Plan Your Target and Learn Your Skills: Transferable State-Only Imitation Learning via Decoupled Policy Optimization

Minghuan Liu [1] Zhengbang Zhu [1] Yuzheng Zhuang [2] Weinan Zhang [1] Jianye Hao [2 3] Yong Yu [1] Jun Wang [2]

Abstract

Recent progress in state-only imitation learning extends the scope of applicability of imitation learning to real-world settings by relieving the need for observing expert actions. However, existing solutions only learn to extract a state-to-action mapping policy from the data, without considering how the expert plans to the target. This hinders the ability to leverage demonstrations and limits the flexibility of the policy. In this paper, we introduce Decoupled Policy Optimization (DePO), which explicitly decouples the policy as a high-level state planner and an inverse dynamics model. With embedded decoupled policy gradient and generative adversarial training, DePO enables knowledge transfer to different action spaces or state transition dynamics, and can generalize the planner to out-of-demonstration state regions. Our in-depth experimental analysis shows the effectiveness of DePO on learning a generalized target state planner while achieving the best imitation performance. We demonstrate the appealing usage of DePO for transferring across different tasks by pre-training, and the potential for co-training agents with various skills.

1. Introduction

Imitation Learning (IL) offers a way to train an agent from demonstrations by mimicking the expert's behaviors without constructing hand-crafted reward functions (Hussein et al., 2017; Liu et al., 2021), which requires the expert demonstrations to include information of not only states but also actions. Unfortunately, the action information is absent from many real-world demonstration resources, e.g., traffic surveillance and sport records. To tackle this challenge, one of the potential solutions is state-only imitation learning (SOIL), also known as learning from observations (LfO) (Torabi et al., 2019b), which extends the scope of applicability of IL by relieving the memorization of low-level actions. However, existing SOIL methods (Torabi et al., 2018; Yang et al., 2019) only attempt to match the expert state sequences implicitly by determining feasible actions for each pair of consecutive states, which is an ad-hoc solution for a specific task and ignores understanding the high-level target planning strategy of the expert.

Evidence from humans' cognition and learning in the physical world suggests that human intelligence is effective at transferring high-level knowledge via observations by extracting invariant characteristics across tasks and performing long-term planning that requires different skills. For example, a courier can learn from peers about the optimal delivery path while inheriting his own skills and habits for driving an electric vehicle or riding a bike. Inspired by this insight, to model a generalized target planner and allow transferring to various action spaces and dynamics, we propose Decoupled Policy Optimization (DePO), a novel architecture that decouples a state-to-action policy as two modules – a state planner that generates the consecutive target state, followed by an inverse dynamics model that delivers action to achieve the target state. Intuitively, the state planner is prompted to perform action-aware planning with respect to the inverse dynamics modelling. Furthermore, to prevent the state planner from compounding error by vanilla supervised learning the state transitions from demonstrations, DePO combines the two modules as an integral policy function and incorporates generative adversarial trainining with the induced decoupled policy gradient.

As such, DePO provides generalized plannable state predictions on out-of-demonstration state regions, guiding the agent to match the expert state sequences. The flexibility of DePO supports transferring the planner over homogeneous tasks with different skills. In experiments, we conduct in-depth analysis showing that DePO enjoys several advantages:

- DePO has a generalized ability of state planning on out-of-demonstration states.
- DePO learns accurate state predictions while keeping

[1]Shanghai Jiao Tong University [2]Huawei Noah's Ark Lab [3]Tianjin University. Correspondence to: Weinan Zhang <wnzhang@sjtu.edu.cn>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

the best imitation performances.

- DePO allows pre-training and co-training with less sampling cost for agents over various action spaces and dynamics.

2. Related Work

SOIL endows the agent with the ability to learn from expert states. Although lacking expert decision information, most of previous works still optimize a state-to-action mapping policy to match the expert state transition occupancy measure (OM) (Ho & Ermon, 2016). Torabi et al. (2018) trained an inverse model to label the action information and applied behavioral cloning, while Torabi et al. (2019a) generalized GAIL to match the state transition OM. Yang et al. (2019) analyzed the inverse dynamics mismatch in SOIL and introduced mutual information to narrow it. Huang et al. (2020) applied SOIL on autonomous driving tasks by utilizing a hierarchical policy with a neural decision module and a non-differentiable execution module.

Table 1: Comparison of different methods.

Method	Inverse Dynamics	State Planner	Decoupled Policy	Main Task
BCO (Torabi et al., 2018)	✓	✗	✗	SOIL
GAIfO (Torabi et al., 2019a)	✗	✗	✗	SOIL
IDDM (Yang et al., 2019)	✗	✗	✗	SOIL
OPOLO (Zhu et al., 2020)	✓	✗	✗	SOIL
PID-GAIL (Huang et al., 2020)	✗	✗	✓	IL
QSS (Edwards et al., 2020)	✓	✓	✓	RL
SAIL (Liu et al., 2020)	✓	✓	✗	IL
GSP (Pathak et al., 2018)	✓	✗	✗	SOIL
IMO (Kimura et al., 2018)	✗	✓	✗	SOIL
DePO (Ours)	✓	✓	✓	SOIL

Our work decouples the state-to-action policy into two modules, i.e., the inverse dynamics model and the state transition planner. Both modules have been widely used by many previous works on RL and IL tasks. For instance, Torabi et al. (2018) and Guo et al. (2019) proposed Behavioral Cloning from Observations (BCO), which trained an inverse dynamics model to label the state-only demonstrations with inferred actions. Nair et al. (2017) proposed to match a human-specified image sequence of ropes manipulating with an inverse dynamics model. Pathak et al. (2018) proposed GSP, which trained a multi-step inverse dynamics model from exploration which is regularized by cycle consistency for image-based imitation. From the task perspective, they try to exactly match an expert demonstration sequence, with a binary classifier judging whether each goal is reached. Kimura et al. (2018) utilized a state transition predictor to fit the state transition probability in the expert data, which is further used to compute a predefined reward function. Liu et al. (2020) constructed a policy prior using the inverse dynamics and the state transition predictor, but the policy prior is trained in a supervised learning style and only used for regularizing the policy network. However, as shown in this paper, the policy can be exactly decoupled as these two parts without keeping an extra policy, where the state planner is optimized through policy gradient. Edwards et al. (2020) proposed a deterministic decoupled policy and updated the policy through a deterministic policy gradient similar to DDPG (Lillicrap et al., 2016), based on the proposed $Q(s, s')$ rather than $Q(s, a)$; however, as revealed in their paper and our following test, the performance of $Q(s, s')$ is rather limited thus we omitted in our comparison. To sort out the difference between these methods and ours, we summarize the key factors in Tab. 1. In Appendix B, we discuss further works on transferable imitation learning.

3. Preliminaries

Markov decision process. Consider a γ-discounted infinite horizon Markov decision process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \rho_0, r, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environment dynamics distribution, $\rho_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor. The agent makes decisions through a policy $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and receives rewards $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In our paper, we will assume the environment dynamics $\mathcal{T}$ is a deterministic function such that $s' = \mathcal{T}(s, a)$, and can have redundant actions, i.e., the transition probabilities can be written as linear combination of other actions'. Formally, this refers to the existence of a state $s_m \in \mathcal{S}$, an action $a_n \in \mathcal{A}$ and a distribution p defined on $\mathcal{A} \setminus \{a_n\}$ such that $\int_{\mathcal{A}\setminus\{a_n\}} p(a)\mathcal{T}(s'|s_m, a)\,\mathrm{d}a = \mathcal{T}(s'|s_m, a_n)$. We call two MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ share the same state transition when $\forall s, s' \in \mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}$, $\exists a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2$ such that $s' = \mathcal{T}_1(s, a_1) = \mathcal{T}_2(s, a_2)$, and $\mathcal{A}_1 \neq \mathcal{A}_2$ suggests different action dynamics. This constructs an important assumption for the transferring challenges in this paper.

Occupancy measure. The concept of occupancy measure (OM) (Ho & Ermon, 2016) is proposed to characterize the statistical properties of a certain policy interacting with a MDP. Specifically, the state OM is defined as the time-discounted cumulative stationary density over the states under a given policy π: $\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi)$. Following such a definition we can define different OM:

a) State-action OM: $\rho_\pi(s, a) = \pi(a|s)\rho_\pi(s)$

b) State transition OM: $\rho_\pi(s, s') = \int_{\mathcal{A}} \rho_\pi(s, a)\mathcal{T}(s'|s, a)\,\mathrm{d}a$

c) Joint OM: $\rho_\pi(s, a, s') = \rho_\pi(s, a)\mathcal{T}(s'|s, a)$

Imitation learning from state-only demonstrations. Imitation learning (IL) (Hussein et al., 2017) studies the task of learning from demonstrations (LfD), which aims to learn a policy from expert demonstrations without getting access to the reward signals. The expert demonstrations typically consist of expert state-action pairs. General IL

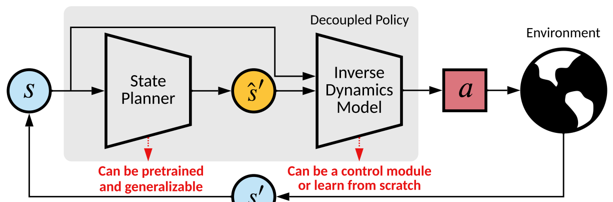


Figure 1: The architecture of Decoupled Policy Optimization (DePO), which consists of a state planner (to plan where to go) followed by an inverse dynamics model (to decide how to reach).

objective minimizes the state-action OM discrepancy:

$$\begin{aligned} \pi^* &= \arg\min_{\pi} \mathbb{E}_{s\sim\rho_\pi^s}\left[\ell\left(\pi_E(\cdot|s), \pi(\cdot|s)\right)\right] \\ &\Rightarrow \arg\min_{\pi} \ell\left(\rho_{\pi_E}(s,a), \rho_\pi(s,a)\right) \ , \end{aligned} \tag{1}$$

where ℓ denotes some distance metric. For example, GAIL (Ho & Ermon, 2016) chooses to minimize the JS divergence $\mathrm{D}_{\mathrm{JS}}(\rho_{\pi_E}(s,a)\|\rho_\pi(s,a))$, and AIRL (Fu et al., 2018) utilizes the KL divergence $\mathrm{D}_{\mathrm{KL}}(\rho_{\pi_E}(s,a)\|\rho_\pi(s,a))$, which also corresponds to a maximum entropy solution with the recovered reward (Liu et al., 2021). However, for the scenario studied in this paper, the action information is absent in demonstrations, which prevents usage of typical IL solutions. A popular solution (Torabi et al., 2019a) is to instead optimize the discrepancy of the state transition OM with the state-to-action policy $\pi(a|s)$ as

$$\pi^* = \arg\min_{\pi}[\ell\left(\rho_{\pi_E}(s,s'), \rho_\pi(s,s')\right)] \ . \tag{2}$$

4. Decoupled Policy Optimization Framework

Previous works on SOIL focus on learning appropriate actions are ad-hoc for a specific task, ignoring understanding the high-level target planning strategy thus limiting the generalized ability. In this section, we present a novel architecture called Decoupled Policy Optimization (DePO) to recover a planner that predicts the neighboring targets where the expert aims to reach. The interaction with the environment further depends on a control module that can be formulated as an inverse dynamics model. An overview of the whole architecture is illustrated in Fig. 1. To make the framework work as desired, we first formalize the decoupled policy; then, we introduce how to learn a generalized planner via both supervised learning and decoupled policy gradient; to alleviate the agnostic problem of the learned state plan, we further propose calibrated decoupled policy gradient to obtain a generalizable and accurate state planner.

4.1. Decoupling the Policy

In the formulation of SOIL, with absent information of actions, the optimality is defined via the state-transition OM. Unlike in standard IL tasks where the expert actions are accessible and perfectly imitating the expert policy corresponds to matching the state-action OM due to the one-to-one correspondence between π and $\rho_\pi(s,a)$ (Ho & Ermon, 2016; Syed et al., 2008), Proposition 4.1 states that finding a policy matching this objective is ambiguous and indirect.

Proposition 4.1. *Suppose Π is the policy space and $\mathcal{P}$ is a valid set of state transition OMs such that $\mathcal{P} = \{\rho : \rho \geq 0$ and $\exists \pi \in \Pi$, s.t. $\rho(s,s') = \rho_0(s)\int_a \pi(a|s)\mathcal{T}(s'|s,a)\,\mathrm{d}a + \int_{\tilde{s},a}\pi(a|s)\mathcal{T}(s'|s,a)\rho(\tilde{s},s)\,\mathrm{d}\tilde{s}\,\mathrm{d}a\}$, then a policy $\pi \in \Pi$ corresponds to one state transition OM $\rho_\pi \in \mathcal{P}$. Instead, under the action-redundant assumption about the dynamics $\mathcal{T}$, a state transition OM $\rho \in \mathcal{P}$ can correspond to more than one policy in Π.*

The proof is in Appendix D. In practical optimization, ambiguity is not really a critical issue since gradient descent-based algorithms have the ability to converge to one of multiple optima. Nevertheless, without explicit modeling the expert's target planner (state transition), learning such an optimal state-to-action mapping policy is ad-hoc. To enable the agent learn a high-level planning strategy from demonstrations that can be transferred to various low-level action spaces, we decouple the policy structure by a planner (formulated by a state transition planner) and a control module (formulated by an inverse dynamics model) by finding a one-to-one corresponding solution for SOIL. Before continuing, we introduce the definition of *hyper-policy*.

Definition 4.2. A hyper-policy $\Omega \in \Lambda$ is a maximal set of policies sharing the same state transition occupancy such that for any $\pi_1, \pi_2 \in \Omega$, we have $\rho_{\pi_1}(s,s') = \rho_{\pi_2}(s,s')$.

Then by definition, Proposition 4.3 shows the one-to-one correspondence between Ω and $\rho_\Omega(s,s')$. Similar to the normal state-to-action mapping policy, a hyper-policy Ω can be represented as a state-to-state mapping function $h_\Omega(s'|s)$ which predicts the state transition such that for any $\pi \in \Omega$,

$$h_\Omega(s'|s) = \frac{\rho_\Omega(s,s')}{\int_{\tilde{s}}\rho_\Omega(s,\tilde{s})\,\mathrm{d}\tilde{s}} = \int_a \pi(a|s)\mathcal{T}(s'|s,a)\,\mathrm{d}a \ . \tag{3}$$

Proposition 4.3. *Suppose the state planner h_Ω is defined as in Eq. (3) and $\Gamma = \{h_\Omega : \Omega \in \Lambda\}$ is a valid set of the state transition predictor, $\mathcal{P}$ is a valid set of the state-transition OM defined as in Proposition 4.3, then a state planner $h_\Omega \in \Gamma$ corresponds to one state transition OM, where $\pi \in \Omega$; and a state transition OM $\rho \in \mathcal{P}$ only corresponds to one hyper-policy state planner such that $h_\rho = \rho(s,s')/\int_{\tilde{s}}\rho(s,\tilde{s})\,\mathrm{d}\tilde{s}$.*

The proof can follow the Theorem 2 of Syed et al. (2008), and for completeness, we include it in Appendix D. Therefore, we find a one-to-one correspondence between the optimization term $\rho(s,s')$ and a state transition planner

$h_\Omega(s'|s)$, which indicates that under state-only demonstrations we only need to recover the state transition prediction of the hyper-policy Ω_E as

$$\begin{aligned}&\arg\min_{\Omega} \ell\left(\rho_{\Omega_E}(s,s'), \rho_\Omega(s,s')\right)\\ \Rightarrow &\arg\min_{h_\Omega} \mathbb{E}_{s\sim\Omega}[\ell\left(h_{\Omega_E}(s'|s), h_\Omega(s'|s)\right)]\,.\end{aligned} \tag{4}$$

Nonetheless, we still require to learn a policy to interact with the MDP environment to match the state transition OM of the expert. This can be achieved by learning any policy $\pi \in \Omega_E$ according to Eq. (4) instead of recovering the expert policy π_E exactly.

Intuitively, the state planner tells the agent the *target* that the expert will reach without informing any feasible *skill* that require the agent to learn. Therefore, to recover a $\pi \in \Omega_E$, we can construct an inverse dynamics such that

$$\pi = \underbrace{\mathcal{T}_\pi^{-1}}_{\text{inverse dynamics}} \left(\underbrace{\mathcal{T}(\pi_E)}_{\text{state planner}} \right)\,. \tag{5}$$

Formally, the expert policy can be decoupled as

$$\begin{aligned}\pi_E(a|s) &= \int_{s'} \mathcal{T}(s'|s,a)\pi_E(a|s)\,\mathrm{d}s'\\ &= \int_{s'} \frac{\rho_{\pi_E}(s,s')I_{\pi_E}(a|s,s')}{\rho_{\pi_E}(s)}\,\mathrm{d}s'\\ &= \int_{s'} h_{\pi_E}(s'|s)I_{\pi_E}(a|s,s')\,\mathrm{d}s'\,.\end{aligned} \tag{6}$$

where both the state planner h and the inverse dynamics model I are policy dependent. When the environment dynamics is deterministic, we have a simpler form on the probability of taking the action a as

$$\pi_E(a|s) = h_{\pi_E}(s'|s)I_{\pi_E}(a|s,s'). \tag{7}$$

In optimality, we should have $s' = \mathcal{T}(s,a)$, yet in the inference stage we actually first predict $\hat{s}'$ from h and then take the action a. Fortunately, the optimality in SOIL only requires us to recover $\pi \in \Omega_E$, we do not have to learn about I_{π_E} but just one feasible skill $I(a|s,s')$. Then the policy distribution given the state s can be recovered by

$$\pi(\cdot|s) = \mathbb{E}_{\underbrace{\hat{s}' \sim h_{\Omega_E}(\hat{s}'|s)}_{\text{target}}}\Big[\underbrace{I(\cdot|s,\hat{s}')}_{\text{skill}}\Big]\,. \tag{8}$$

Here the inverse dynamics model I serves as a control module that offers an arbitrary *skill* to reach the expected *target* state provided by the state planner h, and does not depend on the hyper-policy Ω_E. In the cases where it must be learned from scratch, we need a sampling policy $\pi_\mathcal{B}$ to construct $I = I_{\pi_\mathcal{B}}$, and a mild requirement for $\pi_\mathcal{B}$ that it covers the support of $\rho_{\Omega_E}(s,s')$ so that the learned I can provide a possible action to achieve the target state. Furthermore, if the environment and the expert policy are both deterministic, the state transition is a Dirac delta function, and h is also a simple deterministic function.

4.2. Supervised Learning from Data

We provide a quick view on the simplest way to learn both modules by directly supervised learning from data.

Inverse dynamics model. The requirement of interacting with the environment asks for a control module for reaching the target predicted by the planner, which can be formulated as an inverse dynamics model predicting the action given two consecutive states. Formally, let the ϕ-parameterized inverse dynamics model I_ϕ take the input of a state pair and predict a feasible action to achieve the state transition: $\hat{a} = I_\phi(s,s')$. We note that the inverse dynamics model can either be pre-trained in advance, or a prior ground-truth function, or any differentiable control module that plays the role of planning. If we must learn the model online, we can choose to minimize the divergence (for example, KL) between the inverse dynamics of a sampling policy $\pi_\mathcal{B}$ and I_ϕ, i.e.,

$$\min_{\psi} L^I = \mathbb{E}_{(s,s')\sim\pi_\mathcal{B}}[\mathrm{D_f}(I_{\pi_\mathcal{B}}(a|s,s')\|I_\phi(a|s,s'))]\,. \tag{9}$$

The accuracy of this module is important, otherwise, the agent cannot take specific action to reach the target state. Fortunately, recall that we only need the support of learned $I(a|s,s')$ to cover the support of the expert state transition OM at convergence, from which we can infer at least one possible action to accomplish the transition. Besides, due to the principle of the inverse dynamics serving as a local control module, when learning from experience data, the model only has to focus on inferring accurate actions on states encountered by the current policy instead of the overall state space. Thence, in practice, we train the inverse dynamics model every time before updating the policy.

State planner. The state planner is constructed as modeling the explicit information of the expert by predicting the subsequent state of the expert under the current state. Thus, to learn a parameterized module h_ψ, a direct way is to match the expert data as minimizing the divergence to achieve Eq. (4):

$$\min_{\psi} L^h = \mathbb{E}_{(s,s')\sim\Omega_E}[\mathrm{D_f}(h_{\Omega_E}(s'|s)\|h_\psi(s'|s))]\,. \tag{10}$$

However, simply mimicking the state-transition in dataset tends to result in serious compounding error problems, as behavior cloning (BC) in standard imitation learning (Ross et al., 2011). To alleviate this problem, we can resort to the help of policy gradient and incorporate generative adversarial objective into the decoupled policy learning.

4.3. Decoupled Policy Gradient

By now, we have explained why and how to decouple the policy as two modules, and they are naturally coherent as an integral policy function, thereafter, with a well-defined inverse dynamics, we can derive policy gradient (PG) for optimizing the high-level state planner, named Decoupled PG (DePG) . The predicted target states by the planner will further drive the inverse dynamics to provide desirable actions leading to match the expert state-transition OM.

Assume we can obtain the reward of the task, and denote the state-action value function as $Q(s,a)$, then by chain rule, DePG can be accomplished by

$$\begin{aligned}\nabla_{\phi,\psi}\mathcal{L}^\pi &= \mathbb{E}_{(s,a)\sim\pi}\left[Q(s,a)\nabla_{\phi,\psi}\log\pi_{\phi,\psi}(a|s)\right]\\ &= \mathbb{E}_{(s,a)\sim\pi}\Big[\frac{Q(s,a)}{\pi(a|s)}\Big(\int_{s'} I(a|s,s')\nabla_\psi h_\psi(s'|s)\,\mathrm{d}s'\\ &\quad + \mathbb{E}_{s'\sim h}\left[\nabla_\phi I_\phi(a|s,s')\right]\Big)\Big]\ . \end{aligned} \tag{11}$$

In our formulation, since we have assumed the inverse dynamics to be an accurate control module at least for the current policy, the inverse dynamics function I is static when optimizing the policy function. As such, the gradient form in Eq. (11) only updates ψ (the state planner) and the gradient on ϕ (the inverse dynamics) should be dropped. In addition, since the state planner is approximated by neural networks (NNs) in our case, it is convenient to apply the reparameterization trick and bypass explicitly computing the integral over s' as

$$s' = h(\epsilon; s),\ \ \pi(a|s) = \mathbb{E}_{\epsilon\sim\mathcal{N}}\left[I(a|s,h(\epsilon;s))\right]\ , \tag{12}$$

where ϵ is an input noise vector sampled from some fixed distribution, like a Gaussian. Then Eq. (11) becomes

$$\nabla_\psi\mathcal{L}^\pi = \mathbb{E}_{(s,a)\sim\pi,\epsilon\sim\mathcal{N}}\left[\frac{Q(s,a)}{\pi(a|s)}\left(\nabla_h I(a|s,h_\psi(\epsilon;s))\nabla_\psi h_\psi(\epsilon;s)\right)\right] \tag{13}$$

In this form, we are taking the knowledge from the inverse dynamics about action a to update parameters of the state planner by updating the prediction about the next state with error $\Delta s' = \alpha\nabla_h I(a|s,h(\epsilon;s))$ and learning rate α.

Agnostic decoupled PG. Till now, we take for granted the existence of an accurate control module for the current policy. This makes sense if we have the ground-truth inverse dynamics function of the environment. In more general scenarios, the skill model has to be learned during the training stage. Unfortunately, simply applying DePG (Eq. (11)) to obtain the desired high-level target planner is faced with serious learning challenges, especially when the inverse dynamics model is approximated by NNs.

The problem comes from a constraint on the inverse dynamics modelling's inputs, which requires the prediction of the target planner to be a legal neighbor state that follows the inputs training data distribution. Otherwise the state transition pair (s,s') will be an illegal transition and leads the corresponding output actions to be agnostic. An illegal state transition could still be a legal input to the approximated inverse dynamics model if it may still provide a feasible action to interact with the environment. However, since we are utilizing the gradient of the inverse dynamics model $\nabla_h I(a|s,h_\psi(\epsilon;s))$, we do not expect a generalization on illegal transitions to get a feasible action. In other words, simply optimizing ψ through Eq. (11), we cannot constrain the gradient provided by the inverse dynamics within pointing to a reachable target state, and the state planner will be trained to predict an arbitrary and unreasonable state that can still lead the inverse dynamics model to give a feasible action. This is further revealed in Section 5.1.

Calibrated decoupled PG. To alleviate such a problem, we should dive into the nature of the decoupled structure. The goal of DePO is to learn a planner that determines the high-level target to reach, yet taking which action to achieve is inessential. In the following theorem, we draw insights on regularizing the planner to provide a feasible state plan.

Theorem 4.1. *Let $\mathcal{A}(s)$ be an action set that for all a in $\mathcal{A}(s)$, a deterministic transition function leads to the same state $s' = \mathcal{T}(s,a)$. If there exists an optimal policy π^* and a state $\hat{s}$ such that $\pi^*(\cdot|\hat{s})$ is a distribution over $\mathcal{A}(\hat{s})$, then we can replace $\pi^*(\cdot|\hat{s})$ with any distributions over $\mathcal{A}(\hat{s})$ which does not affect the optimality.*

The proof is in Appendix D. Therefore, no matter what action a the inverse dynamics takes, as long as it leads the agent from s to s', the integrated policies are all optimal policies on s and their value $Q_\pi(s,a)$ should be the same. On this condition, give a state-action pair (s,a) and a legal transition s', we can simplify the inverse dynamics as a deterministic distribution on a with the probability of one, i.e., $I(a|s,s') = 1$, then the probability of the decoupled policy $\pi(a|s) = h(s'|s)$, where s' is the target state, and DePG (Eq. (11)) can be simplified as

$$\nabla_\psi\mathcal{L}^\pi = \mathbb{E}_{(s,a,s')\sim\pi}\left[Q(s,a)\nabla_\psi\log h_\psi(s'|s)\right]\ , \tag{14}$$

which is named Calibrated Decoupled PG (CDePG). Optimizing with CDePG can be realized as maximizing the probability to target state s' on state s if a is a good action regarding the inverse dynamics is accurate. However, solely updating Eq. (14) leads to a severe exploration problem since the planner is only allowed to predict a visited state. Fortunately, DePG provides a way to explore the most promising actions although it is not responsible for getting legal state transition. Thereafter, in practice, we choose to jointly optimize DePG (Eq. (13)) with CDePG (Eq. (14)) and find it can achieve a good balance between exploration and accurate prediction on legal targets. More detailed analyses are provided in Appendix C.4. Note that our derivation does not only limit in the literature of SOIL but also can be applied to general RL tasks. Yet in this paper, we focus on SOIL tasks since our motivation is to understand the high-level planning strategy of experts.

Generative adversarial training. To reach the optimality, the inverse dynamics model must converge to cover the support of the expert hyper-policy, which is essential especially when learning the inverse dynamics from scratch. This asks for the data-collecting policy to sample around the expert's occupancy. This is easy to achieve on simple low-dimensional tasks, but may not be satisfied in high-dimensional continuous environments. Also, due to the limited amount of the demonstration data, simply supervised learning the state planner module can cause serious compounding error due to the one-step optimization and the agent may not know where to go in an unseen state. To this end, we incorporate the GAN-like informative rewards similar to GAIfO (Torabi et al., 2019a) and update the policy using the induced decoupled PGs. In detail, we construct a parameterized discriminator $D_\omega(s, s')$ to compute the reward $r(s, a) \triangleq r(s, s')$ as $\log D_\omega(s, s')$ while the decoupled policy serves as the generator.

The optimization can be conducted using any PG-based RL learning algorithms (e.g., TRPO, PPO, SAC). As the training proceeds, we expect the agent to sample more transition data around Ω_E, and thus the support of the sampling policy progressively covers the support of $\rho_{\Omega_E}(s, s')$ and the agent can generalize well on unseen states.

4.4. Overall Algorithm

The algorithm is composed with three essential parts: the state planner h used for predicting next states; the inverse dynamics model I used for inferring the possible actions conditioned on two adjacent states; and the discriminator D offers intermediate reward signals for training the decoupled policy π. When the inverse dynamics model I is obtained in advance, we only have to learn h; yet when we need to learn both modules from scratch: first learn I and then train h with I fixed. The overall objective of DePO is

$$\min_{\phi} L^I \quad \text{(if } I \text{ needs to be learned)}, \tag{15}$$

$$\min_{\psi} \mathcal{L}^{\pi,h} = \mathcal{L}^{\pi} + \lambda_h \mathcal{L}^h \ , \tag{16}$$

where λ_h is the hyperparameter for trading off the loss. The detailed algorithm is summarized in Appendix A. The flexibility of DePO comes from the decoupled structure, such that the state planner can be transferred to different MDPs without training if they share the same state transition, and only the inverse dynamics module needs to be retrained, which yields significant gain of sample efficiency as shown in the experiments (Section 5).

4.5. Analysis of The Compounding Error

In our formulation, we have decoupled the state-to-action mapping policy as a state-to-state mapping function and a state-pair-to-action mapping function. This introduces a new compounding error challenge such that the agent cannot reach where it plans due to the generalization errors of these two modules.

Theorem 4.2 (Error Bound of DePO)**.** *Consider a deterministic environment whose dynamics transition function $\mathcal{T}(s, a)$ is deterministic and L-Lipschitz. Assume the ground-truth state transition $h_{\Omega_E}(s)$ is deterministic, and for each policy $\pi \in \Pi$, its inverse dynamics I_π is also deterministic and C-Lipschitz. Then for any state s, the distance between the desired state s'_E and reaching state s' sampled by the decoupled policy is bounded by*

$$\|s' - s'_E\| \le LC\|h_{\Omega_E}(s) - h_\psi(s)\| + L\|I_{\pi_\mathcal{B}}(s, \hat{s}') - I_\phi(s, \hat{s}')\| \ ,$$

where $\pi_\mathcal{B}$ is a sampling policy that covers the state transition support of the expert hyper-policy and $\hat{s}' = h_\psi(s)$ is the predicted next state.

The proof can be found in Appendix D. From Theorem 4.2 we know that the compounding error can be enlarged due to each part's generalization error, where the first term corresponds to the error of predicted states and the second term indicates whether the agent can reach where it plans to.

5. Experiments

In this section, we conduct in-depth analyses of our proposed DePO method. We first conduct a simple 2D grid world environment and validate the planning ability of DePO on out-of-demonstration state regions (Section 5.1); then, we show that DePO can achieve the best imitation performance compared with baselines (Section 5.2). Afterwards, we investigate intriguing usages of DePO, which includes transferring by pre-training (Section 5.3), and co-training (Section 5.4) for agents over various action spaces or dynamics. Due to the page limit, we leave more details, additional results and ablation studies in Appendix C.

5.1. Generalized Planning Ability

The key technical contribution of DePO includes the decoupled structure of policy that provides the ability to plan the state transition to match the expert. Therefore, we are keen to verify the non-trivial generalized planning ability, especially on out-of-demonstration states. We generate expert demonstrations in a 2D 6×6 grid world environment, in which the expert starts at the left bottom corner (0,0) and aims to reach the upper right corner (5,5) (Fig. 2(a)). The agent starts at any grid randomly except the shaded zone, and in each grid, it has $k \times 4$ actions, which means the agent has k available actions to reach a neighboring block. All

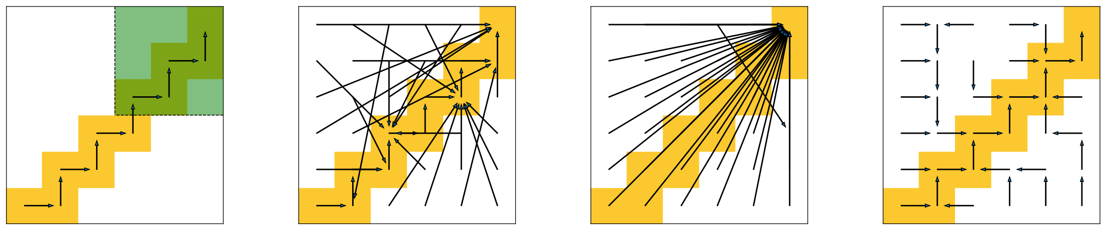


Figure 2: Grid world environment and the prediction of the learned state planner. (a) The expert starts from the left-bottom corner (0,0) to the right-upper (6,6) and the arrows on the yellow grid depicts the path of the expert. The agent is required to start at any grid on the map except the shaded zone. We test three variants under the decoupled policy structure where both modules are learned from scratch. (b) Supervised learning only from the dataset results in predictions of the target state on expert paths, even if not a neighboring (legal) one. (c) Agnostic DePG learns to predict arbitrary states, while the inverse dynamics can still give a legal action to reach a neighbor state. (d) The proposed DePO algorithm, which generalizes the planning into every out-of-demonstration state (white blocks) with legal transitions.

functions are learned from scratch. We first employ simple supervised learning on the two modules, shown in Fig. 2(b), which indicates that the agent can only predict the target state on the demonstrated path, even if it is not a neighboring one; then, we illustrate the agnostic problem of DePG in Fig. 2(c), where it learns to predict arbitrary states, even if it is not a legal transition. However, with the CDePG proposed in Eq. (14), DePO shows great planning ability on the legal transition to match the expert occupancy measure, even on out-of-demonstration states.

5.2. Imitation Evaluations

We show that DePO keeps the best imitation performance by comparing against other baseline methods on easy-to-hard continuous control benchmarking environments (Fig. 4). In each environment, besides GAIfO and BCO, we also evaluate GAIfO with decoupled policy (denoted as GAIfO-DP) and the supervised learning version of DePO. For fairness, we re-implement all algorithms and adopt Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as the underlying RL learning algorithm for GAIfO and DePO. For each environment, we train an SAC agent to collect 4 state-only trajectories as expert data. All algorithms are trained with the same frequency and gradient steps.

It is easily concluded that for simple environments as InvertedPendulum, supervised learning algorithms can achieve better learning efficiency since the state space is limited. However, on difficult tasks, BCO and DePO (Supervised) both fail due to compounding errors, yet policy gradient methods like DePO and GAIfO own better imitation efficacy, and DePO can gain the best performance against its counterparts. In Appendix C.2, we reveal that DePO enables the agent to reach exactly where it plans to, and the state planner is even such accurate that it can be used for long (more than one)-step planning. The corresponding curves of the errors between the predicted state and the reached state are further shown in Appendix C.3.1. To demonstrate the planning ability of DePO, we illustrate the images of

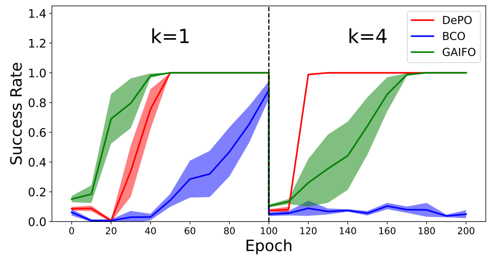


Figure 3: Transferring experiment on grid world environment. The y-axis denotes the success rate of reaching (5,5). The solid line and the shade shown in this and following figures represent the mean and the standard deviation of the results over 5 random seeds.

the imagined rollout states provided by the learned state planner, compared with the reaching states achieved during interaction with the environment, shown in Appendix C.3.2.

5.3. Few-Shot Transferring by Pre-training

We are curious about the potential usage of DePO in transferring the policy to a different skill space (e.g., different action space or different dynamcis) with a pre-trained state planner. We conduct the experiments on both the discrete grid-world task (Fig. 3) and the continuous control benchmarks (Fig. 5). In particular, on grid-world, we train each agent with the action space parameter $k = 1$ for 100 epochs, then we reset the environment and change $k = 4$ for the transferring training stage. On Mujoco tasks, we use the pre-trained state planner in previous imitation tasks (Section 5.2) and start the training on a complex action dynamics (see Appendix C.1.3 for details) for the transferring stage.

For BCO and GAIfO, since the policy is a state-to-action mapping function, it must be re-trained from scratch and need much more exploration data. However, with the decoupled policy structure, DePO can keep the state-predictor and is only required to re-train the inverse dynamics model. Learning curves on all experiments reveal the fast sample efficiency and favorable stability in the transferring stage, which is even more efficient than online SAC RL agents

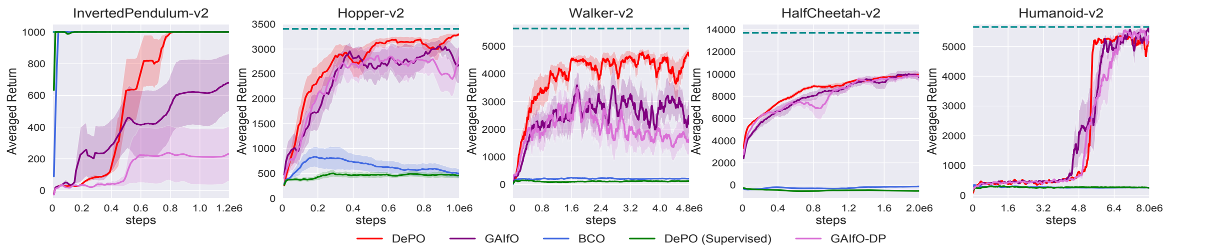


Figure 4: Learning curves on easy-to-hard continuous control benchmarks, where the dash lines represent the expert performance.

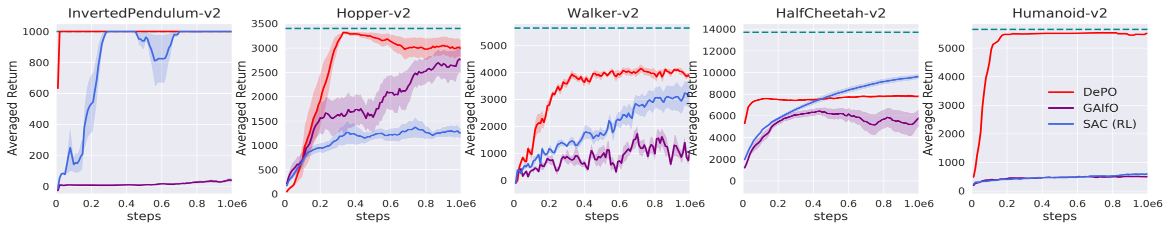


Figure 5: Transferring by pre-training on Mujoco tasks with complex action dynamics within 1e6 interaction steps.

(Fig. 5). Moreover, consider if we have a ground truth inverse dynamics function, DePO even does not need any training and can be directly deployed for tasks with different skills (see Appendix C.5). This motivates an attractive application of DePO for pre-training in a simple task and deployed into a more complex skill space. And in Appendix C.6 we show DePO can be pre-trained on RL agents (without demonstration but reward instead) and is still able to be transferred to learn different skills with much higher efficiency compared with the state-of-the-art RL algorithm.

5.4. Co-training and Transferring for Real-World Application

The rapid development of autonomous driving has brought huge demand on high-fidelity traffic participants interactions simulation for close-loop training and testing (Zhou et al., 2020). However, driver's detailed actions are not easy to obtain, yet we adopt SOIL from a traffic surveillance recording dataset (NGSIM I-80 (Halkias & Colyar, 2006)) that contains kinds of recorded human driving trajectories. For less training time, we only utilize half of the dataset. In addition, for simulating different kinds of vehicles (such as cars and trucks), we should train different policies due to the difference in the underlying controlling strategies of different vehicles. However, note that they share the high-level planning mode of feasible states transitions. To this end, we show another appealing usage of DePO by co-training agents with different action dynamics but learning one single shared state planner.

We utilize the simulator provided by Henaff et al. (2019) as our simulation platform. Specifically, based on the original dynamics (which we refer to as *Normal*), we design two additional action dynamics: *Inverse* – the action is taken additive inverse before stepping in the environment; and *Transpose* – the two dimensions of the action are swapped

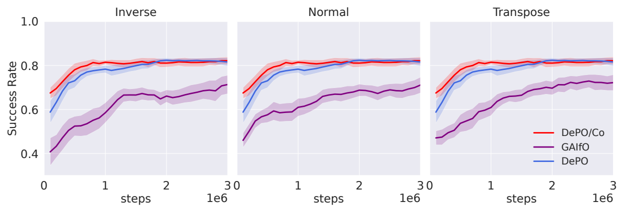


Figure 6: Co-training for different vehicles.

before stepping. In this experiment, we construct a direct inverse dynamics function without training, and compare DePO and GAIfO, which independently trains three different policies for this task, with the co-training version of DePO (denoted as DePO/Co). We plot the *Success Rate* during the training in Fig. 6, denoting the percentage of driving across the entire area without crashing into other vehicles or driving off the road among all evaluation vehicles. We observe DePO benefits a lot efficiency from learning the high-level planner with ground truth inverse dynamics functions; and DePO/Co can take advantage of the shared training of all agents, achieving much less sampling cost and the fastest convergence rate. Additional statistic results for more metrics and transferring experiments can be further referred to Appendix C.5.

6. Conclusion

We propose Decoupled Policy Optimization (DePO) for transferable state-only imitation learning, which decouples the state-to-action mapping policy into a state-to-state mapping state planner and a state-pair-to-action mapping inverse dynamics model. DePO allows for adapting to different agents and generalizing on out-of-demo states with decoupled policy gradient and generative adversarial objective. Extensive experiments demonstrate the power of DePO with appealing usages as transferring by pre-training and effective co-training for different skilled agents. For future works, we plan to generalize DePO to image-based problems, goal-conditioned tasks and apply to real-world robotics.

Acknoledgement

We thank Cheng Chen, Jian Shen, Zhengyu Yang, Menghui Zhu, Hanye Zhao, Minkai Xu and Yuxuan Song for helpful discussions. The SJTU team is supported by "New Generation of AI 2030" Major Project (2018AAA0100900), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62076161). The work is also sponsored by Huawei Innovation Research Program. The author Minghuan Liu is also supported by Wu Wen Jun Honorary Doctoral Scholarship, AI Institute, Shanghai Jiao Tong University.

References

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Edwards, A. D., Sahni, H., Liu, R., Hung, J., Jain, A., Wang, R., Ecoffet, A., Miconi, T., Isbell, C., and Yosinski, J. Estimating q(s,s') with deep deterministic dynamics gradients. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, 2020.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adverserial inverse reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Guo, X., Chang, S., Yu, M., Tesauro, G., and Campbell, M. Hybrid reinforcement learning with expert state sequences. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pp. 3739–3746, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pp. 1856–1865, 2018.

Halkias, J. and Colyar, J. Next generation simulation fact sheet. *US Department of Transportation: Federal Highway Administration*, 2006.

Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., and Silver, D. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Henaff, M., Canziani, A., and LeCun, Y. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*, pp. 4565–4573, 2016.

Huang, J., Xie, S., Sun, J., Ma, Q., Liu, C., Lin, D., and Zhou, B. Learning a decision module by imitating driver's control behaviors. In *Proceedings of the Conference on Robot Learning (CoRL) 2020*, 2020.

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Jiang, S., Pang, J., and Yu, Y. Offline imitation learning with a misspecified simulator. *Advances in neural information processing systems*, 33, 2020.

Kimura, D., Chaudhury, S., Tachibana, R., and Dasgupta, S. Internal model from observations for reward shaping. *arXiv preprint arXiv:1806.01267*, 2018.

Konidaris, G. D. and Barto, A. G. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pp. 895–900, 2007.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Lee, Y., Szot, A., Sun, S.-H., and Lim, J. J. Generalizable imitation learning from observation via inferring goal proximity. *Advances in Neural Information Processing Systems*, 34, 2021.

Li, A. C., Florensa, C., Clavera, I., and Abbeel, P. Sub-policy adaptation for hierarchical reinforcement learning. *International Conference on Learning Representations*, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016,*, 2016.

Liu, F., Ling, Z., Mu, T., and Su, H. State alignment-based imitation learning. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Liu, M., He, T., Xu, M., and Zhang, W. Energy-based imitation learning. In *20th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2021*, 2021.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems NeurIPS 2018*, 31, 2018.

Nair, A., Chen, D., Agrawal, P., Isola, P., Abbeel, P., Malik, J., and Levine, S. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017*, pp. 2146–2153, 2017.

Pathak, D., Mahmoudieh, P., Luo, G., Agrawal, P., Chen, D., Shentu, Y., Shelhamer, E., Malik, J., Efros, A. A., and Darrell, T. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 2050–2053, 2018.

Ross, S., Gordon, G. J., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011*, pp. 627–635, 2011.

Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pp. 4732–4741. PMLR, 2018.

Syed, U., Bowling, M. H., and Schapire, R. E. Apprenticeship learning using linear programming. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference ICML 2008*, pp. 1032–1039, 2008.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pp. 4950–4957, 2018.

Torabi, F., Warnell, G., and Stone, P. Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2019*, pp. 2229–2231, 2019a.

Torabi, F., Warnell, G., and Stone, P. Recent advances in imitation learning from observation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pp. 6325–6331, 2019b.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549. PMLR, 2017.

Yang, C., Ma, X., Huang, W., Sun, F., Liu, H., Huang, J., and Gan, C. Imitation learning from observations by minimizing inverse dynamics disagreement. In *Advances in Neural Information Processing Systems, NeurIPS 2019*, pp. 239–249, 2019.

Zhang, J., Yu, H., and Xu, W. Hierarchical reinforcement learning by discovering intrinsic options. *International Conference on Learning Representations*, 2022.

Zhang, S. and Whiteson, S. Dac: The double actor-critic architecture for learning options. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhou, M., Luo, J., Villela, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., et al. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. In *Conference on Robot Learning, CoRL 2020*, 2020.

Zhu, Z., Lin, K., Dai, B., and Zhou, J. Off-policy imitation learning from observations. In *Advances in Neural Information Processing Systems, NeurIPS 2020*, 2020.

A. Algorithm

Algorithm 1 Decoupled Policy Optimization (DePO)

```
1: Input: State-only expert demonstration data $\mathcal{D} = \{(s_i)\}_{i=1}^N$, empty replay buffer $\mathcal{B}$, randomly initialized discriminator model $D_\omega$, state transition predictor $h_\psi$ and parameterized inverse dynamics model $I_\phi$
2: for $k = 0, 1, 2, \cdots$ do
3:     ▷ Pre-training stage
4:     Collect trajectories $\{(s, a, s', r, \text{done})\}$ using a random initialized policy $\pi = \mathbb{E}_{\epsilon\sim\mathcal{N}}\left[I_\phi(a|s, h_\psi(\epsilon; s))\right]$ and store in $\mathcal{B}$
5:     Sample $(s, a, s') \sim \mathcal{B}$ and update $\phi$ by $L^I$ (Eq. (9))
6: end for
7: for $k = 0, 1, 2, \cdots$ do
8:     ▷ Online training stage
9:     Collect trajectories $\{(s, a, s', r, \text{done})\}$ using current policy $\pi = \mathbb{E}_{\epsilon\sim\mathcal{N}}\left[I_\phi(a|s, h_\psi(\epsilon; s))\right]$ and store in $\mathcal{B}$
10:    Sample $(s, a, s') \sim \mathcal{B}, (s, s') \sim \mathcal{D}$
11:    if Learn inverse dynamics function then
12:        repeat
13:            Update $\phi$ by $L^I$ (Eq. (9))
14:        until Converged
15:    end if
16:    Update the discriminator $D_\omega$ with the loss:
           $$\mathcal{L}_\omega^D = -\mathbb{E}_{(s,s')\sim\mathcal{B}}[\log D_\omega(s, s')] - \mathbb{E}_{(s,s')\sim\mathcal{D}}[\log(1 - D_\omega(s, s'))]\ ,$$
17:    Update $\psi$ by $\mathcal{L}_\psi^{\pi,h}$ (Eq. (17))
18: end for
```

B. Additional Related Works

B.1. Transferable Imitation Learning

Throughout this work, we've discussed several solutions in SOIL and works that utilize either inverse dynamics or state predictor in Section 2 and summarized the difference of these works in Tab. 1. Moreover, in our paper, we pointed out the most important feature of DePO is that it allows transferable imitation learning to different action spaces or action dynamics. Before DePO, there are a few works have investigated transferable imitation learning. For instance, SOIL works can be essentially used for mimicking the expert state transition sampled with different action dynamics (Liu et al., 2020; Jiang et al., 2020), but they still learn an ad-hoc state-to-action policy which cannot be further transferred to other tasks; (Lee et al., 2021) proposed generalizable imitation learning for goal-directed tasks, but their generalized ability only limits in unseen states and goals in the same environment. Besides, (Srinivas et al., 2018) proposed an interesting reward transferring solution. Specifically, they imitate an action sequence using a gradient descent planner, which consists of an encoder module mapping the pixels into latent spaces. The encoder can be further used for constructing an obstacles-aware reward function by measuring the distance from the goal to the current observation. The authors conduct a series of experiments indicating that such an encoder and the corresponding reward function can be transferred to different environments, and even different robots to train different ad-hoc RL agents.

In comparison, the transferring ability of DePO comes from the pre-trained state planner model, which can be directly applied to different action spaces or dynamics, with little sampling cost on training the inverse dynamics model (or even no training at all if we have a ground truth inverse dynamics function serving as a control module).

B.2. Relation to Hierarchical Reinforcement Learning

Our work also relates a lot to Hierarchical Reinforcement Learning (HRL), since the role of the state planner behaves like a high-level policy and the inverse dynamics can be regarded as the low-level policy. In recent HRL works, a typical paradigm is to train the high-level policy using ground-truth rewards to predict the current goal for the low-level policy to achieve, while the low-level policy is trained by a handcrafted goal-matching reward function and provided the action to interact with the environment. Notably, many HRL algorithms represent the goals within a learned embedding space (or know as option) (Konidaris & Barto, 2007; Heess et al., 2016; Kulkarni et al., 2016; Vezhnevets et al., 2017; Zhang et al., 2022), instead of the original state space; and these algorithms either keep a high-level action (option) for a fixed timesteps (Nachum et al., 2018; Vezhnevets et al., 2017) or learn a function to change the option (Zhang & Whiteson, 2019; Bacon et al., 2017). The most similar HRL work is (Nachum et al., 2018), whose high-level policy predicts in the raw form. However, there are lots of differences between their work and ours. First, they still train the high-level and low-level policies with separate reward functions and RL objectives, while our DePO optimizes both modules end-to-end; furthermore, their high-level policy holds a fixed goal for c steps indicating the desire for the low-level policy to yield after c steps, making it hard to be transferred; finally, their main contribution lies in improving the learning efficiency on complicated tasks yet we provide a way for transferring the high-level state planner to different action spaces and dynamics. In (Heess et al., 2016) they also try to transfer after learning the hierarchical modules; nevertheless, they pre-train and transfer the low-level controller which is fixed when training a different high-level controller. Like DePO, (Li et al., 2020) also optimized the hierarchical policy in an end-to-end way and derived the hierarchical policy gradient for optimizing the two-level policy jointly, but their high-level policy predicts a latent skill and fixes the decision for c timesteps. In comparison, DePO obtains and fixes the low-level inverse dynamics (controllers) before optimizing the high-level state planner.

C. Experiment

C.1. Experiment Settings

C.1.1. Real-World Traffic Dataset

NGSIM I-80 dataset includes three videos with a total length of 45 minutes recorded in a fixed area, from which 5596 driving trajectories of different vehicles can be obtained. The original data is equally split into three time intervals, each of which has the length of 15 minutes. We use the first split, among which we choose the first 1000 trajectories for our experiment. We choose 85% of these trajectories as the training set and the remaining 15% as the test set. In our experiment, the state space includes the position and velocity vectors of the ego vehicle and six neighbor vehicles, and the actions are vectors of acceleration and the change in steering angle.

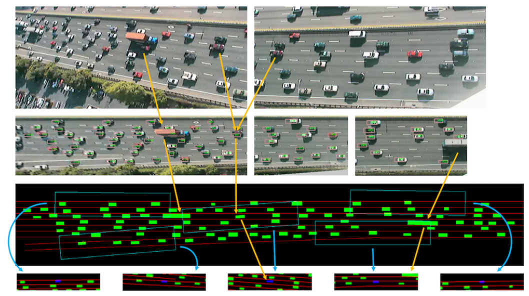


Figure 7: Visualization of NGSIM I-80 data set and its mapping on the simulator. This figure is borrowed from (Henaff et al., 2019).

C.1.2. Implementation Details

For all experiments, we implement the value network as two-layer MLPs with 256 hidden units and the discriminator as 128 hidden units. For DePO and GAIfO-DP, the state predictor is a two-layer MLP with 256 hidden units and the inverse dynamics model is a four-layer MLP with 512 hidden units. For GAIfO and BCO, the policy network is a two-layer MLP with 256 hidden units. For fairness, we re-implement all the algorithms based on a Pytorch code framework[1] and adopt Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as the RL learning algorithm for GAIfO and DePO. All algorithms are trained with the same frequency and gradient steps.

For Mujoco benchmarks, we train an SAC agent to collect expert data, and take it for training the imitation learning agents. At training time we remove the terminal states and each episode will not end until 1000 steps. At testing time the terminal states are set for fair comparison. All algorithms are evaluated by deterministic policies. The codes are now public at

[1]https://github.com/Ericonaldo/ILSwiss.

https://github.com/apexrl/DePO.

For the NGSIM driving experiment, the original state contains the information of other cars, which is hard to predict. Therefore, we ignore it when predicting the state transition and the action of inverse dynamics. Formally, denote the state of all vehicles as s and the state of the ego vehicle as s_e, the state planner predicts s'_e from s: $s'_e \sim h(s'_e|s)$. The inverse dynamics model predicts the controllable action a based on the consecutive states of ego vehicle: $a \sim I(a|s_e, s'_e)$. The discriminator is also constructed as differentiating the states of all vehicles $D(s, s)$. The codes are now public at https://github.com/apexrl/DePO_NGSIM.

During training, we randomly pick one car to be controlled by the policy at the beginning of every episode, and we replay the other cars by data. The episode ends when cars collide or successfully get through the road. To reduce the sampling time in the driving simulator, we implement parallel sampling using Python *multiprocessing* library. In practice, we run 25 simulators to collect samples at the same time.

For training DePO on both Mujoco and NGSIM driving experiment, we normalize the Q value into an interval of $[0, 1]$ for CDePG and keeps the Q value for DePG. Since CDePG (Eq. (14)) can actually be treated as a weighted MLE objective as discussed in Appendix C.4 and the supervised learning loss of the state planner (Eq. (10)) can also be implemented as an MLE objective, thereafter, in our implementation, we additionally allocate the maximum Q value (which is exactly 1 after normalization) as the weight for the supervised loss, and λ_h also works for CDePG. In other words, we optimize the following objectives in these experiments:

$$\begin{aligned} &\min_{\phi} L^I \quad \text{(if } I \text{ need to be learned)}, \\ &\min_{\psi} \mathcal{L}^{\pi,h} = \mathcal{L}^{\pi}_{\text{DePG}} + \lambda_h(\mathcal{L}^h + \mathcal{L}^{\pi}_{\text{CDePG}}) \ , \end{aligned} \tag{17}$$

C.1.3. Transferring Settings

For transferring experiments on Mujoco, we try two kinds of dynamics setting: i) **simple** transfer (results can be referred to Appendix C.3), we invert the original action dynamics in each task; in other words, the original environment transition function takes the addictive inverse of the input action. For example, an action of -0.5 in the transferring experiments will lead to the same results as the action of 0.5 in the imitation experiments under the same state. ii) **complex** transfer (results can be referred to Section 5.3): we take an 80% of the original gravity (exactly your proposal 1) with a complicated dynamics for the transferring experiment (different both action space and dynamics). Particularly, given the original action space dimension m and dynamics $s' = f_s(a)$ on state s, the new action dimension and dynamics become $n = 2m$ and $s' = f_s(h(a))$, where h is constructed as:

$$h = -\exp(a[0:n/2]+1) + \exp(a[n/2:-1]))/1.5 \ ,$$

here $a[i:j]$ selects the i-th to $(j-1)$-th elements from the action vector a. In other words, we transfer to a different gravity setting while doubling the action space and construct a more complicated action dynamics for agent to learn.

C.1.4. Hyperparameters

We list the key hyperparameters of the best performance of DePO on each task in Tab. 2. For each task, except the additional parameter – state planner coefficient λ_h, the other hyperparameters are the same for all tested methods (if they need such parameters).

C.2. Additional Experiment Results

C.3. Simple Transfer on Mujoco

In our early submission, we first try a naive transfer setting, i.e, inverted the action dynamics as stated in Appendix C.1.3, to illustrate the advantage of DePO, shown in Fig. 8. By comparing the results of two transferring scenarios, we can observe that the complex setting does not affect much to the performance of DePO, while the other baselines do not perform in complex transfer setting as well as they do in simple setting.

C.3.1. Does DePO Reach Where it Predicts?

In this section, we aim to study whether the agent can reach where it plans to go. Therefore, we analyze the distance of the reaching states and the predicted consecutive states and draw the mean square error (MSE) along the imitation training

Table 2: Hyperparameters of DePO. Note that in our experiment, except the additional state planner coefficient λ_h, other hyperparameters are the same for all tested methods.

Environments	Invert.	Hop.	Walk.	Half.	Human.	NGSIM.
Trajectory maximum length	1000					1500
Optimizer	AdamOptimizer					
Discount factor γ	0.99					
Replay buffer size	2e5					2e6
Batch size	256					1024
State planner coefficient λ_h	0.1			0.01		1.0
Q learning rate	3e-4					
π learning rate	3e-4					
D learning rate	3e-4					
I learning rate	1e-4					
I learning interval (epochs)	10					
Gradient penalty weight	4.0		8.0	0.5	16.0	4.0
Reward scale	2.0					

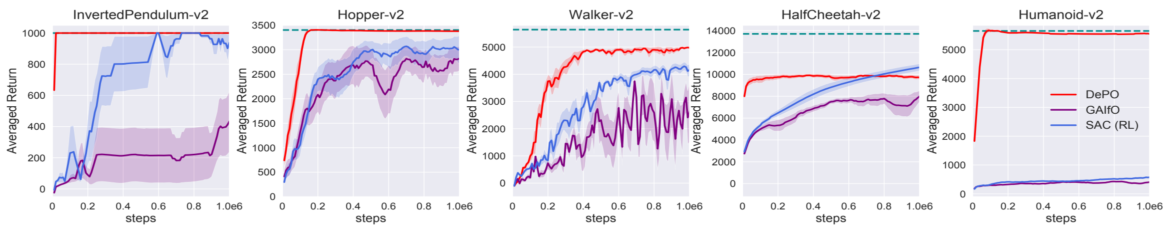


Figure 8: Transferring by pre-training on Mujoco tasks with inverted action dynamics within 1e6 interaction steps.

procedure in Fig. 9. As shown in the figure, as the training goes, the averaged gap between the planned states and the states to achieve becomes smaller. On the contrary, the two variants, i.e., DePO (Supervised) and GAIfO-DP does not show any consistency. Particularly, DePO (Supervised) has little prediction error on two simple domains, but from the return curves, we believe that it stays at the initial state without any exploring; on the other tasks, the error does not even converge. Likewise, the prediction errors of GAIfO-DP diverge in most cases since there is no signal for the state planner to learn to predict the target state. The only exception is on Humanoid, where GAIfO-DP surprisingly converged as DePO does. We guess the reason may be the shallow layers of the policy (layers of state planner) learn an identical mapping, and the errors between two consecutive states are small. As for DePO, we further illustrate that the state planner does predict reasonable state transitions (Appendix C.3.2), which is even accurate enough for multi-step planning.

From our experience, the agent must have adequate exploration during the early learning stage, which may cause a large difference between the desired targets and the reached ones. If not, the agent tends to spin in a small local area, thus deteriorating the final performance. A piece of evidence showing such problems is the performance of DePO (Supervised). Except on HalfCheetah where the MSE explodes, the differences between the reaching states and the predictions on the other tasks are all limited. However, on Hopper and Walker, the imitation performance is bad since the prediction is only around the neighbor of the initialized state.

We also show the MSE curves during the transferring experiment in Fig. 10 and Fig. 11. As expected, the target state predictions of DePO are still stably accurate along the whole training stage.

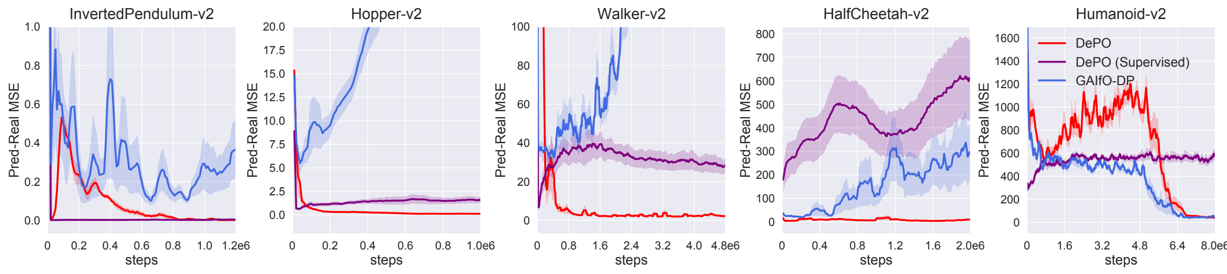


Figure 9: MSE curves of the one-step prediction of the state planner and the real state achieved in the environment for imitation experiments. The target state predictions are accurate and the differences are getting smaller during the training.

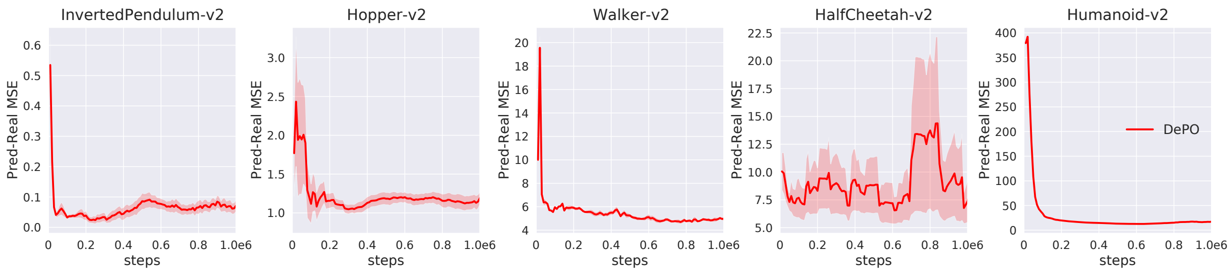


Figure 10: MSE curves of the one-step prediction of the state planner and the real state that the agent achieves in the environment in **complex** transfer experiments. The target state predictions are stably accurate along the whole training stage.

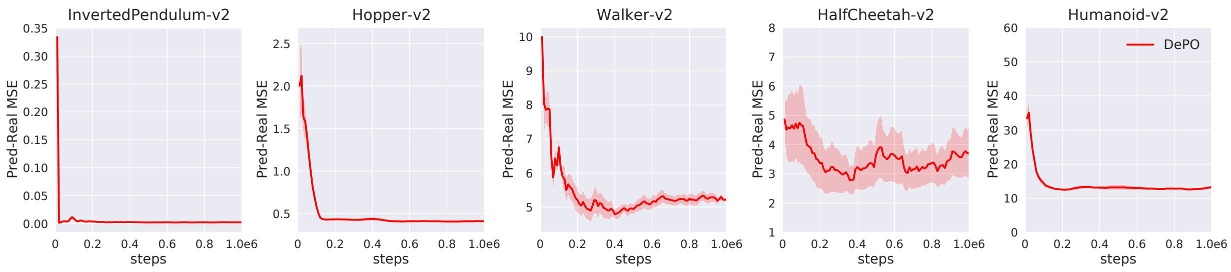


Figure 11: MSE curves of the one-step prediction of the state planner and the real state that the agent achieves in the environment in **simple** transfer experiments. The target state predictions are stably accurate along the whole training stage.

C.3.2. How Can the State Predictor be Used for Planning?

In our experiments, we find the state planner is rather accurate for predicting the next state with small compounding errors, and therefore we want to know how it can be used for multi-step planning without interacting with the environment. In this section, we compare the image of the imagined rollouts provided by the state planner and the real rollout that the agent achieves during its interactions with the environments. Note that the imagined rollouts are generated by consecutively predicting from predicted states. Formally, on a state s, h_ψ predicts the next possible state $\hat{s}'$ to reach, then $\hat{s}'$ is further taken as an input to the predictor and predicts the two-step away state $\hat{s}''$. We repeat the cycle of prediction on prediction progressively until the imagined state makes no sense, i.e., the compounding error explodes. The results on Mujoco tasks are shown in Fig. 12 and NGSIM is in Fig. 13. It is surprising that the planner can rollout at least tens or even hundreds of steps accurately.

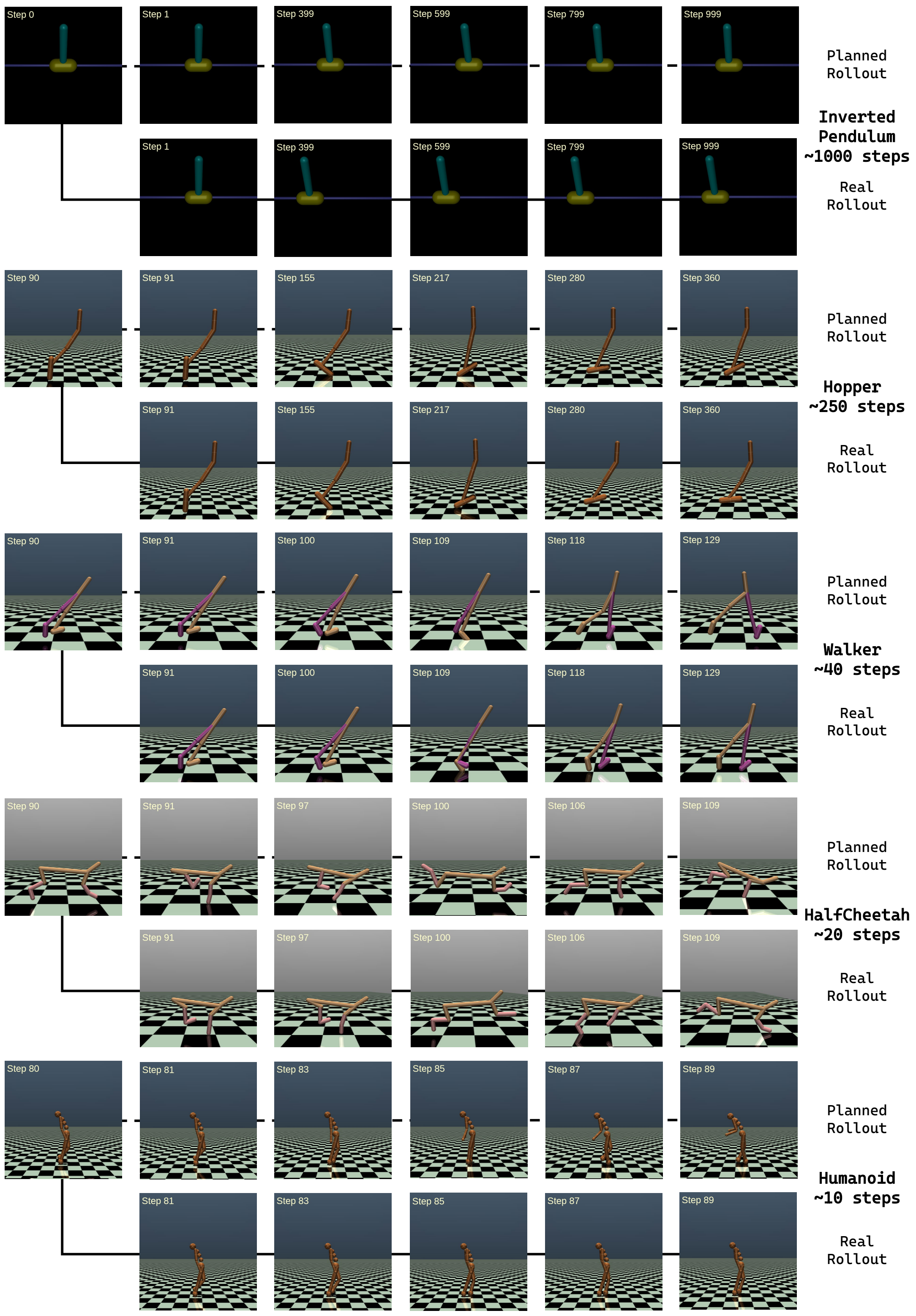


Figure 12: Imagined rollout by state planner and the real rollout that was achieved during the interaction with the environment on Mujoco tasks. We also provide a corresponding demo video in `https://youtu.be/WahVjjvcYYM`.

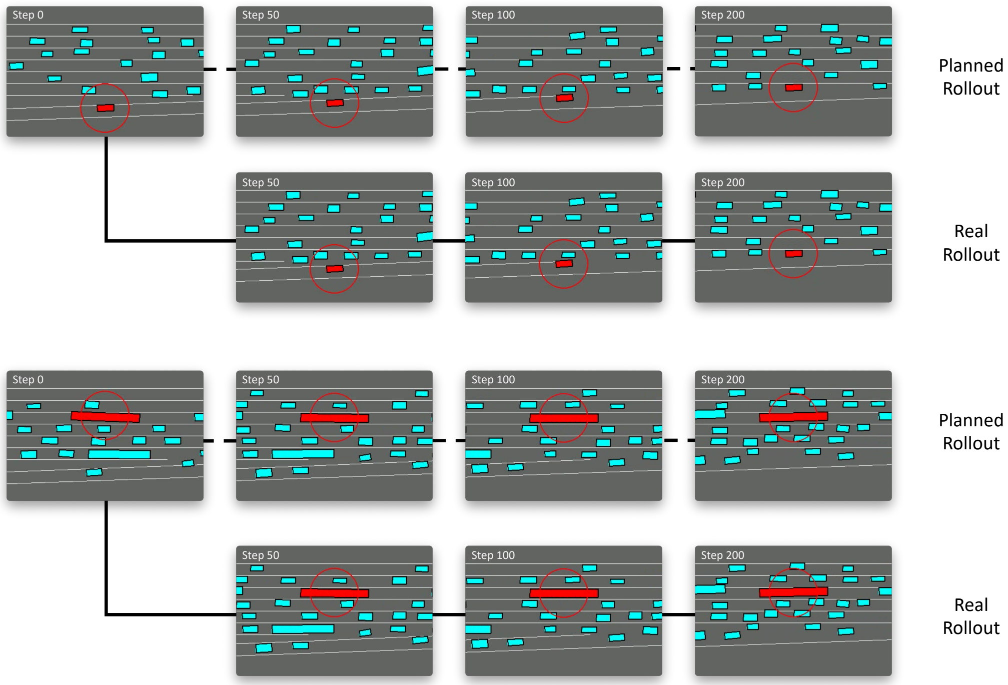


Figure 13: Imagined rollout by state planner and the real rollout that achieved during the interaction with the environment on NGSIM dataset.

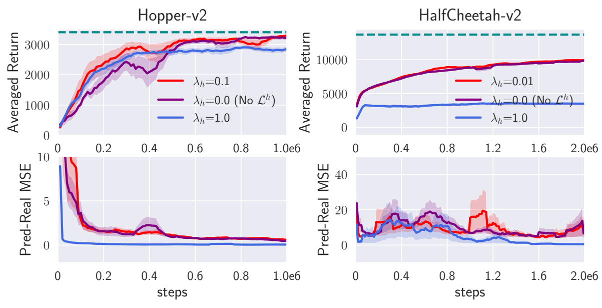
C.4. Ablation Studies

Ablation on λ_h. For DePO, the only important hyperparameter is the supervised coefficient of state planner λ_h. In our method, we combine supervised learning with policy gradient to train the state planner. Therefore, we first investigate the effect of the supervised learning objective by setting $\lambda_h = 0.0$ (i.e., removing $\mathcal{L}^h$). The results are shown Fig. 14(a), which indicates that the supervised learning actually does not affect much about the training procedure of DePO, and therefore can be removed. However, on some tasks like Hopper, augmenting expert supervision can lead to better learning efficiency. In addition, a large λ_h actually deteriorates the final performance, especially on harder tasks, while the prediction error can be even smaller. Therefore, we believe the deterioration comes from the poor exploration ability of the large λ_h, which asks high accuracy of the predictions on expert data even at the beginning of training.

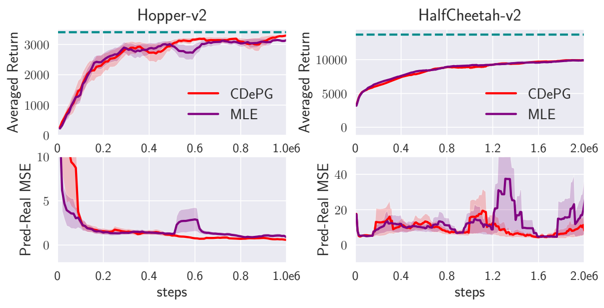
Ablation on CDePG. In Section 4.3, we propose CDePG, which is used to alleviate the agnostic challenge of DePG by training the state planner with real experience sampled from the environment. From another point of view, updating CDePG (Eq. (14)) is learning towards a weighted MLE objective (which is actually also the case for vanilla policy gradient), where the value function $Q(s, a)$ serving as the weight. To better understand how CDePG works, we conduct an ablation experiment on removing the weight Q, i.e., we are now optimizing an MLE objective (but also with coefficient λ_h for better exploration) instead of CDePG. The results are shown in Fig. 14(a), indicating that MLE actually achieves similar results to CDePG, although the MSE errors can be worse. From such a perspective, DePO can also be realized as regularizing an MLE constraint on DePG. In addition, without such regularization, DePO simply works bad with large MSE errors on all environments, due to the agnostic update of the state planner.

C.5. Additional Results for NGSIM Experiments

Statistical evaluation metrics. We utilize *Success Rate*, *Mean Distance* and *KL Divergence* as evaluation metrics for evaluating the final performances. Specifically, *Success Rate* is the percentage of driving across the entire area without crashing into other vehicles or driving off the road, *Mean Distance* is the distance traveled before the episode ends, and *KL Divergence* measures the position distribution distance between the expert and the agent. The results are presented in Tab. 3. All metrics are taken average over the aforementioned three different action dynamics. Note that for each method, the performances are evaluated using the best model with the highest success rate.



(a) Ablation study on λ_h.



(b) Ablation study on CDePG.

Figure 14: Ablation studies.

Table 3: Statistical performance on NGSIM I-80 driving task over 5 random seeds.

Method	Success Rate (%)	Mean Distance (m)	KL Divergence
GAIfO	70.3 ± 2.5	180.0 ± 0.9	23.0 ± 10.2
DePO/Co	84.2 ± 5.2	192.4 ± 4.6	12.4 ± 7.1
Expert	100.0	210.0	0.0

One-shot transferring by pre-training. We illustrate the transfer ability of DePO with ground-truth inverse dynamics function on NGSIM I-80 datasets. Specifically, we train the high-level state-to-state state planner on the *Normal* action dynamics, and then transferred to the *Inverse* and *Transpose* action dynamics. The results are shown in Fig. 15. As expected, with a known inverse dynamics function, the state planner can be directly deployed on different skills, which does not require additional sampling and training cost. However, the compared baseline method (GAIfO) needs to learn a state-to-action mapping policy for every agent separately.

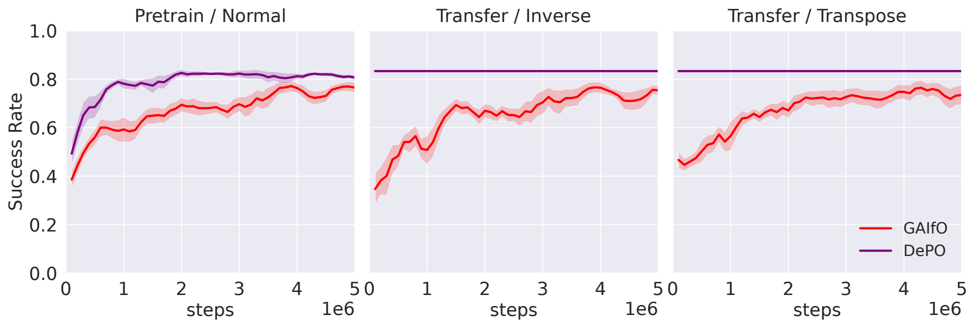


Figure 15: One-shot Transferring by pre-training on NGSIM dataset with different action dynamics.

C.6. RL Training and Transferring

RL training. We have mentioned that the key induction of DePG and CDePG does not limit in the literature of SOIL such that DePO can be applied to general RL tasks. In this section, we conduct RL experiments on Mujoco tasks comparing with SAC using the reward provided by the environments instead of learning from expert demonstrations. In this case, we do not have to keep a discriminator D or optimize the supervised loss $\mathcal{L}^h$. The learning curves are shown in Fig. 16, revealing the considerable performance of DePO. Although the learning efficiency and final performance may be lightly inferior to SAC, DePO can learn accurate high-level state planners that can be used for transferring (Fig. 17).

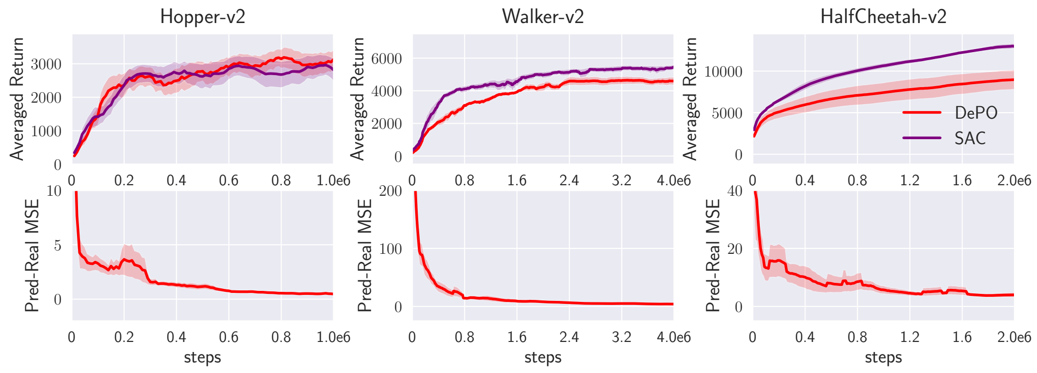


Figure 16: RL experiments on Mujoco tasks over 5 random seeds.

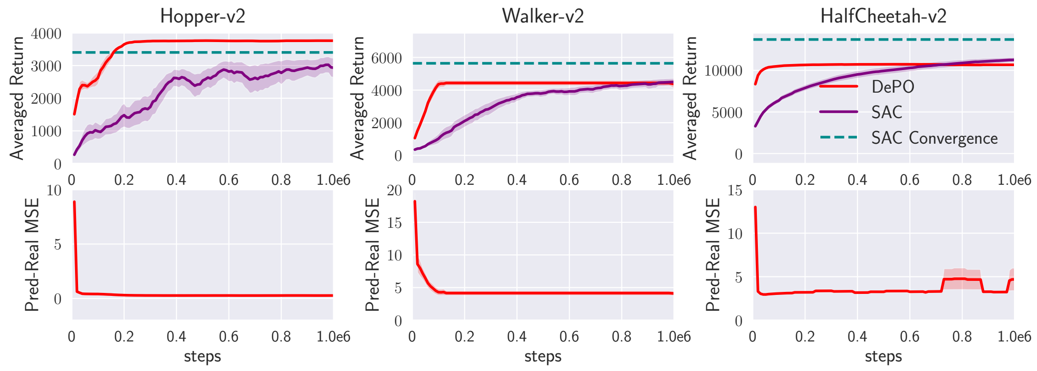


Figure 17: One-shot Transferring by pre-training RL agents on Mujoco tasks with inverted action dynamics over 5 random seeds.

One-shot transferring by pre-training From the bottom of Fig. 16, we know that DePO still learns accurate high-level state planners, and therefore we are able to transfer the learned planners to tasks that require different skills. We also test transferring on inverted action dynamics(Appendix C.3), and the results are shown in Fig. 17. By pre-training a great state planner, DePO can achieve a stable performance on each task, and can even outperform the averaged convergence performance of SAC (on Hopper).

D. Proofs

In our proofs we will work in finite state and action spaces $\mathcal{S}$ and $\mathcal{A}$ to avoid technical machinery out of the scope of this paper.

Proposition D.1. *Suppose Π is the policy space and $\mathcal{P}$ is a valid set of state transition OMs such that $\mathcal{P} = \{\rho : \rho \geq 0$ and $\exists \pi \in \Pi, s.t.\ \rho(s,s') = \rho_0(s)\int_a \pi(a|s)\mathcal{T}(s'|s,a)\,\mathrm{d}a + \int_{s'',a}\pi(a|s)\mathcal{T}(s'|s,a)\rho(s'',s)\,\mathrm{d}s''\,\mathrm{d}a\}$, then a policy $\pi \in \Pi$ corresponds to one state transition OM $\rho_\pi \in \mathcal{P}$. However, under the action-redundant assumption about the dynamics $\mathcal{T}$, a state transition OM $\rho \in \mathcal{P}$ can correspond to more than one policy in Π.*

Proof. We first provide the proof for the one-to-one correspondence between marginal distribution $\sum_a \pi(a|s)\mathcal{T}(s'|s,a)$ and state transition OM $\rho(s,s') \in \mathcal{P}$.

For a given policy π, by definition of state transition OM, we have

$$\begin{aligned}\rho_\pi(s,s') &= \sum_a \mathcal{T}(s'|s,a)\rho_\pi(s,a) \\ &= \sum_a \pi(a|s)\mathcal{T}(s'|s,a)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)\,.\end{aligned} \tag{18}$$

For all t greater than or equal to 1, we have

$$P(s_t = s|\pi) = \sum_{s''} P(s_{t-1} = s'', s_t = s|\pi) \ . \tag{19}$$

Take Eq. (19) into Eq. (18), we have

$$\begin{aligned}
\rho_\pi(s, s'|\pi) &= P(s_0 = s, s_1 = s'|\pi) + \sum_a \pi(a|s)\mathcal{T}(s'|s,a) \sum_{t=1}^{\infty} \gamma^t \sum_{s''} P(s_{t-1} = s'', s_t = s|\pi) \\
&= P(s_0 = s, s_1 = s'|\pi) + \gamma \sum_a \pi(a|s)\mathcal{T}(s'|s,a) \sum_{s''} \sum_{t=0}^{\infty} \gamma^t P(s_t = s'', s_{t+1} = s|\pi) \\
&= P(s_0 = s, s_1 = s'|\pi) + \gamma \sum_a \pi(a|s)\mathcal{T}(s'|s,a) \sum_{s''} \rho_\pi(s'', s) \\
&= \rho_0(s) \sum_a \pi(a|s)\mathcal{T}(s'|s,a) + \gamma \sum_{s'',a} \pi(a|s)\mathcal{T}(s'|s,a)\rho_\pi(s'', s)
\end{aligned} \tag{20}$$

Consider the following equation of variable ρ:

$$\rho(s, s') = \rho_0(s) \sum_a \pi(a|s)\mathcal{T}(s'|s,a) + \gamma \sum_{s'',a} \pi(a|s)\mathcal{T}(s'|s,a)\rho(s'', s) \ . \tag{21}$$

According to Eq. (20), ρ_π is a solution of Eq. (21). Now we proceed to prove ρ_π as the unique solution of Eq. (21). Define the matrix

$$A_{(ss',s''s)} \triangleq \begin{cases} 1 - \gamma \sum_a \pi(a|s)\mathcal{T}(s'|s,a) & \text{if } (s,s') = (s'',s) \\ -\gamma \sum_a \pi(a|s)\mathcal{T}(s'|s,a) & \text{otherwise .} \end{cases}$$

Note that A is a two-dimensional matrix indexed by state transition pairs. Also define the vector

$$b_{s,s'} \triangleq \rho_0(s) \sum_a \pi(a|s)\mathcal{T}(s'|s,a) \ .$$

We can rewrite Eq. (21) equivalently as

$$A\rho = b \ . \tag{22}$$

Since $\sum_{s',a} \pi(a|s)\mathcal{T}(s'|s,a) = 1$ and $\gamma < 1$, for all (s'', s), we have

$$\begin{aligned}
&\sum_{s,s'} \gamma \sum_a \pi(a|s)\mathcal{T}(s'|s,a) = \gamma < 1 \\
&\Rightarrow 1 - \gamma \sum_a \pi(a|s'')\mathcal{T}(s|s'',a) > \sum_{(s,s')\neq(s'',s)} \gamma \sum_a \pi(a|s)\mathcal{T}(s'|s,a) \\
&\Rightarrow \left|A_{(s''s,s''s)}\right| \geq \sum_{(s,s')\neq(s'',s)} \left|A_{(ss',s''s)}\right| \ .
\end{aligned}$$

Therefore, we have proven A as column-wise strictly diagonally dominant, which implies that A is non-singular, so Eq. (21) has at most one solution. Since for all ρ in $\mathcal{P}$, it must satisfy the constraint Eq. (21), which means that for any marginal distribution $\sum_a \pi(a|s)\mathcal{T}(s'|s,a)$, there is only one corresponding ρ in $\mathcal{P}$.

Now, we proceed to prove that for every ρ in $\mathcal{P}$, there is only one corresponding marginal distribution $\sum_a \pi(a|s)\mathcal{T}(s'|s,a)$ such that $\rho_\pi = \rho$. By definition of $\mathcal{P}$, ρ is the solution of Eq. (21) for some policy π. By rewriting Eq. (21), the marginal distribution can be written in the form of a function expression of ρ as

$$\sum_a \pi(a|s)\mathcal{T}(s'|s,a) = \frac{\rho(s,s')}{\rho_0(s) + \gamma \sum_{s''} \rho(s'',s)} \ . \tag{23}$$

This means every $\rho \in \mathcal{P}$ only corresponds to one marginal distribution $\sum_a \pi(a|s)\mathcal{T}(s'|s,a)$. As we discussed before, ρ is the state transition OM of π, i.e., $\rho = \rho_\pi$.

By establishing the one-to-one correspondence between the marginal distribution $\sum_a \pi(a|s)\mathcal{T}(s'|s,a)$ and state transition OM $\rho \in \mathcal{P}$, we can alternatively study the correspondence between the marginal distribution and policy. Obviously, one policy can only correspond to one marginal distribution. We now prove that if the dynamics $\mathcal{T}$ has redundant actions, one marginal distribution can correspond to more than one policy in π.

We prove the statement by counterexample construction. If the dynamics $\mathcal{T}$ has redundant actions, there exist $s_m \in \mathcal{S}$, $a_n \in \mathcal{A}$ and distribution p defined on $\mathcal{A} \setminus \{a_n\}$ such that $\sum_{a\in\mathcal{A}\setminus\{a_n\}} p(a)\mathcal{T}(s'|s_m,a) = \mathcal{T}(s'|s_m,a_n)$. Consider two policy π_0 and π_1 such that

$$\begin{cases} \pi_0(a|s) = \pi_1(a|s) & \text{if } s \neq s_m \\ \pi_0(a_n|s_m) = 1 & \\ \pi_0(a|s_m) = 0 & \text{if } a \neq a_n \\ \pi_1(a_n|s_m) = 0 & \\ \pi_1(a|s_m) = p(a) & \text{if } a \neq a_n \ . \end{cases} \tag{24}$$

From Eq. (24), we know that π_0 and π_1 are two different policies. However, they share the same marginal distribution $\sum_a \pi(a|s)\mathcal{T}(s'|s,a)$. To justify this, we first consider the case when s equals to s_m, where we have

$$\begin{aligned} \sum_a \pi_0(a|s_m)\mathcal{T}(s'|s_m,a) &= \pi_0(a_n|s_m)\mathcal{T}(s'|s_m,a_n) + \sum_{a\in\mathcal{A}\setminus\{a_n\}} \pi_0(a|s_m)\mathcal{T}(s'|s_m,a) \\ &= \mathcal{T}(s'|s_m,a_n) \\ &= \sum_{a\in\mathcal{A}\setminus\{a_n\}} \pi_1(a|s_m)\mathcal{T}(s'|s_m,a) \\ &= \sum_{a\in\mathcal{A}\setminus\{a_n\}} \pi_1(a|s_m)\mathcal{T}(s'|s_m,a) + \pi_1(a_n|s_m)\mathcal{T}(s'|s_m,a_n) \\ &= \sum_a \pi_1(a|s_m)\mathcal{T}(s'|s_m,a) \ . \end{aligned} \tag{25}$$

When s does not equal to s_m, the equality holds trivially, since the action selection probability of π_0 and π_1 defined on these states are exactly the same. Thus, one marginal distribution can correspond to more than one policy in π when there are redundant actions.

□

Proposition D.2. *Suppose the state transition predictor h_Ω is defined as in Eq. (3) and $\Gamma = \{h_\Omega : \Omega \in \Lambda\}$ is a valid set of the state transition predictors, $\mathcal{P}$ is a valid set of the state-transition OMs defined as in Proposition 4.3, then a state transition predictor $h_\Omega \in \Gamma$ corresponds to one state transition OM $\rho_\Omega \in \mathcal{P}$; and a state transition OM $\rho \in \mathcal{P}$ only corresponds to one hyper-policy state transition predictor such that $h_\rho = \rho(s,s')/\int_{s'} \rho(s,s')\,\mathrm{d}s'$.*

Proof. During the proof of Proposition 4.3, we have an intermediate result that there is one-to-one correspondence between the marginal distribution $\sum_a \pi(a|s)\mathcal{T}(s'|s,a)$ and state transition OM $\rho \in \mathcal{P}$. Since the definition of state transition predictor is exactly $h_\Omega(s'|s) = \sum_a \pi(a|s)\mathcal{T}(s'|s,a)$ $(\forall \pi \in \Omega)$, the one-to-one correspondence naturally holds between state transition predictor $h(s'|s)$ and state transition OM $\rho \in \mathcal{P}$.

□

Theorem D.3. *Let $\mathcal{A}(s)$ be an action set that for all a in $\mathcal{A}(s)$, a deterministic transition function leads to the same state $s' = \mathcal{T}(s,a)$. If there exists an optimal policy π^* and a state $\hat{s}$ such that $\pi^*(\cdot|\hat{s})$ is a distribution over $\mathcal{A}(\hat{s})$, then we can replace $\pi^*(\cdot|\hat{s})$ with any distributions over $\mathcal{A}(\hat{s})$ which does not affect the optimality.*

Proof. We denote the policy after replacing $\pi^*(\cdot|\hat{s})$ with a distribution p over $\mathcal{A}(\hat{s})$ as π, and we have

$$\pi(a|s) = \begin{cases} \pi^*(a|s), & s \neq \hat{s}, \\ p(a), & s = \hat{s}. \end{cases} \tag{26}$$

The Bellman equation of V^{π^*} can be written as

$$V^{\pi^*}(s) = \mathbb{E}_{a\sim\pi^*(\cdot|s), s'\sim\mathcal{T}(\cdot|s,a)}[r(s,a) + V^{\pi^*}(s')] \ . \tag{27}$$

Since π and π^* only differ at state $\hat{s}$, for all $s \neq \hat{s}$, we have

$$\begin{aligned} V^{\pi^*}(s) &= \mathbb{E}_{a\sim\pi^*(\cdot|s), s'\sim\mathcal{T}(\cdot|s,a)}[r(s,a) + V^{\pi^*}(s')] \\ &= \mathbb{E}_{a\sim\pi(\cdot|s), s'\sim\mathcal{T}(\cdot|s,a)}[r(s,a) + V^{\pi^*}(s')] \ . \end{aligned} \tag{28}$$

Notice that for all a in $\mathcal{A}(\hat{s})$, the deterministic transition function lead to the same state $\hat{s}' = \mathcal{T}(\hat{s},a)$. Since our reward function is state-only and defined on (s,s'), the reward $r(\hat{s},a) = r(\hat{s},\hat{s}')$ must be the same for all a in $\mathcal{A}(\hat{s})$. Therefore, we have

$$\begin{aligned} V^{\pi^*}(\hat{s}) &= \mathbb{E}_{a\sim\pi^*(\cdot|\hat{s})}[r(\hat{s},a)] + V^{\pi^*}(\hat{s}') \\ &= r(\hat{s},\hat{s}') + V^{\pi^*}(\hat{s}') \\ &= \mathbb{E}_{a\sim p(\cdot)}[r(\hat{s},a)] + V^{\pi^*}(s') \ , \end{aligned} \tag{29}$$

which means Eq. (28) also holds for $\hat{s}$. Thus we reach the result that

$$\forall s,\ V^{\pi^*}(s) = \mathbb{E}_{a\sim\pi(\cdot|s), s'\sim\mathcal{T}(\cdot|s,a)}[r(s,a) + V^{\pi^*}(s')] \ . \tag{30}$$

According to policy evaluation theorem, we know that V^{π^*} is also the value function of policy π. Since V^{π} is the optimal value function, we complete the proof that π is also an optimal policy.

□

Theorem D.4 (Error Bound of DePO). *Consider a deterministic environment whose transition function $\mathcal{T}(s,a)$ is deterministic and L-Lipschitz. Assume the ground-truth state transition $h_{\Omega_E}(s)$ is deterministic, and for each policy $\pi \in \Pi$, its inverse dynamics I_π is also deterministic and C-Lipschitz. Then for any state s, the distance between the desired state s'_E and reaching state s' sampled by the decoupled policy is bounded by:*

$$\|s' - s'_E\| \leq LC\|h_{\Omega_E}(s) - h_\psi(s)\| + L\|I_{\tilde{\pi}}(s,\hat{s}') - I_\phi(s,\hat{s}')\| \ , \tag{31}$$

where $\tilde{\pi}$ is a sampling policy that covers the state transition support of the expert hyper-policy and $\hat{s}' = h_\psi(s)$ is the predicted consecutive state.

Proof. Given a state s, the expert takes a step in a deterministic environment and get s'. We assume that the expert Ω_E can use any feasible policy $\tilde{\pi}$ that covers the support of Ω_E to reach s:

$$s'_E = \mathcal{T}(s, I_{\tilde{\pi}}(s, h_\Omega(s))) \tag{32}$$

Similarly, using decoupled policy, the agent predict $\hat{s}' = h_\psi(s)$ and infer an executing action by an inverse dynamics model $a = I_\phi(s, s')$, which is learned from the sampling policy $\tilde{\pi}$. Denote the reaching state of the agent as s':

$$s' = \mathcal{T}(s, I_\phi(s, h_\psi(s))) \tag{33}$$

Therefore, the distance between s' and s'_E is:

$$\|s' - s'_E\| = \|\mathcal{T}(s, I_{\tilde{\pi}}(s, h_\Omega(s))) - \mathcal{T}(s, I_\phi(s, h_\psi(s)))\|$$

Let's consider the deterministic transition on s is a function of a such that $s' = \mathcal{T}^s(a)$, then we continue the deviation:

$$\begin{aligned}\|s' - s'_E\| &\le \|\mathcal{T}^s(I_{\tilde{\pi}}(s, h_\Omega(s))) - \mathcal{T}^s(I_\phi(s, h_\psi(s)))\| \\ &\le L\|I_{\tilde{\pi}}(s, h_\Omega(s))) - I_\phi(s, h_\psi(s))\| \\ &\le L\|I_{\tilde{\pi}}(s, h_\Omega(s))) - I_{\tilde{\pi}}(s, h_\psi(s))) + I_{\tilde{\pi}}(s, h_\psi(s))) - I_\phi(s, h_\psi(s))\|\end{aligned}$$

Similarly we also take the inverse transition on s is a function of s' such that $a = I^s(s')$, then we have that:

$$\begin{aligned}\|s' - s'_E\| &\le L\|I^s_{\tilde{\pi}}(h_\Omega(s))) - I^s_{\tilde{\pi}}(h_\psi(s))) \\ &\qquad + I^s_{\tilde{\pi}}(h_\psi(s))) - I^s_\phi(h_\psi(s))\| \\ &\le L\|I^s_{\tilde{\pi}}(h_\Omega(s))) - I^s_{\tilde{\pi}}(h_\psi(s)))\| + L\|I^s_{\tilde{\pi}}(h_\psi(s))) - I^s_\phi(h_\psi(s))\| \\ &\le LC\|h_\Omega(s)) - h_\psi(s))\| + L\|I^s_{\tilde{\pi}}(\hat{s}') - I^s_\phi(\hat{s}')\| \ .\end{aligned} \tag{34}$$

□

Theorem D.5 (Error Bound of BCO). *Consider a deterministic environment whose transition function $\mathcal{T}(s, a)$ is deterministic and L-Lipschitz, and a parameterized policy $\pi_\psi(a|s)$ that learns from the label provided by a parameterized inverse dynamics model I_ϕ. Then for any state s, the distance between the desired state s'_E and reaching state s' sampled by a state-to-action policy as BCO (Torabi et al., 2018) is bounded by:*

$$\begin{aligned}\|s' - s'_E\| \le L &\left\| \pi_\psi(a|s) - \int_{s'^*} p_{\pi_E}(s'^*|s) I_\phi(a|s, s'^*) \,\mathrm{d}s'^*) \right\| \\ &+ L \left\| \int_{s'^*} p_{\pi_E}(s'^*|s) I_{\tilde{\pi}}(a|s, s'^*)) - p_{\pi_E}(s'^*|s) I_\phi(a|s, s'^*) \,\mathrm{d}s'^* \right\| \ ,\end{aligned} \tag{35}$$

where $\tilde{\pi} \in \omega_E$ is a policy instance of the expert hyper-policy ω_E such that $\mathcal{T}(s, \tilde{\pi}(s)) = s'_E$.

Proof.

$$\begin{aligned}\|s' - s'_E\| &= \|\mathcal{T}(s, \pi_\psi(s)) - \mathcal{T}(s, \tilde{\pi}(s))\| \\ &= \|\mathcal{T}^s(\pi_\psi(s)) - \mathcal{T}^s(\tilde{\pi}(s))\| \\ &\le L\|\tilde{\pi}(a|s) - \pi_\psi(a|s)\| \\ &= L \left\| \pi_\psi(a|s) - \int_{s'^*} p_{\pi_E}(s'^*|s) I_\phi(a|s, s'^*) \,\mathrm{d}s'^* \right. \\ &\qquad \left. + \int_{s'^*} p_{\pi_E}(s'^*|s) I_\phi(a|s, s'^*) \,\mathrm{d}s'^* - \int_{s'^*} p_{\pi_E}(s'^*|s) I_{\tilde{\pi}}(a|s, s'^*)) \,\mathrm{d}s'^* \right\| \\ &\le L \left\| \pi_\psi(a|s) - \int_{s'^*} p_{\pi_E}(s'^*|s) I_\phi(a|s, s'^*) \,\mathrm{d}s'^*) \right\| \\ &\qquad + L \left\| \int_{s'^*} p_{\pi_E}(s'^*|s) I_{\tilde{\pi}}(a|s, s'^*)) - p_{\pi_E}(s'^*|s) I_\phi(a|s, s'^*) \,\mathrm{d}s'^* \right\|\end{aligned} \tag{36}$$

□

An intuitive explanation for the bound is that BCO (Torabi et al., 2018) first seeks to recover a policy that shares the same hyper-policy with π_E via learning an inverse dynamics model and then try to conduct behavior cloning. Therefore the errors come from the reconstruction error of $\tilde{\pi}$ using I_ϕ (the second term) and the fitting error of behavior cloning (the first term).

By comparing Theorem D.4 and Theorem D.5, it is observed that for reaching each state, BCO requires a good inverse dynamics model over the state space to construct $\tilde{\pi}$ and then conduct imitation learning to $\tilde{\pi}$, while DePO only requires to learn a good inverse dynamics model on the predicted state and directly construct $\tilde{\pi}$ without the second behavior cloning step.