Meta-Controller: Few-Shot Imitation of Unseen Embodiments and Tasks in Continuous Control

Seongwoong Cho* **Donggyun Kim*** **Jinwoo Lee** **Seunghoon Hong**
School of Computing, KAIST
{seongwoongjo, kdgyun425, bestgenius10, seunghoon.hong}@kaist.ac.kr

Abstract

Generalizing across robot embodiments and tasks is crucial for adaptive robotic systems. Modular policy learning approaches adapt to new embodiments but are limited to specific tasks, while few-shot imitation learning (IL) approaches often focus on a single embodiment. In this paper, we introduce a few-shot behavior cloning framework to simultaneously generalize to unseen embodiments and tasks using a few (*e.g.,* five) reward-free demonstrations. Our framework leverages a joint-level input-output representation to unify the state and action spaces of heterogeneous embodiments and employs a novel structure-motion state encoder that is parameterized to capture both shared knowledge across all embodiments and embodiment-specific knowledge. A matching-based policy network then predicts actions from a few demonstrations, producing an adaptive policy that is robust to over-fitting. Evaluated in the DeepMind Control suite, our framework termed Meta-Controller demonstrates superior few-shot generalization to unseen embodiments and tasks over modular policy learning and few-shot IL approaches. Codes are available at https://github.com/SeongwoongCho/meta-controller.

1 Introduction

Generalizing across different robot embodiments and tasks with only a few demonstrations is a fundamental challenge in continuous control [6, 11, 40, 29, 33, 14]. This capability is crucial for developing versatile and adaptive robotic systems that can operate effectively in diverse and dynamic environments. The high diversity of embodiments and tasks, however, makes this particularly challenging. Robot embodiments vary widely in their morphological (*e.g.,* number and connectivity of joints) and dynamic (*e.g.,* motor gear ratios, damping coefficients) configurations, complicating the design of a unified architecture capable of handling heterogeneous input states and output actions. Furthermore, the variety of tasks—such as locomotion, object manipulation, and navigation—requires the learning of transferable skills that can efficiently generalize across different tasks.

Despite significant advancements in reinforcement learning (RL) and imitation learning (IL), achieving simultaneous generalization across diverse embodiments and tasks with a few demonstrations remains largely underexplored. Modular policy learning approaches [34, 18, 16, 12, 10] have shown promise by learning modular policies that can be shared across embodiments with different morphologies. However, these methods primarily focus on specific task types such as locomotion, and lack the flexibility to adapt to a wide range of control tasks, limiting their broader applicability. Conversely, few-shot IL approaches [8, 7, 39, 13, 37, 36] aim to learn novel tasks from a few demonstrations. These techniques excel in scenarios with sparse training data but typically concentrate on a single embodiment with fixed morphological and dynamic structures, restricting their ability to generalize across various embodiments and tasks. As a result, these two fields have developed independently, and their integration remains an open area of research.

To address these challenges, we propose a novel framework that flexibly learns arbitrary control tasks on unseen embodiments using a few (*e.g.,* five) reward-free demonstrations. To handle hetero-

*Equal contribution

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

geneous embodiments within a unified architecture, we tokenize states and actions into joint-level representations, since joints serve as the fundamental building blocks of robots and provide a modular representation for compositional generalization to unseen embodiments [12, 10]. Given this unified I/O, we employ a state encoder to capture both embodiment-specific knowledge about morphology and dynamics and shared knowledge about the physics governing the environment. Then we design a matching-based policy network that predicts actions from the encoded states, conditioned on a few demonstrations. Our model is trained using episodic meta-learning on a dataset comprising various embodiments and tasks, followed by few-shot behavior cloning on unseen embodiments and tasks using a few reward-free demonstrations.

Our key contributions are as follows: (1) We design a novel structure-motion encoder that operates on joint-level state representations, efficiently disentangling embodiment-specific and task-specific knowledge. (2) We propose a matching-based meta-learning framework that efficiently transfers knowledge of local motions to quickly learn unseen tasks with a few demonstrations. (3) We evaluate our framework in various environments within the DeepMind Control suite, encompassing diverse embodiments and tasks, demonstrating superior few-shot learning performance over existing baselines.

2 Problem Setup

A reinforcement learning (RL) problem involves an agent interacting with an environment, typically modeled as a Markov Decision Process (MDP). An MDP is represented by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. In conventional RL, an agent learns a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes expected cumulative rewards $\mathbb{E}_\pi[\gamma^t R(s_t, a_t)]$, where $\gamma^t \in [0, 1]$ is a discount factor. Training such an RL agent typically requires numerous interactions with the environment and a carefully designed reward function, making it burdensome to learn new tasks. Behavior cloning (BC) addresses these challenges by using supervised learning techniques to imitate an expert policy from offline demonstrations. We focus on the few-shot BC setting, where the training data consists of a few reward-free demonstrations $\mathcal{D} = \{\tau_i\}_{i \leq N}$, with each demonstration $\tau_i = \{(s_t^i, a_t^i)\}_{t \leq T}$ being a temporal sequence of states and actions performed by an expert model.

In this paper, we consider continuous control problems of multi-joint robots that involve various embodiments and tasks. An embodiment $\mathcal{E}$ refers to the physical configuration of robots, which includes (1) the morphology, *i.e.,* the shape, size, and arrangement of the components such as limbs, joints, and sensors, and (2) the dynamics parameters that affect the robot's behavior, *e.g.,* motion inertia, mass, gear ratios, and damping. In terms of MDP, different embodiments can have different dimensionality of state and action spaces, *e.g.*, $\dim(\mathcal{S}_{\mathcal{E}_i}) \neq \dim(\mathcal{S}_{\mathcal{E}_j})$ and $\dim(\mathcal{A}_{\mathcal{E}_i}) \neq \dim(\mathcal{A}_{\mathcal{E}_j})$ for $\mathcal{E}_i \neq \mathcal{E}_j$, as well as distinct transition probabilities $P_{\mathcal{E}}$ that determine the kinematics of the robot. A task $\mathcal{T}$ is defined by a specific goal or objective that the robot must achieve within its environment, characterized by a reward function $R_{\mathcal{T}}$. Tasks can vary widely, ranging from locomotion and manipulation to complex interactions with dynamic environments. The combination of different embodiments $\mathcal{E}$ and tasks $\mathcal{T}$ creates a broad class of continuous control problems.

Our objective is to achieve simultaneous generalization to unseen embodiments and tasks of continuous control with a few-shot behavior cloning framework. In other words, the model has to learn a policy for a novel continuous control problem from only a few demonstrations $\mathcal{D}$, where both embodiment $\mathcal{E}$ and task $\mathcal{T}$ can be arbitrary and previously unseen.

2.1 Challenges and Desiderata

Despite achieving the simultaneous few-shot generalization to unseen embodiments and tasks is crucial for developing versatile and adaptive robotic systems, this problem remains less explored. We characterize two distinct challenges and desiderata to address each challenge.

Handling Heterogeneous Embodiments. To generalize to arbitrary embodiments in continuous control, the model must possess an architecture capable of universally handling heterogeneous states and actions of various embodiments. This necessitates a unified input/output (I/O) representation for states and actions, allowing for the sharing of structural characteristics (e.g., dimensionality) and semantics (e.g., input attributes or output control types) across different embodiments. Additionally,

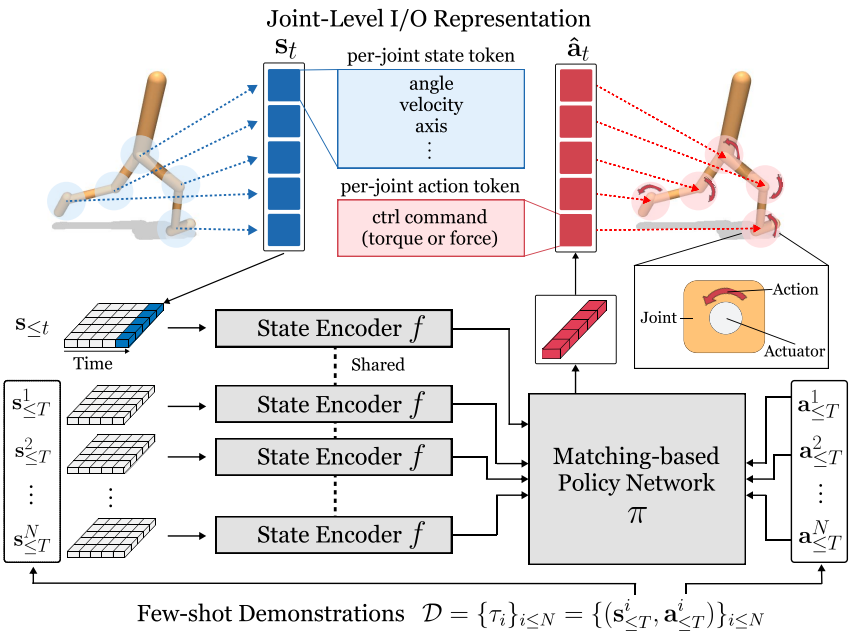


Figure 1: The overall framework of Meta-Controller. First, the states and actions of various robot embodiments are tokenized into joint-level representations. The state tokens are then encoded by the state encoder to capture knowledge about the embodiments. Finally, a matching-based policy network uses few-shot demonstrations with the encoded state features to predict per-joint actions.

the encoder to extract state features should be able to capture transferable knowledge across different embodiments as well as embodiment-specific knowledge to flexibly adapt to distinct morphologies and dynamics of each embodiment.

Few-shot Policy Adaptation. To achieve robust few-shot learning on unseen tasks, an efficient and flexible policy adaptation mechanism is essential. Given the diverse behaviors required by continuous control problems across various embodiments and tasks, the policy network must learn transferable skills shared by different tasks. Simultaneously, the model must dynamically adapt its policy by inferring the task based on a few demonstrations, ensuring it captures the underlying structure of previously unseen tasks. Additionally, the adaptation mechanism must be robust to overfitting.

3 Method

In this section, we introduce Meta-Controller, a few-shot behavior cloning framework for simultaneous generalization of embodiments and tasks of continuous control. Figure 1 illustrates the overall framework. Meta-Controller addresses heterogeneous embodiments by tokenizing the states and actions into joint-level I/O (Section 3.1) and employing a state encoder that captures knowledge about the structure and dynamics of the embodiment (Section 3.2.1). Then, a matching-based policy network (Section 3.2.2) predicts the action by leveraging a few given demonstrations. The training protocol of Meta-Controller consists of episodic meta-learning and few-shot fine-tuning (Section 3.3).

3.1 Joint-Level I/O Representation

To unify the state and action spaces of different embodiments, we adopt joint-level tokenization. Joints are fundamental components of robots, and their primary source of action is the torque or force generated by actuators attached to each joint. This allows us to standardize the states and actions of a robotic agent into per-joint observations and control commands. Consequently, joint-level states and actions provide a natural modular representation, facilitating the compositional generalization of various robot embodiments[2].

[2]We consider a pre-defined set of joints whose *compositions* differ per embodiment.

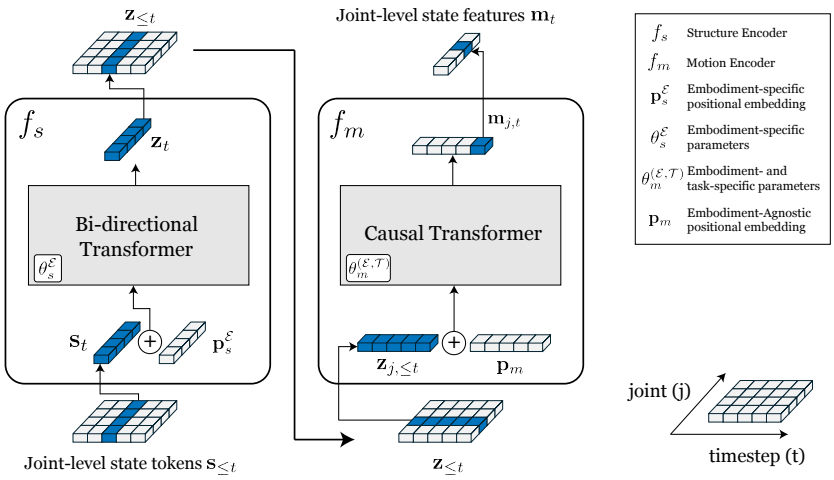


Figure 2: The state encoder f consists of two component transformers. Joint-level state tokens are first encoded by the structure encoder f_s along the joint axis, where the positional embedding and a part of backbone parameters adapt the model to each embodiment. The features are then passed to the motion encoder f_m, which computes causal attentions of per-joint features along the temporal axis, where a part of backbone parameters adapt the model both to the embodiment and task.

For any given embodiment $\mathcal{E}$, we represent the corresponding states $\mathbf{s}_t$ and actions $\mathbf{a}_t$ as arrays of joint-level tokens as follows:

$$\mathbf{s}_t = [\mathbf{s}_{j,t}]_{j \leq J_\mathcal{E}} \in \mathbb{R}^{J_\mathcal{E} \times d}, \quad \mathbf{a}_t = [\mathbf{a}_{j,t}]_{j \leq J_\mathcal{E}} \in \mathbb{R}^{J_\mathcal{E} \times 1}, \tag{1}$$

where $J_\mathcal{E}$ is the number of joints in $\mathcal{E}$ and t indicates the time. The state token $\mathbf{s}_{j,t} \in \mathbb{R}^d$ represents per-joint information, such as position, velocity, movement axis, and motion types (*i.e.,* linear or angular). The action token $\mathbf{a}_{j,t} \in [-1, 1]$ represents the control command of j-th joint, where we assign zero value for free joints. This joint-level representation ensures consistency across heterogeneous robotic embodiments, enabling unified learning of different continuous control problems.

3.2 Meta-Controller

Given the tokenized representation of joint states and actions, we aim to build an adaptive controller for arbitrary robot embodiments to perform arbitrary continuous control tasks based on a set of few demonstrations $\mathcal{D} = \{\tau^i\}_{i \leq N}$. To address the challenge discussed in Section 2.1, we employ an encoder f to extract the embodiment-aware state features and an adaptive policy network π that efficiently leverages the demonstrations $\mathcal{D}$ to predict the actions.

3.2.1 State Encoder for Embodiment Generalization

The state encoder f encodes the tokens of the current state with history $\mathbf{s}_{\leq t}$ into state features $\mathbf{m}_t$.

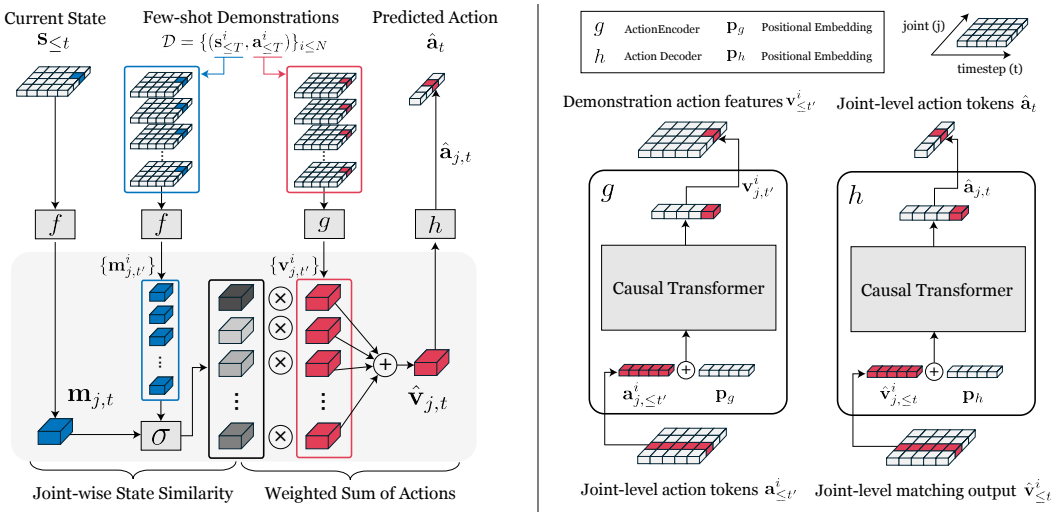
$$\mathbf{m}_t = f(\mathbf{s}_{\leq t}; \theta). \tag{2}$$

To effectively encode the states of each embodiment, we decompose our state encoder into two components: a structure encoder f_s that captures morphological knowledge, and a motion encoder f_m that captures dynamics knowledge.

Structure Encoder. The structure encoder models the relationships among the joints within each embodiment. As shown in Figure 2, we use a bi-directional transformer on the joint-level state tokens $\mathbf{s}_t$ at each timestep t to extract the structure features $\mathbf{z}_t$:

$$\mathbf{z}_t = f_s(\mathbf{s}_t + \mathbf{p}_s^\mathcal{E}; \theta_s, \theta_s^\mathcal{E}), \tag{3}$$

where the embodiment-specific positional embedding $\mathbf{p}_s^\mathcal{E}$ is added to the input tokens. Note that we decompose the parameters of f_s into adaptive parameters $\left(\mathbf{p}_s^\mathcal{E}, \theta_s^\mathcal{E}\right)$ that capture embodiment-specific knowledge and shared parameters θ_s that captures common knowledge across embodiments. The



(a) Matching-based Policy Network (b) Action Encoder and Decoder

Figure 3: An illustration of the matching-based policy network π. (a) Each state and action token in few-shot demonstrations is encoded by the corresponding encoders f and g, where we use the same encoder f used for the current state. A matching module σ then computes the weighted sum of action features based on the joint-wise similarity between state features. Finally, an action decoder h decodes the joint-wise matching output to predict the current action. (b) Both the action encoder g and decoder h are causal transformers operating along the temporal axis of action tokens and features.

positional embedding $\mathbf{p}_s^{\mathcal{E}}$ is crucial for adapting to local configurations (*e.g.,* length, movement range) of joints in $\mathcal{E}$ not explicitly given in the state $\mathbf{s}_t$. Global configurations (*e.g.,* control timestep) shared by all joints are handled through the embodiment-specific parameters $\theta_s^{\mathcal{E}}$ in the transformer backbone. Shared parameters θ_s capture common knowledge, such as the physics governing the environment.

To enable efficient yet flexible adaptation during few-shot behavior cloning, we designate only a small portion of the backbone parameters to be embodiment-specific. Inspired by parameter-efficient fine-tuning (PEFT) approaches that effectively modulate transformers with only a few parameters, we employ bias parameters [2], low-rank projection matrices [17], and also layer-scale parameters for $\theta_s^{\mathcal{E}}$. As explained in Section 3.3, only the adaptive parameters $\left(\mathbf{p}_s^{\mathcal{E}}, \theta_s^{\mathcal{E}}\right)$ are updated during few-shot learning on unseen embodiments, ensuring robustness to overfitting the few demonstrations.

Motion Encoder. While the state encoder f_s encodes structural information about the embodiments, it does not model the temporal dynamics of states which is crucial for understanding continuous control tasks. Therefore, we introduce a motion encoder f_m, which is a causal transformer that encodes the state features along the temporal axis. As illustrated in Figure 2, f_m rearranges the encoded structure features $\mathbf{z}_{\leq t}$ into separate temporal sequences of joint-level features $\mathbf{z}_{j,\leq t}$, then produce the motion features for each joint $\mathbf{m}_{j,t}$ auto-regressively:

$$\mathbf{m}_{j,t} = f_m(\mathbf{z}_{j,\leq t} + \mathbf{p}_m; \theta_m, \theta_m^{(\mathcal{E},\mathcal{T})}), \quad \forall j \leq J_{\mathcal{E}} \tag{4}$$

where $\mathbf{p}_m$ denotes the positional embedding for specifying the timesteps t. Additionally, we introduce a small portion of adaptive parameters $\theta_m^{(\mathcal{E},\mathcal{T})}$ in the causal transformer backbone, which is both specific to embodiment $\mathcal{E}$ and task $\mathcal{T}$. These parameters help the model understand the unique motions in the few-shot demonstrations that are specific to each task and embodiment. We use the same PEFT techniques employed in the structure encoder for the adaptive parameters.

3.2.2 Few-shot Policy Adaptation for Task Generalization

To learn unseen tasks $\mathcal{T}$, we design an adaptive policy network π with a matching framework [20, 19] that incorporates the demonstrations $\mathcal{D} = \{(\mathbf{s}^i_{\leq T}, \mathbf{a}^i_{\leq T})\}_{i \leq N}$ to produce an action $\hat{\mathbf{a}}_t$.

$$\hat{\mathbf{a}}_t = \pi(\mathbf{m}_{\leq t}, \mathcal{D}; \phi). \tag{5}$$

Figure 3 illustrates the architecture of the policy network, which consists of an action encoder g, a matching module σ, and an action decoder h.

Matching-based Policy Network. To incorporate the demonstrations, we encode the state and action tokens in the demonstrations using the state encoder f introduced in Section 3.2.1 and an additional action encoder g, respectively.

$$\mathbf{m}^i_{j,t'} = f(\mathbf{s}^i_{j,\leq t'}; \theta), \qquad \forall j \leq J_{\mathcal{E}},\ \forall t' \leq T,\ \forall i \leq N, \tag{6}$$

$$\mathbf{v}^i_{j,t'} = g(\mathbf{a}^i_{j,\leq t'} + \mathbf{p}_g; \phi_g), \quad \forall j \leq J_{\mathcal{E}},\ \forall t' \leq T,\ \forall i \leq N, \tag{7}$$

where we employ a causal transformer for g with temporal position embedding $\mathbf{p}_g$. Then π incorporates the motion features $\mathbf{m}_t$ of the current state (Eq. (2)) and the encoded demonstration features (Eq. (6) and (7)) via joint-wise matching as follows:

$$\hat{\mathbf{v}}_{j,t} = \sum_{t' \leq T} \sum_{i \leq N} \sigma(\mathbf{m}_{j,t}, \mathbf{m}^i_{j,t'}) \cdot \mathbf{v}^i_{j,t'}, \quad \forall j \leq J_{\mathcal{E}}, \tag{8}$$

where σ is a similarity function (*e.g.*, cosine similarity). Finally, we employ an action decoder h, a causal transformer that decodes the joint-wise matching output $\hat{\mathbf{v}}_{j,t}$ into j-th action token $\hat{\mathbf{a}}_{j,t}$.

$$\hat{\mathbf{a}}_{j,t} = h(\hat{\mathbf{v}}_{j,\leq t}; \phi_h), \quad \forall j \leq J_{\mathcal{E}}. \tag{9}$$

The matching-based policy network offers significant benefits for few-shot behavior cloning, particularly when dealing with unseen tasks and embodiments. Its robust adaptation capabilities stem from effectively incorporating demonstration data through a similarity function, which dynamically matches current state features with those from demonstrations. This non-parametric approach minimizes overfitting, enhancing generalization from limited examples.

Hierarchical IL Interpretation. Eq. (8) can also be interpreted as hierarchical imitation learning that generalizes to unseen tasks using a transferable skill set. Since the action encoder g extracts a pool of temporal action features that are composed to produce the current action $\mathbf{a}_t$ in the feature space, we can treat the action features $\mathbf{v}^i_{j,t'}$ as *local motor skills*, *i.e.,* building blocks of joint behavior for various control tasks. This modular approach lets the network recombine these skills to efficiently tackle new challenges. By assigning high similarity scores to relevant demonstrations, the policy network ensures accurate imitation of expert behavior, improving performance on novel tasks.

3.3 Training & Inference

The training protocol of Meta-Controller consists of two stages: episodic meta-learning and few-shot fine-tuning, where we train whole parameters (θ, ϕ) of the model during the first stage while we train only the adaptive parameters $(\mathbf{p}^{\mathcal{E}}_s, \theta^{\mathcal{E}}_s, \theta^{(\mathcal{E},\mathcal{T})}_m)$ during the second stage.

Episodic Meta-Learning. During episodic meta-learning, the model acquires general knowledge of continuous control through a number of episodes that mimic few-shot behavior cloning scenarios to ensure effective adaptation to unseen embodiments and tasks. To this end, we leverage a meta-training dataset that consists of demonstrations $\mathcal{B}_{(\mathcal{E},\mathcal{T})}$ by expert agents with various embodiments $\mathcal{E}$ and tasks $\mathcal{T}$. At each episode, we first sample a continuous control problem $(\mathcal{E}, \mathcal{T})$, then sample two subsets of $\mathcal{B}_{(\mathcal{E},\mathcal{T})}$: a set of support data $\mathcal{D}_S$ and query data $\mathcal{D}_Q$. Then the model is trained to imitate query data using the demonstrations.

$$\min_{(\theta,\phi)} \mathbb{E}_{p(\mathcal{E},\mathcal{T})} \left[\mathbb{E}_{\mathcal{D}_S, \mathcal{D}_Q \sim \mathcal{B}_{(\mathcal{E},\mathcal{T})}} \left[\frac{1}{|\mathcal{D}_Q|} \sum_{(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{D}_Q} \left\| \mathbf{a}_t - \hat{\mathbf{a}}_t \right\|^2 \right] \right], \tag{10}$$

where $\hat{\mathbf{a}}_t$ is produced by Eq. (5) using $\mathcal{D}_S$, and $p(\mathcal{E}, \mathcal{T})$ is a uniform distribution over all control problems within the dataset.

Few-shot Fine-Tuning. After acquiring the meta-knowledge about continuous control problems, we apply our Meta-Controller in a few-shot behavior cloning setup, where it should adapt to both unseen embodiments and tasks with a few demonstrations $\mathcal{D}$. To this end, we randomly split $\mathcal{D}$ into two disjoint subsets, and fine-tune the model with Eq. (10) but *with only respect to* the embodiment-specific and task-specific parameters $(\mathbf{p}^{\mathcal{E}}_s, \theta^{\mathcal{E}}_s, \theta^{(\mathcal{E},\mathcal{T})}_m)$ while freezing the rest. After fine-tuning, the model uses the whole demonstrations $\mathcal{D}$ in the policy network to produce the actions at evaluation.

4 Related Work

Modular Policy Learning. Modular policy learning aims to develop modular policies for multi-joint robots that are adaptable to various morphologies. NerveNet [34] uses a Graph Neural Network (GNN) to model structural relationships between joint-level features. SWAT [16] leverages graph features like the normalized graph Laplacian for improved structural learning in reinforcement learning (RL). Amorpheous [23] and MetaMorph [12] use transformer architectures, treating the morphological graph as fully connected to exploit transformers' capabilities. These works focus on RL and zero-shot generalization to specific locomotion tasks. Furuta et al.[10] proposes an imitation learning framework with a benchmark environment for extensive morphologies, showing promising zero-shot generalization but limited to specific tasks like reaching a goal position, and not handling general continuous control tasks.

Few-shot Imitation Learning. Few-shot Imitation Learning (IL) approaches focus on generalizing novel RL tasks with only a few demonstrations. Gradient-based meta-learning algorithms like MAML [8] and Bayesian MAML [39] have been explored for rapid task adaptation. Duan et al. [7] addressed one-shot imitation learning by incorporating an attention model over the query state and demonstrations. FIST [13] is a hierarchical skill transition model that learns to extract transferable high-level skill sets from demonstrations and leverages this skill knowledge during adaptation. Recently, PDT [37] introduced a transformer-based few-shot learner, using the few-shot demonstration as a prompt token. Similarly, HDT [36] trains a hyper-network to generate adapter parameters for a pre-trained decision transformer, adapting to few-shot demonstrations. Despite notable generalization abilities, these few-shot IL approaches have only been studied within a single embodiment, limiting their applicability to real-world scenarios with heterogeneous embodiments.

5 Experiments

5.1 Experimental Setup

Environment and Dataset. We evaluate the few-shot behavior cloning of unseen embodiments and tasks within the DeepMind Control (DMC) suite [31], which includes continuous control tasks featuring diverse kinematic structures. We select 30 tasks from 10 embodiments as training tasks and 8 tasks from 4 embodiments as held-out evaluation tasks. The evaluation tasks include three unseen embodiments (`hopper`, `reacher-four`, `wolf`) and one seen embodiment (`walker`), with the `wolf` being an additional embodiment introduced in [34]. Our meta-training dataset is constructed using a replay buffer of an expert agent [38], consisting of up to 2000 demonstration trajectories for each task and embodiment. Each demonstration consists of state-action pairs over $T = 500$ timesteps, with rewards discarded for both episodic meta-learning and few-shot behavior cloning. For N-shot few-shot behavior cloning, we use the last N demonstrations from the buffer. A full detail of the dataset is included in Appendix B.1.

Evaluation Protocol. We evaluate all models with 20 different initial states and report the mean and standard error. For evaluation metric, we use a *normalized score* [9] calculated by $\frac{\text{score}-\text{random score}}{\text{expert score}-\text{random score}}$, where each score represents the average cumulative rewards during the evaluation. The random score is obtained by evaluating a random agent using a uniform distribution policy over the action spaces. We evaluate the models every 1000 iterations of few-shot behavior cloning and report the best score. For models that use task-specific low-rank projection matrices, we search the rank parameter over $\{4, 8, 16\}$ and report the best one. Throughout the experiments, we present 5-shot learning results ($N = 5$) unless otherwise specified.

Baselines. We compare our model with various Decision Transformer [4] (DT)-based few-shot imitation learning approaches and two transformer-based modular policy learning approaches. In the DT-based models, we exclude return-to-go tokens from the input tokens to simulate behavior cloning. Since the DT architecture is not inherently designed to handle heterogeneous state and action spaces, we modify its input and output linear heads in an embodiment-specific manner and fine-tune them for unseen embodiments.

- **DT-based Models.** From-Scratch Decision Transformer **(FS-DT)** is a decision transformer that trains a downstream task directly from randomly initialized weights. **Multi-Task Decision Transformer (MT-DT)** is a variant of the decision transformer that trains multiple tasks with task-specific

Table 1: 5-shot behavior cloning results on DeepMind Control (DMC) suite.

	Unseen Emb.							Seen Emb.	Avg.
Emb. ($\mathcal{E}$)	hopper			wolf		reacher-four		walker	
Task ($\mathcal{T}$)	hop	hop-bwd.	stand	walk	run	easy	hard	walk-bwd.	
FS-DT [4]	1.6±1.5	56.9±6.2	9.8±0.9	44.7±10.8	35.8±7.2	-0.7±4.2	5.5±4.9	20.6±8.6	21.8
MT-DT	3.2±2.3	64.3±7.0	10.1±1.1	53.4±11.8	46.2±10.2	10.2±7.4	8.4±5.1	**86.4±5.0**	35.3
PDT [37]	0.9±0.7	29.7±9.3	6.5±1.9	47.0±11.4	35.7±8.3	3.3±4.7	5.8±3.5	5.0±1.5	16.7
PDT+PEFT	2.3±1.5	61.7±7.7	12.3±4.0	54.9±8.4	52.7±6.4	-1.7±3.3	9.1±5.9	74.1±8.5	33.2
HDT [36]	0.8±0.4	51.6±7.9	13.7±3.7	56.4±11.0	38.7±9.1	1.2±5.3	8.5±5.5	3.3±1.0	21.8
L2M [28]	4.5±1.7	45.1±7.9	4.0±1.2	50.6±10.7	65.9±3.8	3.7±6.3	15.5±7.4	40.0±9.8	28.7
MetaMorph [12]	33.4±4.9	81.8±2.4	54.1±4.7	73.0±4.4	29.7±4.6	-4.1±2.6	3.8±3.1	31.8±8.6	37.9
MTGv2 [10]	21.6±3.3	83.1±2.2	40.7±6.3	68.2±7.3	29.1±4.3	-1.4±2.7	4.6±4.8	35.5±5.9	35.2
Ours	**49.1±6.1**	**87.2±1.6**	**82.5±4.9**	**91.7±5.1**	**67.3±3.1**	**56.1±8.8**	**50.8±10.6**	84.3±5.7	**71.1**

parameters and fine-tunes only these parameters for few-shot adaptation. For the task-specific parameters, we use the same parameters as $\theta_m^{(\mathcal{E},\mathcal{T})}$ used in our motion encoder. Prompt-based Decision Transformer **(PDT)** [37] adapts its policy by conditioning on the few-shot demonstrations through prompting. We also report the performance of a variant of PDT that fine-tunes task-specific parameters (**PDT+PEFT**), similar to MT-DT. Hyper Decision Transformer **(HDT)** [36] employs a hyper-network conditioned on few-shot demonstrations to generate parameters of the Adapter [25] module applied to DT. Learning To Modulate **(L2M)**[28] incorporates parameter-efficient fine-tuning (PEFT) techniques to DT architecture. While L2M is not directly proposed for few-shot imitation learning, we include this baseline since it uses a similar PEFT technique as ours.

- **Modular Policy-based Models.** We include two modular policy learning approaches, **MetaMorph**[12] and **MTGv2**[10], which utilize a transformer architecture to encode joint-level states. Originally designed for zero-shot learning of locomotion tasks, we adapt these approaches by incorporating task-specific linear heads and fine-tuning them on few-shot demonstrations. Both models are trained using the behavior cloning (BC) objective as described in [10].

Implementation Details. We implement both the structure encoder and the motion encoder using a 6-layer transformer with 4 attention heads and a hidden dimension of 512. Due to the quadratic computation cost of the transformer, we set the maximum history size of causal attention layers in the encoders to 10. The baseline models use the same transformer backbone. All models are trained for 200,000 iterations on the meta-training dataset and fine-tuned for 10,000 iterations on the few-shot demonstrations. For downstream embodiments that are structurally similar to a training task (*e.g.*, `reacher-three` and `reacher-four`), we initialize the encoder's embodiment-specific parameters using the trained parameters during fine-tuning. More details are included in Appendix B.2.

5.2 Main Results

Table 1 shows the 5-shot behavior cloning results of the models on both unseen and seen embodiments. We observe that Meta-Controller consistently outperforms all baselines across various continuous control problems, demonstrating its effectiveness. Existing few-shot imitation learning approaches that lack an embodiment generalization mechanism, such as PDT and HDT, struggle to adapt to unseen embodiments, often performing comparably to the naive from-scratch baseline (DT-FS). In contrast, modular policy learning approaches like MetaMorph and MTGv2 show better adaptation to unseen embodiments (*e.g.*, `hopper`). However, their performance is still inferior to Meta-Controller, which can be attributed to the absence of a few-shot adaptation mechanism for unseen tasks.

Notably, our model significantly outperforms all baselines in the challenging `reacher-four` embodiment, where models must understand the notion of a *goal position* given only five demonstrations. Figure 4 shows that while baseline models converge to the pose in the demonstrations regardless of the goal position, our model successfully reaches the goal position and converges with a unique pose. We attribute this success to the modular nature of our model. Our state encoder effectively shares knowledge about joint-level motions thanks to its parametrization, and the matching-based policy network flexibly exploits the local motor skills from a few demonstrations. We provide more results and analysis in Appendix C.

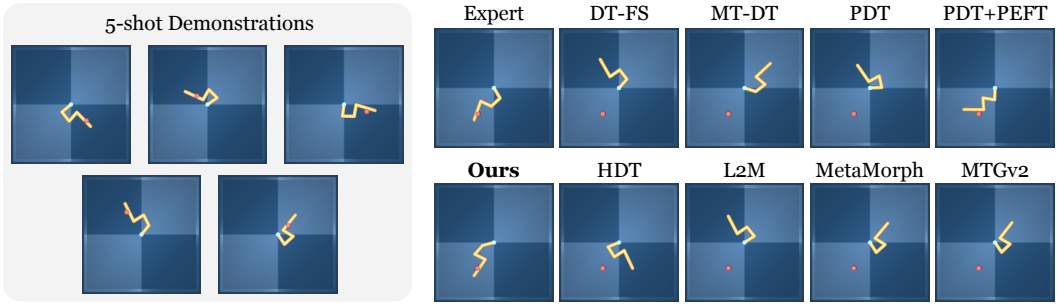


Figure 4: Qualitative comparison on the `hard` task of the `reacher-four` embodiment, visualizing the final states of the demonstrations and the rollout trajectories of each model. In this task, the robot must move its limb tip to the goal position (visualized as a red ball). While most of the baselines converge to one of the poses in the demonstrations and ignore the goal position, our model accurately solves the task with a distinct pose from the demonstrations.

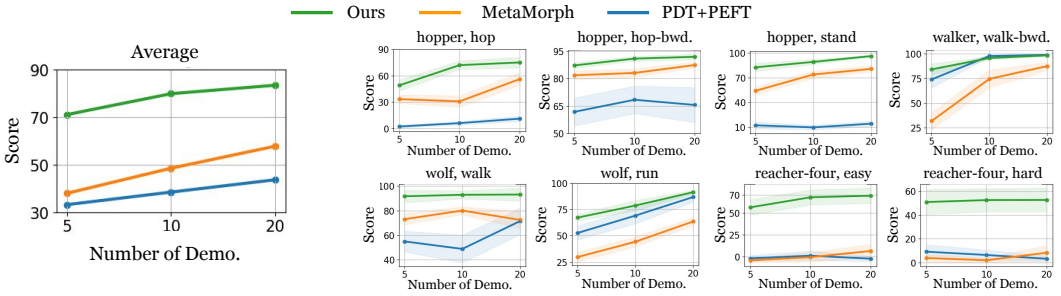


Figure 5: Ablation study on the number of demonstrations. We plot the normalized scores for each pair of embodiment and task $(\mathcal{E}, \mathcal{T})$ and their average, varying the number of shots as 5, 10, and 20.

5.3 Ablation Studies

Ablation on the Architecture. To verify the effectiveness of each architectural component introduced in Section 3, we conduct an ablation study by progressively replacing each module—the structure encoder f_s, the motion encoder f_m, and the matching module σ—with a linear layer. To isolate the effect of adaptation parameters, we replace f_s with task-specific linear layers and f_m with embodiment- and task-specific linear layers. Table 2 summarizes the results. We observe that the structure encoder f_s is crucial for generalization to unseen embodiments, as performance drops drastically when it is removed (row 1 vs. row 3). This indicates that the structure encoder captures modular knowledge about various morphologies, transferrable to unseen embodiments. Combined with the structure encoder, the motion encoder further improves performance (row 2 vs. row 3), especially on the seen embodiment (`walker`). This shows that modeling the temporal relationships of joints is beneficial when the model understands the embodiment. Finally, the matching module consistently improves performance (row 3 vs. row 4), particularly on challenging tasks such as the `hop` task of `hopper` and the tasks of `reacher-four` This demonstrates the effectiveness of the matching architecture in preventing overfitting with few demonstrations. We provide more ablation studies on architectural components in Appendix D.1.

Ablation on the Parametrization. To analyze the impact of the embodiment-specific and task-specific parametrization introduced in the state encoder, we conduct an ablation study by removing the adaptive parameters $\mathbf{p}_s^{\mathcal{E}}, \theta_s^{\mathcal{E}}$ in the structure encoder and $\theta_m^{(\mathcal{E},\mathcal{T})}$ the motion encoder. In this study, the matching module σ is used in all models, where we provide additional results without using it in Appendix D.2. The results show that the model without adaptive parameters in the structure encoder (row 1) performs well in many tasks, likely due to the universality of joint-level input/output representation, which allows for compositional generalization. However, adding embodiment-specific parameters consistently improves performance across all tasks, indicating the benefit of capturing embodiment-specific knowledge. The model without adaptive parameters in the motion encoder (row 2) remains competitive with the model including them (row 3) in many tasks but fails in the `hop` task

Table 2: Ablation study on the architectural components.

f_s	f_m	σ	Unseen Emb.							Seen Emb.	Avg.
			hopper			wolf		reacher-four		walker	
			hop	hop-bwd.	stand	walk	run	easy	hard	walk-bwd.	
✗	✓	✗	5.3±3.8	46.7±9.3	14.9±5.9	64.5±11.1	53.5±8.0	-3.2±3.6	4.3±5.1	66.6±10.5	31.6
✓	✗	✗	13.1±5.6	**88.1±1.7**	75.0±5.7	71.8±6.9	53.7±8.0	4.1±7.2	3.4±4.1	48.1±5.4	44.7
✓	✓	✗	22.8±5.9	86.3±1.6	**83.4**±4.2	79.7±6.4	57.3±6.7	54.9±9.5	24.8±8.2	73.4±5.0	60.3
✓	✓	✓	**49.1±6.1**	87.2±1.6	82.5±4.9	**91.7±5.1**	**67.3±3.1**	**56.1±8.8**	**50.8±10.6**	**84.3±5.7**	**71.1**

Table 3: Ablation study on the adaptive parameters in the state encoder.

$\mathbf{p}_s^{\mathcal{E}}, \theta_s^{\mathcal{E}}$	$\theta_m^{(\mathcal{E},\mathcal{T})}$	Unseen Emb.							Seen Emb.	Avg.
		hopper			wolf		reacher-four		walker	
		hop	hop-bwd.	stand	walk	run	easy	hard	walk-bwd.	
✗	✓	37.6±5.6	76.8±3.6	68.0±5.6	85.4±4.5	47.2±2.9	17.4±8.9	0.4±0.9	59.2±8.1	49.0
✓	✗	16.4±6.4	84.7±2.7	**84.0±3.7**	89.2±2.6	**71.0±3.1**	**70.8±7.9**	48.1±9.4	5.3±1.1	58.7
✓	✓	**49.1±6.1**	**87.2±1.6**	82.5±4.9	**91.7±5.1**	67.3±3.1	56.1±8.8	**50.8±10.6**	**84.3±5.7**	**71.1**

of `hopper` and the `walk-bwd.` task of `walker`. This failure, even in a seen embodiment (`walker`), suggests that certain unique motions cannot be adequately captured without adaptive parameters. Overall, introducing adaptive parameters in both encoders yields the best performance.

Ablation on the Number of Demonstrations. In Figure 5, we plot the performance of our model and the two best-performing baselines—one from modular policy learning and the other from few-shot IL—by varying the number of demonstrations. As expected, the performance of our model consistently increases with the number of demonstrations provided. The consistent superiority of our model in few-shot behavior cloning, across varying numbers of demonstrations, highlights the necessity of (1) a powerful encoder capable of handling unseen embodiments, and (2) a few-shot policy adaptation mechanism that enables robust few-shot learning on unseen tasks. These components are crucial for achieving simultaneous few-shot generalization to both unseen embodiments and tasks.

We provide more ablation studies on meta-training task composition in Appendix D.3

6 Conclusion

We addressed the challenging problem of few-shot behavior cloning with unseen embodiments and tasks in continuous control. Our framework, Meta-Controller, effectively handles diverse embodiments using two key components: the state encoder and the matching-based policy network. Leveraging the modular nature of joint-level input/output representations, our state encoder extracts transferable features about the morphology and dynamics of the embodiment, capturing both specific and shared knowledge. The matching-based policy network uses these features to infer task structures from few-shot demonstrations, enabling robust imitation without overfitting. Experiments showed that our model generalizes well to unseen embodiments and tasks with only five demonstrations.

Acknowledgments and Disclosure of Funding

This work was in part supported by the National Research Foundation of Korea (RS-2024-00351212 and RS-2024-00436165), the Institute of Information & communications Technology Planning & Evaluation (IITP) (RS-2022-II220926, RS-2024-00509279, RS-2021-II212068, RS-2022-II220959, and RS-2019-II190075) funded by the Korea government (MSIT), and NAVER-Intel Co-Lab.

References

[1] Pytorch lightning. *GitHub*, 3, 2019.

[2] E. Ben Zaken, Y. Goldberg, and S. Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2022.

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

[4] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[5] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

[6] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. 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.

[7] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

[8] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[9] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[10] H. Furuta, Y. Iwasawa, Y. Matsuo, and S. S. Gu. A system for morphology-task generalization via unified representation and behavior distillation. In *The Eleventh International Conference on Learning Representations*, 2022.

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

[12] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei. Metamorph: Learning universal controllers with transformers. In *International Conference on Learning Representations*, 2021.

[13] K. Hakhamaneshi, R. Zhao, A. Zhan, P. Abbeel, and M. Laskin. Hierarchical few-shot imitation with skill transition models. In *International Conference on Learning Representations*, 2021.

[14] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2023.

[15] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[16] S. Hong, D. Yoon, and K.-E. Kim. Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning. In *International Conference on Learning Representations*, 2021.

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

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

[19] D. Kim, S. Cho, S. Kim, C. Luo, and S. Hong. Chameleon: A data-efficient generalist for dense visual prediction in the wild. *arXiv preprint arXiv:2404.18459*, 2024.

[20] D. Kim, J. Kim, S. Cho, C. Luo, and S. Hong. Universal few-shot learning of dense prediction tasks with visual token matching. In *The Eleventh International Conference on Learning Representations*, 2022.

[21] S. Kim, S. Shen, D. Thorsley, A. Gholami, W. Kwon, J. Hassoun, and K. Keutzer. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794, 2022.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. In *International Conference on Learning Representations*, 2021.

[24] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[25] R. K. Mahabadi, S. Ruder, M. Dehghani, and J. 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)*, pages 565–576, 2021.

[26] A. Majumdar, K. Yadav, S. Arnaud, J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, T. Wu, J. Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36:655–677, 2023.

[27] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*, 2019.

[28] T. Schmied, M. Hofmarcher, F. Paischer, R. Pascanu, and S. Hochreiter. Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] I. Schubert, J. Zhang, J. Bruce, S. Bechtle, E. Parisotto, M. Riedmiller, J. T. Springenberg, A. Byravan, L. Hasenclever, and N. Heess. A generalist dynamics model for control. *arXiv preprint arXiv:2305.10912*, 2023.

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

[31] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 2020.

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

[33] Q. Vuong, S. Levine, H. R. Walke, K. Pertsch, A. Singh, R. Doshi, C. Xu, J. Luo, L. Tan, D. Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

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

[35] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[36] M. Xu, Y. Lu, Y. Shen, S. Zhang, D. Zhao, and C. Gan. Hyper-decision transformer for efficient online policy adaptation. In *The Eleventh International Conference on Learning Representations*, 2022.

[37] M. Xu, Y. Shen, S. Zhang, Y. Lu, D. Zhao, J. Tenenbaum, and C. Gan. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR, 2022.

[38] D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[39] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

[40] Y. Zhou, S. Sonawani, M. Phielipp, S. Stepputtis, and H. Amor. Modularity through attention: Efficient training and transfer of language-conditioned policies for robot manipulation. In *6th Annual Conference on Robot Learning*, 2022.

Appendix

This document provides the contents that are not included in the main text due to the page limit.

A Limitations and Broader Impacts

Limitations. Despite the promising results, the Meta-Controller framework has several limitations that warrant further investigation. One significant limitation is its reliance on simulated environments for training and evaluation, which may not fully capture the complexities and variabilities of real-world scenarios. This gap between simulation and reality could hinder the direct application of the framework to practical robotic tasks. Additionally, while the framework shows robust performance in the few-shot learning setting, it may still face challenges in environments with highly stochastic dynamics or in tasks that require extensive long-term planning. Another limitation is the assumption of a unified joint-level representation, which, although effective for the embodiments tested, may not generalize well to robots with significantly different morphologies or actuation mechanisms. Furthermore, the computational complexity of the transformer-based architecture could pose scalability issues for real-time applications, especially on resource-constrained robotic platforms. Lastly, the ethical implications of deploying such adaptable robotic systems need to be carefully considered to prevent potential misuse in sensitive areas.

Broader Impacts. The proposed Meta-Controller framework for few-shot imitation learning has several significant broader impacts. Firstly, its ability to generalize across various robot embodiments and tasks with minimal demonstrations can greatly enhance the adaptability and versatility of robotic systems in dynamic environments. This flexibility is crucial for deploying robots in real-world scenarios where they must quickly learn and adapt to new tasks without extensive retraining. Secondly, by leveraging a modular approach that decouples embodiment-specific and task-specific knowledge, the framework promotes efficient knowledge transfer, potentially reducing the computational resources and time required for training new robotic tasks. However, the deployment of such adaptive systems also raises concerns about their potential misuse. For instance, advanced robotic systems equipped with this technology could be employed in surveillance or military applications, leading to ethical and privacy issues. Therefore, it is essential to implement safeguards and ethical guidelines to ensure the responsible use of these advancements. Furthermore, the reliance on simulated environments for training and evaluation might limit the transferability of the results to real-world conditions, necessitating further research to bridge this gap.

B More Details on Experiments

In this section, we describe the details of the tasks and embodiments in the used dataset and its collection processes.

B.1 More details on Datasets and Environment

Embodiments and Tasks in DeepMind Control (DMC) suite. The DeepMind Control (DMC) suite benchmark [31] is a collection of continuous control tasks implemented using the MuJoCo physics engine [30]. It provides a variety of simulated environments to test and develop reinforcement learning algorithms. The environments in DMC are designed to be diverse and challenging, promoting the development of agents capable of handling a wide range of tasks. To reduce the complexity of the problem and facilitate knowledge acquisition from diverse tasks and embodiments, we only consider embodiments that operate on a 2D coordinate space. We then augment the dataset with a number of tasks used in [34, 14], including two tasks from newly added embodiments, `walk` and `run`) task of `wolf`, where we use the same reward function as the `walk` and `run`) task of `walker`, respectively. From a total of 38 tasks from 13 embodiments, we select an unseen embodiment with seen tasks (`reacher-four`, `wolf`), a seen embodiment with an unseen task (`walker`), and an unseen embodiment with unseen tasks (`hopper`) as held-out evaluation tasks for comprehensive analysis. For the exact setup of the newly added embodiments and tasks, please refer to the github repository of NerveNet [3] and TD-MPC2 [4]. We list all embodiments and tasks used in our experiments in Table 4.

[3] `https://github.com/WilsonWangTHU/NerveNet`

[4] `https://github.com/nicklashansen/tdmpc2/tree/main`

Table 4: List of all embodiments and tasks used in our experiments.

Embodiment ($\mathcal{E}$)	Task ($\mathcal{T}$)
acrobot	swingup
ball in cup	catch
	spin
cartpole	balance
	swingup
cartpole two	poles
cheetah	flip
	flip backwards
	jump
	legs up
	lie down
	run back
	run backwards
	run front
	stand back
	stand front
hopper	hop
	hop backwards
	stand
pendulum	spin
	swingup
pointmass	easy
reacher	hard
reacher four	easy
	hard
reacher three	hard
walker	arabesque
	backflip
	flip
	headstand
	legs up
	lie down
	run
	stand
	walk
	walk backwards
wolf	run
	walk

Dataset Collection. For each task, we train the DrQ-v2 agent [38] using an online replay buffer. The agent is trained with the hyperparameters specified in [38], generating 1000 to 2000 demonstration trajectories for each task. For the `wolf` embodiment that was not originally included in the experiments of the DrQ-v2, we use the same hyperparameters as those used for the corresponding tasks of `walker`. After training, we filter out low-quality demonstrations with cumulative rewards smaller than 10 to ensure the reliability of the training data.

Data Processing. The states of various embodiments and tasks in DMC consist of two joint-level observations, such as the relative position and velocity of each joint, as well as extrasensory observations that are required to solve each task (*e.g.,* in `reacher` embodiment, the distance between its tip position and the goal position is given as an extrasensory observation). The other per-joint attributes such as the movement axis or motion type can be extracted by the kinematic tree of each embodiment provided in XML format. To ensure compatibility in practical applications, we use

only the states defined in the default DMC setting. There are two different types of joints in the embodiments we consider: hinge joints and slide joints. Since these joints have distinct motion characteristics (angular motion and linear motion, respectively), we encode the states of each type separately to avoid semantic conflict. We standardize the relative position using a fixed axis (*e.g.,* the z-axis in the xz plane) instead of using raw, unstandardized input. For joints with partially existing states (e.g., the free joint of the torso), we zero-pad the non-existent states. To handle extrasensory states provided by a specific embodiment, we concatenate all extrasensory observations into a single vector and project them into a single joint token $\mathbf{g}_t \in \mathbb{R}^d$. To this end, we introduce embodiment-specific linear projection layers for the embodiments having extrasensory observations. Lastly, for each embodiment, we normalize the states by scaling them with min-max statistics calculated across the data for each embodiment.

B.2 More Details on Implementation

Episode Sampling Strategy. In the episode sampling procedure in Eq. (10), it is crucial to ensure consistency between the query $\mathcal{Q}$ and the few-shot demonstration $\mathcal{D}$ for effective learning from demonstrations. Since we use a replay buffer as our meta-training dataset which encompasses demonstrations obtained by learning agents at diverse stages, naively applying a uniform sampling strategy on the entire replay buffer results in a high probability of selecting inconsistent queries and demonstrations. Instead, to ensure task consistency in each episode during episodic meta-learning (Section 3.3), we sample the conditioning demonstrations $\mathcal{D}$ and the query data $\mathcal{Q}$ for Eq. (10) from a temporal segment of each replay buffer $\mathcal{B}_{(\mathcal{E},\mathcal{T})}$. In other words, the demonstrations in $\mathcal{D}$ and $\mathcal{Q}$ are obtained from expert agents at adjacent training epochs. In all experiments, we use a temporal segment size of 10 for episodic meta-learning.

Training Details. All models are trained for 200,000 iterations using the Adam optimizer [22] and a *poly* learning rate scheduler [24] with a base learning rate of 2×10^{-4}. After training, we fine-tune all models for 10,000 iterations with a fixed learning rate of 2×10^{-4}, except for HDT [36], which requires a higher learning rate of 10^{-2}. For meta-training, we train the model with $8 \times$ RTX A6000 GPUs for approximately 25 hours, and we fine-tune the model on each task with $1 \times$ RTX A6000 GPU for approximately 2 hours.

As the baselines used in our experiments have not been demonstrated in our few-shot behavior cloning setting with unseen embodiments and tasks, we modify their base architecture or learning framework for fair comparison. When fine-tuning HDT, we first adapt the embodiment-specific head parameters of HDT with the episodic meta-learning objective, as it does not generate appropriate adapter parameters with untrained head parameters of novel embodiment. We then obtain adapter parameters by forwarding every temporal chunk of the few-shot demonstration with the hyper-network and initialize adapter parameters by the mean of the outputs. Then, following the HDT procedure, we fine-tune only the adapter parameters using the BC objective. For fine-tuning the PDT+PEFT baseline, we apply PEFT techniques to the PDT model meta-trained without task-specific parameters. For the few-shot adaptation of L2M, the learnable modulation pool and the corresponding low-rank projection matrices are fine-tuned.

Architectural Details. For transformer models, we employ a Pre-LN transformer layer [35], which consists of a self-attention layer followed by a 2-layer MLP with $4\times$ hidden dimension size. We use GELU [15] activation for the self-attention layers. To enhance the expressiveness of our matching-based policy network, we implement the matching module σ in Eq (8) with multi-head cross-attention [32], following Kim et. al. [20] and Kim et. al. [19]. We list the hyper-parameters of Meta-Controller in Table 5.

Implementation Framework. We implemented our model based on PyTorch Lightning [1] which supports both Intel Gaudi-v2 (HABANA) and NVIDIA AI accelerators (CUDA). We provide the code for both systems on separate branches in the GitHub repository.

C Additional Results

In this section, we provide additional results and analysis on computational efficiency and robustness.

Table 5: Hyper-parameters of Meta-Controller used in our experiments.

Hyper-parameters	Value
Number of demonstrations used in each episode	4
Global Batch Size	64
Hidden dimension	512
Attention heads	4
Low-rank for structure encoder f_s	16
Low-rank for motion encoder f_m	16
Layerscale initialization	1
Training iteration	200,000
Learning rate warmup iterations	1000
Base learning rate	2×10^{-4}

Table 6: 3-shot behavior cloning results on DeepMind Control suite.

	Unseen Emb.							Seen Emb.	Avg.
Emb. ($\mathcal{E}$)	`hopper`			`wolf`		`reacher-four`		`walker`	
Task ($\mathcal{T}$)	`hop`	`hop-bwd.`	`stand`	`walk`	`run`	`easy`	`hard`	`walk-bwd.`	
MT-DT	7.9 ± 1.5	40.0 ± 8.2	5.6 ± 0.9	49.8 ± 9.9	32.2 ± 7.1	10.1 ± 5.0	4.5 ± 3.7	58.2 ± 6.7	25.2
PDT+PEFT	2.5 ± 1.2	42.6 ± 7.1	8.3 ± 2.2	39.0 ± 10.8	37.9 ± 5.7	8.3 ± 5.6	6.7 ± 5.1	**60.0 ± 10.7**	25.7
MetaMorph	13.4 ± 2.8	76.8 ± 2.5	33.9 ± 4.3	51.8 ± 8.4	26.4 ± 2.5	2.5 ± 3.7	4.0 ± 4.2	18.8 ± 8.2	30.1
MTGv2	15.5 ± 3.0	**77.4 ± 2.7**	29.0 ± 2.7	65.0 ± 5.9	25.1 ± 4.0	-4.4 ± 1.9	2.8 ± 2.4	19.4 ± 6.6	27.9
Ours	**34.8 ± 4.0**	75.3 ± 5.9	**68.3 ± 6.5**	**78.2 ± 6.0**	**43.6 ± 4.7**	**51.3 ± 8.9**	**73.2 ± 9.1**	53.5 ± 7.8	**57.0**

Table 7: 5-shot behavior cloning results on variations of embodiments in DeepMind Control suite.

Target Env. ($\mathcal{E}, \mathcal{T}$)	`hopper-stand`				`wolf-walk`		Avg.
Emb. Variations	foot length		calf-thigh ratio		front-back leg ratio		
	50%	200%	2:1	1:2	2:1	1:2	
MT-DT	7.5 ± 5.4	23.0 ± 8.2	2.1 ± 0.7	11.0 ± 2.3	73.1 ± 5.8	52.1 ± 11.9	28.1
PDT+PEFT	3.2 ± 1.2	14.4 ± 7.8	4.4 ± 3.4	15.0 ± 6.1	65.3 ± 6.8	47.8 ± 10.8	25.0
MetaMorph	5.8 ± 1.7	39.9 ± 9.5	5.7 ± 4.7	29.0 ± 6.2	60.3 ± 7.8	71.2 ± 7.9	35.3
MTGv2	8.2 ± 2.2	13.3 ± 5.1	4.6 ± 1.6	24.6 ± 3.9	73.8 ± 5.5	4.0 ± 2.1	21.4
Ours	**15.7 ± 6.8**	**43.0 ± 10.0**	**12.5 ± 6.9**	**75.3 ± 6.6**	**90.3 ± 3.3**	**73.2 ± 9.7**	**51.7**

C.1 Additional Results on 3-Shot Behavior Cloning

To investigate performance in lower-shot settings, we evaluate our model with 3-shot behavior cloning on tasks presented in table 1. As shown in Table 6, our model consistently outperforms baseline approaches, demonstrating robustness even with fewer demonstrations. This result highlights the model's adaptability to fewer examples while maintaining reliable generalization across different tasks and embodiments.

C.2 Additional Results on Embodiment Variations

We further assess model performance on six additional embodiments by adjusting physical parameters, such as joint lengths (e.g., foot length of hopper) and ratios among joints (e.g., calf-thigh ratio of hopper, front leg-back leg ratio of wolf). These modifications can make the tasks harder, as the original embodiments are optimized for specific tasks. In Table 7, we compare Meta-Controller with the high-performing baselines from Table 1. Our model consistently outperforms all baselines in these challenging variants, demonstrating the robustness and adaptability of our approach to variations in embodiment configurations.

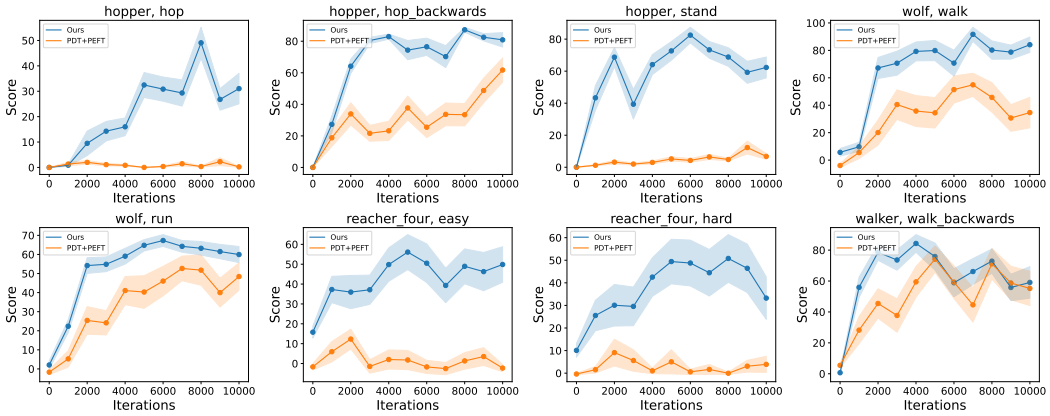


Figure 6: Learning curves of ours and PDT+PEFT in 5-shot settings.

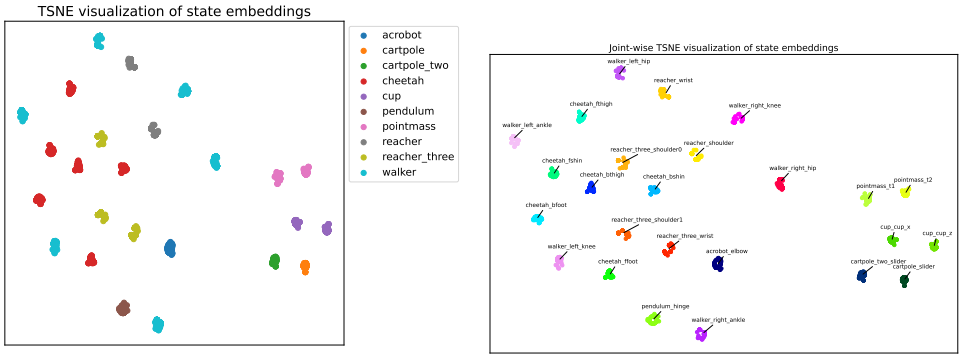


Figure 7: t-SNE visualization of embeddings of state encoder f. In the left figure, embeddings of the same embodiment have the same color. In the right figure, embeddings of the same joint have the same color.

C.3 Additional Qualitative Results

We provide additional qualitative results about the behavior of the few-shot agents in the tasks we used for evaluation. In Figure 11-18, we plot the rendered images of a single evaluation rollout by the `dm_control` library [31] from the initial state ($t = 0$) to the states unrolled by each agent, by skipping every ten timesteps in the visualization.

C.4 Visualization of Learning Curves

We provide learning curves of Meta-Controller and PDT+PEFT in Figure 6. Due to the modular nature of our structure encoder, our model not only achieves better performance but also converges much more quickly than PDT+PEFT in every task.

C.5 Visualization of Learned Embeddings

To gain insights into how our model represents different embodiments, we provide a t-SNE visualization of embedding space of features obtained by structure encoder f_s in Figure 7. First, we observe that the embeddings are clustered by the joints of each embodiment, and the clusters corresponding the same embodiment are located nearby. Also, we note that the embeddings of slide joints (e.g., `cartpole`, `cartpole-two`, `cup`, `pointmass`) and the embeddings of hinge joints (e.g., `reacher`, `reacher-three`, `walker`, `acrobot`, `cheetah`, `pendulum`) are separated in the right and the left regions. Thus, the state encoder captures both embodiment-specific and joint-specific knowledge, providing rich features to the policy network.

Table 8: Inference time of ours and VC-1 [26], measured in a single NVIDIA RTX 3090 GPU.

Model	Ours	VC-1 (ViT-L backbone)
Inference Time (ms)	17.8	26.8

Table 9: Relative inference time of each module of ours, measured in a single NVIDIA RTX 3090 GPU.

Module	State encoder (f)	Action encoder (g)	Action decoder (h)	Matching (σ)
Inference Time (%)	31.54	31.83	34.08	2.55

C.6 Computational Complexity Analysis

Table 8 presents the inference time comparisons between our model and VC-1 [26], which is a ViT-based foundation model designed for behavior cloning of continuous control tasks. Despite VC-1's high performance in visual tasks, its transformer-based architecture incurs notable computational costs. Our model achieves faster inference times, highlighting its efficiency relative to VC-1. We also note that models like RT-2 [3], which are designed for real-time robotic manipulation, typically involve architectures such as a 40-layer ViT and a large language model (LLM) with 3 billion parameters, requiring significantly higher computational resources than our approach.

Additionally, Table 9 provides a breakdown of relative inference times for each module in our model, demonstrating that the majority of computation time is allocated to the transformer-based encoders and decoders. In this context, advancements in optimizing transformers, such as sparse attention mechanisms [5], model pruning [21], and knowledge distillation [27], can enhance inference speeds. Since these techniques are orthogonal to ours, they can be naturally incorporated into our method for resource-constrained robotic platforms that require high speed inference.

C.7 Failure Case Analysis

We present visualizations of scenarios where the Meta-Controller performs less effectively and cumulative rewards over time for each scenario in Figures 8 and 9, respectively. In these failure cases, we observed that agents struggle to obtain rewards until they reach a specific posture. Once they achieve this posture (highlighted by the red box in Figure 8), they begin to solve the task effectively. This pattern is also reflected in Figure 9, where rewards remain near zero until a certain timestep, after which they rise consistently. This result implies that encouraging the agent to reach states similar to those in the demonstrations (e.g., via exploration strategies) would improve performance in challenging few-shot scenarios.

C.8 Robustness Analysis under Noise

To assess robustness in noisy environments, we introduce varying levels of noise to the transition dynamics and measure the resulting performance. Random noise sampled from $\mathcal{U}[-n, n]$ was added to the agent's action at each timestep, with three noise levels $n \in [2\%, 5\%, 10\%]$ of the action range. Figure 10 plots the rewards of our model at each noise level compared to experts.

The results indicate that our method maintains its performance across many tasks as noise levels increase, showing its robustness under stochastic environments. Interestingly, for tasks such as reacher-four, the performance increases with higher noise levels, likely due to the exploration effect induced by stochastic transitions. This robustness under stochastic dynamics indicates potential for the model's application in real-world scenarios where environmental variability is common.

D Additional Ablation Studies

In this section, we provide additional results on ablation studies.

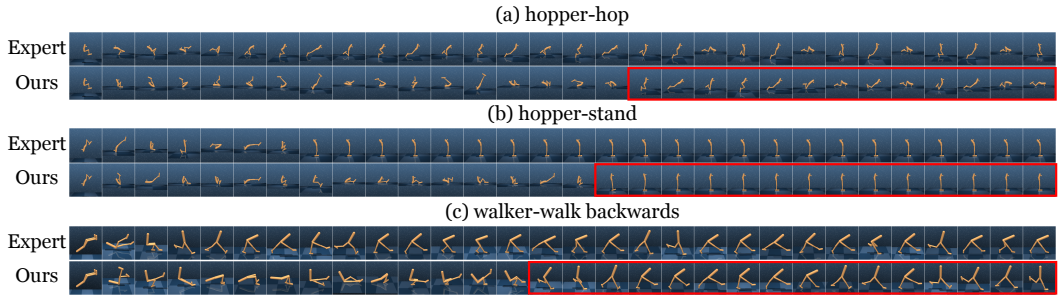


Figure 8: Failure cases of our model. As indicated by red boxes, the agent begins to solve tasks effectively after it reaches a specific posture.

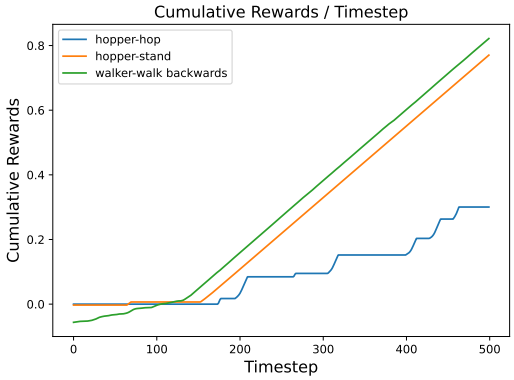


Figure 9: Cumulative rewards of the failure cases in Figure 8.

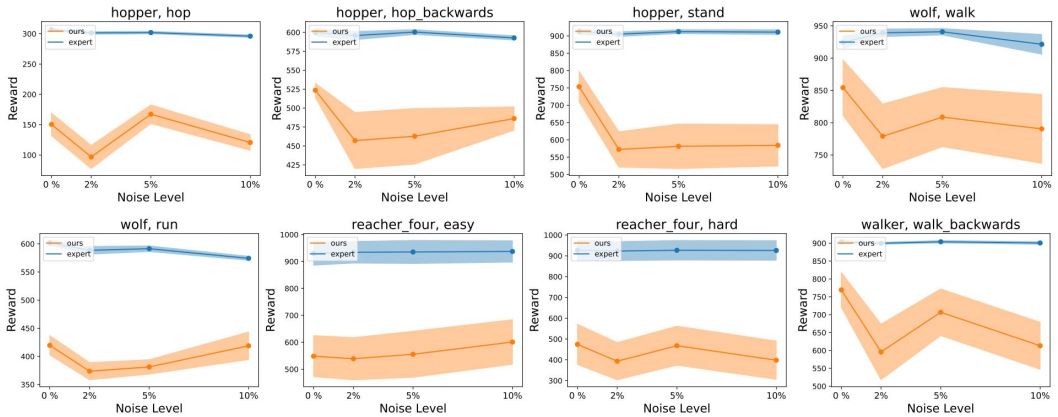


Figure 10: Robustness analysis of our model in the presence of various noise levels in transition dynamics. Random noise $\epsilon \sim \mathcal{U}[-n, n]$ is added to each action, where n varies over 2%, 5%, and 10% of the action range. The shaded region represents the standard error.

Table 10: Additional ablation study on the architectural components. Model variants are evaluated in 5-shot setting.

f_s	f_m	σ	g	h	Unseen Emb.							Seen Emb.	Avg.
					hopper			wolf		reacher-four		walker	
					hop	hop-bwd.	stand	walk	run	easy	hard	walk-bwd.	
✗	✗	✗	✗	✗	6.4 ± 2.4	58.2 ± 8.4	7.7 ± 1.1	32.4 ± 8.7	27.4 ± 6.4	14.6 ± 7.5	4.9 ± 4.1	0.2 ± 1.7	19.0
✗	✓	✓	✓	✓	5.7±2.7	44.7±10.3	15.5±3.9	64.5±9.9	53.3±8.5	9.3±7.0	0.0±0.8	48.4±10.4	30.2
✓	✓	✓	✗	✗	25.7 ± 5.5	**89.5 ± 1.4**	42.5 ± 9.3	83.9 ± 6.1	60.7 ± 3.4	53.7 ± 8.9	**53.2 ± 10.7**	71.6 ± 6.0	60.1
✓	✓	✓	✓	✓	**49.1±6.1**	87.2±1.6	**82.5±4.9**	**91.7±5.1**	**67.3±3.1**	**56.1±8.8**	50.8±10.6	**84.3±5.7**	**71.1**

Table 11: Ablation study on the adaptive parameters in the state encoder *without* the matching module in the policy network.

$\mathbf{p}_s^{\mathcal{E}}, \theta_s^{\mathcal{E}}$	$\theta_m^{(\mathcal{E},\mathcal{T})}$	Unseen Emb.							Seen Emb.	Avg.
		hopper			wolf		reacher-four		walker	
		hop	hop-bwd.	stand	walk	run	easy	hard	walk-bwd.	
✗	✓	22.6±3.0	68.1±3.0	48.1±5.7	71.7±4.7	32.6±3.9	13.7±6.5	10.3±5.2	53.1±9.2	40.0
✓	✗	**40.0±5.8**	**91.6±0.8**	53.2±9.8	78.3±8.4	**61.8±5.1**	**78.1±7.7**	**36.1±8.7**	2.3±1.8	55.2
✓	✓	22.8±5.9	86.3±1.6	**83.4±4.2**	**79.7±6.4**	57.3±6.7	54.9±9.5	24.8±8.2	**73.4±5.0**	**60.3**

D.1 Additional Ablation Studies on Architectural Components

Table 10 presents the additional ablation studies on key architectural components:structure encoder f_s, motion encoder f_m, action encoder g, action decoder h and matching module σ. Consistent with the discussion in Section 5.3, we observe that removing the structure encoder f_s (row 2) significantly reduces performance on both seen and unseen embodiments. This indicates that the structure encoder captures essential morphology-related knowledge that enables cross-embodiment generalization.

We also ablate the action encoder g and action decoder h (row 3), observing that removing them result in decreased adaptability and performance. The action encoder plays a key role in transforming raw action into a latent representation, which effectively enhancing expressibility of the policy network and enables adaptation to various unseen tasks with non-convex relationships between states and actions. Additionally, by encoding actions along the temporal axis, the model can construct a pool of transferrable action features related to local motor skills, which facilitates efficient transfer to unseen tasks that share modular skills but involve different skill combinations.

D.2 Additional Ablation studies on Adaptation Mechanism

To further analyze the adaptation mechanism of our model, we conduct an additional ablation study by ablating the adaptive parameters in the state encoder *without* using the matching module in the policy network. On average, using both adaptive parameters (row 3) achieves the best performance. However, we note that the model variant without adaptive parameters in the motion encoder (row 2) achieves higher performance than using the parameters (row 3) in many tasks. We conjecture that this is due to over-fitting on the few-shot demonstrations, as the adaptive parameters in the motion encoder are both specific to the embodiment and task and more prone to over-fitting. However, as we discussed with Table 3 in Section 5.3, such trends are not found in general (except for the `easy` task of `reacher-four`). This indicates that in few-shot learning settings, the PEFT techniques must be employed together with a robust architecture such as matching to exploit its effectiveness maximally.

D.3 Additional Ablation Studies on Meta-Training Task Composition

To further understand how the composition of meta-training tasks affects performance, we conduct experiments with varying subsets of embodiments and tasks in Table 12. We select 3 combinations of training tasks, where we remove 4 embodiments from the original 10 embodiments. Then, we performed 5-shot behavior cloning experiments on the 8 tasks presented in table 1.

According to Table 12, we observe that using all embodiments outperforms the baselines in most tasks, indicating that a diverse set of embodiments makes the model robust to unseen embodiments. Our

Table 12: Ablation study on embodiments included in the meta-training data. We report the 5-shot score by removing four embodiments from the meta-training data, where we choose three different subsets of the removing embodiments. $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ denote {`cartpole-two`, `cup`, `pendulum`, `pointmass`}, {`cartpole-two`, `pendulum`, `pointmass`, `reacher-three`}, and {`cartpole-two`, `cheetah`, `cup`, `reacher-three`}, respectively.

	Unseen Emb.							Seen Emb.	Avg.
Emb. ($\mathcal{E}$)	`hopper`			`wolf`		`reacher-four`		`walker`	
Task ($\mathcal{T}$)	`hop`	`hop-bwd.`	`stand`	`walk`	`run`	`easy`	`hard`	`walk-bwd.`	
w/o $\mathcal{E}_1$	15.9 ± 5.5	76.6 ± 6.8	71.5 ± 7.1	83.2 ± 4.1	62.1 ± 5.0	**82.1 ± 7.6**	41.8 ± 9.8	70.7 ± 6.0	63.0
w/o $\mathcal{E}_2$	16.3 ± 6.0	85.2 ± 2.8	81.6 ± 7.3	83.9 ± 5.2	**70.1 ± 3.4**	5.6 ± 6.9	4.6 ± 5.1	78.8 ± 7.3	53.3
w/o $\mathcal{E}_3$	26.6 ± 8.0	**88.8 ± 1.1**	68.6 ± 6.1	83.6 ± 6.4	59.0 ± 2.9	4.6 ± 6.5	3.8 ± 4.1	81.8 ± 5.1	52.1
Use All Embodiments	**49.1± 6.1**	87.2 ± 1.6	**82.5 ± 4.9**	**91.7 ± 5.1**	67.3 ± 3.1	56.1 ± 8.8	**50.8 ± 10.6**	**84.3 ± 5.7**	**71.1**

findings also show that it is crucial to include embodiments with morphology and dynamics similar to the downstream tasks in the meta-training dataset. For example, removing the `reacher-three` task (as seen in row 2 and row 3) significantly drops the performance of the `reacher-four` task. This result reveals that embodiments with similar dynamics or morphological features can facilitate more effective knowledge transfer during meta-testing, suggesting that the diversity of data greatly impacts performance.

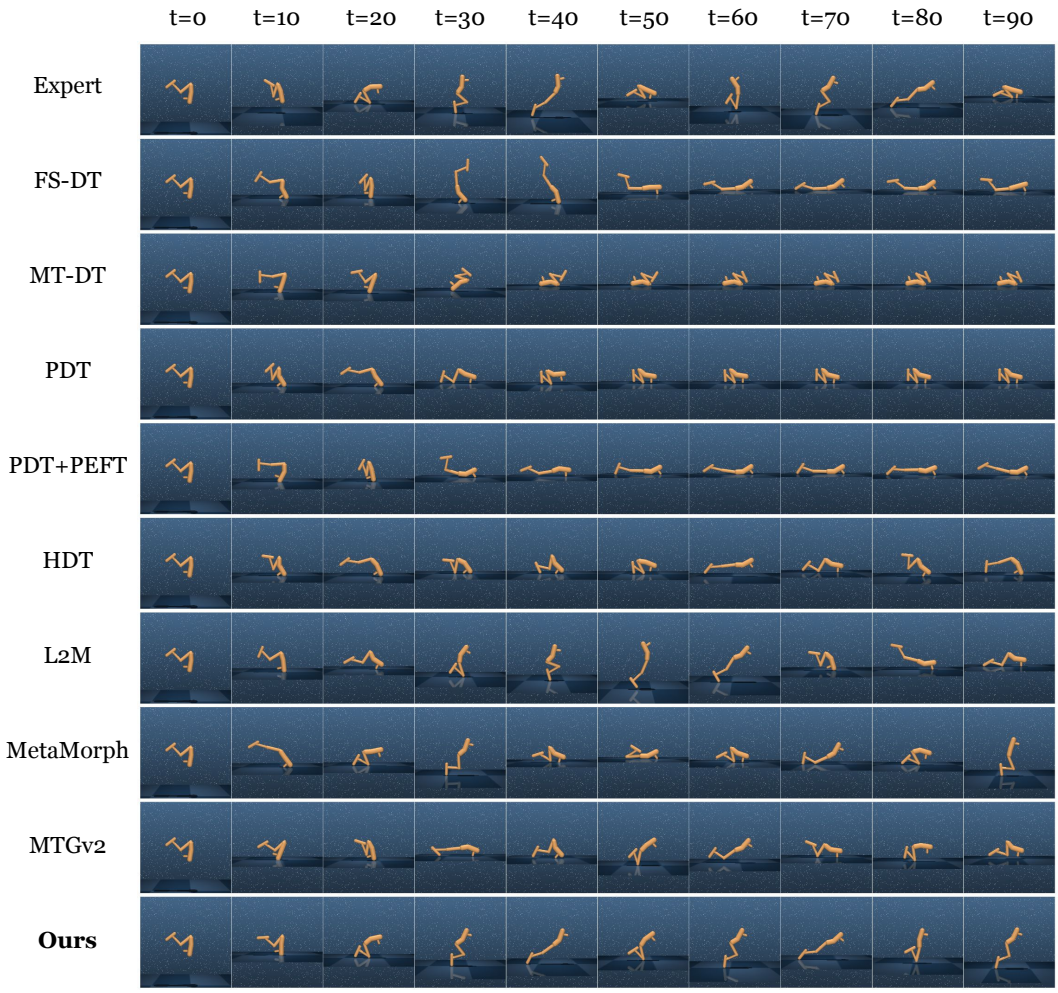


Figure 11: Additional Qualitative Results on `hopper-hop` task.

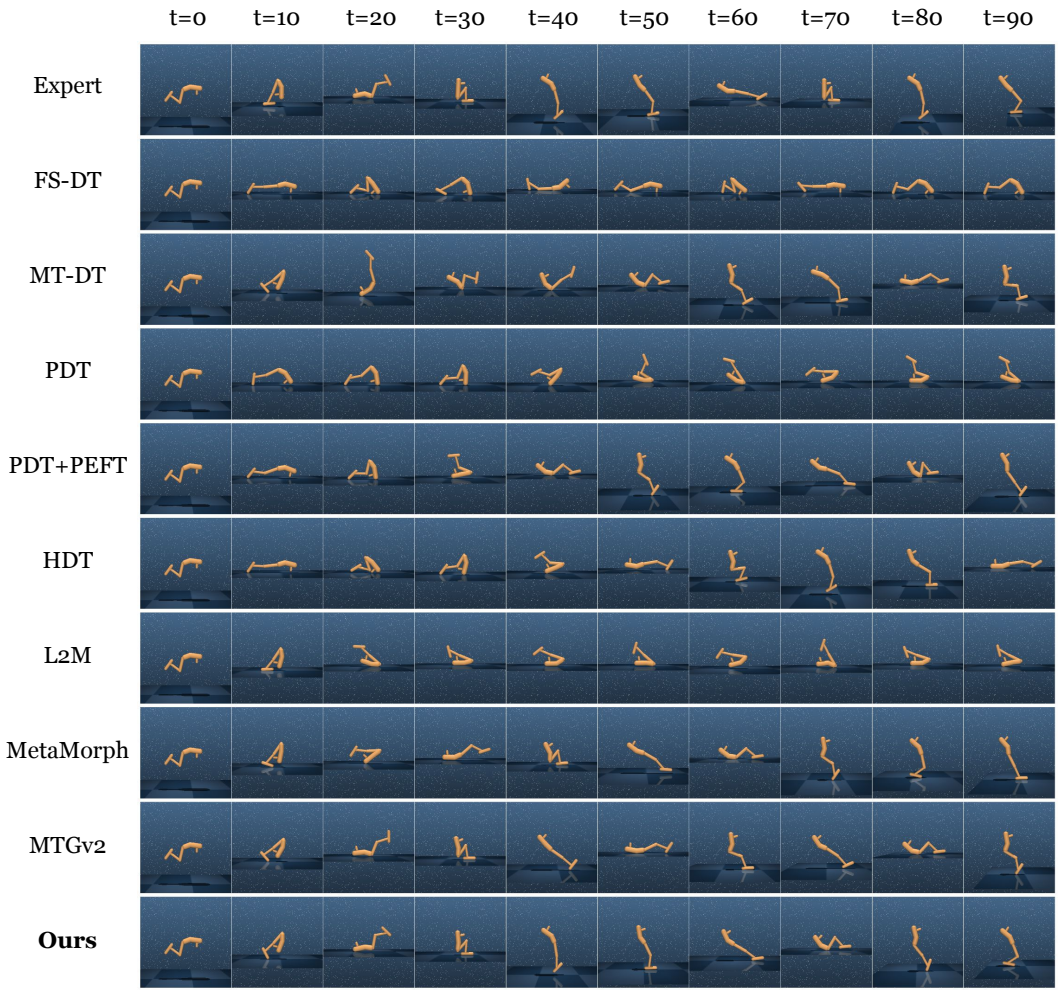


Figure 12: Additional Qualitative Results on `hopper-hop backwards` task.

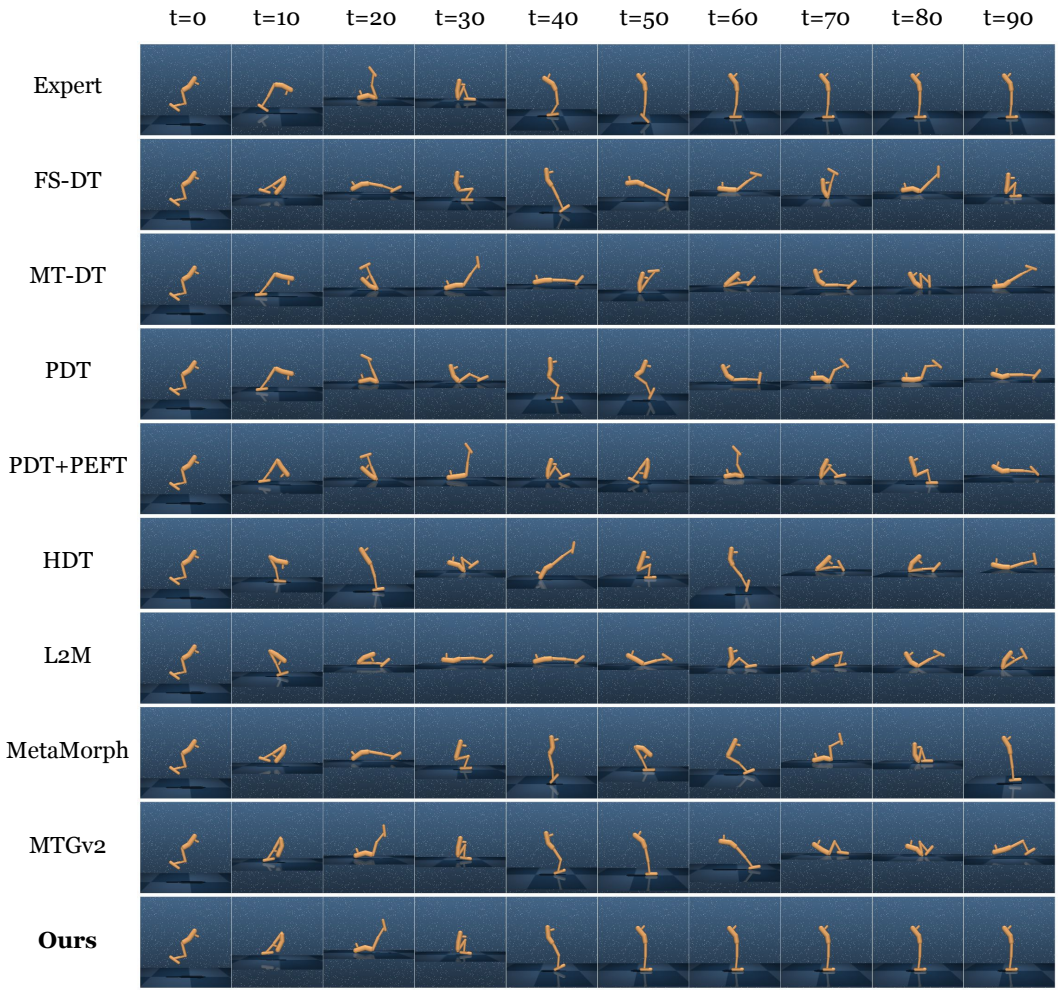


Figure 13: Additional Qualitative Results on `hopper-stand` task.

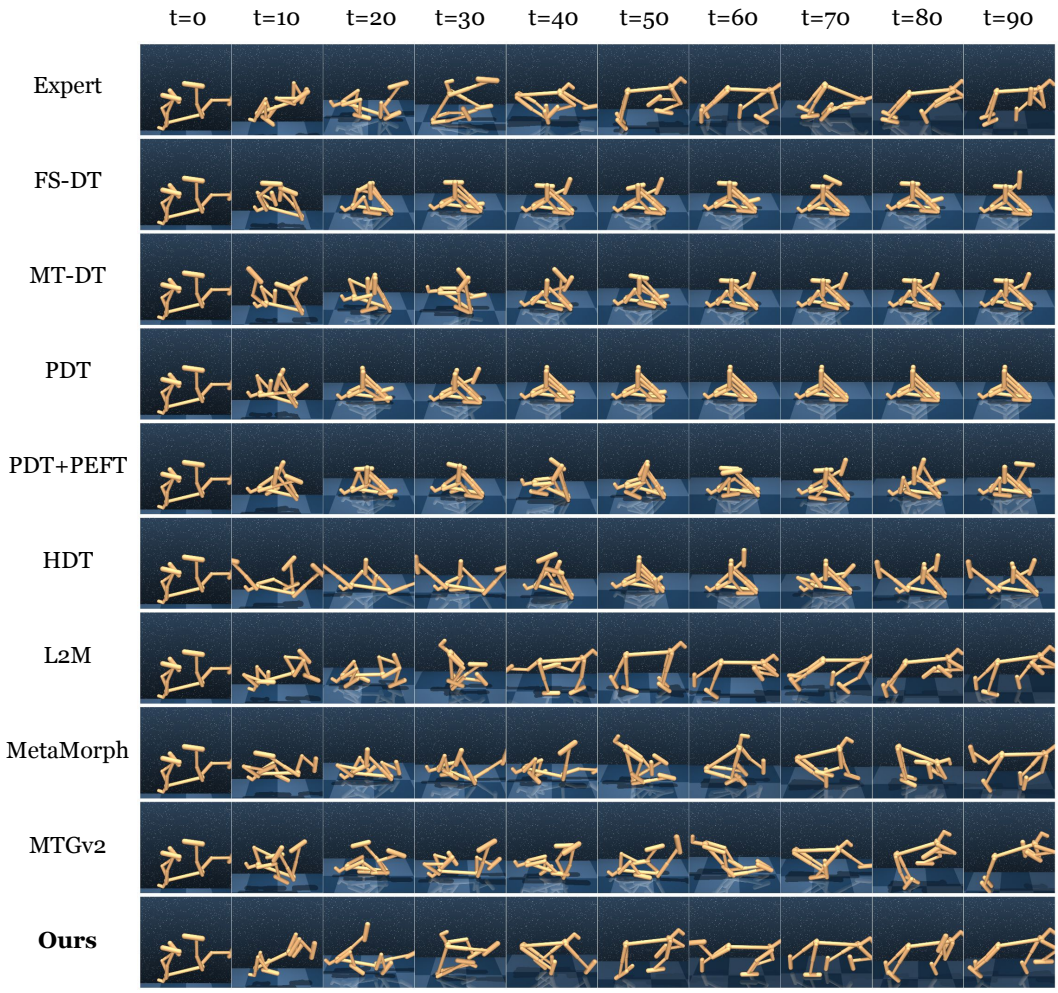


Figure 14: Additional Qualitative Results on `wolf-walk` task.

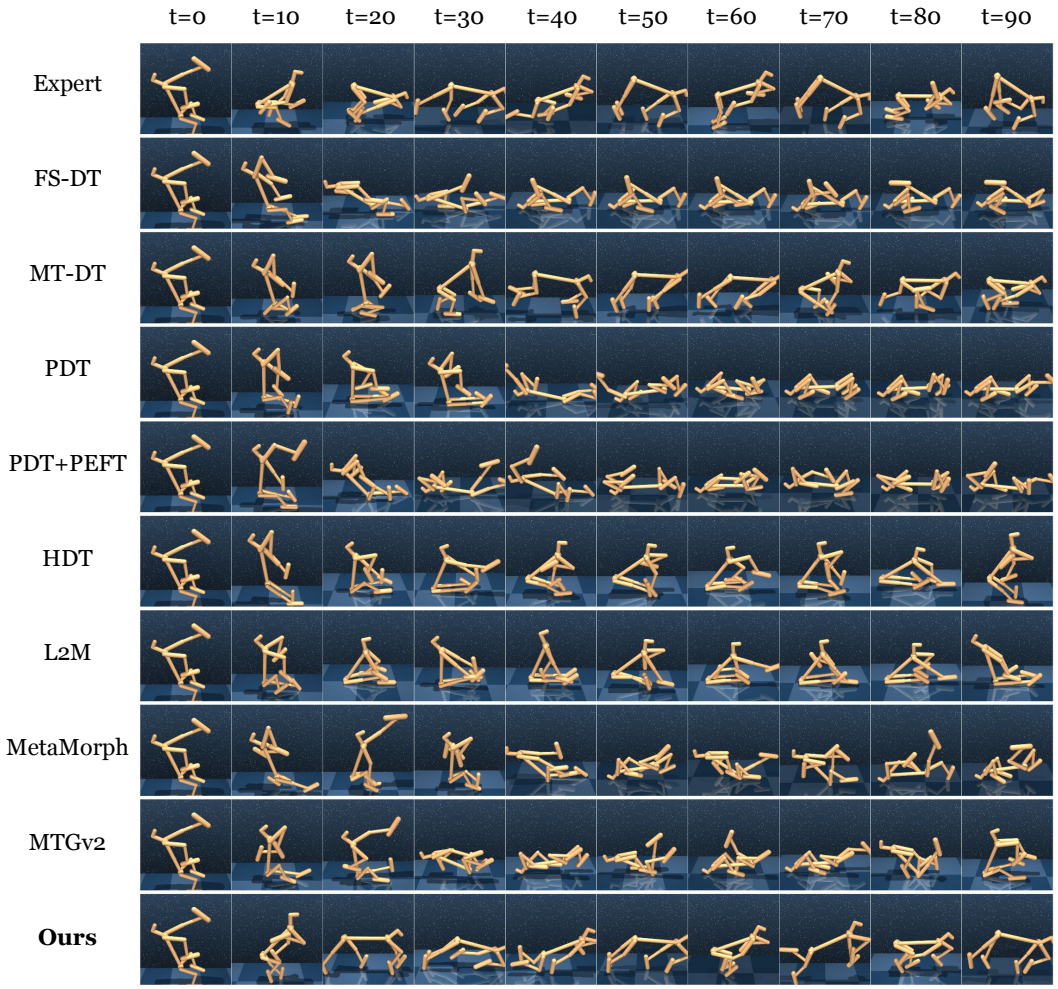


Figure 15: Additional Qualitative Results on `wolf-run` task.

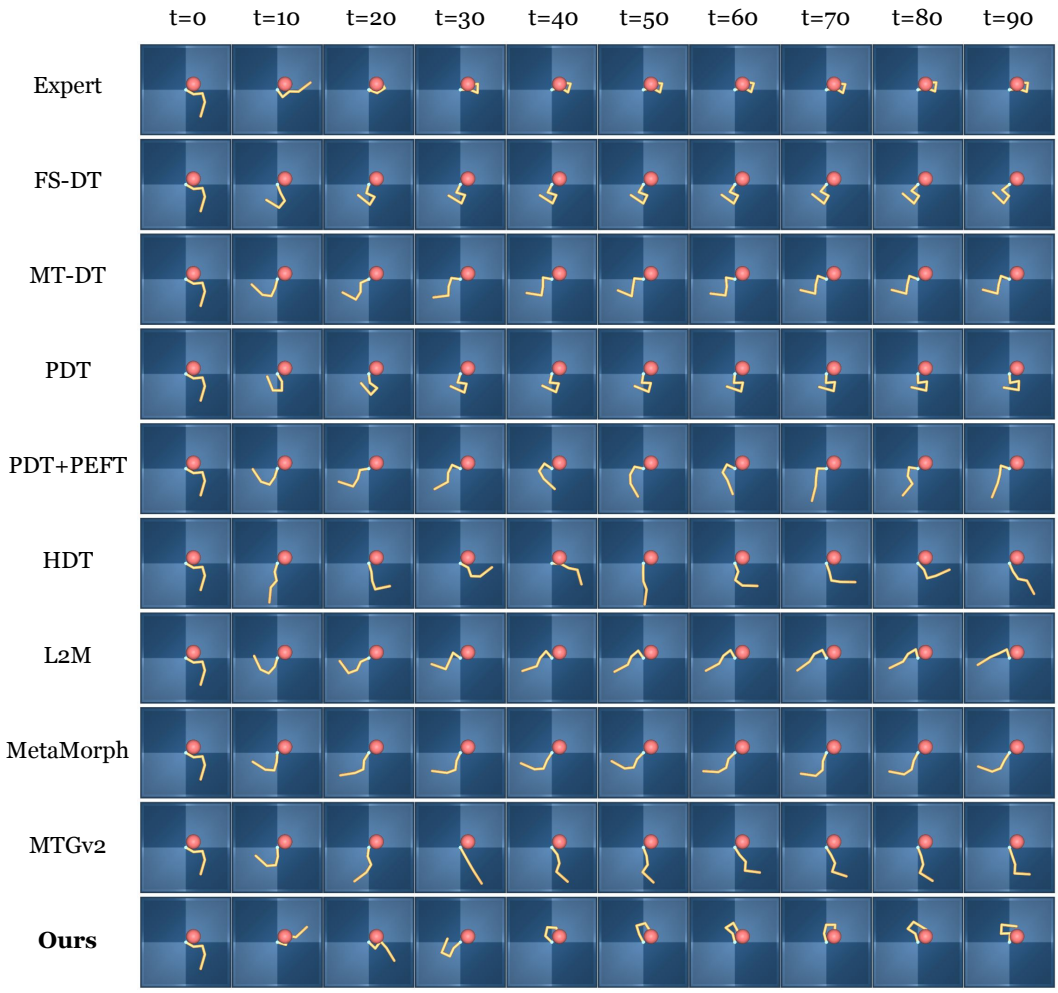


Figure 16: Additional Qualitative Results on `reacher four-easy` task.

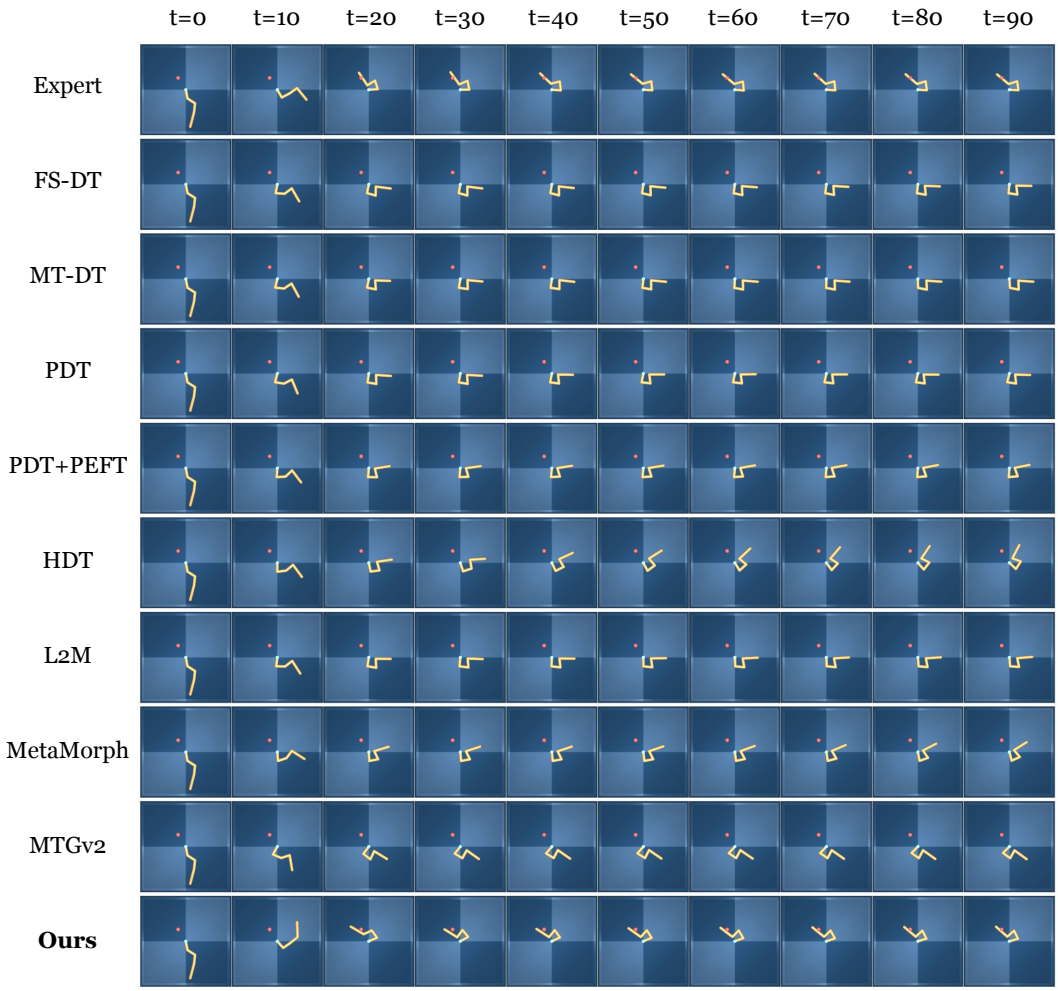


Figure 17: Additional Qualitative Results on `reacher four-hard` task.

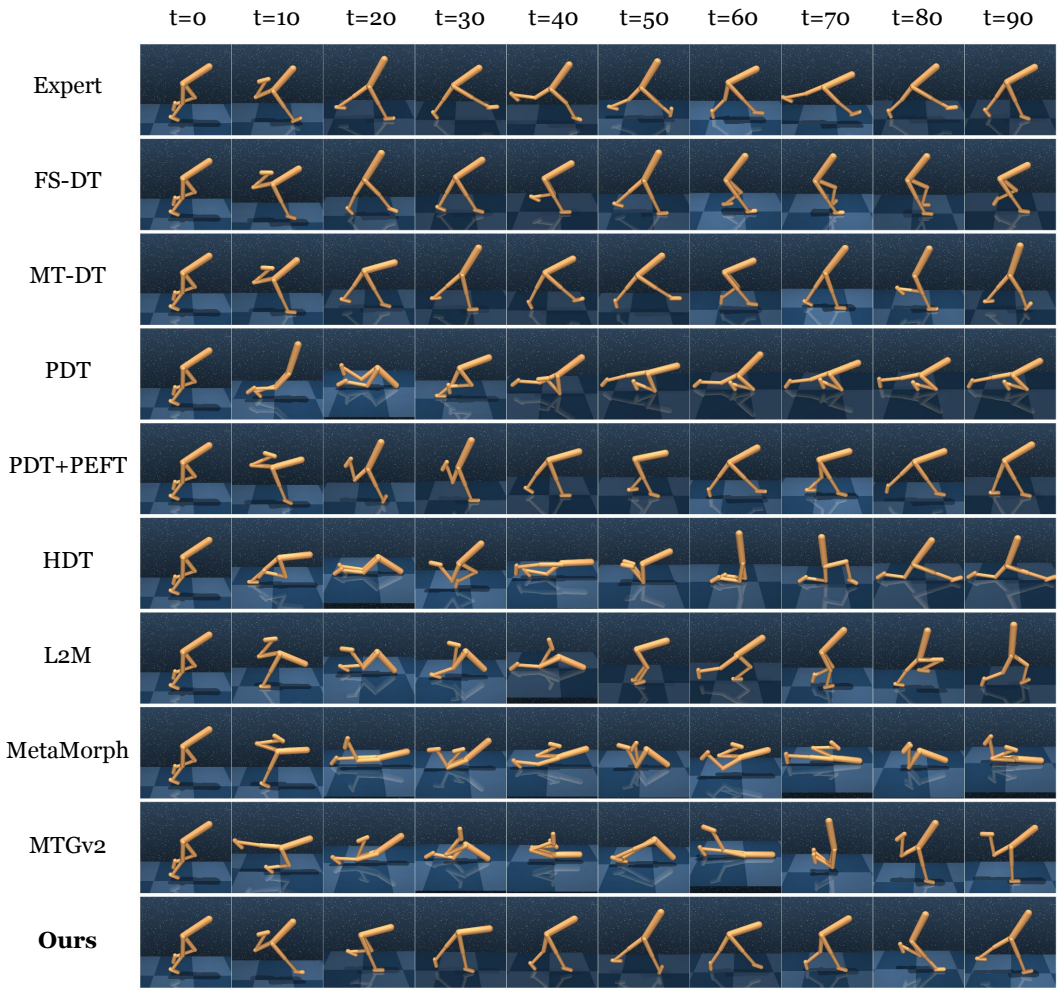


Figure 18: Additional Qualitative Results on `walker-walk backwards` task.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In both the abstract and the introduction, we claim our scope as a few-shot behavior cloning to unseen embodiments and tasks and our contribution as the novel state encoder and a matching-based policy network, as well as experiments in DMC benchmark.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discuss the limitations of our work in Appendix A.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Our paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the details about the dataset, evaluation protocol, and implementations in the experiments section 5 and the appendix.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We have not cleared up the code in a readable format yet.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details about the dataset, evaluation protocol, and implementations in the experiments section 5 and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the standard error in all experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the information on the computer resources needed to reproduce the experiments in Appendix B.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of our work in Appendix A.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

Justification: The experiments are conducted in a physics simulator, which does not have any societal impacts, to the best of our knowledge.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We mentioned and cited the original owners of all assets we used in the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not introduce any new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We do not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.