Distilling Morphology-Conditioned Hypernetworks for Efficient Universal Morphology Control

Zheng Xiong [1] **Risto Vuorio** [1] **Jacob Beck** [1] **Matthieu Zimmer** [2] **Kun Shao** [2] **Shimon Whiteson** [1]

Abstract

Learning a universal policy across different robot morphologies can significantly improve learning efficiency and enable zero-shot generalization to unseen morphologies. However, learning a highly performant universal policy requires sophisticated architectures like transformers (TF) that have larger memory and computational cost than simpler multi-layer perceptrons (MLP). To achieve both good performance like TF and high efficiency like MLP at inference time, we propose HyperDistill, which consists of: (1) A morphology-conditioned hypernetwork (HN) that generates robot-wise MLP policies, and (2) A policy distillation approach that is essential for successful training of the HN. We show that on UNIMAL, a benchmark with hundreds of diverse morphologies, HyperDistill performs as well as a universal TF teacher policy on both training and unseen test robots, but reduces model size by 6-14 times, and computational cost by 67-160 times in different environments. Our analysis attributes the efficiency advantage of HyperDistill at inference time to knowledge decoupling, i.e., the ability to decouple inter-task and intra-task knowledge, a general principle that could also be applied to improve inference efficiency in other domains. The code is publicly available at https://github.com/MasterXiong/Universal-Morphology-Control.

1. Introduction

Reinforcement learning (RL) for robotic control has made great progress in recent years (Levine et al., 2016; Kalashnikov et al., 2018; Andrychowicz et al., 2020; Brohan et al., 2022). However, generalization across robots remains a key challenge, as the policy trained for one robot transfers poorly to another robot with a different morphology, i.e., the topology graph of a robot and the hardware parameters of each body part. Furthermore, it is too sample inefficient to train a separate policy for each new robot from scratch.

To tackle this challenge, universal morphology control aims to learn a universal policy to control different robots. By training on a set of diverse morphologies, this constitutes a form of multi-task RL (Vithayathil Varghese & Mahmoud, 2020) where controlling each robot is a separate task. The learned policy is expected to not only improve learning efficiency on the training robots, but also enable zero-shot generalization to test robots with unseen morphologies.

Training a multi-layer perceptron (MLP) policy is usually sufficient to achieve good performance on a single robot, but often generalizes poorly to other robots (Wang et al., 2018). While it is feasible to train a multi-robot MLP policy for universal morphology control, it is significantly outperformed by graph neural networks (GNN) (Wang et al., 2018; Pathak et al., 2019; Huang et al., 2020) and transformers (TF) (Kurin et al., 2020; Gupta et al., 2022) w.r.t. both training and generalization performance. Moreover, TF outperforms GNN by better modeling interactions between distant nodes in the morphology graph (Kurin et al., 2020), and has thus been adopted by most recent approaches (Dong et al., 2022; Furuta et al., 2022; Gupta et al., 2022; Hong et al., 2022; Trabucco et al., 2022; Chen et al., 2023; Xiong et al., 2023).

However, TF 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., 2021; Brohan et al., 2022; Leal et al., 2023). For example, a state-of-the-art TF-based method for universal morphology control (Xiong et al., 2023) requires about 40M FLOPs for a single step of inference 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 performance of TF but the inference efficiency of MLP?*

[1]Department of Computer Science, University of Oxford, Oxford, UK [2]Huawei Noah's Ark Lab, London, UK. Correspondence to: Zheng Xiong <zheng.xiong@cs.ox.ac.uk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

In this paper, we answer this question affirmatively. To get the best of both worlds, we introduce a hypernetwork (HN) (Ha et al., 2016), i.e., a network that takes context features as input to generate the parameters of a base network. Our key intuition is that, compared to TF, HN can better decouple inter-task knowledge and intra-task knowledge via its hybrid architecture for more efficient inference, which we call the *knowledge decoupling hypothesis*. Under our problem setting, the context-conditioned HN can provide sufficient model capacity to accommodate inter-robot knowledge, while the generated base network only needs to encode task-specific knowledge about the robot it controls. By contrast, TF encodes both kinds of knowledge with a single large model, which introduces high model redundancy when deployed on a specific robot.

Specifically, suppose we train a set of 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.

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. In this paper, 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 high inference efficiency via knowledge decoupling, and (2) Training via policy distillation to successfully learn such an HN policy. We experiment on a challenging universal morphology control benchmark called UNIMAL (Gupta et al., 2021), which includes hundreds of diverse morphologies to evaluate both multi-robot training and zero-shot generalization. HyperDistill achieves performance similar to the universal TF teacher 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 further support our knowledge decoupling hypothesis, which could serve as a general principle to improve inference efficiency in other domains.

2. Background

2.1. Problem Formulation

Learning a universal policy to control different morphologies can be seen as solving a contextual Markov Decision Process (CMDP) (Hallak et al., 2015), where the task context space $\mathcal{C}$ is defined over all possible morphology configurations. For robot (task) k, we use $\mathcal{S}_k$, $\mathcal{A}_k$, T_k and R_k to represent its state space, action space, transition function and reward function respectively.

We assume that all the robots are drawn from a modular design space, i.e., each robot can be seen as a morphology tree over a set of basic nodes (limbs), and the node-level state and action space are homogeneous across different robots' limbs. Based on this assumption, we have $\mathcal{S}_k = \mathcal{S}_k^1 \times \cdots \times \mathcal{S}_k^{N_k}$ and $\mathcal{A}_k = \mathcal{A}_k^1 \times \cdots \times \mathcal{A}_k^{N_k}$, where N_k is the number of limbs in robot k. The task context c_k includes morphology information about the robot, consisting of node-wise context features $\{c_k^i | i = 1, \ldots, N_k\}$ (see Appendix A for more details), and a topology tree of the robot.

We use $s_{k,t}, a_{k,t}, r_{k,t}$ to represent the state, action and reward at time step t for robot k. The training objective is to learn a universal policy $\pi_\theta(a_{k,t}|s_{k,t}, c_k)$ to maximize the average return over a set of K training robots, i.e., $\max_\theta \left[\frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{H} r_{k,t}\right]$, where H is the task horizon for all different robots. In addition to good training performance, we also expect the learned policy to generalize well on unseen test morphologies in a zero-shot manner.

2.2. Architectures 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 single 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).

When learning a multi-robot MLP policy, to handle the vari-

able 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. As the universal policy also conditions on the morphology context, the context features c_k^i are usually concatenated with the state features $s_{k,t}^i$ as node-wise input (Gupta et al., 2022). So the final input to the multi-robot MLP policy is $x = \left[x_1, x_2, \ldots, x_{N_k}, \vec{0}, \ldots, \vec{0}\right]$, where $x_i = [s_i, c_i]$, and we omit the robot index k and time index t thereafter for simplicity of notation. For the input layer of the MLP, we have $h^{(0)} = \sigma\left(W^{\text{in}}x + b^{\text{in}}\right) = \sigma\left(\sum_{i=1}^{N_k} W_i^{\text{in}} x_i + b^{\text{in}}\right)$, where σ is the activation function, W_i^{in} is the subset of the weight matrix that encodes the input features of node i. For the l-th hidden layer, we have $h^{(l+1)} = \sigma\left(W^{(l)}h^{(l)} + b^{(l)}\right)$. For the output layer, we have $a = W^{\text{out}}h^{(L)} + b^{\text{out}}$, which can be decomposed into node-wise action $a_i = W_i^{\text{out}}h^{(L)} + b_i^{\text{out}}$.

Unlike MLP, GNN (Wu et al., 2020) and TF (Vaswani et al., 2017) can naturally handle variable numbers of homogeneous elements. GNN can be directly applied to different morphology graphs, while TF treats each limb as a token. Unlike in many domains where tokens are processed sequentially, in universal morphology control, TF processes different nodes in parallel to model their spatial interactions. A typical TF architecture for universal morphology control consists of a linear embedding layer that embeds node-wise input features, multiple attention blocks that model limb interactions, and an MLP decoder that maps the node-wise embedding outputed by the attention blocks to node-wise actions (Gupta et al., 2022; Xiong et al., 2023).

2.3. Hypernetworks

A hypernetwork (Ha et al., 2016) is a network that generates the parameters of a base network conditioned on some context c. We can decompose an HN into a context encoder f and several output heads (parameterized as linear layers) that take the context embedding $e = f(c)$ as input, and output parameters 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., $W = \text{HN}_W(e)$, $b = \text{HN}_b(e)$. If W is a $M \times N$ matrix, and the context embedding dimension is E, then HN_W is a linear layer with input dimension E and output dimension $M \times N$, while HN_b is a linear layer with input dimension E and output dimension N.

3. HyperDistill

This section introduces HyperDistill, which achieves both good performance and high efficiency at inference time. In Section 3.1, we analyze the limitations of existing methods for universal morphology control, which motivates us to introduce HN as a solution in Section 3.2. In Section 3.3, we discuss why and how we train the HN via policy distillation, and analyze some of its critical algorithmic designs.

3.1. Limitations of Existing Architectures

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 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.

3.2. 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

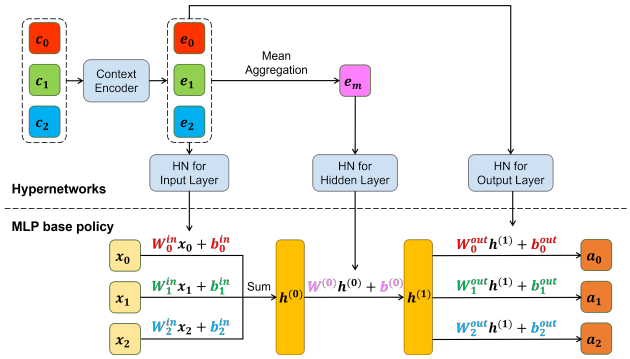

Figure 1. The architecture of HyperDistill. Different colors highlight the correspondence between the parameters in the base network and the context embedding they condition on via HN. We only show one hidden layer in the base MLP for ease of illustration. More hidden layers can be easily added in a similar way.

generate a task-specific base policy for each new task, while the much smaller base network is sufficient to efficiently solve the task. 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 is shown in Figure 1. Unlike prior work in HN, a unique challenge in applying HN to universal morphology control is that the input and output dimensions of the MLP base network vary across robots due to the variable number of limbs. We tackle this challenge by generating limb-wise parameters for the input and output layers, as the subset of parameters corresponding to each limb still has fixed dimension based on the modular space assumption in Section 2.1. Specifically, for the input layer, we have $h^{(0)} = \sigma\left(\sum_i W_i^{\text{in}} x_i + b_i^{\text{in}}\right)$, where $W_i^{\text{in}} = \text{HN}_W^{\text{in}}(e_i)$, $b_i^{\text{in}} = \text{HN}_b^{\text{in}}(e_i)$, 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, 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 A). 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 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.

3.3. HyperDistill Training via Policy Distillation

In principle, we can train a universal HN policy from scratch via RL. However, empirically we find that the 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 (Parisotto et al., 2015; Rusu et al., 2015) by first training a universal TF policy, then distilling it into an HN policy via behavior cloning, 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 pre-trained teacher policy for more efficient evaluation of different algorithmic choices.

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:

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

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., 2022; Wan et al., 2023), none of them systematically investigate how different algorithmic choices in PD influence the student's generalization performance. In this paper, we highlight four key factors that may influence the generalization gap between teach and student 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., 2022). 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 (Xiong et al., 2023), 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 pre-trained 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 we leave 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.

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 4.2)

(2) How do different algorithmic and architecture choices in HyperDistill influence its training and generalization performance? (Section 4.3)

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. (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 (Xiong et al., 2023) as the TF teacher, as it achieves SOTA performance on the UNIMAL benchmark. ModuMorph differs from a standard TF architecture in two ways: (1) In the attention blocks, it computes a

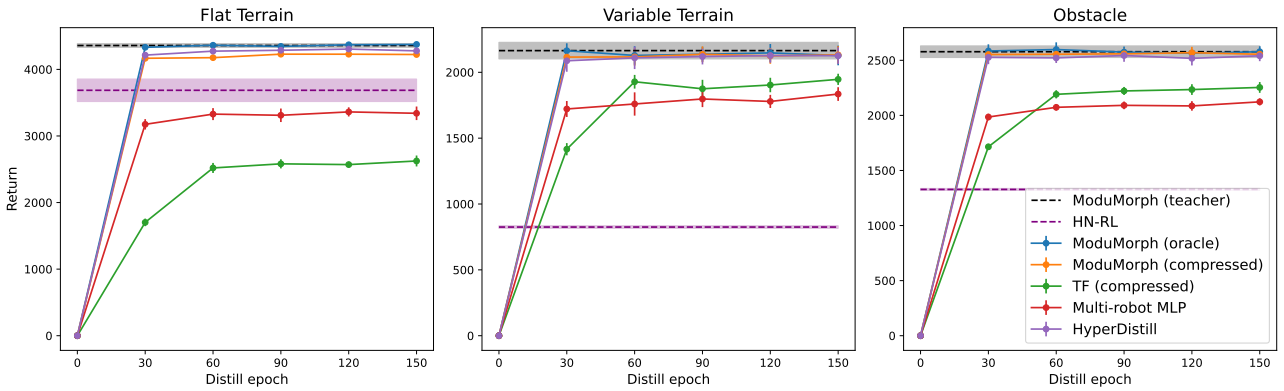

Figure 2. The performance of different methods on the *training* robots in each environment.

fixed attention matrix solely conditioned on the morphology context, which performs better than computing dynamic attention weights based on limb observations. (2) The embedding layer and the MLP decoder are generated by HNs in ModuMorph. Although ModuMorph and HyperDistill both use HNs, their motivations are significantly different. ModuMorph adopts HNs to better model the diverse behaviors across limbs, but is still a large TF-based model with high inference costs. By contrast, HyperDistill utilizes HNs to enable knowledge decoupling for efficient inference. This novel perspective on the role of HNs is a key contribution of our method. Furthermore, as we show in Section 4.2, a TF-based model 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.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 with a similar number of parameters as the base MLP in HyperDistill. It is used to validate that standard TF cannot achieve both good performance and high efficiency at the same time like HyperDistill. **(3) ModuMorph (compressed):** A 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 ability than standard 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 MLP as discussed in Section 3.2. Finally, we also compare with training an HN policy via RL (**HN-RL**) to show the importance of distillation.

Distillation Setup The distillation process runs for 150 epochs, with a mini batch size of 5120. We use Adam 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.3 for the size of each student model in each environment.

4.2. Main Results

Figures 2 and 3 illustrate how different methods perform on the training and test robots during the distillation process. Table 1 compares the model size and FLOPs of different methods at inference time.

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.

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 gen-

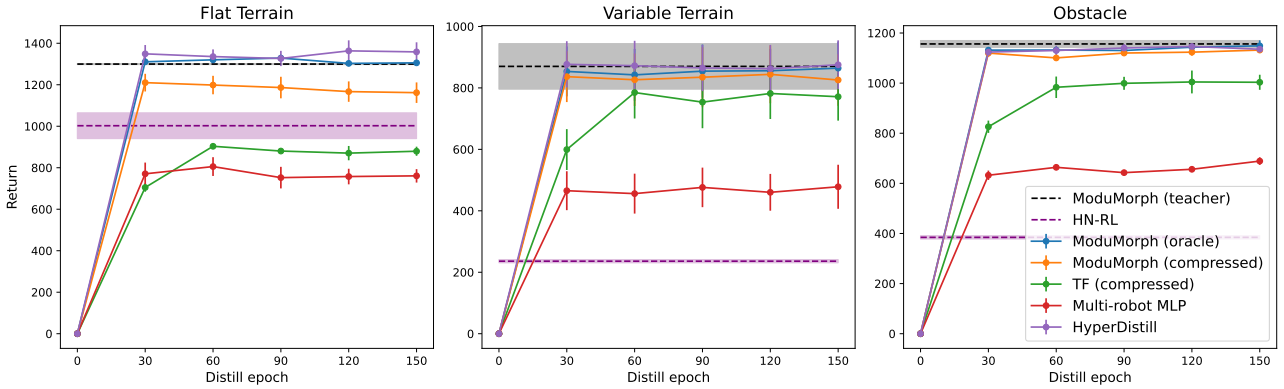

Figure 3. The performance of different methods on the *test* robots in each environment.

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 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 for how we compute the FLOPs, and C.1 for more analysis on the results in this table.

eralization 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., 2020; 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 3.1.

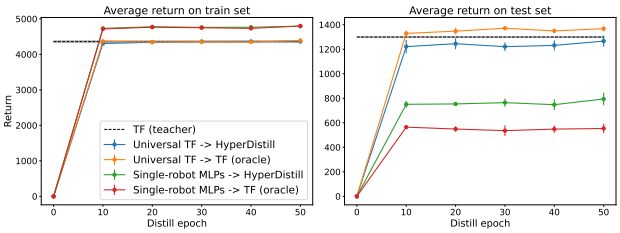

Figure 4. The student's learning curves under different teacher choices. In the figure legend, "X → Y" means that we distill from teacher X into student Y.

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.

4.3. Ablation Studies

In this section, we investigate how the algorithmic choices in PD influence generalization performance of the student. Due to space limitations, see Appendix C.2 for ablations on architecture choices in HyperDistill. 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.

The Choice of PD Teacher(s) We train (1) A universal TF teacher on all the training robots; (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 4 shows how the student policies perform on the training and test robots with different teacher choices. As expected, when using single-

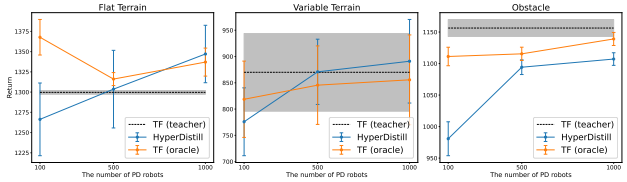

Figure 5. Final generalization performance of HyperDistill and TF (oracle) w.r.t. the number of PD robots in different environments.

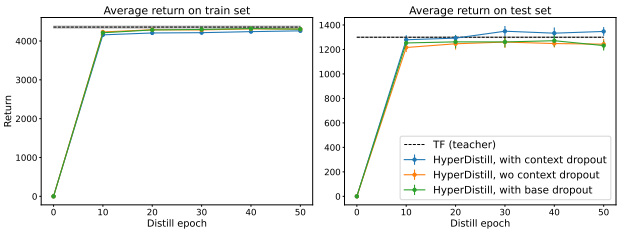

Figure 6. Learning curves of HyperDistill in FT with different ways of incorporating dropout regularization.

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 3.3.

Student-Teacher Architecture Alignment We reexamine the results in Figure 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 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 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 6, 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.

5. Discussion

We reexamine 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 enough 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, 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 (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 at a time to solve a given task (Barth & 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.

6. Related Work

Universal Morphology Control While generalization across robots with variations only in kinematics or dynamics parameters has been extensively studied in prior work (Tobin et al., 2017; Peng et al., 2018; Clavera et al., 2019; Ghadirzadeh et al., 2021; Feng et al., 2022), universal morphology control poses a more challenging task of generalization across morphologies. While learning an MLP policy with zero-padding is feasible to work across morphologies, it is significantly outperformed by GNN (Wang et al., 2018; Pathak et al., 2019; Huang et al., 2020) and TF (Kurin et al., 2020; Dong et al., 2022; Furuta et al., 2022; Gupta et al., 2022; Hong et al., 2022; Trabucco et al., 2022; Chen et al., 2023; Xiong et al., 2023). Furthermore, TF outperforms GNN by better modeling the interactions between distant limbs in a robot (Kurin et al., 2020), thus has been adopted by most recent works. However, these more sophisticated models introduce higher computational cost during deployment, which we aim to tackle by generating a compact MLP controller at inference time with morphology-conditioned HN. Similar to our work, ModuMorph (Xiong et al., 2023) also utilizes HN. However, ModuMorph introduces HNs to better model the diverse behaviors across different limbs and only uses HNs to generate embedding and decoder layers inside a TF, while our method introduces HNs to improve inference efficiency via knowledge decoupling and generate a whole MLP controller with HNs.

Hypernetworks A hypernetwork (Ha et al., 2016) takes context features as input to generate the parameters of a base network. In the context of RL, HNs provide a powerful way to model the complex dependency between task context and the optimal control policy for the task (Galanti & 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., 2022; Rezaei-Shoshtari et al., 2022; Beck 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.

Policy Distillation Distillation (Buciluǎ 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). However, less attention is paid to measuring and minimizing the generalization gap between the teacher and the student. Igl et al. (2020) 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 (tasks) 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 generalization gap. Most similar to our work is Furuta et al. (2022), which distill multiple single-robot MLP teachers into a TF student for universal morphology control. 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. Conclusion

In this paper, we propose HyperDistill to achieve both good performance and high efficiency at inference time for universal morphology control. While training HyperDistill via RL is hard, we take a policy distillation approach, and systematically investigate how some key algorithmic choices influence task generalization of the student policy. HyperDistill matches the performance of the 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 (Achiam et al., 2023; Brohan et al., 2022; 2023; Padalkar et al., 2023), the findings in our paper 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 on 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 inference efficiency gain may decrease as the HN will be called more frequently. How to tackle this challenge is an interesting direction for future work.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Specifically, as the control policy learned by our method can be deployed on real-world robots for tasks like locomotion and manipulation, issues like safety in operation need to be carefully considered, whose impacts have been extensively discussed in prior work.

Acknowledgements

We would like to thank Matthew Jackson and Alex Goldie for their helpful discussion on the work. We would also like to thank the conference reviewers for their constructive feedback on the paper. Zheng Xiong is supported by UK EPSRC CDT in Autonomous Intelligent Machines and Systems (grant number EP/S024050/1) and AWS. Risto Vuorio is supported by EPSRC Doctoral Training Partnership Scholarship and Department of Computer Science Scholarship. Jacob Beck is supported by the Oxford-Google DeepMind Doctoral Scholarship. The experiments were made possible by a generous equipment grant from NVIDIA.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

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

Beck, J., Jackson, M. T., Vuorio, R., and Whiteson, S. Hypernetworks in meta-reinforcement learning. In *6th Annual Conference on Robot Learning*, 2022.

Beck, J., Vuorio, R., Xiong, Z., and Whiteson, S. Recurrent hypernetworks are surprisingly strong in meta-rl. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. 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, pp. 3438–3445, 2020.

Chen, R., Han, J., Sun, F., and Huang, W. Subequivariant graph reinforcement learning in 3d environments. *arXiv preprint arXiv:2305.18951*, 2023.

Chen, T., Xu, J., and Agrawal, P. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pp. 297–307. PMLR, 2022.

Clavera, I., Nagabandi, A., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. 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.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, 2019.

Czarnecki, W. M., Pascanu, R., Osindero, S., Jayakumar, S., Swirszcz, G., and Jaderberg, M. Distilling policy distillation. In *The 22nd international conference on artificial intelligence and statistics*, pp. 1331–1340. PMLR, 2019.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., and Levine, S. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33: 13049–13061, 2020.

Dong, H., Wang, T., Liu, J., and Zhang, C. Low-rank modular reinforcement learning via muscle synergy. *Advances in Neural Information Processing Systems*, 35: 19861–19873, 2022.

Feng, G., Zhang, H., Li, Z., Peng, X. B., Basireddy, B., Yue, L., SONG, Z., Yang, L., Liu, Y., Sreenath, K., et al. Genloco: Generalized locomotion controllers for quadrupedal robots. In *6th Annual Conference on Robot Learning*, 2022.

Furuta, H., Iwasawa, Y., Matsuo, Y., and Gu, S. S. A system for morphology-task generalization via unified representation and behavior distillation. *arXiv preprint arXiv:2211.14296*, 2022.

Galanti, T. and Wolf, L. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33:10409–10419, 2020.

Ghadirzadeh, A., Chen, X., Poklukar, P., Finn, C., Björkman, M., and Kragic, D. Bayesian meta-learning for few-shot policy adaptation across robotic platforms. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1274–1280. IEEE, 2021.

Gupta, A., Savarese, S., Ganguli, S., and Fei-Fei, L. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):1–12, 2021.

Gupta, A., Fan, L., Ganguli, S., and Fei-Fei, L. Metamorph: Learning universal controllers with transformers. *arXiv preprint arXiv:2203.11931*, 2022.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Hallak, A., Di Castro, D., and Mannor, S. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Hao, Z., Guo, J., Han, K., Tang, Y., Hu, H., Wang, Y., and Xu, C. One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation. *arXiv preprint arXiv:2310.19444*, 2023.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. 2015.

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

Hong, S., Yoon, D., and Kim, K.-E. 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.

Huang, W., Mordatch, I., and Pathak, D. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 38–44. IEEE, 2016.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018.

Kurin, V., Igl, M., Rocktäschel, T., Boehmer, W., and Whiteson, S. My body is a cage: the role of morphology in graph-based incompatible control. *arXiv preprint arXiv:2010.01856*, 2020.

Leal, I., Choromanski, K., Jain, D., Dubey, A., Varley, J., Ryoo, M., Lu, Y., Liu, F., Sindhwani, V., Vuong, Q., et al. Sara-rt: Scaling up robotics transformers with self-adaptive robust attention. *arXiv preprint arXiv:2312.01990*, 2023.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431, 2020.

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Parisotto, E., Ba, J. L., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Pathak, D., Lu, C., Darrell, T., Isola, P., and Efros, A. A. Learning to control self-assembling morphologies: a study of generalization via modularity. *Advances in Neural Information Processing Systems*, 32, 2019.

Peng, M., Zhu, B., and Jiao, J. Linear representation meta-reinforcement learning for instant adaptation. *arXiv preprint arXiv:2101.04750*, 2021.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Rezaei-Shoshtari, S., Morissette, C., Hogan, F. R., Dudek, G., and Meger, D. Hypernetworks for zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:2211.15457*, 2022.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34: 8583–8595, 2021.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Sarafian, E., Keynan, S., and Kraus, S. Recomposing the reinforcement learning building blocks with hypernetworks. In *International Conference on Machine Learning*, pp. 9301–9312. PMLR, 2021.

Shen, S., Yao, Z., Li, C., Darrell, T., Keutzer, K., and He, Y. Scaling vision-language models with sparse mixture of experts. *arXiv preprint arXiv:2303.07226*, 2023.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. 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)*, pp. 23–30. IEEE, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Trabucco, B., Phielipp, M., and Berseth, G. Anymorph: Learning transferable polices by inferring agent morphology. In *International Conference on Machine Learning*, pp. 21677–21691. PMLR, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vithayathil Varghese, N. and Mahmoud, Q. H. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9): 1363, 2020.

Wan, W., Geng, H., Liu, Y., Shan, Z., Yang, Y., Yi, L., and Wang, H. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. *arXiv preprint arXiv:2304.00464*, 2023.

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Xiong, Z., Beck, J., and Whiteson, S. Universal morphology control via contextual modulation. *arXiv preprint arXiv:2302.11070*, 2023.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Multi-task reinforcement learning without interference. In *Proc. Optim. Found. Reinforcement Learn. Workshop NeurIPS*, 2019.

Zhao, Y., Gao, Y., Sun, Q., Tian, Y., Mao, L., and Gao, F. A real-time low-computation cost human-following framework in outdoor environment for legged robots. *Robotics and Autonomous Systems*, 146:103899, 2021.

A. 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; (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 3.2, 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 validate the importance of this feature transformation.

B. Further Experimental Setup

B.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. 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 & 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 and etc., as these operations only have linear complexity, which is negligible compared to the quadratic complexity of linear layers.

B.3. Architecture Details of Different Methods

Table 2 shows the size of different architectures in each environment. For the ModuMorph (teacher) and ModuMorph (oracle), we use the same architecture hyperparameters as in Xiong et al. (2023). 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 num.	Hidden size	Layer num.	Head	Hidden dim	Layer num.	Head	Hidden dim	Layer num.	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 2. The size of different models in each environment.

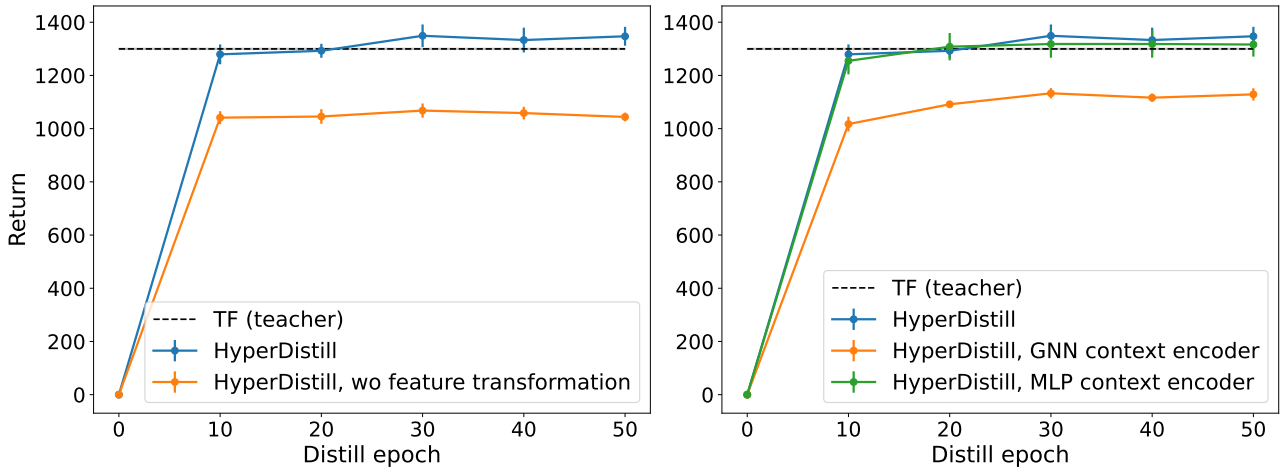

Figure 7. How context representation learning in HyperDistill influences generalization performance in FT. Left: context feature transformation. Right: architecture of the context encoder.

C. Further Experimental Results and Analysis

C.1. Further Analysis on Inference Efficiency

As shown in Table 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.2. Ablation Study on Context Representation Learning

Context Feature Transformation As shown in Figure 7 (left), conducting feature transformations to improve feature discrimination plays an important role in improving performance.

Architecture of Context Encoder As shown in Figure 7 (right), for different choices of the context encoder, 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 MLP and TF context encoder in the other two more challenging environments (Figure 8), and find that the TF context encoder performs better, which validates the benefits of modeling node interaction for context representation learning.

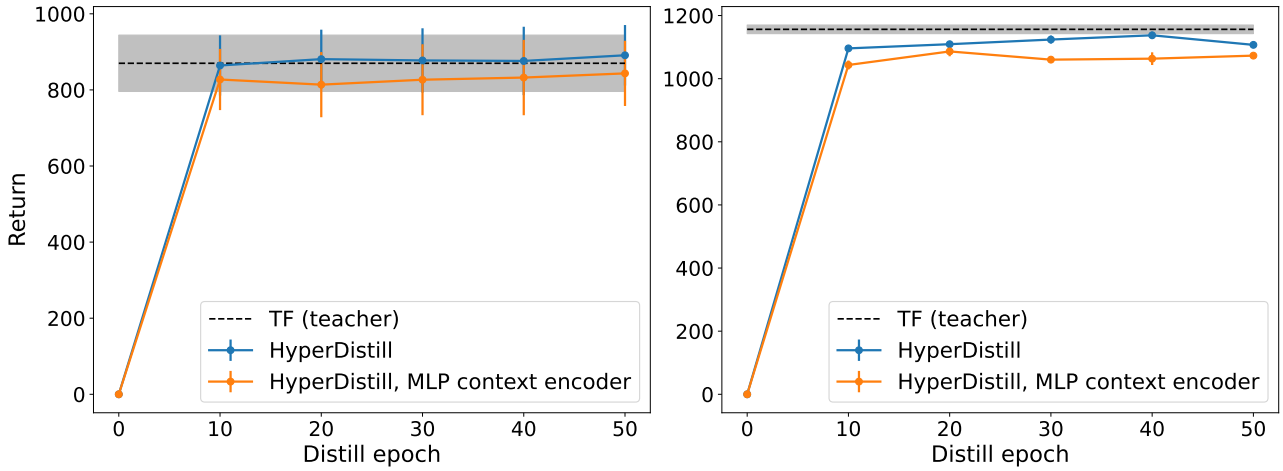

Figure 8. How TF and MLP context encoder influence HyperDistill's generalization performance. Left: VT; Right: Obstacle.