneck of robotic learning in the near future, we expect that learning from a broader task distribution and designing

algorithms that can better transfer knowledge across these richer but heterogeneous data sources will be a trend in future work.

10.1.1 What is a Useful Embodiment Distribution?

To learn a generalist policy, a key question is what is the distribution of tasks that we expect it to generalize to. For skill generalization, this question may be easy to answer, as we just want the robot to do whatever human can do in different environments. However, for embodiment generalization, the answer may not be very clear. In principle, training on cross-embodiment data improves data efficiency of learning on the training embodiments, and enables better generalization or more efficient adaptation to a new embodiment that is similar to the training ones. So given the huge morphology space of all possible robot embodiments, we need to carefully choose the training embodiments, which have a huge influence on: (1) Where and how we collect the training data; and (2) To what new embodiments can we expect good generalization.

In this thesis, we adopt the UNIMAL benchmark [Gupta et al., 2022] as the embodiment distribution for the problem setting of universal morphology control. But this is mainly for research purpose, as (1) We need a scalable and systematical approach to generate diverse morphologies for multi-robot training and evaluation; and (2) It is better to use the same benchmark as in previous work for a fair comparison. However, most morphologies in the UNIMAL benchmark, like those shown in Figure 6.4, are rarely used in real robots. In other words, although generalization across embodiments is a valuable question and the experimental results on the UNIMAL benchmark validate the effectiveness of our methods, the embodiment distribution defined by the benchmark may not fully align with the cross-embodiment scenarios we really care about in the real world.

Consequently, to better reflect the problem settings that will be encountered in real-world scenarios, future work on cross-embodiment control should focus more on the embodiments that are commonly seen and used in real world, such as robot arms

with grippers and dexterous hands for manipulation, quadruped robots, wheeled robots and humanoids for locomotion and navigation, etc.

Furthermore, in addition to training on data collected from these commonly used robot embodiments, human videos is another important data source with a huge volume as introduced in Section 5.2.3. Designing methods to mitigate the embodiment gap between human and robot to better utilize the abundant human videos is an important direction for future work, and could borrow insights from cross-domain transfer learning as introduced in Section 4.2.1. Many useful representations can be learned from human videos at different levels, such as pretrained visual representation [Nair et al., 2023, Wu et al., 2024a, Cheang et al., 2024], reward and value function [Ma et al., 2023a,b], latent actions [Bruce et al., 2024, NVIDIA et al., 2025, Bu et al., 2025a,c, Ye et al., 2025, Bi et al., 2025a], etc.

As a final remark, although we have discussed about the importance of benchmarking on a more practical embodiment distribution that better aligns with real-world robots from a problem-driven perspective, we also want to highlight the value of a method-driven perspective for research that identifies the limitations of existing methods under a specific problem setting to motivate new algorithms. For example, although the UNIMAL benchmark itself may not perfectly reflect real-world embodiment distribution, the key ideas we develop when tackling the challenges in this benchmark, such as using HNs to learn more expressive multi-task policy and the knowledge decoupling principle, have been proven very useful in the VLA setting. In summary, research is an exploration process in the unknown space with high uncertainty, and sometimes we need to do some less "useful" research in short term to gain important insights that may help with long-term research just like in RL.

10.2 Towards Self-Improving Generalist Robots

Although great progress has been made for learning generalist robots in recent years, their performance is still far from perfection for large-scale real-world deployment. While fine-tuning enables better performance on downstream tasks, sometimes even achieving a high enough success rate for real-world deployment on some specific

tasks, it usually comes at the cost of losing the broad generalization ability of the pretrained model due to catastrophic forgetting, which contradicts with our original motivation of learning generalist robots. In general, existing robots mainly follow a "train-(fine-tune)-deploy" paradigm by first learning on a fixed set of pretraining or fine-tuning data, then deploying in the real world without further model update. In contrast, humans can continually learn from experience to be more skilled at different tasks during their lifetime. So we believe it is an important direction for future work to learn self-improving generalist robots that can continually optimize their policies with new experience accumulated during deployment.

To achieve this goal, we discuss about two promising directions for future work as below:

Self-Correction during Task Execution The ability to correct mistakes when solving a task may significantly improve a generalist robot's performance, e.g., if a robot arm fails to pick an object due to grasping on the object's edge, it can self-correct its behaviors by slightly adjusting the location to close its gripper so that it can grasp the object's body in the next trial.

However, most existing generalist robots cannot correct mistakes in such a way as they are memoryless, i.e, if we input the same state/observation sequence to the policy, it will still output the same action that leads to failure. Although some VLAs have been claimed to show error recovery ability [Black et al., 2025b], they work not by really updating the policy for correction, but because the mistaken action takes the robot to a next state where the policy happens to know how to act correctly. So this kind of "error recovery" ability is still limited.

Context-based meta-RL methods [Beck et al., 2025] may provide a promising solution to this challenge, where the policy further conditions its behaviors on a context encoding of the interaction history with the environment so far, so that the policy can try new ways to fix its mistake by conditioning on a continually evolving history context. Memory-based VLAs [Li et al., 2024b, Shi et al., 2025a] have been explored to better solve long-horizon and temporally dependent tasks, which

could also be adapted for self-correction. Instead of improving upon a single policy, quality-diversity methods pretrain a set of qualitatively different policies that have similar performance on the training tasks, then adaptively try out these policies on a new task to find the most suitable one [Cully et al., 2015, Mouret and Clune, 2015, Margolis and Agrawal, 2023]. In other words, they self-improve by searching in a a pre-given policy space, instead of in the parameter space of a given policy.

A key challenge here is how to collect training data with self-correction behaviors to either learn the history context encoder or fill the memory buffer with relevant transitions, as most exiting robotic datasets only contain successful demonstrations from experts. Another key challenge is how to enable broad and robust self-correction ability if the the robot encounters new failure modes that are out-of-distribution [Mendonca et al., 2020, Xiong et al., 2021]. It is also interesting and important to investigate how to utilize VLM's strong prior knowledge about the world to provide more guidance on self-correction instead of learning simply via trial and error.

Lifelong Self-Improvement In addition to intra-task self-correction, another key desiderata of self-improving generalist robots is the ability to continually learn from newly accumulated experience, no matter successful or failed, in a lifelong fashion to achieve better performance on future tasks. As discussed before, fine-tuning with IL or RL can improve performance on specific tasks, but sacrifices generalization performance due to catastrophic forgetting. While we can co-fine-tune with both pretraining and newly acquired data to achieve both better performance and maintain generalization, this is usually time-consuming and computationally expensive to run, and will involve privacy issue if the users want to keep the data collected in their deployment scenarios private.

Lifelong learning methods [Parisi et al., 2019, Wang et al., 2024b] provide a natural solution to this challenge, but how to scale these methods to a broader task distribution that will be encountered by a generalist robot remains underexplored [Liu et al., 2023a]. Learning task-specific LoRA modules [Liu et al., 2023c], i.e., the single-task LoRA method we discussed in the HyperLoRA chapter, is another way

to self-improve without catastrophic forgetting, but its scalability and generalization remains a challenge as discussed in Chapter 9. From a model architecture perspective, modular/hierarchical architectures like dual-system VLAs may be more suitable for lifelong learning compared to monolithic architectures, as they can decompose the functionality of different modules and only update task-relevant modules without influencing the behaviors of other modules, such as learning a set of separate skills modules without interference.

10.3 Hypernetworks in Foundation Models

In this thesis, we have frequently used HNs under two main motivations and achieved promising results accordingly. First, we utilize the strong expressive power of HNs to better model the complicated dependency between the task context and the correspondingly optimal control policy, which significantly boosts model performance as shown in ModuMorph and HyperLoRA. Second, compared to monolithic multi-task policies that need to activate the whole model during inference, we utilize the knowledge decoupling ability provided by HN's hierarchical architecture to enable both high model capacity during pretraining, and efficient inference by only activating a compact HN-generated task-specific policy at test time, which achieves similar performance and significantly higher inference efficiency compared to monolithic models in HyperDistill and HyperVLA.

While HNs also introduce new challenges like more complicated optimization and higher overfitting risks as discussed in Section 3.2.2, in principle we believe these do not constitute fundamental limitations of HNs and can be better solved if more research focuses on addressing these issues in the future. For example, we have proposed how to alleviate optimization challenges in HNs via policy distillation in HyperDistill and context normalization in HyperVLA, and how to improve task generalization of the HN in HyperDistill.

Consequently, we believe HNs could serve as an important building block in future robotic foundation models. In addition to their advantages in model capacity and inference efficiency as investigated in this thesis, future work could

further utilize other characteristics of HNs, such as their ability to measure model uncertainty [Krueger et al., 2017, Ratzlaff and Fuxin, 2019, Chauhan et al., 2024b] and reduce catastrophic forgetting in continual learning [von Oswald et al., 2020], to learn better generalist robots.

Furthermore, while this thesis only utilizes HNs for robotic control, another promising direction for future work is to further extend their application to foundation models in other domains, especially for more efficient inference. For example, HNs can be applied to VLMs for image generation or understanding, where the language query serves as the task context to generate a compact base network that only extracts query-relevant visual features from the image input. How to scale HN learning to a even larger model size to match the performance of SOTA foundation models in different domains remains an open question for future work.

11 Conclusion

This thesis investigated how to learn robots that can generalize across a broad distribution of tasks and fast adapt to new tasks by training on large-scale multi-task robotic data.

We started by formulating multi-task robotic control as a Contextual Markov Decision Process in Chapter 2, which can be solved by learning a context-conditioned policy via either RL or IL. By defining the task context in different ways, we investigated two different dimensions of generalization in robotics: (1) Generalization across robot embodiments, i.e., universal morphology control; and (2) Generalization across skills specified by language instructions.

In Part II of the thesis, we focused on how to improve pretraining and inference efficiency in cross-embodiment control. In chapter 6, we proposed ModuMorph, a Transformer-based universal controller that better models the complicated dependency between a robot's morphology and its optimal behaviors via contextual modulation, which includes: (1) Generating morphology-conditioned policy parameters via HN; and (2) Fixed attention weights that solely condition on the robot morphology. By better modeling how the policy conditions on the morphology context, we achieved consistent performance improvement on both training and test morphologies.

In Chapter 7, we investigated if we can combine strong generalization of Transformer and high efficiency of MLP in universal morphology control. We proposed a key principle called *knowledge decoupling*, i.e., a generalist policy can be significantly accelerated at inference time if it can decouple the knowledge required to solve different tasks and only activate a specialist policy to solve each specific task. We utilized HN's hierarchical architecture to realize knowledge decoupling, and designed a morphology-conditioned HN to generate a compact MLP base policy to control each robot. We further investigated how to successfully train this HN via policy distillation and improve its generalization to unseen morphologies. HyperDistill, our proposed method that combines these two key components, achieves similar performance as ModuMorph, while significantly accelerating inference speed by two orders of magnitude, which validates the power of knowledge decoupling.

In Part III of the thesis, we focused on cross-skill control of how to learn a VLA that follows language instructions to solve different manipulation tasks. In Chapter 8, we applied the knowledge decoupling principle to VLAs, as inference efficiency is a key bottleneck of existing VLAs. We proposed HyperVLA to generate a compact policy via an HN that conditions on language instruction and image context, and investigated several key algorithm design choices to improve the performance of HyperVLA. It achieves similar and even better performance compared to SOTA monolithic VLAs, while significantly improving inference efficiency by two orders of magnitude, which again validates the effectiveness of knowledge decoupling.

Finally in Chapter 9, we investigated how to adapt a pretrained VLA to a new domain with many different tasks in a sample- and computation-efficient way. We proposed HyperLoRA, which utilizes the strong expressive power of HNs again to generate task-conditioned LoRA parameters, and significantly outperforms task-agnostic LoRA adaptation w.r.t. adaptation performance and sample efficiency.

In summary, the contributions in this thesis significantly improve model pre-training, inference efficiency or adaptation performance when learning a generalist robot that can generalize across a broad range of embodiments and skills. We

hope the key ideas developed in this thesis, such as the knowledge decoupling principle and utilizing HNs as a key building block in learning generalist robots, could motivate more future work in these directions to help build more versatile and efficient foundation models for robotics and other domains.

Appendices

A

Appendix for Chapter 6, ModuMorph

Contents

A.1 Implementation Details

A.1.1 Architecture Details of Contextual Modulation

Intuitively, using GNN or TF as the context encoder to model node interactions may learn better context representations. However, in practice we find that using simple MLPs as the context encoder achieves similar or even better performance. The reason might be that we have many fewer training samples for the context encoder, which equals the number of robots we have for training, as the morphology context does not change on a robot. So using models with high capacity may not be helpful here and even lead to overfitting. Moreover, HNs are known to be hard to optimize [Chang et al., 2020], and we find that using TF as the context encoder makes HN

training unstable. Consequently, we just use MLPs as the context encoder shared over different nodes. Specifically, we train two separate context encoders for HN and FA respectively. The context encoder is a 2-layer MLP for HN, and a 3-layer MLP for FA, both with 128 units in each hidden layer.

The HN output layer is implemented as a linear mapping from context encoding to the parameters in the base network, with one independent output head for each modulated layer in the base controller. We initialize it with the *Bias-HyperInit* method proposed by Beck et al. [2023a], i.e., the weights are set to 0, and the biases are sampled from the same distribution that is used to initialize the modulated layer in the base network. In this way, all the nodes share the same control parameters just like MetaMorph at the beginning, and gradually develop node-wise diversity while the HN weights are updated.

A.1.2 Proprioceptive and Context Features

We use the same proprioceptive and context features as in MetaMorph for a fair comparison, which can be found in Appendix A.1 of Gupta et al. [2022].

A.2 PPO and Early Stopping

PPO [Schulman et al., 2017] is an on-policy RL algorithm that takes multiple steps of update on the current data we have, while also trying not to exceed some trust region boundary to avoid performance collapse. For a state-action pair (s, a), the clipping objective function of PPO is defined as

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s, a)\right),$$

where π_{θ_k} is the behavior policy used to collect on-policy data for policy update in iteration k, π_θ is the target policy we want to learn, $A^{\pi_{\theta_k}}(s, a)$ is the advantage function of π_{θ_k}, and ϵ is a hyperparameter that constrains the updated policy to not be too far away from the behavior policy.

In many RL libraries, one PPO iteration is implemented by first collecting N steps of rollout data on M workers in parallel with π_{θ_k}, then dividing the collected

data into B batches and repeating minibatch update for T epochs. Consequently, for one PPO iteration, we do $T \cdot B$ times of minibatch update.

However, the clipping objective function alone can not guarantee policy update within the trust region. Early stopping is a solution to this problem by first checking the approximate KL divergence between π_θ and π_{θ_k} before each minibatch update, and terminating the current update iteration if the divergence exceeds a threshold value δ. The policies' KL divergence is approximated as $D_{\mathrm{KL}}(\pi_{\theta_k}||\pi_\theta) \approx \sum_{(s,a)\in\mathcal{B}} \log\left(\frac{\pi_{\theta_k}(a|s)}{\pi_\theta(a|s)}\right)$, where $\mathcal{B}$ is the current data minibatch used for policy update.

The early stopping threshold δ is a critical hyperparameter in PPO, as it gives a measurement of the range of the trust region we allow for policy update. We tune it over the candidate set of $\{0.03, 0.05\}$, and report the optimal value of δ for each method in each environment in Table A.1. We do not tune over a wider range, as empirically we found that a even smaller value of δ usually causes early stopping to happen too early, while a larger value allows for a too large trust region, both harming the learning performance.

Environment	MetaMorph*	FA	HN	FA+HN
FT	0.05	0.05	0.05	0.05
Incline	0.03	0.05	0.05	0.05
Exploration	0.03	0.03	0.03	0.03
VT	0.03	0.03	0.03	0.03
Obstacles	0.03	0.03	0.03	0.03

Table A.1: Optimal value of the early stopping threshold for each method in each environment.

A.3 Analysis on MetaMorph

In this section, we introduce the issues found when reproducing MetaMorph results with its source code, which motivates us to propose the MetaMorph* variant as an alternative baseline.

The MetaMorph paper reports significant improvement in learning performance by adding positional encoding (PE) to the node embedding. Specifically, MetaMorph

adopts learned PE, i.e., the PE for each position is a vector that is learned during training instead of hard-coded in advance. However, a robot morphology is structured as a tree, which does not hold sequential information about each node by nature. So MetaMorph first traverses each morphology tree via depth-first search to turn it into a 1D sequence, then index each node by its position in the sequence. One PE vector is learned for each position in the sequence, and the nodes with the same index across different robots will share the same PE vector.

However, we found that in the source code of MetaMorph, PE is implemented as $e'_i = \text{dropout}(e_i + \text{PE}_i)$, where e_i is the embedding of node i. To investigate which operation actually contributes to the performance improvement, we experiment with $e'_i = \text{dropout}(e_i)$ and $e'_i = e_i + \text{PE}_i$ respectively, and surprisingly find that the dropout operation is the main contributor here, while PE alone makes little difference in training performance (Figure A.1).

A.3.1 Why PE Does Not Help?

Our hypothesis here is that PE has the benefit of enabling more diverse behaviors across different nodes, but also has the drawback of adding the same PE vector to nodes with different physical meanings across robots, i.e., PE is not consistent across morphologies (Figure A.2 shows two sources of inconsistency in PE). And when training on multiple robots, the drawback outweighs the benefit, so PE provides no performance gain overall.

To validate this hypothesis, we investigate the effect of PE in single-task (ST) training. If our hypothesis holds, we should observe better performance by using PE compared to not, as there is no inconsistency issue on a single robot, while the benefits of PE maintains. We experiment on 30 morphologies and show their average learning curve in Figure A.3. As expected, PE indeed improves training performance in ST training, which proves that using PE to distinguish between different nodes in a single morphology is helpful. However, the inconsistency issue breaks its effectiveness in the multi-morphology training setting.

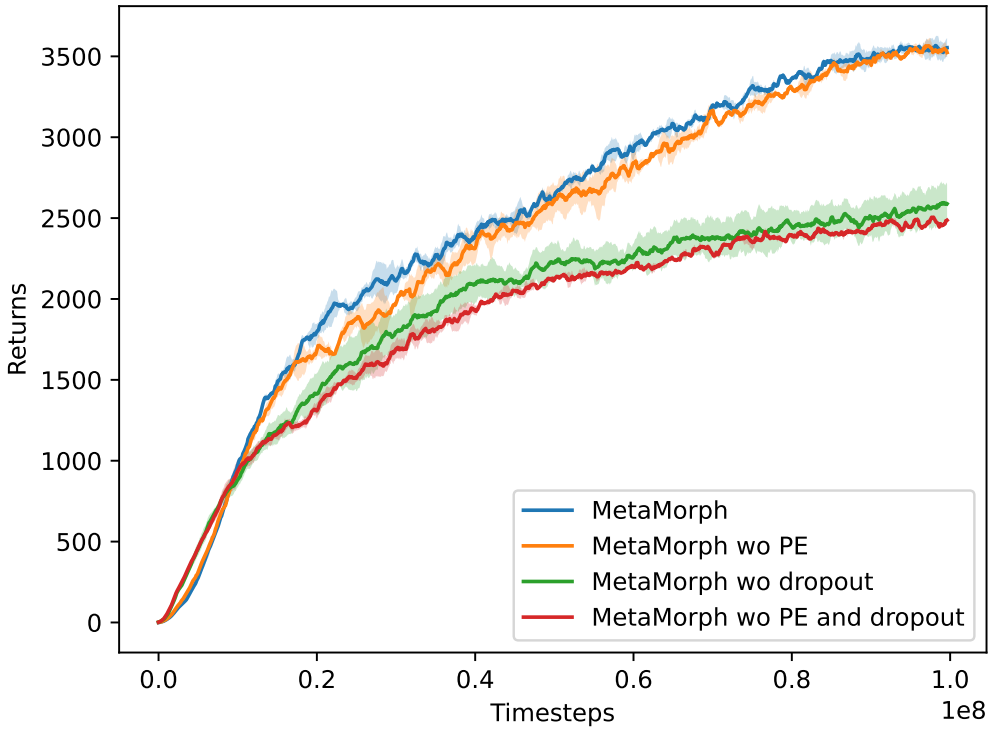

Figure A.1: The effect of PE and dropout on the performance of MetaMorph in the FT environment. The shaded area in each curve represents the standard deviation across three random seeds.

Based on the above analysis, we conclude that the PE implementation in MetaMorph is not essential for good performance, thus decide to not include it in MetaMorph*.

A.3.2 Why Dropout Helps?

It's quite surprising that removing the dropout operation causes such a significant performance drop in MetaMorph, as dropout is not believed to be a very useful regularizer for on-policy RL algorithms [Liu et al., 2021c]. Furthermore, the dropout operation is implemented in an inconsistent way in MetaMorph, which introduces significant noise to the action probability ratio $r = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$. Specifically, if a dropout mask m is applied to a state s during data collection, then the same mask should be used when s is sampled during policy update, i.e., $r = \frac{\pi_\theta(a|s;m)}{\pi_{\theta_k}(a|s;m)}$, to maintain a consistent ratio computation. However, in the MetaMorph code, a different

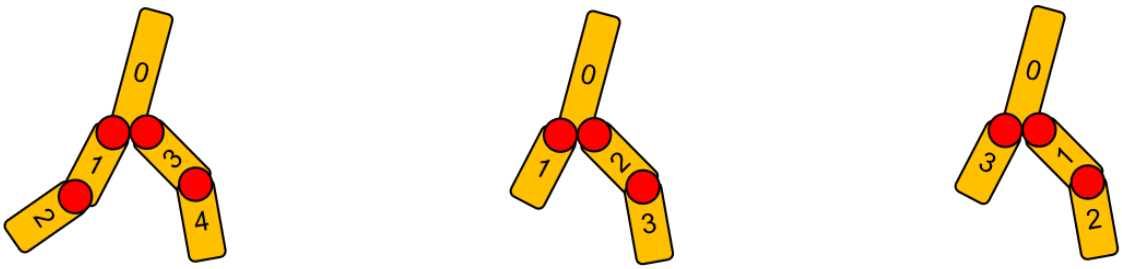

Figure A.2: Two sources of inconsistency in PE across morphologies. (1) The nodes that play different roles in different morphologies may share the same PE, such as the two nodes indexed by 2 in the left and middle robot. (2) There is no intrinsic order between the children of a parent node in the robot morphology, so PE is sensitive to how we choose which child node to expand first, such as the middle and right robot which have the same morphology but totally different PE for each non-root node.

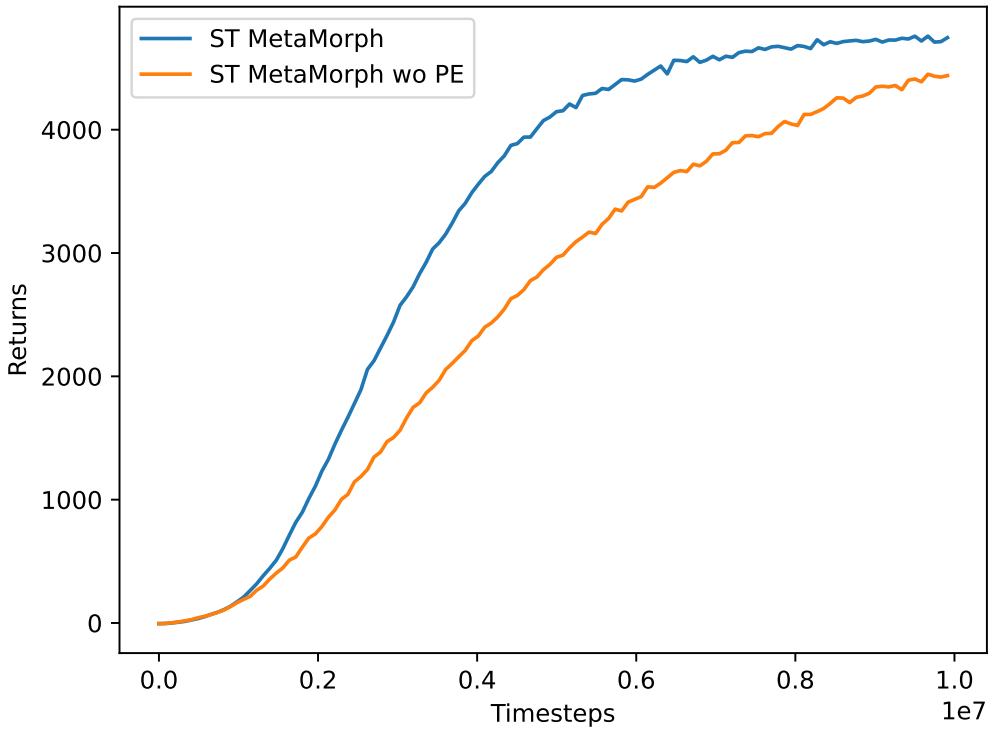

Figure A.3: The effect of PE on single-task MetaMorph in the FT environment.

dropout mask m' is randomly sampled whenever s is used for policy update, i.e., $r' = \frac{\pi_\theta(a|s;m')}{\pi_{\theta_k}(a|s;m)}$, which introduces significant noise. For example, before the first minibatch update in a PPO iteration, we expect r to be 1 for each state-action pair in the minibatch, as $\pi_\theta = \pi_{\theta_k}$ before policy update. However, using inconsistent dropout masks will make r unequal to 1 even before any policy update, which

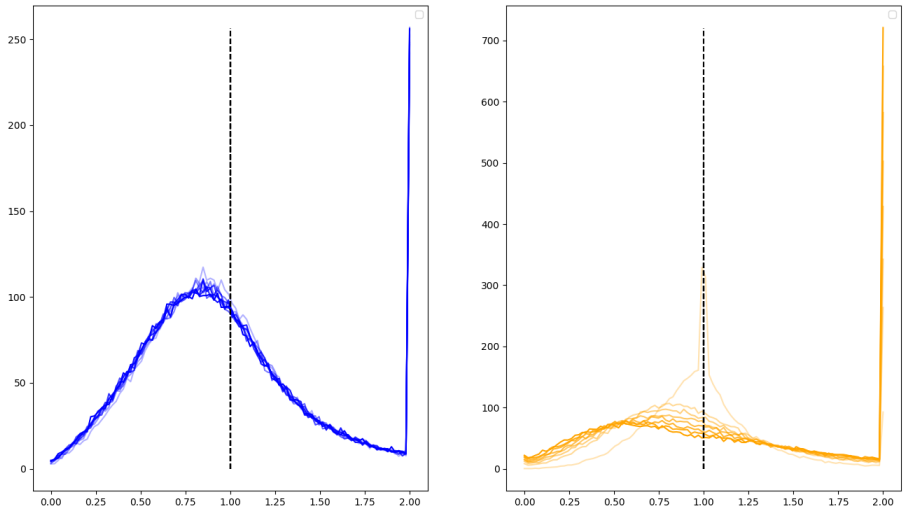

Figure A.4: Ratio distribution of each epoch during one PPO update iteration. Left: MetaMorph; Right: MetaMorph without dropout. Lighter color represents earlier epoch during one update iteration. There is a spike in the right of each subplot because we clip the ratios to be within $[0, 2]$.

does not make sense intuitively.

We thus look deeper into the training statistics of PPO to better understand why this inconsistent dropout operation works, and notice that using dropout or not causes a significant difference in the distribution shift of r during the learning process. Figure A.4 shows how the distribution of r changes on each epoch during one PPO update iteration. We can see that the inconsistent dropout operation somehow maintains the distribution stable across epochs, while the distribution significantly changes if learning without dropout. This implies that when learning without dropout, the policy has very likely exceeded the trust region and thus performs worse.

A natural solution to restricting the distribution shift is the early stopping method proposed in Appendix A.2. MetaMorph actually already uses early stopping in its code, but the threshold is set to 0.2, which is too large and in practice seldom triggers early stopping. We set it to a smaller value of 0.03 or 0.05, so that we can achieve similar performance as MetaMorph without using the inconsistent dropout operation, which is the second modification we make in MetaMorph*.

B

Appendix for Chapter 7, HyperDistill

Contents

B.1 Morphology Context Features and Transformations

For each robot, its morphology context includes the topology graph of the robot and limb-wise context features. Limb-wise context features include: (1) The initially relative position of the limb w.r.t. its parent node; (2) The initially geometric orientation of the limb w.r.t. its parent node; (3) The mass and shape parameters of the limb; and (4) Parameters about the joints that connect the limb to its parent node, including joint type, joint range and axis, and motor gear [Gupta et al., 2022].

As discussed in Section 7.3.1, the context features need to be distinguishable enough to learn a good morphology-conditioned HN. Among the original context features, we find that the relative position feature does not provide sufficient

discrimination, as different limbs, especially symmetric ones within a robot, can have the same relative position to their parents, but actually locate in different parts of the robot and play different roles. To solve this issue, we transform relative position features into absolute position features to better distinguish different limbs' positions in the morphology, which can be easily computed by following the path from the torso (the root of the morphology tree) to each limb node. Experimental results in Figure 7.7 validate the importance of this feature transformation.

B.2 Further Experimental Setup

B.2.1 Generation of PD Robots

We mutate the 100 training robots in the UNIMAL benchmark to get an augmented set of 1000 PD robots, on which we collect expert data for policy distillation. Specifically, for each training robot, we first uniformly sample a number m from $[1, 2, 3]$, then sequentially apply m mutation steps to the robot to get a new morphology. See Gupta et al. [2021] for the set of feasible mutations allowed in the UNIMAL benchmark. We generate 9 variants for each training robot, yielding an augmented set of 1000 PD robots in total.

B.2.2 FLOPs Computation

We compute the FLOPs of a linear layer with input dimension M and output dimension N as $2 \times M \times N$ [Hobbhahn and Sevilla, 2021]. Then the FLOPs of an MLP or a TF can be analytically computed by recursively decomposing its FLOPs into the summation of basic linear layers' FLOPs. We omit the FLOPs of other operations like activation functions, layer normalization, etc., as these operations only have linear complexity, which is negligible compared to the quadratic complexity of linear layers.

B.2.3 Architecture Details of Different Methods

Table B.1 shows the size of different architectures in each environment. For ModuMorph (teacher) and ModuMorph (oracle), we use the same architecture

hyperparameters as in ModuMorph. For the base MLP in HyperDistill, we fix the hidden layer size to 256, and do a grid search over hidden layer number to find the smallest base MLP that can achieve on-par performance with the teacher policy in each environment. Then we set multi-robot MLP's architecture identical to the base MLP in HyperDistill. For ModuMorph (compressed) and TF (compressed), we tune the number of attention layers, the number of attention heads, and the dimension of the linear layer's hidden dimension inside the attention module, so that the compressed model has a similar number of parameters as HyperDistill's base MLP. The token embedding dimension is 128 for all different TF architectures.

Environment	Base MLP in HyperDistill & Multi-robot MLP		ModuMorph (teacher) & ModuMorph (oracle)			ModuMorph (compressed)			TF (compressed)		
	# Layer	Hidden size	# Layer	Head	Hidden dim	# Layer	Head	Hidden dim	# Layer	Head	Hidden dim
FT	2	256	5	2	1024	1	1	128	1	1	256
VT	3	256	5	2	1024	1	1	128	2	1	128
Obstacle	3	256	5	2	1024	1	1	128	2	1	128

Table B.1: The size of different models in each environment.

B.3 Further Analysis on Inference Efficiency

As shown in Table 7.1, HyperDistill's efficiency advantage is more significant in the FT environment, as in the other two environments, there is an additional high-dimensional terrain information input to the policy, which needs to be processed by a large MLP encoder. This adds a large constant to the model size and FLOPs of all methods, and thus reduces the efficiency ratio of HyperDistill to the other methods.

Although multi-robot MLP and the base network of HyperDistill have the same number of hidden layers and hidden units, the model size and FLOPs of multi-robot MLP are still a bit larger than those of HyperDistill, as it needs to condition on the morphology context by concatenating limb-wise context features to the policy input, which introduces some additional cost.

In HyperDistill, generating the base MLP with HN takes 43M FLOPs in the FT environment, and 65M FLOPs in the VT and Obstacle environment. The FLOPs of generating the base MLP with HN is just slightly larger than the FLOPs of a single inference step with a universal TF controller.

C

Appendix for Chapter 8, HyperVLA

Contents

C.1 Further Ablation Studies

We report further ablation results in Table C.1 and C.2 to validate the importance of the following design choices in HyperVLA. To reduce computational cost, we run each ablation experiment with only one seed, while the performance gap is significant enough to draw conclusions with high confidence.

Smaller Learning Rate for DINOv2 Fine-Tuning To ablate the importance of fine-tuning DINOv2 with a smaller learning rate as introduced in Section 8.2.3, we increase the learning rate for DINOv2 by 10 times, and use the same learning rate of 0.0003 for both HN training and DINOv2 fine-tuning. This variant performs poorly, which validates the importance of fine-tuning DINOv2 with a smaller learning rate to maintain its strong prior knowledge.

Fine-tuning versus Freezing DINOv2 To validate the importance of fine-tuning DINOv2 in the base network, we ablate by freezing it while keeping the remaining settings unchanged. This variant underperforms HyperVLA, which validates the importance of fine-tuning DINOv2 during HyperVLA pretraining.

Choice of the Image Encoder As introduced in Section 8.2.3, HyperVLA can support different image encoders, and empirically we find DINOv2 to perform best. We ablate by using either CLIP [Radford et al., 2021] or SigLIP [Zhai et al., 2023] as the image encoder in the base network. As our code is implemented in JAX and we can only find a PyTorch version of SigLIP, our code does not support fine-tuning SigLIP during training. The ablation results show that using fine-tuned DINOv2 outperforms fine-tuned CLIP, while frozen DINOv2 outperforms frozen SigLIP.

Importance of the HN We run this ablation experiment to validate that inference acceleration can not be achieved by training a small base network alone, and the HN in our method is essential to maintain high model capacity and achieve good performance. We experiment by removing the HN in HyperVLA and train the base network alone. This variant performs poorly, validating the importance of using HN.

	pick	move	close drawer	Avg
HyperVLA (Full)	58 ± 3	73 ± 1	58 ± 7	63 ± 3
Larger learning rate for DINOv2	3	13	17	11
Frozen DINOv2	56	47	58	54
Fine-tuned CLIP	58	61	59	59
Frozen SigLIP	20	34	49	34
Train base net alone	5	11	12	9

Table C.1: Ablation results on how different algorithm designs in HyperVLA influence its performance on Google Robot tasks.

	spoon on towel	carrot on plate	stack cube	eggplant in basket	Avg
HyperVLA (Full)	48 ± 3	21 ± 5	39 ± 8	52 ± 13	40 ± 5
Larger learning rate for DINOv2	0	0	0	0	0
Frozen DINOv2	18	40	0	3	15
Fine-tuned CLIP	63	30	13	0	27
Frozen SigLIP	3	0	0	0	1
Train base net alone	3	0	0	0	1

Table C.2: Ablation results on how different algorithm designs in HyperVLA influence its performance on WidowX tasks.

Bibliography

Sayanta Adhikari, Dupati Srikar Chandra, PK Srijith, Pankaj Wasnik, and Naoyuki Oneo. Adaprefix++: Integrating adapters, prefixes and hypernetwork for continual learning. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 7298–7307. IEEE, 2025.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Sayantan Auddy, Sebastian Bergner, and Justus Piater. Effect of optimizer, initializer, and architecture of hypernetworks on continual learning from demonstration. In *European Robotics Forum*, pages 315–320. Springer, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

David Barber and Felix Agakov. The im algorithm: a variational approach to information maximization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'03, page 201–208, Cambridge, MA, USA, 2003. MIT Press.

Alison L Barth and James FA Poulet. Experimental evidence for sparse firing in the neocortex. *Trends in neurosciences*, 35(6):345–355, 2012.

Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1478–1487. PMLR, 2023a. URL https://proceedings.mlr.press/v205/beck23a.html.

Jacob Beck, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Recurrent hypernetworks are surprisingly strong in meta-rl. *Advances in Neural Information Processing Systems*, 36:62121–62138, 2023b.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A tutorial on meta-reinforcement learning. *Foundations and Trends® in Machine Learning*, 18(2-3):224–384, 2025.

Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint. *URL https://github. com/Stanford-ILIAD/openvla-mini*, 2024.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. In *Robotics: Science and Systems*, 2024.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Álvaro Belmonte-Baeza, Joonho Lee, Giorgio Valsecchi, and Marco Hutter. Meta reinforcement learning for optimal design of legged robots. *IEEE Robotics and Automation Letters*, 7(4):12134–12141, 2022.

Eseoghene Ben-Iwhiwhu, Jeffery Dick, Nicholas A Ketz, Praveen K Pilly, and Andrea Soltoggio. Context meta-reinforcement learning via neuromodulation. *Neural Networks*, 152:70–79, 2022.

Carolin Benjamins, Theresa Eimer, Frederik Schubert, Aditya Mohan, Sebastian Döhler, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. Contextualize me – the case for context in reinforcement learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=Y42xVBQusn.

Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. In *1st Workshop on X-Embodiment Robot Learning*, 2024. URL https://openreview.net/forum?id=59U5Sl6nyF.

Hongzhe Bi, Lingxuan Wu, Tianwei Lin, Hengkai Tan, Zhizhong Su, Hang Su, and Jun Zhu. H-rdt: Human manipulation enhanced bimanual robotic manipulation. *arXiv preprint arXiv:2507.23523*, 2025a.

Jianxin Bi, Kevin Yuchen Ma, Ce Hao, Mike Zheng Shou, and Harold Soh. Vla-touch: Enhancing vision-language-action models with dual-level tactile feedback, 2025b. URL https://arxiv.org/abs/2507.17294.

Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414, 2019.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Rich Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=c0chJTSbci.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025a. URL https://openreview.net/forum?id=vlhoswksBO.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Nico Bohlinger, Grzegorz Czechmanowski, Maciej Piotr Krupka, Piotr Kicki, Krzysztof Walas, Jan Peters, and Davide Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 3356–3378. PMLR, 2025. URL https://proceedings.mlr.press/v270/bohlinger25a.html.

David Bonet, Daniel Mas Montserrat, Xavier Giró-i Nieto, and Alexander G. Ioannidis. Hyperfast: instant classification for tabular data. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i10.28988. URL https://doi.org/10.1609/aaai.v38i10.28988.

Andrew Brock, Theo Lim, J.M. Ritchie, and Nick Weston. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rydeCEhs-.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023. doi: 10.15607/RSS.2023.XIX.025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4603–4623. PMLR, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, Shu Jiang, Yuxin Jiang, Cheng Jing, Hongyang Li, Jialu Li, Chiming Liu, Yi Liu, Yuxiang Lu, Jianlan Luo, Ping Luo, Yao Mu, Yuehan Niu, Yixuan Pan, Jiangmiao Pang, Yu Qiao, Guanghui Ren, Cheng Ruan, Jiaqi Shan, Yongjian Shen, Chengshi Shi, Mingkang Shi, Modi Shi, Chonghao Sima, Jianheng Song, Huijie Wang, Wenhao Wang, Dafeng Wei, Chengen Xie, Guo Xu, Junchi Yan, Cunbiao Yang, Lei Yang, Shukai Yang, Maoqing Yao, Jia Zeng, Chi Zhang, Qinglin Zhang, Bin Zhao, Chengyue Zhao, Jiaqi Zhao, and Jianchao Zhu. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems, 2025a. URL https://arxiv.org/abs/2503.06669.

Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation, 2025b. URL https://arxiv.org/abs/2410.08001.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to Act Anywhere with Task-centric Latent Actions. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, 2025c. doi: 10.15607/RSS.2025.XXI.014.

Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150464. URL https://doi.org/10.1145/1150402.1150464.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lma24tPB.

Vinod Kumar Chauhan, Jiandong Zhou, Ghadeer Ghosheh, Soheila Molaei, and David A Clifton. Dynamic inter-treatment information sharing for individualized treatment effects estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 3529–3537. PMLR, 2024a.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):250, 2024b.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation, 2024. URL https://arxiv.org/abs/2410.06158.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. In S. Bengio, H. Wallach,

H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/b8cfbf77a3d250a4523ba67a65a7d031-Paper.pdf.

Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 297–307. PMLR, 2022. URL https://proceedings.mlr.press/v164/chen22a.html.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, Siamak Shakeri, Mostafa Dehghani, Daniel Salz, Mario Lucic, Michael Tschannen, Arsha Nagrani, Hexiang Hu, Mandar Joshi, Bo Pang, Ceslee Montgomery, Paulina Pietrzyk, Marvin Ritter, AJ Piergiovanni, Matthias Minderer, Filip Pavetic, Austin Waters, Gang Li, Ibrahim Alabdulmohsin, Lucas Beyer, Julien Amelot, Kenton Lee, Andreas Peter Steiner, Yang Li, Daniel Keysers, Anurag Arnab, Yuanzhong Xu, Keran Rong, Alexander Kolesnikov, Mojtaba Seyedhosseini, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. On scaling up a multilingual vision and language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14432–14444, 2024.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. ConRFT: A Reinforced Fine-tuning Method for VLA Models via Consistency Policy. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, 2025. doi: 10.15607/RSS.2025.XXI.019.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023. doi: 10.15607/RSS.2023.XIX.026.

Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyztsoC5Y7.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 2019. URL https://proceedings.mlr.press/v97/cobbe19a.html.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms. In *ICLR Worskhop on Reproducibility*, 2019.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153:235–256, 2007.

Can Cui, Pengxiang Ding, Wenxuan Song, Shuanghao Bai, Xinyang Tong, Zirui Ge, Runze Suo, Wanqi Zhou, Yang Liu, Bofang Jia, Han Zhao, Siteng Huang, and Donglin Wang. Openhelix: A short survey, empirical analysis, and open-source dual-system vla model for robotic manipulation, 2025. URL https://arxiv.org/abs/2505.03912.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd international conference on artificial intelligence and statistics*, pages 1331–1340. PMLR, 2019.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, 130(1):33–55, 2022.

Shengliang Deng, Mi Yan, Songlin Wei, Haixin Ma, Yuxin Yang, Jiayi Chen, Zhiqi Zhang, Taoyu Yang, Xuheng Zhang, Heming Cui, Zhizheng Zhang, and He Wang. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 1004–1029. PMLR, 2025. URL https://proceedings.mlr.press/v305/deng25a.html.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13049–13061. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/985e9a46e10005356bbaf194249f6856-Paper.pdf.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *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, 2019.

Jacopo Di Ventura, Dylan R Ashley, Vincent Herrmann, Francesco Faccio, and Jürgen Schmidhuber. Upside down reinforcement learning with policy generators. *arXiv preprint arXiv:2501.16288*, 2025.

Muhayy Ud Din, Waseem Akram, Lyes Saad Saoud, Jan Rosell, and Irfan Hussain. Vision language action models in robotic manipulation: A systematic review, 2025. URL https://arxiv.org/abs/2507.10672.

Heng Dong, Tonghan Wang, Jiayuan Liu, and Chongjie Zhang. Low-rank modular reinforcement learning via muscle synergy. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 19861–19873. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/7da6005a8d6942e8b328357da2872aed-Paper-Conference.pdf.

Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=AuJnXGq3AL.

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.

Rousslan Fernand Julien Dossa, Shengyi Huang, Santiago Ontañón, and Takashi Matsubara. An empirical investigation of early stopping optimizations in proximal policy optimization. *IEEE Access*, 9:117981–117992, 2021.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-e: An embodied multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 2023. URL https://proceedings.mlr.press/v202/driess23a.html.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

Laurens Engwegen, Daan Brinks, and Wendelin Böhmer. Modular recurrence in contextual mdps for universal morphology control, 2025. URL https://arxiv.org/abs/2506.08630.

Francesco Faccio, Vincent Herrmann, Aditya Ramesh, Louis Kirsch, and Jürgen Schmidhuber. Goal-conditioned generators of deep policies. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i6.25912. URL https://doi.org/10.1609/aaai.v37i6.25912.

Gilbert Feng, Hongbo Zhang, Zhongyu Li, Xue Bin Peng, Bhuvan Basireddy, Linzhu Yue, ZHITAO SONG, Lizhi Yang, Yunhui Liu, Koushil Sreenath, and Sergey Levine. Genloco: Generalized locomotion controllers for quadrupedal robots. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1893–1903. PMLR, 2023. URL https://proceedings.mlr.press/v205/feng23a.html.

Arnaud Fickinger, Samuel Cohen, Stuart Russell, and Brandon Amos. Cross-domain imitation learning via optimal transport. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xP3cPq2hQC.

AI Figure. Helix: A vision-language-action model for generalist humanoid control. *Figure AI News*, 2024.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation using low-cost whole-body teleoperation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=FO6tePGRZj.

Hiroki Furuta, Yusuke Iwasawa, Yutaka Matsuo, and Shixiang Shane Gu. A system for morphology-task generalization via unified representation and behavior distillation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=HcUf-QwZeFh.

Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10409–10419. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/75c58d36157505a600e0695ed0b3a22d-Paper.pdf.

Chongkai Gao, Haozhuo Zhang, Zhixuan Xu, Cai Zhehao, and Lin Shao. FLIP: Flow-centric generative planning as general-purpose manipulation world model. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=B2N0nCVC91.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.090.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

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.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012, 2022.

Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19383–19400, 2024.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b_CQDy9vrD1.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. RT-trajectory: Robotic task generalization via hindsight trajectory sketches. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F1TKzG8LJO.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning, 2025. URL https://arxiv.org/abs/2501.16664.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hyq4yhile.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721, 2021.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Opmqtk_GvYL.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015. URL https://arxiv.org/abs/1502.02259.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

ByungOk Han, Jaehong Kim, and Jinhyeok Jang. A dual process vla: Efficient robotic manipulation leveraging vlm, 2024. URL https://arxiv.org/abs/2410.15549.

Peng Hao, Chaofan Zhang, Dingzhe Li, Xiaoge Cao, Xiaoshuai Hao, Shaowei Cui, and Shuo Wang. Tla: Tactile-language-action model for contact-rich manipulation, 2025. URL https://arxiv.org/abs/2503.08548.

YiFan Hao, Yang Yang, Junru Song, Wei Peng, Weien Zhou, Tingsong Jiang, and Wen Yao. Heteromorpheus: Universal control based on morphological heterogeneity modeling. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2024.

Zhiwei Hao, Jianyuan Guo, Kai Han, Yehui Tang, Han Hu, Yunhe Wang, and Chang Xu. One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 79570–79582. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fb8e5f198c7a5dcd48860354e38c0edc-Paper-Conference.pdf.

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.

Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, Yaguang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. HyperPrompt: Prompt-based task-conditioning of transformers. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8678–8690. PMLR, 2022. URL https://proceedings.mlr.press/v162/he22f.html.

Donald Hejna, Lerrel Pinto, and Pieter Abbeel. Hierarchically decoupled imitation for morphological transfer. 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 4159–4171. PMLR, 2020. URL https://proceedings.mlr.press/v119/hejna20a.html.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Marius Hobbhahn and Jaime Sevilla. What's the backward-forward flop ratio for neural networks?, 2021. URL https://epochai.org/blog/backward-forward-FLOP-ratio. Accessed: 2024-02-01.

Sunghoon Hong, Deunsol Yoon, and Kee-Eung Kim. Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fy_XRVHqly.

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 Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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.

Jiaheng Hu, Rose Hendrix, Ali Farhadi, Aniruddha Kembhavi, Roberto Martín-Martín, Peter Stone, Kuo-Hao Zeng, and Kiana Ehsani. Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3617–3624, 2025a. doi: 10.1109/ICRA55743.2025.11127934.

Yang Hu and Giovanni Montana. Skill transfer in deep reinforcement learning under morphological heterogeneity, 2019. URL https://arxiv.org/abs/1908.05265.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=c0dhw1du33.

Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-vla: Unlocking vision-language-action model's physical knowledge for tactile generalization, 2025a. URL https://arxiv.org/abs/2507.09160.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. 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 4455–4464. PMLR, 2020. URL https://proceedings.mlr.press/v119/huang20d.html.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 540–562. PMLR, 2023a. URL https://proceedings.mlr.press/v229/huang23b.html.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR, 2023b. URL https://proceedings.mlr.press/v205/huang23c.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 4573–4602. PMLR, 2025b. URL https://proceedings.mlr.press/v270/huang25g.html.

Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 799–805. IEEE, 2021.

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 38–44. IEEE, 2016.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Qun8fv4qSby.

Donald Joseph Hejna III, Pieter Abbeel, and Lerrel Pinto. Task-agnostic morphology evolution. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=CGQ6ENUMX6.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

Mathias Jackermeier and Alessandro Abate. DeepLTL: Learning to efficiently satisfy complex LTL specifications for multi-task RL. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9pW2J49flQ.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative interactions and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rylnK6VtDH.

Jingtian Ji, Buqing Nie, and Yue Gao. Daga: Dynamics aware reinforcement learning with graph-based rapid adaptation. *IEEE Robotics and Automation Letters*, 8(4):2189–2196, 2023.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *CoRL Workshop on Learning Robot Fine and Dexterous Manipulation: Perception and Control*, 2024.

Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

Dmitry Kalashnikov, Jake Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Scaling up multi-task robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 557–575. PMLR, 2022. URL https://proceedings.mlr.press/v164/kalashnikov22a.html.

Kento Kawaharazuka, Jihoon Oh, Jun Yamada, Ingmar Posner, and Yuke Zhu. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 13:162467–162504, 2025. doi: 10.1109/ACCESS.2025.3609980.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.120.

Kuno Kim, Yihong Gu, Jiaming Song, Shengjia Zhao, and Stefano Ermon. Domain adaptive imitation learning. 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 5286–5295. PMLR, 2020. URL https://proceedings.mlr.press/v119/kim20c.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ZMnD6QZAE6.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

Yeachan Kim, Junho Kim, Wing-Lam Mok, Jun-Hyung Park, and SangKeun Lee. Client-customized adaptation for parameter-efficient federated learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1159–1172, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, 22(30):1–82, 2021.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *Proceedings of Robotics: Science and Systems*, Virtual, 2021. doi: 10.15607/RSS.2021.XVII.011.

Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=N3zUDGN5lO.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and Pawan K Mudigonda. In defense of the unitary scalarization for deep multi-task learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 12169–12183. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/4f301ae934f396086bfefd1139039dbd-Paper-Conference.pdf.

Isabel Leal, Krzysztof Choromanski, Deepali Jain, Avinava Dubey, Jake Varley, Michael Ryoo, Yao Lu, Frederick Liu, Vikas Sindhwani, Quan Vuong, Tamas Sarlos, Ken Oslund, Karol Hausman, and Kanishka Rao. Sara-rt: Scaling up robotics transformers with self-adaptive robust attention. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6920–6927, 2024. doi: 10.1109/ICRA57147.2024.10611597.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

Boyu Li, Haoran Li, Yuanheng Zhu, and Dongbin Zhao. Mat: Morphological adaptive transformer for universal morphology policy learning. *IEEE Transactions on Cognitive and Developmental Systems*, 16(4):1611–1621, 2024a.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024b. URL https://arxiv.org/abs/2411.19650.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 3705–3728. PMLR, 2025a. URL https://proceedings.mlr.press/v270/li25c.html.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=h7aQxzKbq6.

Yichao Liang, Kevin Ellis, and Joao Henriques. Rapid motor adaptation for robotic manipulator arms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16404–16413, 2024a.

Yongyuan Liang, Tingqiang Xu, Kaizhe Hu, Guangqi Jiang, Furong Huang, and Huazhe Xu. Make-an-agent: A generalizable policy network generator with behavior-prompted diffusion. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 19288–19306. Curran Associates, Inc., 2024b. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/223eb69a2f2fb97fde58eaa958babb7a-Paper-Conference.pdf.

Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2488–2494. IEEE, 2019.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pISLZG7ktL.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in*

Neural Information Processing Systems, volume 34, pages 18878–18890. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9d27fdf2477ffbff837d73ef7ae23db9-Paper.pdf.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International conference on machine learning*, pages 6925–6935. PMLR, 2021b.

Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Grounding complex natural language commands for temporal tasks in unseen environments. In *Conference on Robot Learning*, pages 1084–1110. PMLR, 2023b.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study, 2025a. URL https://arxiv.org/abs/2505.19789.

Min Liu, Deepak Pathak, and Ananye Agarwal. Locoformer: Generalist locomotion via long-context adaptation. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 532–546. PMLR, 2025b. URL https://proceedings.mlr.press/v305/liu25a.html.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1b: a diffusion foundation model for bimanual manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=yAzN4tz7oI.

Xingyu Liu, Deepak Pathak, and Kris Kitani. REvolveR: Continuous evolutionary models for robot-to-robot policy transfer. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13995–14007. PMLR, 2022. URL https://proceedings.mlr.press/v162/liu22p.html.

Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. Regularization matters in policy optimization - an empirical study on continuous control. In *International Conference on Learning Representations*, 2021c. URL https://openreview.net/forum?id=yr1mzrH3IC.

Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. TAIL: Task-specific adapters for imitation learning with large pretrained models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023c. URL https://openreview.net/forum?id=jnzjnYTSlO.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18719.

Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pages 854–869. PMLR, 2020.

Yingbo Luo, Meibao Yao, and Xueming Xiao. Gcnt: Graph-based transformer policies for morphology-agnostic reinforcement learning, 2025. URL https://arxiv.org/abs/2505.15211.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 23301–23320. PMLR, 2023a. URL https://proceedings.mlr.press/v202/ma23b.html.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=YJ7o2wetJ2.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai, 2025. URL https://arxiv.org/abs/2405.14093.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2021.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1820–1864. PMLR, 2023. URL https://proceedings.mlr.press/v229/mandlekar23a.html.

Gabriel B. Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 2023. URL https://proceedings.mlr.press/v205/margolis23a.html.

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.

Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Dieter Fox, and Abhishek Gupta. ASID: Active exploration for system identification in robotic manipulation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jNR6s6OSBT.

Russell Mendonca, Xinyang Geng, Chelsea Finn, and Sergey Levine. Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling. *arXiv preprint arXiv:2006.07178*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 2016. PMLR. URL https://proceedings.mlr.press/v48/mniha16.html.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 25081–25094. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4ec43957eda1126ad4887995d05fae3b-Paper-Conference.pdf.

Andreas C Mueller, Carlo A Curino, and Raghu Ramakrishnan. Mothernet: Fast training and inference via hyper-network transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6H4jRWKFc3.

Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation Letters*, 6(2):911–918, 2021.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 892–909. PMLR, 2023. URL https://proceedings.mlr.press/v205/nair23a.html.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: a framework and survey. *J. Mach. Learn. Res.*, 21(1), 2020. ISSN 1532-4435.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *RSS 2024 Workshop: Data Generation for Robotics*, 2024.

Aviv Navon, Aviv Shamsian, Ethan Fetaya, and Gal Chechik. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NjF772F4ZZR.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

Fei Ni, Jianye Hao, Shiguang Wu, Longxin Kou, Jiashun Liu, Yan Zheng, Bin Wang, and Yuzheng Zhuang. Generate subgoal images before act: Unlocking the chain-of-thought reasoning in diffusion model for robot manipulation with multimodal prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13991–14000, 2024.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Ji" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019. URL https://arxiv.org/abs/1910.07113.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey,

Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu,

Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

Jiayi Pan, Glen Chou, and Dmitry Berenson. Data-efficient learning of natural language to linear temporal logic translators for robot task specification. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11554–11561. IEEE, 2023.

Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao

Dong. Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17359–17369, 2025.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: A study of generalization via modularity. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c26820b8a4c1b3c2aa868d6d57e14a79-Paper.pdf.

Matt Peng, Banghua Zhu, and Jiantao Jiao. Linear representation meta-reinforcement learning for instant adaptation, 2021. URL https://arxiv.org/abs/2101.04750.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Edoardo M Ponti, Alessandro Sordoni, and Siva Reddy. Combining modular skills in multitask learning. *arXiv preprint arXiv:2202.13914*, 2022.

Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1722–1732. PMLR, 2023. URL https://proceedings.mlr.press/v205/qi23a.html.

Wenjie Qiu, Wensen Mao, and He Zhu. Instructing goal-conditioned reinforcement learning agents with temporal logic objectives. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 39147–39175. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/7b35a69f434b5eb07ed1b1ef16ace52c-Paper-Conference.pdf.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

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.

Neale Ratzlaff and Li Fuxin. HyperGAN: A generative model for diverse, performant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5361–5369. PMLR, 2019. URL https://proceedings.mlr.press/v97/ratzlaff19a.html.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=1ikK0kHjvj. Featured Certification, Outstanding Certification.

Sahand Rezaei-Shoshtari, Charlotte Morissette, Francois R. Hogan, Gregory Dudek, and David Meger. Hypernetworks for zero-shot transfer in reinforcement learning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i8.26146. URL https://doi.org/10.1609/aaai.v37i8.26146.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8583–8595. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJgklhAcK7.

Elad Sarafian, Shai Keynan, and Sarit Kraus. Recomposing the reinforcement learning building blocks with hypernetworks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9301–9312. PMLR, 2021. URL https://proceedings.mlr.press/v139/sarafian21a.html.

Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805. IEEE, 2019.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

Gerrit Schoettler, Ashvin Nair, Juan Aparicio Ojea, Sergey Levine, and Eugen Solowjow. Meta-reinforcement learning for robotic industrial insertion tasks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9728–9735. IEEE, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 2015a. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Marcin Sendera, Marcin Przewięźlikowski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Hypershot: Few-shot learning by kernel hypernetworks. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2469–2478, 2023.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I*, page 362–373, Berlin, Heidelberg, 2018. Springer-Verlag. ISBN 978-3-030-04166-3. doi: 10.1007/978-3-030-04167-0_33. URL https://doi.org/10.1007/978-3-030-04167-0_33.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=agTr-vRQsa.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9489–9502. PMLR, 2021. URL https://proceedings.mlr.press/v139/shamsian21a.html.

Tanmay Shankar, Yixin Lin, Aravind Rajeswaran, Vikash Kumar, Stuart Anderson, and Jean Oh. Translating robot skills: Learning unsupervised skill correspondences across robots. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19626–19644. PMLR, 2022. URL https://proceedings.mlr.press/v162/shankar22a.html.

Sheng Shen, Zhewei Yao, Chunyuan Li, Trevor Darrell, Kurt Keutzer, and Yuxiong He. Scaling vision-language models with sparse mixture of experts. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=IpJ5rAFLv7.

Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From llms to actions: latent codes as bridges in hierarchical robot control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8539–8546. IEEE, 2024.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025a.

Lucy Xiaoyang Shi, brian ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=lNVHg9npif.

Modi Shi, Li Chen, Jin Chen, Yuxiang Lu, Chiming Liu, Guanghui Ren, Ping Luo, Di Huang, Maoqing Yao, and Hongyang Li. Is diversity all you need for scalable robotic manipulation?, 2025c. URL https://arxiv.org/abs/2507.06219.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Harshit Sikchi, Siddhant Agarwal, Pranaya Jajoo, Samyak Parajuli, Caleb Chuck, Max Rudolph, Peter Stone, Amy Zhang, and Scott Niekum. Rlzero: Direct policy inference from language without in-domain supervision. *arXiv preprint arXiv:2412.05718*, 2024.

Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, 1994.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9767–9779. PMLR, 2021. URL https://proceedings.mlr.press/v139/sodhani21a.html.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Jun Ma, and Haoang Li. Accelerating vision-language-action model integrated with action chunking via parallel decoding, 2025. URL https://arxiv.org/abs/2503.02310.

Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 387–414. Springer, 2012.

Nicolas Stifani. Motor neurons and the generation of spinal motor neuron diversity. *Frontiers in cellular neuroscience*, 8:293, 2014.

Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Sean Kirmani, Brianna Zitkovich, Fei Xia, Chelsea Finn, and Karol Hausman. Open-world object manipulation using pre-trained vision-language models. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 3397–3417. PMLR, 2023. URL https://proceedings.mlr.press/v229/stone23a.html.

Mingfei Sun, Vitaly Kurin, Guoqing Liu, Sam Devlin, Tao Qin, Katja Hofmann, and Shimon Whiteson. You may not need ratio clipping in ppo. *arXiv preprint arXiv:2202.00079*, 2022.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019. URL https://proceedings.mlr.press/v97/tan19a.html.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(56):1633–1685, 2009. URL http://jmlr.org/papers/v10/taylor09a.html.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Kale-ab Abebe Tessera, Arrasy Rahman, Amos Storkey, and Stefano V. Albrecht. Hypermarl: Adaptive hypernetworks for multi-agent rl, 2025. URL https://arxiv.org/abs/2412.04233.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

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. URL https://arxiv.org/abs/2307.09288.

Brandon Trabucco, Mariano Phielipp, and Glen Berseth. AnyMorph: Learning transferable polices by inferring agent morphology. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 21677–21691. PMLR, 2022. URL https://proceedings.mlr.press/v162/trabucco22b.html.

Pashootan Vaezipoor, Andrew C Li, Rodrigo A Toro Icarte, and Sheila A. Mcilraith. Ltl2action: Generalizing ltl instructions for multi-task rl. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10497–10508. PMLR, 2021. URL https://proceedings.mlr.press/v139/vaezipoor21a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgwNerKvB.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1723–1736. PMLR, 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3891–3902, 2023.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 124420–124450. Curran Associates, Inc., 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e0f393e7980a24fd12fa6f15adfa25fb-Paper-Conference.pdf.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024b.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1sqHMZCb.

Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. Neural graph evolution: Automatic robot design. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkgWHnR5tm.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. In *Forty-first International Conference on Machine Learning*, 2024c. URL https://openreview.net/forum?id=SQIDlJd3hN.

Yuxing Wang, Shuang Wu, Haobo Fu, QIANG FU, Tiantian Zhang, Yongzhe Chang, and Xueqian Wang. Curriculum-based co-design of morphology and control of voxel-based soft robots. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=r9fX833CsuN.

Hayato Watahiki, Ryo Iwase, Ryosuke Unno, and Yoshimasa Tsuruoka. Learning a domain-agnostic policy through adversarial representation matching for cross-domain policy transfer. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=NxoFmGgWC9.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024b.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Youguang Xing, Xu Luo, Junlin Xie, Lianli Gao, Hengtao Shen, and Jingkuan Song. Shortcut learning in generalist robot policies: The role of dataset diversity and fragmentation, 2025. URL https://arxiv.org/abs/2508.06426.

Zheng Xiong, Luisa M Zintgraf, Jacob Austin Beck, Risto Vuorio, and Shimon Whiteson. On the practical consistency of meta-reinforcement learning algorithms. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=xwQgKphwhFA.

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

Zheng Xiong, Siddhant Sharma, Kang Li, Risto Vuorio, and Shimon Whiteson. Efficient domain adaptation of robotic foundation models via hypernetwork-generated loRA. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*, 2024a. URL https://openreview.net/forum?id=SEvvWs3CAL.

Zheng Xiong, Risto Vuorio, Jacob Beck, Matthieu Zimmer, Kun Shao, and Shimon Whiteson. Distilling morphology-conditioned hypernetworks for efficient universal morphology control. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=WjvEvYTy3w.

Zheng Xiong, Kang Li, Zilin Wang, Matthew Jackson, Jakob Foerster, and Shimon Whiteson. Hypervla: Efficient inference in vision-language-action models via hypernetworks, 2025. URL https://arxiv.org/abs/2510.04898.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic Generalist Policy Distillation via Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, 2025a. doi: 10.15607/RSS.2025.XXI.028.

Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2475–2499. PMLR, 2025b. URL https://proceedings.mlr.press/v270/xu25a.html.

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.

Jonathan Heewon Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the Limits of Cross-Embodiment Learning for Manipulation and Navigation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.093.

Ruihan Yang, Huazhe Xu, YI WU, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4767–4777. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/32cfdce9631d8c7906e8e9d6e68b514b-Paper.pdf.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VYOe2eBQeh.

Chen Yu, Weinan Zhang, Hang Lai, Zheng Tian, Laurent Kneip, and Jun Wang. Multi-embodiment legged robot control as a sequence modeling problem. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7250–7257. IEEE, 2023.

Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, Cewu Lu, and Wenqiang Zhang. Forcevla: Enhancing vla models with a force-aware moe for contact-rich manipulation, 2025. URL https://arxiv.org/abs/2505.22159.

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.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf.

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 Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 2020b. URL https://proceedings.mlr.press/v100/yu20a.html.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, 2017. doi: 10.15607/RSS.2017.XIII.048.

Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris M. Kitani. Transform2act: Learning a transform-and-control policy for efficient agent design. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=UcDUxjPYWSr.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *2nd Workshop on Dexterous Manipulation: Design, Perception and Control (RSS)*, 2024. URL https://openreview.net/forum?id=KwwJuZIBXH.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rkgW0oA9FX.

Hongyin Zhang, Pengxiang Ding, Shangke Lyu, Ying Peng, and Donglin Wang. GEVRM: Goal-expressive video generation model for robust visual manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=hPWWXpCaJ7.

Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. Hirt: Enhancing robotic control with hierarchical robot transformers. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 933–946. PMLR, 2025b. URL https://proceedings.mlr.press/v270/zhang25b.html.

Qiang Zhang, Tete Xiao, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Learning cross-domain correspondence for control with dynamics cycle-consistency. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QIRlze3I6hX.

Wenbo Zhang, Tianrun Hu, Yanyuan Qiao, Hanbo Zhang, Yuchu Qin, Yang Li, Jiajun Liu, Tao Kong, Lingqiao Liu, and Xiao Ma. Chain-of-action: Trajectory autoregressive modeling for robotic manipulation. *arXiv preprint arXiv:2506.09990*, 2025c.

Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020a.

Hao Zhao, Jie Fu, and Zhaofeng He. Prototype-based hyperadapter for sample-efficient multi-task tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4603–4615, 2023a.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023b. doi: 10.15607/RSS.2023.XIX.016.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020b.

Yue Zhao, Yue Gao, Qiao Sun, Yuan Tian, Liheng Mao, and Feng Gao. A real-time low-computation cost human-following framework in outdoor environment for legged robots. *Robotics and Autonomous Systems*, 146:103899, 2021a.

Zihao Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. Meld: Meta-reinforcement learning from images via latent state models. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1246–1261. PMLR, 2021b. URL https://proceedings.mlr.press/v155/zhao21d.html.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D vision-language-action generative world model. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 61229–61245. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhen24a.html.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

Andrey Zhmoginov, Mark Sandler, and Maksym Vladymyrov. HyperTransformer: Model generation for supervised and semi-supervised few-shot learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27075–27098. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhmoginov22a.html.

Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, Zhiquan Qi,

Yitao Liang, Yuanpei Chen, and Yaodong Yang. A survey on vision-language-action models: An action tokenization perspective, 2025a. URL https://arxiv.org/abs/2507.01925.

Yifan Zhong, Xuchuan Huang, Ruochong Li, Ceyao Zhang, Zhang Chen, Tianrui Guan, Fanlian Zeng, Ka Num Lui, Yuyao Ye, Yitao Liang, Yaodong Yang, and Yuanpei Chen. Dexgraspvla: A vision-language-action framework towards general dexterous grasping, 2025b. URL https://arxiv.org/abs/2502.20900.

Zhongyi Zhou, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Vision-language-action model with open-world embodied reasoning from pretrained knowledge. *arXiv preprint arXiv:2505.21906*, 2025.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(11):13344–13362, 2023. ISSN 0162-8828. doi: 10.1109/TPAMI.2023.3292075. URL https://doi.org/10.1109/TPAMI.2023.3292075.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkl9JlBYvr.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R Sanketi, Grecia Salazar, Michael S Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J Joshi, Alex Irpan, brian ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=XMQgwiJ7KSX.