PUSHING THE LIMIT OF SAMPLE-EFFICIENT OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning (RL) has achieved significant progress in recent years. However, most existing offline RL methods require a large amount of training data to achieve reasonable performance and offer limited generalizability in out-of-distribution (OOD) regions due to conservative data-related regularizations. This seriously hinders the usability of offline RL in solving many real-world applications, where the available data are often limited. In this study, we introduce a highly sample-efficient offline RL algorithm that enables state-stitching in a compact latent space regulated by the fundamental time-reversal symmetry (T-symmetry) of dynamical systems. Specifically, we introduce a T-symmetry enforced inverse dynamics model (TS-IDM) to derive well-regulated latent state representations that greatly facilitate OOD generalization. A guide-policy can then be learned entirely in the latent space to output the next state that maximizes the reward, bypassing the conservative action-level behavior constraints as adopted in most offline RL methods. Finally, the optimized action can be easily extracted by using the guide-policy's output as the goal state in the learned TS-IDM. We call our method Offline RL via **T**-symmetry **E**nforced **L**atent **S**tate-Stitching (TELS). Our approach achieves amazing sample efficiency and OOD generalizability, significantly outperforming existing offline RL methods in a wide range of challenging small-sample tasks, even using as few as 1% of the data samples in D4RL datasets.

1 INTRODUCTION

Offline reinforcement learning (RL) has seen rapid progress in recent years. It bypasses the reliance on environment interactions as in online RL, directly utilizing pre-collected offline data for policy learning, thus being ideal for many real-world tasks that lack high-fidelity simulators or have environment interaction restrictions (Levine et al., 2020; Zhan et al., 2022; 2025). However, offline RL is also known to be prone to value overestimation, caused by extrapolation error when evaluating out-of-distribution (OOD) samples and amplified through the bootstrapped update procedure in RL (Kumar et al., 2019; Fujimoto et al., 2019).

In the past few years, quite a few offline RL methods have been proposed, which commonly adopt the pessimism principle using strategies such as adding explicit or implicit policy constraints to prevent the selection of OOD actions (Kumar et al., 2019; Fujimoto et al., 2019; Wu et al., 2019; Fujimoto and Gu, 2021), penalizing value function on unseen samples (Kumar et al., 2020; Xu et al., 2022b; Bai et al., 2021; Lyu et al., 2022), or adopting in-sample learning to implicit regularize policy optimization (Kostrikov et al., 2022; Xu et al., 2023; Mao et al., 2024b). What's in common with these methods is the use of some kind of action-level constraints to avoid OOD exploitation. Although this could stabilize offline value and policy learning, it inevitably leads to over-conservatism and crippled OOD generalization performance (Li et al., 2022; Cheng et al., 2023). Most of the existing offline RL methods only perform well when trained in sufficiently large amounts of offline data with reasonable state-action space coverage (e.g., 1 million samples for simple D4RL tasks (Fu et al., 2020)). This forms a stark contrast to the reality in most real-world scenarios, where the historical data are often limited and scaling up data collection can be rather costly (Zhan et al., 2022; 2025; Cheng et al., 2023). Although offline RL was initially proposed to address a broad spectrum of practical tasks, its successful real-world deployments remain limited to date.

Enhancing sample efficiency and OOD generalization capability is essential to making offline RL widely applicable to real-world applications. This is particularly important for small dataset settings, as most of the state-action space will become OOD regions. Some recent attempts have been made to improve the generalization performance of offline RL, which mainly follows three directions. The first direction builds upon the empirical observation that deep value functions interpolate well but struggle to extrapolate, thus allowing exploitation on interpolated OOD actions to promote generalization (Li et al., 2022). However, this method has a smoothness assumption on the offline dataset geometry and only applies to continuous action space. The second class of methods avoids the conservative action-level constraint and instead performs reward maximization on the state-space (Xu et al., 2022a; Park et al., 2024), which allows exploitation of OOD actions as long as the corresponding state transitions are reachable (also referred to as "*state-stitching*" (Xu et al., 2022a)). Although such methods offer some promising generalization capabilities, they still require the state-action space to have reasonable data coverage to enable valid state-stitching. The last and also most explored direction is to learn compact and robust latent representations to enhance sample efficiency (Laskin et al., 2020; Agarwal et al., 2021; Yang and Nachum, 2021; Weissenbacher et al., 2022; Cheng et al., 2023). Most of these methods only focus on extracting statistical-level information from the data, using techniques such as contrastive learning (Laskin et al., 2020; Agarwal et al., 2021; Yang and Nachum, 2021; Uehara et al., 2021). Due to the lack of in-depth modeling of the underlying dynamics patterns inside the sequential data, these methods still struggle to provide generalizable information beyond data distribution. Some recent methods (Weissenbacher et al., 2022; Cheng et al., 2023; Zhan et al., 2025) propose to extract fundamental symmetries of dynamics to facilitate policy learning, such as the time-reversal symmetry (T-symmetry) (Cheng et al., 2023; Zhan et al., 2025), i.e., the underlying physical laws should not change under the time-reversal transformation: $t \to -t$. If we can find and leverage such universally held symmetries in the dataset, then it is possible to maximally promote OOD generalization without being restrained by data distribution-related information. Although promising, these methods are built upon existing action-level constraint offline RL backbone algorithms like CQL (Kumar et al., 2020) or TD3+BC (Fujimoto and Gu, 2021), which still suffer from the over-conservatism issue.

In this paper, we find that enabling state-stitching in a coherent, fundamental symmetry-enforced latent space can actually lead to a surprisingly strong sample-efficient offline RL algorithm. We refer to our method as Offline RL via **T**-symmetry **E**nforced **L**atent **S**tate-Stitching (TELS). Specifically, we introduce a T-symmetry enforced inverse dynamics model (TS-IDM) to not only learn well-behaved latent representations that greatly alleviate the difficulty of OOD generalization, but can also facilitate effective action inference. Within the learned latent state space, we can optimize a T-symmetry regularized guide-policy to output the next latent state that maximizes the reward, bypassing the conservative action-level behavioral regularization as adopted by most existing offline RL algorithms. Lastly, the optimized action can be easily extracted by plugging the output of the guide-policy as the goal state in the learned TS-IDM. The resulting algorithm achieves incredible sample efficiency and OOD generalization capability, significantly outperforming existing offline RL algorithms in a wide range of challenging reduced-size D4RL benchmark datasets, even using as few as 1% of the original samples. Our method greatly pushes the performance limit of offline RL under low data regimes, offering a new opportunity to tackle many previously unsolvable real-world tasks.

2 PRELIMINARIES

Offline RL. We consider the standard Markov decision process (MDP) setting (Sutton and Barto, 2018), which is represented as a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, \mathcal{P}, \rho, \gamma\}$, and a dataset $\mathcal{D}$, which consists of trajectories $\tau = \{s_0, a_0, s_1, a_1, ..., s_T\}$. Here $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $r(s, a)$ is a scalar reward function, $\mathcal{P}(s'|s, a)$ and ρ denote the transition dynamics and initial state distribution respectively, and $\gamma \in (0, 1)$ is a discount factor. Our goal is to learn a policy $\pi(a|s)$ based on dataset $\mathcal{D}$ by maximizing the expected return in the MDP: $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \cdot r(s_t, a_t)]$.

Offline policy optimization in the state space. Instead of adopting conservative action-level constraints for offline policy learning, Policy-guided Offline RL (POR) (Xu et al., 2022a) proposes an alternative scheme, which decomposes the conventional reward-maximizing policy into a guide-policy and an execute policy. The guide-policy only works in the state space to find the optimal next state that maximizes the state-value function, and the execute-policy is learned as an inverse dynamics model (Xu et al., 2022a) or a goal-conditioned imitative policy (Park et al., 2024). Such methods only need to learn a state-only value function V using the IQL-style expectile regression (Kostrikov

et al., 2022) or the sparse value learning objective as discussed in Xu et al. (2023). We present the former as follows:

$$V = \arg\min_{V} \mathbb{E}_{(s,r,s')\sim\mathcal{D}} \left[L_2^\tau \left(r(s) + \gamma \bar{V}(s') - V(s) \right) \right] \tag{1}$$

where $L_2^\tau(x) = |\tau - \mathbb{1}(x < 0)|x^2$ is the asymmetric expectile regression loss and $\bar{V}$ denotes the target value network. Based on the learned state-value function, we can learn a guide-policy $\pi_g(s'|s)$ to serve as a prophet by telling which state the agent should (high reward) and can (logical generalization) go to, without being constrained to state-action transitions seen in the dataset. This can be achieved by leveraging the advantage weighted regression (AWR) objective (Neumann and Peters, 2008; Peng et al., 2019) to maximize the value while implicitly constraining π_g to $s \to s'$ transitions observed in the dataset (i.e., state-stitching):

$$\pi_g = \arg\max_{\pi_g} \mathbb{E}_{(s,r,s')\sim\mathcal{D}} \Big[\exp(\alpha \cdot A(s,s')) \log \pi_g(s' \mid s) \Big] \tag{2}$$

where the advantage $A(s,s') = r + \gamma V(s') - V(s)$ serves as the behavior cloning weight, and α is the temperature parameter to prioritize value maximization over state-wise imitation learning.

For the execute-policy π_e, POR employs a supervised learning framework and trains π_e by maximizing the likelihood of the actions given the states and next states: $\max_{\pi_e} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[\log \pi_e(a \mid s, s')]$. During evaluation phase, given the current state s, we can sample the optimized next state s' from $\pi_g(s'|s)$, and can get final action simply as $a^* = \pi_e(a \mid s, \pi_g(s'|s))$. **Time-reversal symmetry for generalizable offline RL.** Recently, leveraging fundamental, universally held symmetries of dynamics such as T-symmetry discovered in classical and quantum mechanics (Lamb and Roberts, 1998; Huh et al., 2020) has been shown to be a promising approach to enhance the generalization of offline RL (Cheng et al., 2023; Zhan et al., 2025). Specifically, if we model the system dynamics with measurements $\mathbf{x}$ as a set of non-linear first-order differential equations (ODEs) expressed as $\frac{d\mathbf{x}}{dt} = F(\mathbf{x})$, a dynamical system is said to exhibit *time-reversal symmetry* if there is an invertible transformation Γ that reverses the direction of time: i.e., $d\Gamma(\mathbf{x})/dt = -F(\Gamma(\mathbf{x}))$. For the discrete-time MDP setting, the T-symmetry can be extended as learning a pair of ODE forward $F(s,a) \to \dot{s}$ and reverse dynamics $G(s',a) \to -\dot{s}$, and require them to satisfy $F(s,a) = -G(s',a)$ (Cheng et al., 2023), where the time-derivative of state $\dot{s} = \frac{ds}{dt}$ is approximated as $s' - s$.

Based on this intuition, TSRL (Cheng et al., 2023) constructed an encoder-decoder structured *T-symmetry enforced dynamics model* (TDM) for representation learning, which embeds a pair of latent ODE forward and reverse dynamics to enforce T-symmetry. TSRL achieves impressive performance under small-sample settings and its variant has been successfully used for real-world industrial control (Zhan et al., 2025), but it still has some limitations. First, TSRL only uses the learned encoder from TDM to derive the latent representations, without fully exploiting the rich dynamics-related information in TDM for downstream policy learning. Second, it needs both the state and action as inputs to encode latent representations, forcing TSRL to adopt a Q-function maximization method (i.e., TD3+BC (Fujimoto and Gu, 2021)) for policy optimization, which inevitably requires adding conservative action-level behavior cloning constraints to stabilize training. Moreover, involving action as an input for representation learning is also prone to capturing the biased behaviors in the data-generating policy, which could impede learning fundamental, distribution-agnostic dynamics patterns in data. Please refer to Appendix A for a detailed comparison between TSRL and our method.

3 Methodology

We now present our proposed method, TELS, which comprises a T-symmetry enforced inverse dynamics model (TS-IDM) integrated with an effective offline policy optimization procedure operated in latent state space. TS-IDM overcomes multiple drawbacks of TDM in TSRL (Cheng et al., 2023), which not only extracts the fundamental, T-symmetry preserving representations from the limited data and facilitates OOD generalization, but can also be seamlessly used as an execute-policy for optimal action extraction. The overall framework of TELS is illustrated in Figure 1.

3.1 T-symmetry Enforced Inverse Dynamic Model

If we look at the input and output of our proposed TS-IDM, it functions similarly to a typical inverse dynamics model that takes current and next state (s, s') as input and outputs the predicted

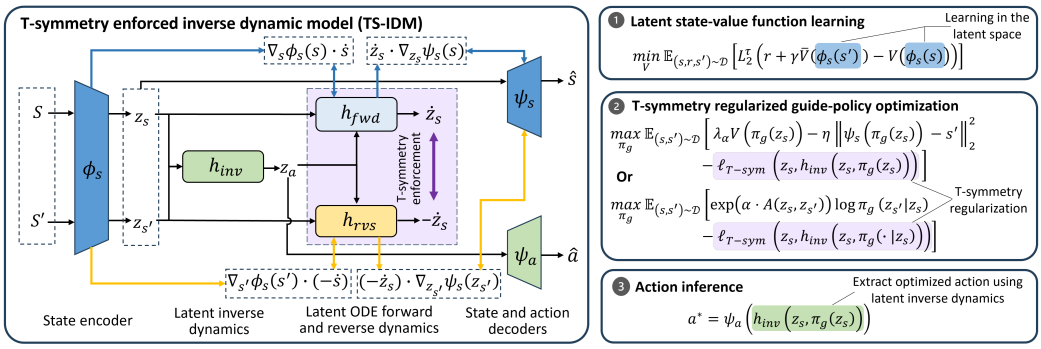


Figure 1: Overview of our proposed **T**-symmetry **E**nforced **L**atent **S**tate-Stitching (TELS) framework.

action a. However, TS-IDM's architecture is special in several aspects. In its interior, it comprises a state encoder $\phi_s(s) = z_s$, a latent inverse dynamics model $h_{inv}(z_s, z_{s'}) = z_a$, a pair of T-symmetry enforced latent ODE forward and reverse dynamics models $h_{fwd}(z_s, z_a) = \dot{z}_s$ and $h_{rvs}(z_{s'}, z_a) = -\dot{z}_s$, an action decoder $\psi_a(z_a) = \hat{a}$, and an extra state decoder $\psi_s(z_s) = \hat{s}$. In the following, we describe their detailed design logic and learning objectives.

Encoding and decoding. As previously discussed, constructing an informative and well-structured latent space is critical for sample-efficient offline policy optimization. To this end, we introduce a state encoder $\phi_s(s) = z_s$ to map the states s into their corresponding latent representations z_s, and also a state decoder $\psi_s(z_s) = s$ to reconstruct the original states from their latent embeddings, ensuring that the learned latent representations remain faithful to the original state space and avoid excessive distortion.

We then construct a latent inverse dynamics model $h_{inv}(z_s, z_{s'}) = z_a$, which infers the latent action z_a from the latent state transitions $(z_s, z_{s'})$. By inferring actions from state transitions, the learned latent space implicitly encodes the underlying dynamics of the environment. Moreover, the inverse dynamic model h_{inv} can be integrated with a pair of latent ODE dynamic models to derive the T-symmetry property of the system, which we will introduce in more detail shortly. Finally, to ensure that the inferred actions are both meaningful and interpretable, we employ an action decoder $\psi_a(z_a) = \hat{a}$ to map the latent action back to its original action space. We can thus formulate the reconstruction loss for the states and actions as follows:

$$\ell_{\text{rec}}(s, a, s') = \underbrace{\|\psi_s(\phi_s(s)) - s\|_2^2}_{\text{reconstruction loss of states}} + \underbrace{\|\psi_a(h_{inv}(z_s, z_{s'})) - a\|_2^2}_{\text{reconstruction loss of actions}} \tag{3}$$

Latent ODE forward and reverse dynamics models. Drawing inspiration from previous research that integrates physics-informed insights into dynamical systems modeling (Brunton et al., 2016; Champion et al., 2019; Huh et al., 2020; Cheng et al., 2023), we embed a pair of latent ODE forward and reverse dynamics $h_{fwd}(z_s, z_a) = \dot{z}_s$ and $h_{rvs}(z_{s'}, z_a) = -\dot{z}_s$ to separately capture the forward and reverse time evolution in the latent states. We are interested in modeling ODE systems because it encourages learning parsimonious models helpful to uncover fundamental properties from the data that can maximally promote generalization (Brunton et al., 2016; Champion et al., 2019). Note that based on the chain rule, we can derive the supervision signal for the latent dynamics models with $\dot{z}_s = \frac{dz}{dt} = \frac{dz_s}{ds} \cdot \frac{ds}{dt} = \nabla_s z_s \cdot \dot{s} = \nabla_s \phi_s(s) \cdot \dot{s}$ to enforce the ODE property. Therefore, we introduce the following training losses for h_{fwd} and h_{rvs}:

$$\begin{aligned}\ell_{\text{dyn}}(s, s') = & \underbrace{\|(\nabla_s z_s)\dot{s} - \dot{z}_s\|_2^2}_{\text{latent ODE forward dynamics}} + \underbrace{\|(\nabla_{s'} z_{s'})(-\dot{s}) - (-\dot{z}_s)\|_2^2}_{\text{latent ODE reverse dynamics}} \\ = & \|\nabla_s \phi_s(s)\dot{s} - h_{fwd}(z_s, z_a)\|_2^2 + \|\nabla_{s'} \phi_s(s')(-\dot{s}) - h_{rvs}(z_{s'}, z_a)\|_2^2,\end{aligned} \tag{4}$$

where the latent action z_a is obtained from the latent inverse dynamics model $h_{inv}(z_s, z_{s'})$.

ODE property enforcement on state decoder. Note that in $\ell_{\text{dyn}}(s, s')$, we actually implicitly enforced the ODE property on the state encoder ϕ_s, the same should also apply to the state decoder ψ_s to ensure compatibility with the T-symmetry formalism, i.e. the time-derivative of the state encoder $\frac{d\phi_s(s)}{dt}$ and decoder $\frac{d\psi_s(z_s)}{dt}$ should behave in the same way as $\dot{z}_s$ and $\dot{s}$. Similar to the

previous treatment on the state encoder, as $\dot{s} = \frac{d\psi_s(z_s)}{dt} = \frac{d\psi_s(z_s)}{dz_s} \cdot \frac{dz_s}{dt} = \nabla_{z_s}\psi_s(z_s) \cdot \dot{z}_s$, we can use the following objective to enforce the ODE property for the state decoder ψ_s:

$$\begin{aligned}\ell_{\text{ode}}(s, s') =& \underbrace{\|\nabla_{z_s}\psi_s(z_s) \cdot \dot{z}_s - \dot{s}\|_2^2}_{\text{enforce ODE of } \psi_s \text{ on } h_{fwd}} + \underbrace{\|\nabla_{z_{s'}}\psi_s(z_{s'}) \cdot (-\dot{z}_s) - (-\dot{s})\|_2^2}_{\text{enforce ODE of } \psi_s \text{ on } h_{rvs}} \\ =& \|\nabla_{z_s}\psi_s(z_s) \cdot h_{fwd}(z_s, z_a) - \dot{s}\|_2^2 + \|\nabla_{z_{s'}}\psi_s(z_{s'}) \cdot h_{rvs}(z_{s'}, z_a) + \dot{s}\|_2^2 \end{aligned} \tag{5}$$

Again, the latent action z_a is obtained from $h_{inv}(z_s, z_{s'})$. Notably, the ODE property enforcement in Eq. (5) is not considered in the T-symmetry enforced dynamics model (TDM) proposed by TSRL (Cheng et al., 2023). In other words, TDM only enforces the ODE properties for encoders but not on decoders. This could lead to inconsistency between the learned dynamics and the underlying ODE structure, leading to inaccurate or misaligned ODE representations.

T-symmetry enforcement. To further regularize the learned latent representations, we incorporate the extended version of T-symmetry (Cheng et al., 2023) by requiring $h_{fwd}(z_s, z_a) = -h_{rvs}(z_{s'}, z_a)$, which corresponds to the following T-symmetry consistency loss:

$$\ell_{\text{T-sym}}(z_s, z_a) = \|h_{fwd}(z_s, z_a) + h_{rvs}(z_s + h_{fwd}(z_s, z_a), z_a)\|_2^2 \tag{6}$$

where we use the fact that $z_{s'} = z_s + \dot{z}_s = z_s + h_{fwd}(z_s, z_a)$ and $h_{rvs}(z_s + h_{fwd}(z_s, z_a), z_a) = -\dot{z}_s = -h_{fwd}(z_s, z_a)$ to further couple the learning process of h_{fwd} and h_{rvs}. Moreover, given a latent state-action pair (z_s, z_a), the above T-symmetry consistency loss can also serve as an evaluation metric to assess their agreement with the learned TS-IDM. A large T-symmetry loss indicates that the latent state-action representation (z_s, z_a) induced by some (s, s') may not satisfy the fundamental dynamics pattern, therefore more likely to be a problematic or non-generalizable sample.

Overall learning objective. Finally, the complete training loss function of TS-IDM is as follows:

$$\mathcal{L}_{\text{TS-IDM}} = \sum_{(s,a,s') \in \mathcal{D}} \Big[\ell_{\text{rec}} + \beta \cdot (\ell_{\text{dyn}} + \ell_{\text{ode}} + \ell_{\text{T-sym}})\Big](s, a, s') \tag{7}$$

where β is a hyperparameter that balances extracting fundamental dynamics properties and ensuring the interpretability of the learned representation. As we can observe from the final learning objective, TS-IDM introduces a series of coupling designs among state encoder ϕ_s, decoder ψ_s, latent inverse dynamics h_{inv}, latent ODE forward and reverse dynamics h_{fwd} and h_{rvs}, forming a strongly consistent, T-symmetry preserving ODE system to capture the fundamental dynamics properties in the offline dataset.

3.2 Latent Space Offline Policy Optimization

Once we have a learned TS-IDM, we can extract three highly useful components from it to facilitate sample-efficient downstream offline policy optimization, including 1) a robust state encoder $\phi(s)$ that provides well-behaved and generalizable latent space ideal for state-stitching; 2) T-symmetry consistency as an additional regularizer to prevent erroneous generalization when learning a guide-policy in the latent state space; and 3) the TS-IDM itself can serve as an execute-policy as in POR (Xu et al., 2022a) to extract optimized action given the learned guide-policy.

Latent state-value functions learning. Based on the state encoder $\phi_s(s)$ from the learned TS-IDM, we can convert the entire offline policy optimization process into the latent state space, which enjoys both the stable learning process and generalizability due to more compact and well-behaved representations. Specifically, we can use a similar IQL-style expectile regression loss as in Eq. (1) to learn a state-value function $V(z_s)$, but in the latent state space:

$$\min_V \mathbb{E}_{(s,r,s') \sim \mathcal{D}} \Big[L_2^\tau \left(r + \gamma \bar{V}\left(\phi_s(s')\right) - V\left(\phi_s(s)\right)\right)\Big] \tag{8}$$

T-symmetry regularized guide-policy optimization. A major benefit of learning within the T-symmetry preserving latent space is that, as T-symmetry captures what is essential and invariant about the dynamical system, thus it can generalize and provide reliable information even for OOD samples beyond the offline dataset. This naturally favors learning a reward-maximizing guide-policy π_g in the latent space, which can enjoy more effective state-stitching. Moreover, different from POR (Xu et al., 2022a), by leveraging the T-symmetry consistency term $\ell_{\text{T-sym}}(\cdot)$ in Eq. (6) as an additional

Algorithm 1 Offline RL via T-symmetry Enforced Latent State-Stitching (TELS).

Require: Offline dataset $\mathcal{D}$.
1: *// TS-IDM learning*
2: Learning the state encoder ϕ_s, state decoder ψ_s, action decoder ψ_a, latent inverse dynamics h_{inv}, latent forward and reverse dynamics h_{fwd} and h_{rvs} using the TS-IDM learning objective Eq. (7).
3: Initialize V_θ, $V_{\theta'}$, π_σ
4: *// Policy training*
5: **for** $t = 1, \cdots, M$ training steps **do**
6: Sample transitions $(s, r, s') \sim \mathcal{D}$ and compute their representations $(z_s, z_{s'})$ using the state encoder ϕ_s.
7: Use $(z_s, r, z_{s'})$ to update the latent state-value function V using Eq.(8).
8: Use $(z_s, z_{s'})$ to update the latent guide-policy π_g using Eq. (9) or (10).
9: **end for**
10: *// Evaluation*
11: Get initial state s from environment
12: **while** not done **do**
13: Get optimized next state $z^*_{s'}$ using guide-policy π_g.
14: Extract action a using Eq. (11).
15: **end while**

regularizer, we can prevent π_g from outputting problematic and non-generalizable latent next state, thereby further enhancing logical state-wise OOD generalization.

In TELS, we provide two instantiations for guide-policy optimization, depending on the choice of using deterministic policy $\pi_g(z_s)$ or stochastic policy $\pi_g(z_{s'}|z_s)$:

- Deterministic policy:

$$\max_{\pi_g} \mathbb{E}_{(s,s')\sim\mathcal{D}}\Big[\lambda_\alpha V(\pi_g(z_s)) - \eta\|\psi_s(\pi_g(z_s)) - s'\|_2^2 - \ell_{\text{T-sym}}\left(z_s, h_{ivs}\left(z_s, \pi_g(z_s)\right)\right)\Big] \tag{9}$$

- Stochastic policy:

$$\max_{\pi_g} \mathbb{E}_{(s,s')\sim\mathcal{D}}\Big[\exp(\alpha \cdot A(z_s, z_{s'}))\log \pi_g(z_{s'} \mid z_s) - \ell_{\text{T-sym}}(z_s, h_{ivs}(z_s, \pi_g(\cdot|z_s))\Big] \tag{10}$$

where $z_s = \phi_s(s)$, $z_{s'} = \phi_s(s')$, and $A(z_s, z_{s'}) = r + \gamma V\left(z_{s'}\right) - V\left(z_s\right)$.

For the deterministic policy $\pi_g(z_s)$, we extract the guide-policy by directly maximizing the latent state-value function V weighted by a normalization term λ_α, together with two additional regularization terms. The first regularizes the next state decoded from the guide-policy using state decoder ψ_s should not deviate too much from the ground truth next state s' in the dataset. The last term regularizes guide-policy induced latent state-action pair (i.e., $(z_s, z_a) = (z_s, h_{inv}(z_s, \pi_g(z_s)))$) to comply with the T-symmetry consistency specified in the learned TS-IDM.

For the stochastic guide-policy $\pi_g(z_{s'}|z_s)$, we adopt the AWR-style (Neumann and Peters, 2008; Peng et al., 2019) policy optimization objective as in Eq. (2), while also incorporating the T-symmetry consistency regularization similar to the deterministic policy version. In our experiments, we find that the deterministic version objective Eq. (9) works well for the MuJoCo locomotion tasks, while the stochastic version Eq. (10) works better for more complex D4RL Antmaze tasks (Fu et al., 2020), potentially due to more stochastic nature of the task environment.

Action inference. After learning the guide-policy π_g, we can further use it to generate the optimized action for control. To do this, we can simply use the optimized latent next state $z^*_{s'}$ obtained from guide-policy $\pi_g(z_s)$ or $\pi_g(\cdot|z_s)$ as the goal state, and plug it into the learned latent inverse dynamics model $h_{inv}(z_s, z_{s'})$ in TS-IDM to replace $z_{s'}$. The final action can be extracted by decoding the resulting latent action from h_{inv} using the action decoder ψ_a :

$$a^* = \psi_a\left(h_{inv}\left(z_s, \pi_g(z_s)\right)\right) \tag{11}$$

Note that there is no training process needed for this stage. We fully utilize the learned TS-IDM to serve our purpose. We present the complete training and inference procedure of TELS in Algorithm 1.

Table 1: Average normalized scores on reduced-size D4RL datasets. The scores are taken over the final 10 evaluations with 5 seeds.

Task	Size (ratio)	BC	TD3+BC	CQL	IQL	DOGE	IDQL	POR	TSRL	TELS
Hopper-m	10k (1%)	29.7±11.7	40.1±18.6	43.1±24.6	46.7±6.5	44.2 ± 10.2	44.2±12.1	46.4 ± 1.7	62.0±3.7	**77.3 ± 10.7**
Hopper-mr	10k (2.5%)	12.1±5.3	7.3±6.1	2.3±1.9	13.4±3.1	17.9 ± 4.5	21.7±7.0	17.4 ± 6.2	21.8±8.2	**43.2 ± 3.5**
Hopper-me	10k (0.5%)	27.8±10.7	17.8±7.9	29.9±4.5	34.3±8.7	50.5 ± 25.2	43.2±4.4	37.9 ± 6.1	50.9±8.6	**100.9 ± 6.8**
Halfcheetah-m	10k (1%)	26.4±7.3	16.4±10.2	35.8±3.8	29.9±0.12	36.2 ± 3.4	36.4±1.5	33.3±3.2	38.4±3.1	**40.8 ± 0.6**
Halfcheetah-mr	10k (5%)	14.3±7.8	17.9±9.5	8.1±9.4	22.7±6.4	23.4 ± 3.6	26.7±1.0	27.5±3.6	28.1±3.5	**33.2 ± 1.0**
Halfcheetah-me	10k (0.5%)	19.1±9.4	15.4±10.7	26.5±10.8	10.5±8.8	26.7 ± 6.6	38.8±1.9	34.7±2.6	39.9±21.1	**40.7 ± 1.2**
Walker2d-m	10k (1%)	15.8±14.1	7.4±13.1	18.8±18.8	22.5±3.8	45.1 ± 10.2	31.7±14.2	22.2±3.6	49.7±10.6	**62.4 ± 5.3**
Walker2d-mr	10k (3.3%)	1.4±1.9	5.7±5.8	8.5±2.19	10.7±11.9	13.5 ± 8.4	12.2±10.5	14.8±4.2	26.0±11.3	**54.8 ± 6.0**
Walker2d-me	10k (0.5%)	21.7±8.2	7.9±9.1	19.1±14.4	26.5±8.6	35.3 ± 11.6	21.8±14.5	20.1±8.6	46.4±17.4	**87.4 ± 13.3**
Antmaze-u	10k (1%)	44.7 ± 42.1	0.7 ± 1.2	0.1 ± 0.0	65.1 ± 19.4	56.3 ± 24.4	67.5 ±12.4	6.1 ± 7.3	76.1 ± 15.6	**88.7 ± 7.7**
Antmaze-u-d	10k (1%)	24.1 ± 22.2	16.27 ± 16.4	0.5 ± 0.1	34.6 ± 18.5	41.7 ± 18.9	55.1 ± 36.8	42.1 ± 14.2	52.2 ± 22.1	**60.9 ± 16.9**
Antmaze-m-d	100k (10%)	0.0	0.0	0.0	4.8 ± 5.9	0.0	9.0 ±3.4	0.0	0.0	**47.2 ± 17.3**
Antmaze-m-p	100k (10%)	0.0	0.0	0.0	12.5 ± 5.4	0.0	9.4 ± 14.7	0.0	0.0	**62.9 ± 17.8**
Antmaze-l-d	100k (10%)	0.0	0.0	0.0	3.6 ± 4.1	0.0	16.1 ± 8.4	0.0	0.0	**39.8 ± 14.1**
Antmaze-l-p	100k (10%)	0.0	0.0	0.0	3.5 ± 4.1	0.0	9.7 ±8.5	0.0	0.0	**47.3 ± 13.1**

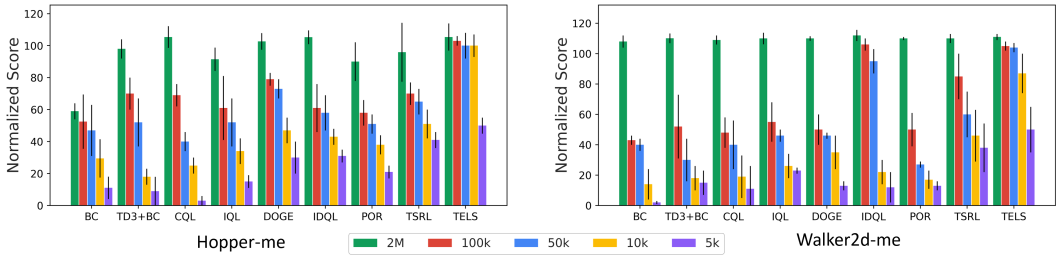


Figure 2: Performance of TELS against baselines under different data sizes

4 Experiments

In this section, we present the evaluation results of TELS on the D4RL MuJoCo-v2 and Antmaze-v1 tasks (Fu et al., 2020) against behavior cloning (BC), and existing offline RL methods: TD3+BC (Fujimoto and Gu, 2021), CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2021), DOGE (Li et al., 2022), POR (Xu et al., 2022a), diffusion-based method IDQL (Hansen-Estruch et al., 2023) and TSRL (Cheng et al., 2023), which is the current SOTA method in small-sample settings. We also conduct additional experiments to further evaluate the OOD generalizability of TELS on a challenging task, as well as the effectiveness of the representations learned with TS-IDM in improving small-sample performance. Performance comparison of the full datasets and implementation details can be found in Appendix B and C.

4.1 Performance Comparison on Small-Sample Setting

In Table 1, we evaluate TELS against baseline methods on reduced-size D4RL datasets (5k∼100k samples, about 0.5∼10% of their original sizes)[1]. These small-sample tasks are particularly challenging for offline RL methods, as the data only sparsely cover the state-action space and require strong OOD generalization capability for algorithms to achieve reasonable performance.

As shown in Table 1, most baselines fail to learn reasonable policies under small datasets, especially in the most challenging 100k Antmaze-medium/large datasets. For example, conventional offline RL methods like TD3+BC and CQL perform poorly on small datasets, primarily due to their over-conservative data-related policy constraints. Baselines that have generalization promotion designs, such as DOGE and TSRL, perform slightly better but still fail miserably in the challenging Antmaze-m/l tasks, as they still adopt conservative action-level constraints to stabilize policy learning. Recent diffusion-based methods like IDQL, although perform well on large datasets, struggle to learn when given limited data. By contrast, TELS dominates the chart and outperforms all other baselines in all tasks, sometimes by a large margin. This is mainly attributed to the leverage of fundamental,

[1]We use the same reduced-size MuJoCo datasets from the TSRL paper, and create 100k randomly sub-sampled Antmaze datasets by ourselves for additional experiments.

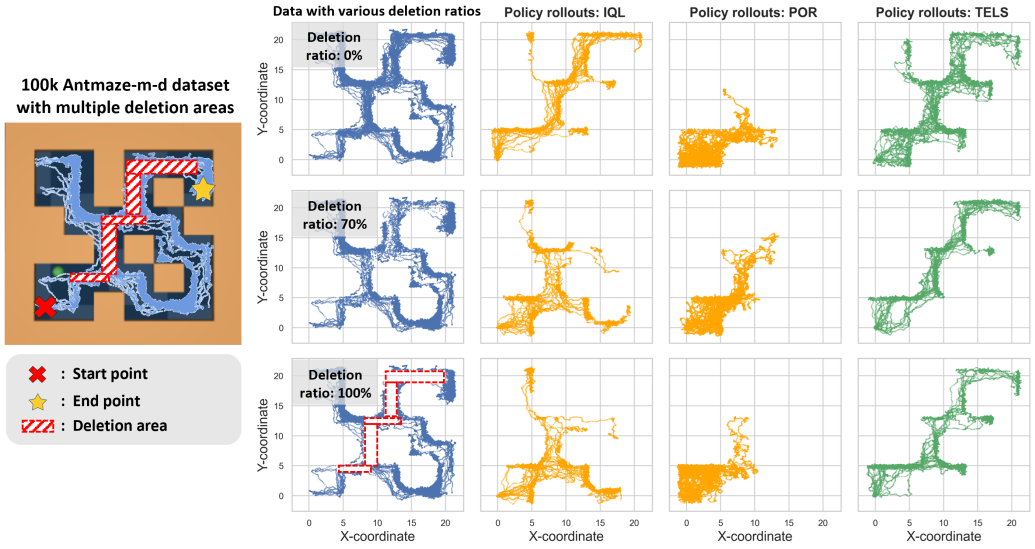


Figure 3: **Left:** Illustration of the 100k Antmaze-m-d task with multiple deletion areas, where the red cross denotes the start point, the yellow star denotes the goal locations, and the red shaded areas denote the data deletion regions. **Right:** Visualization of the training dataset and policy rollout trajectories generated by trained policies from various algorithms under varying deletion ratios.

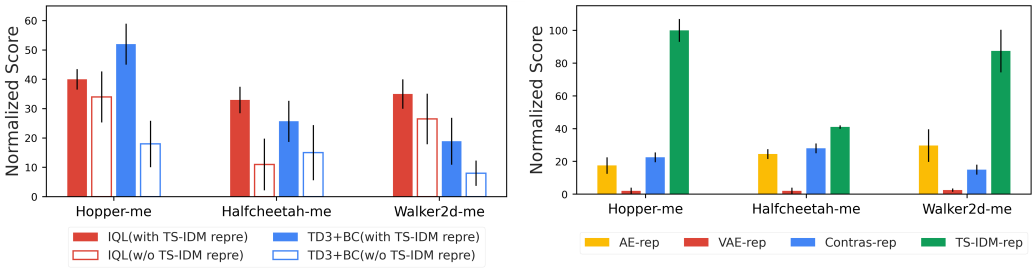


Figure 4: **Left:** The performance of IQL and TD3+BC on 10k datasets with or without using the representation from TS-IDM. **Right:** Performance of TELS with different representation models on 10k datasets, error bars indicate the normalized scores over 5 random seeds.

data distribution-agnostic T-symmetry property for policy learning, which greatly improves the OOD generalization performance. This is evident when observing the huge performance difference between POR and TELS, as the former shares a similar policy optimization procedure with TELS but does not use the T-symmetry enforced representation and policy regularization.

We also evaluate the performance of the algorithms across different dataset sizes in Figure 2. The results show that TELS can robustly maintain reasonable performance even with only 5k samples, surpassing all the other methods, while most baseline methods suffer from significant performance drop when training samples are decreased.

4.2 Investigating the OOD Generalizability of TELS

To further demonstrate the OOD generalizability of TELS, we construct a more challenging task based on the reduced-size 100k Antmaze-m-d dataset, as illustrated in Figure 3. Specifically, we randomly remove samples within 5 critical regions along the critical paths from the start to the goal locations. This task requires extremely strong OOD generalization capability to solve, as the vital information for the optimal trajectory is extremely scarce or completely OOD. We train IQL, POR, and TELS on the remaining data and plot their policy rollouts over 20 episodes for performance evaluation and behavior analyses (due to page limit, we also include results for IDQL, DOGE, TSRL in Appendix B.2).

Table 2: Ablation results on the design components of TS-IDM.

	$\phi/\psi + h_{inv}$	$+ h_{fwd}, h_{rvs} \uparrow$	$+ \ell_{\text{ode}} \uparrow$	$+ \ell_{\text{T-sym}} \uparrow$
Hopper-me	17.2 ± 7.0	35.5 ± 7.3	61.4 ± 23.7	$\mathbf{100.9 \pm 6.8}$
Halfcheetah-me	29.7 ± 3.6	31.3 ± 1.1	31.2 ± 1.2	$\mathbf{40.7 \pm 1.2}$
Walker2d-me	24.5 ± 10.1	33.6 ± 9.2	58.5 ± 18.1	$\mathbf{87.4 \pm 13.3}$

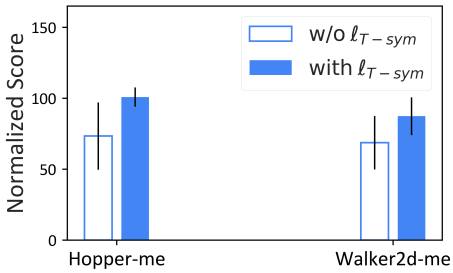


Figure 5: Impact of $\ell_{\text{T-sym}}$ on policy optimization

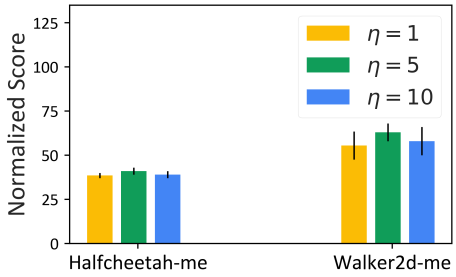


Figure 6: Performance of TELS with different η

As shown in Figure 3, IQL can only achieve some success when the deletion ratio is 0%, and POR fails to reach the goal in all cases. By contrast, TELS consistently learns optimal policy even with 70% and 100% deletion rates. It can effectively utilize the limited information provided in the sparse remaining data samples at the boundaries of the deletion areas for policy learning. These highlight the extraordinary OOD generalization capability of TELS in extremely challenging low-data regimes.

4.3 Effectiveness of the Learned Representations

To verify the effectiveness of the learned latent representation in TS-IDM, we use TS-IDM's state encoder $\phi_s(s)$ as the representation learning module on top of two conventional offline RL methods: IQL and TD3+BC. Figure 4 (left) reveals significant performance improvements and variance reduction when IQL and TD3+BC are trained within the latent state space induced by $\phi_s(s)$, suggesting that TS-IDM indeed learns compact and generalizable representations that benefit policy learning.

To further evaluate the quality of TS-IDM's representations, in Figure 4 (right), we replace TS-IDM in TELS with other representation learning methods, including autoencoder ("AE-rep"), variational autoencoder ("VAE-rep") (Kingma and Welling, 2014), and contrastive learning method SimCLR ("Contras-rep") (Chen et al., 2020). Among these, VAE performs the worst, as it is overly impacted by the simplistic Gaussian prior distribution; policies with AE and contrastive representations obtain some scores but still perform poorly, due to the lack of system dynamics-related information. In contrast, TS-IDM provides an information-rich and well-behaved latent space, significantly enhancing policy performance for small dataset settings.

4.4 Ablation Study

Ablations on the design components of TS-IDM. To examine the impact of each component in TS-IDM, we compare TELS with various variants of TS-IDM, starting with a vanilla latent inverse dynamics model with encoder and decoders, denoted as " $\phi/\psi + h_{inv}$", gradually adding latent forward and reverse dynamics models "h_{fwd}, h_{rvs}", ODE property enforcement "ℓ_{ode}", and eventually the T-symmetry consistency loss "$\ell_{\text{T-sym}}$", resulting in the full TS-IDM. The results on 10k datasets are presented in Table 2.

We observe that the naïve autoencoder-based inverse dynamics model fails to provide reasonable representations. Incorporating latent dynamics models is helpful because some system dynamics-related information is introduced, but the performance gain remains insufficient. Enforcing ODE properties on decoders significantly enhances the reliability of the learned representations, particularly in tasks like Walker2d-me. Lastly, enforcing T-symmetry consistency proves to be the strongest performance improvement factor, which greatly enhances the quality of the learned representations for downstream policy learning.

Ablations on regularizer terms in policy optimization. We also conduct ablation experiments in Figure 5 to validate the effectiveness of the T-symmetry consistency regularizer term $\ell_{\text{T-sym}}$ during the guide-policy optimization process of TELS. The results demonstrate that incorporating this term can effectively enhance performance while reducing variance, highlighting the importance of utilizing T-symmetry consistency regularization to promote OOD generalization and learning stability. Additionally, in Figure 6, we evaluate the hyperparameter robustness of TELS by training it with various values of $\eta = \{1, 5, 10\}$ to examine its sensitivity to the state-level behavioral constraint in Eq. (9). The results show that TELS is robust to different η values, consistently delivering reliable performance across various policy constraint settings.

5 Related Work

Offline RL faces unique challenges in mitigating the risk of OOD exploitation. Evaluating value functions in OOD regions often results in inaccurate estimates, which can lead to severe value overestimation and misguiding policy learning. To mitigate this, most offline RL methods leverage data-related constraints to stabilize the learning process. These include explicit behavior constraint techniques that penalize action divergence (Wu et al., 2019; Kumar et al., 2019; Fujimoto and Gu, 2021), value regularization schemes to discourage policies from selecting OOD actions via modifying Bellman update (Kumar et al., 2020; Xu et al., 2022b; Bai et al., 2021; Lyu et al., 2022) or introducing uncertainty penalities (Wu et al., 2021; An et al., 2021; Bai et al., 2021), and in-sample learning methods (Brandfonbrener et al., 2021; Kostrikov et al., 2022; Xu et al., 2023; Mao et al., 2024b), which stabilize training by only using in-sample data for value and policy learning, avoiding OOD samples. While these methods perform reasonably well on datasets with sufficient state-action coverage, they often struggle in small-sample settings where exploiting OOD generalization is vital for achieving good performance. Recently, leveraging expressive model architectures such as Transformers and diffusion models (Chen et al., 2021; Wang et al., 2022; Ajay et al., 2022; Janner et al., 2022; Hansen-Estruch et al., 2023; Mao et al., 2024a) have gained popularity in offline RL, due to their strong capability to fit complex data distributions. However, these models are overly heavy and require extensive amounts of data to learn, making them impractical for the small-sample setting.

6 Conclusion

In this paper, we propose a highly sample-efficient offline RL algorithm that learns optimized policy within the latent space regulated by the fundamental T-symmetry in the dynamical systems. Specifically, we develop a T-symmetry enforced inverse dynamics model (TS-IDM) to construct a well-behaved and generalizable latent space, effectively mitigating the challenges of OOD generalization. By learning a T-symmetry regularized guide-policy within this latent space, we can obtain the reward-maximizing next state to serve as the goal state input in the learned TS-IDM for optimal action extraction. Through extensive experiments, we show that TELS achieves surprisingly strong OOD generalization capability and SOTA small-sample performance. Moreover, we show empirically that TS-IDM can also function as a representation model to provide informative representations and enhance the performance of existing methods under the small-sample setting. One potential limitation of TELS is that strong ODE and T-symmetry property regularizations, although helpful for extracting fundamental features, sometimes could limit the model's expressive power (see Appendix B.3). Future studies can explore improved designs to balance fundamental pattern extraction and model expressivity perfectly.

References

R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021.

A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

G. An, S. Moon, J.-H. Kim, and H. O. Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

C. Bai, L. Wang, Z. Yang, Z.-H. Deng, A. Garg, P. Liu, and Z. Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

D. Brandfonbrener, W. Whitney, R. Ranganath, and J. Bruna. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021.

S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

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.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

P. Cheng, X. Zhan, W. Zhang, Y. Lin, H. Wang, L. Jiang, et al. Look beneath the surface: Exploiting fundamental symmetry for sample-efficient offline rl. *Advances in Neural Information Processing Systems*, 36, 2023.

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

I. Huh, E. Yang, S. J. Hwang, and J. Shin. Time-reversal symmetric ode network. *Advances in Neural Information Processing Systems*, 33:19016–19027, 2020.

M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR 2014 : International Conference on Learning Representations (ICLR) 2014*, 2014.

I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

J. S. Lamb and J. A. Roberts. Time-reversal symmetry in dynamical systems: a survey. *Physica D: Nonlinear Phenomena*, 112(1-2):1–39, 1998.

M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pages 5639–5650. PMLR, 2020.

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

J. Li, X. Zhan, H. Xu, X. Zhu, J. Liu, and Y.-Q. Zhang. When data geometry meets deep function: Generalizing offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022.

J. Lyu, X. Ma, X. Li, and Z. Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.

L. Mao, H. Xu, X. Zhan, W. Zhang, and A. Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

L. Mao, H. Xu, W. Zhang, and X. Zhan. Odice: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. In *The Twelfth International Conference on Learning Representations*, 2024b.

G. Neumann and J. Peters. Fitted q-iteration by advantage weighted regression. *Advances in neural information processing systems*, 21, 2008.

S. Park, D. Ghosh, B. Eysenbach, and S. Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024.

X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

M. Uehara, X. Zhang, and W. Sun. Representation learning for online and offline rl in low-rank mdps. *arXiv preprint arXiv:2110.04652*, 2021.

Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

M. Weissenbacher, S. Sinha, A. Garg, and K. Yoshinobu. Koopman q-learning: Offline reinforcement learning via symmetries of dynamics. In *International Conference on Machine Learning*, pages 23645–23667. PMLR, 2022.

Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Y. Wu, S. Zhai, N. Srivastava, J. M. Susskind, J. Zhang, R. Salakhutdinov, and H. Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *International Conference on Machine Learning*, pages 11319–11328. PMLR, 2021.

H. Xu, J. Li, J. Li, and X. Zhan. A policy-guided imitation approach for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022a.

H. Xu, X. Zhan, and X. Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022b.

H. Xu, L. Jiang, J. Li, Z. Yang, Z. Wang, V. W. K. Chan, and X. Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *The Eleventh International Conference on Learning Representations*, 2023.

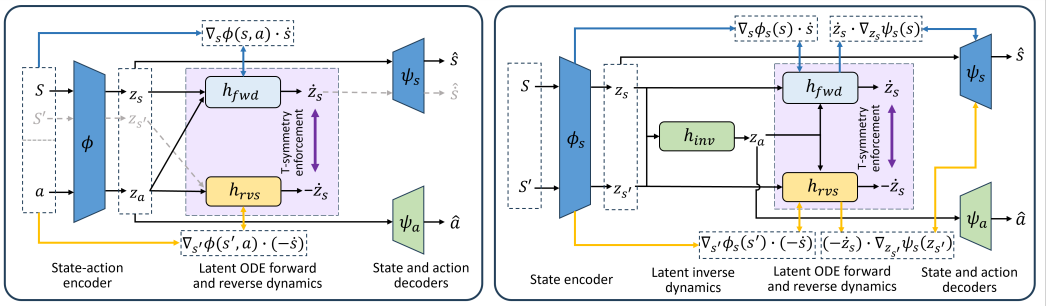
M. Yang and O. Nachum. Representation matters: offline pretraining for sequential decision making. In *International Conference on Machine Learning*, pages 11784–11794. PMLR, 2021.

X. Zhan, H. Xu, Y. Zhang, X. Zhu, H. Yin, and Y. Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

X. Zhan, X. Zhu, P. Cheng, X. Hu, Z. He, H. Geng, J. Leng, H. Zheng, C. Liu, T. Hong, Y. Liang, Y. Liu, and F. Zhao. Data center cooling system optimization using offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

A Additional Discussion on Related Works

In this section, we present a detailed discussion of the connections and differences between our proposed method, TELS, and the existing approaches TSRL (Cheng et al., 2023) and POR (Xu et al., 2022a).



(a) T-symmetry enforced dynamic model (TDM) in TSRL

(b) Our proposed T-symmetry enforced inverse dynamic model (TS-IDM)

Figure 7: Comparison of the architecture between TDM in TSRL and our proposed TS-IDM in TELS.

Connection and Differences with TSRL. As illustrated in Figure 7, both TSRL and TELS leverage the T-symmetry consistency enforcement to construct the latent space. Specifically, in Figure 7 (a), TSRL employs a T-symmetry-enforced dynamics model (TDM), which models system dynamics by incorporating paired latent ODE forward and reverse dynamics to enforce T-symmetry. In contrast, Figure 7 (b) illustrates our proposed T-symmetry-enforced inverse dynamics model (TS-IDM), which integrates T-symmetry constraints into both forward and reverse dynamics while incorporating an inverse dynamics model. We emphasize the main differences between TELS and TSRL as follows:

- **Architecture:** As presented in Figure 7 (a), TDM jointly encodes state-action pairs to form the latent space, which may capture behavioral biases from the dataset (e.g., expert-specific action patterns) and impede learning fundamental, distribution-agnostic dynamics patterns in data. In contrast, Figure 7 (b) illustrates that TS-IDM overcomes these limitations by adopting a state-only modeling approach, focusing on the underlying latent state variations. Additionally, the only useful component of the learned TDM for downstream policy learning is its encoder $\phi(s, a)$, wasting the dynamics-related information captured by the model. In contrast, TS-IDM trains an inverse dynamics model within the T-symmetry-enforced latent space, which can be reused as an execute-policy to extract optimal actions.
- **Detailed Model Design:** As shown in Figure 7 (a), TDM only enforces the ODE property for its encoder but not the decoder, which could lead to inconsistency between the learned dynamics and the underlying ODE structure, resulting in inaccurate or misaligned ODE representations. To address this problem, we introduce the loss term ℓ_{ode} in Eq. (5) specifically to achieve this goal. This design is very important as it can greatly enhance the coupling among the different elements in the model and results in a more stable learning process.
- **Training Procedure:** In TSRL, the TDM encoder and decoders must be pre-trained before joint training on other components to avoid stability issues. In contrast, our proposed TS-IDM does not require pre-training; all components can be learned jointly in a single stage. Additionally, TDM requires adding L1-norm regularization to the parameters of the latent forward and reverse dynamics models to stabilize the learning process. This is unnecessary in TS-IDM (see Eq. 7), as the design of our proposed TS-IDM enables strongly coupled and consistent relationships among all its internal components. The learning curves of TS-IDM can be found in Appendix E.
- **Policy Optimization:** Since TDM requires both state and action inputs to derive latent representations, it is constrained to Q-function maximization for policy optimization. Consequently, TSRL adopts TD3+BC as its backbone for policy learning, which inherently suffers from over-conservative action-level constraints, particularly in small dataset settings. In contrast, TELS performs policy optimization entirely within the compact and generalizable latent state space derived from TS-IDM, enabling state-level optimization that avoids the limitations of action-space constraints.

Table 3: Average normalized scores on full datasets D4RL datasets. The scores are taken over the final 10 evaluations with 5 seeds.

Task	BC	TD3+BC	CQL	IQL	DOGE	IDQL	POR	TSRL	TELS (ours)
Hopper-m	52.9	59.3	58.5	66.3	**98.6 ± 2.1**	63.1	78.6 ± 7.2	86.7±8.7	94.3 ± 2.8
Hopper-mr	18.1	60.9	95.0	94.7	76.2±17.7	82.4	98.9 ± 2.1	78.7±28.1	**99.5 ± 2.3**
Hopper-me	52.5	98.0	**105.4**	91.5	102.7± 5.2	105.3	90.0 ± 12.1	95.9±18.4	**105.4 ± 8.5**
Halfcheetah-m	42.6	**48.3**	44.0	47.4	45.3± 0.6	**49.7**	48.8 ± 0.5	48.2 ±0.7	44.3 ± 0.4
Halfcheetah-mr	**55.2**	44.6	45.5	44.2	42.8 ±0.6	45.1	43.5±0.9	42.2±3.5	41.1 ± 0.1
Halfcheetah-me	55.2	90.7	91.6	86.7	78.7±8.4	**94.4**	**94.7±2.2**	**92.0±1.6**	87.1 ± 2.9
Walker2d-m	75.3	83.7	72.5	78.3	**86.8 ± 0.8**	80.2	81.1 ± 2.3	77.5 ±4.5	81.3± 5.1
Walker2d-mr	26.0	81.8	77.2	73.9	**87.3 ± 2.3**	79.8	76.6 ± 6.9	66.1±12.0	**86.0 ± 3.3**
Walker2d-me	107.5	110.1	108.8	109.6	**110.4±1.5**	**111.6**	109.1 ± 0.7	109.8±3.12	**110.7 ± 1.4**
Antmaze-u	65.0	78.6	84.8	85.5	**97.0 ± 1.8**	93.8	90.6 ± 7.1	81.4 ± 19.2	94.5 ± 10.3
Antmaze-u-d	45.6	71.4	43.4	66.7	63.5 ± 9.3	62.0	71.3 ± 12.1	76.5 ± 29.7	**79.7 ± 15.3**
Antmaze-m-d	0.0	0.0	54.0±11.7	74.6±3.2	77.6±6.1	**86.6**	79.2±3.1	0.0	82.4 ± 4.5
Antmaze-m-p	0.0	0.0	65.2±4.8	70.4±5.3	80.6±6.5	83.5	84.6 ±5.6	0.0	**86.7 ± 5.7**
Antmaze-l-d	0.0	0.0	31.6±9.5	45.6±7.6	36.4 ±9.1	56.4	**73.4 ±8.5**	0.0	41.7 ± 14.2
Antmaze-l-p	0.0	0.0	18.8±15.3	43.5±4.5	48.2±8.1	57.0	58.0 ± 12.4	0.0	**60.7 ± 13.3**

Connection and differences with POR. As discussed in Section 2, while both POR and TELS share similarities in utilizing a state-stitching approach in state space for policy optimization, they exhibit the following fundamental differences:

- **State-Space vs. Latent-Space Optimization:** POR relies on policy optimization in the original state space, which inherently requires sufficient state-action coverage for valid state-stitching. In contrast, TELS mitigates this limitation by constructing a compact and generalizable latent space via TS-IDM.
- **Unregularized T-Symmetry vs. T-Symmetry Regularized Policy Optimization:** POR optimizes the guide-policy solely through an AWR formulation (Neumann and Peters, 2008; Peng et al., 2019), constraining π_g to stay close to the dataset via state-stitching (Eq. 2), but lacks additional regularization to ensure generalizable state transitions. In contrast, TELS enforces an additional T-symmetry consistency regularization $\ell_{\text{T-sym}}$, which plays a critical role in preventing π_g from outputting problematic and non-generalizable latent next states, thereby enhancing its OOD generalizability.

B Additional Results

B.1 Evaluation on the Full Datasets

We also evaluate the performance of TELS on the original full datasets of D4RL tasks, and the results are presented in Table 3. Our proposed method achieves comparable or better performance than existing offline RL methods. Note that although TSRL also adopts a similar T-symmetry regularized representation learning scheme as ours, it performs poorly in Antmaze medium and large datasets. This is primarily due to its use of the conservative TD3+BC backbone for policy optimization, which also behaves similarly in these tasks.

Moreover, we notice that with larger data size and broader state-action space coverage, the strong T-symmetry regularization in the TS-IDM can be properly relaxed, as sufficient data samples can be used to learn the model reasonably well. Therefore, we can trade off some regularization to promote model expressiveness (i.e., lower model learning loss). Specifically, for Antmaze tasks with the full dataset, we set the regularization hyperparameter $\beta = 0.01$ to train the TS-IDM. In Appendix B.3, we provide additional ablation experiments on the influence of the hyperparameter β.

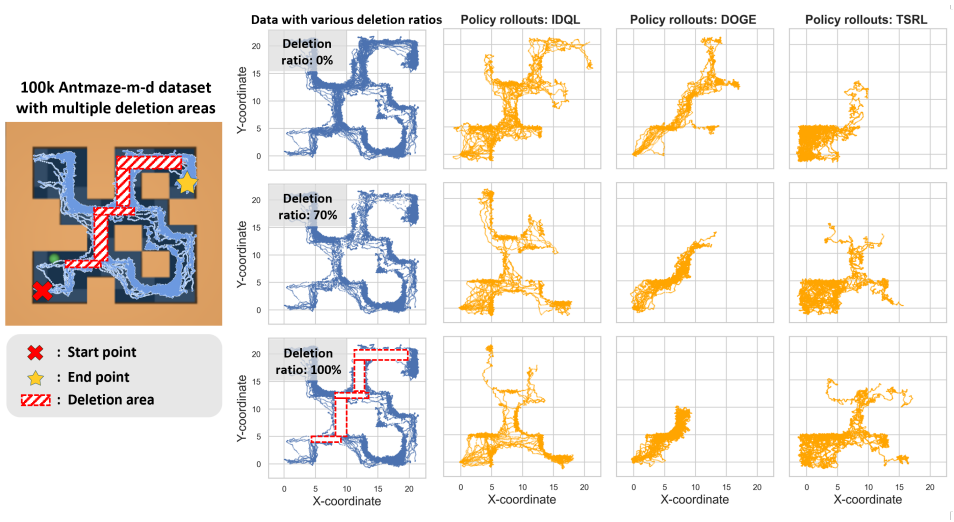


Figure 8: **Left:** Illustration of the 100k Antmaze-m-d task with multiple deletion areas, where the red cross denotes the start point, the yellow star denotes the goal locations, and the red shaded areas denote the data deletion regions. **Right:** Visualization of the training dataset and policy rollout trajectories generated by trained policies from various algorithms under varying deletion ratios.

B.2 Additional OOD Generalizability Validation Experiments

We further evaluate the generalization capabilities of DOGE (Li et al., 2022), IDQL (Hansen-Estruch et al., 2023), and TSRL (Cheng et al., 2023) under the variation deletion AntMaze setting. Specifically, we first train these methods on the remaining data after deletion and then analyze their behaviors by visualizing rollouts over 20 evaluation episodes.

As shown in Figure 8, only IDQL occasionally successfully reaches the goal when no data is deleted (0% deletion), whereas both DOGE and TSRL fail in all cases. As the deletion ratio increases to 70% and 100%, none of the three methods can learn effective policies. These results demonstrate the challenges within this setting, which demands a more compact and expressive latent space and a highly generalizable policy capable of solving the task with severely limited and extremely sparse data. Although TSRL incorporates TDM to capture system dynamics, the available data remains insufficient for its action-constraint-based approach to derive a reasonable policy.

B.3 Additional Ablation Experiments

Impact of T-symmetry regularization on TS-IDM. To investigate the impact of T-symmetry regularization strength controlled by the hyperparameter β in Eq. (7), we conduct additional ablation experiments by varying the value of β to assess how T-symmetry regularization influences the representation learning quality and downstream policy's performance. Specifically, we train TS-IDM on reduced-size 10k D4RL MuJoCo datasets with $\beta = \{0.1, 1, 10\}$, representing different T-symmetry regularization strengths. The learning curves of TS-IDM's overall learning loss "$\mathcal{L}_{\text{TS-IDM}}$" in Eq. (7) are presented in Figure 9. The final policy learning performances with different TS-IDM models are presented in Table 4.

From Figure 9, we observe that choosing a proper β value impacts the learning quality of TS-IDM. A large β (e.g., $\beta = 10$) could impose overly strong regularization and hurt model expressiveness, which is reflected in the high learning loss at convergence. However, when the regularization strength is lowered, maintaining a proper scale of β is important to ensure both the quality and generalizability of the learned representations. As we can see in Figure 9, in the Hopper and Walker2d tasks, choosing $\beta = 1$ provides the lowest "$\mathcal{L}_{\text{TS-IDM}}$" loss; whereas in the Halfcheetah task, "$\mathcal{L}_{\text{TS-IDM}}$" is the lowest when choosing $\beta = 0.1$. If we check the final policy's performance under different TS-IDMs in Table 4, we can see a clear correlation with what we have observed in Figure 9. TELS achieves the highest score on Hopper and Walker2d tasks when $\beta = 1$, but the scores are higher for Halfcheetah tasks when $\beta = 0.1$. This matches exactly with the learning performance of TS-IDM under different β values. The strong correlation between TS-IDM's learning performance and the final policy's

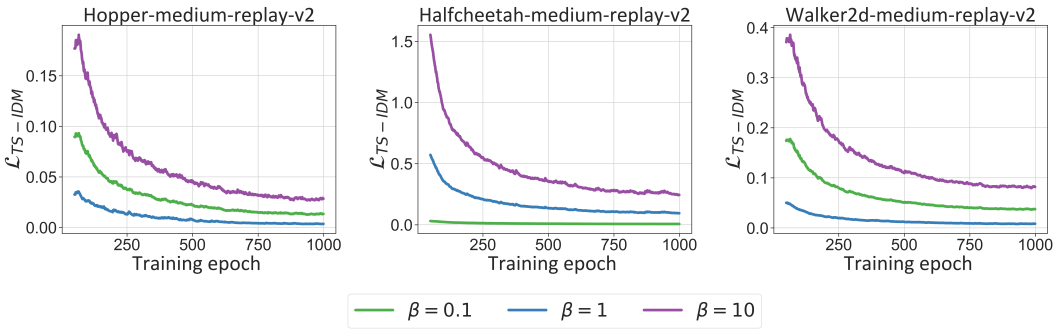


Figure 9: The learning curves for training TS-IDM on 10k dataset with different β hyperparameter.

Table 4: Performance of TELS on 10k D4RL MuJoCo datasets when using TS-IDM with different β hyperparameters.

	$\beta = 10$	$\beta = 1$	$\beta = 0.1$
Hopper-m	77.3 $\pm$ 5.4	**77.3 $\pm$ 10.7**	61.4 $\pm$ 5.6
Hopper-mr	15.3 $\pm$ 6.6	**43.2 $\pm$ 3.5**	19.7 $\pm$3.4
Hopper-me	37.6 $\pm$ 17.9	**100.9 $\pm$ 6.8**	64.7 $\pm$ 3.3
Halfcheetah-m	32.9 $\pm$ 2.3	40.8 $\pm$ 0.6	**41.2 $\pm$ 1.1**
Halfcheetah-mr	8.6 $\pm$ 1.8	33.2 $\pm$ 1.0	**34.0 $\pm$ 2.2**
Halfcheetah-me	7.5 $\pm$ 2.2	**40.7 $\pm$1.2**	39.5 $\pm$ 2.1
Walker2d-m	37.2 $\pm$ 7.9	**62.4 $\pm$ 5.3**	54.6 $\pm$ 8.2
Walker2d-mr	17.1$\pm$2.9	**54.8 $\pm$ 6.0**	39.2 $\pm$ 8.6
Walker2d-me	20.4 $\pm$ 10.4	**87.4 $\pm$ 13.3**	44.7 $\pm$ 9.8

performance of TELS shows that we can select the best β hyperparameter values by simply looking at TS-IDM's training loss and using the one that provides the lowest training loss. This avoids the need to perform potentially unsafe online policy evaluations or unstable offline policy evaluations, which is favorable in real-world deployments.

Impact of components in TS-IDM for stochastic policy optimization. To validate the efficacy of the T-symmetry regularizer $\ell_{\text{T-sym}}$ in Eq. (10), we conduct ablation studies on 100k-sample Antmaze tasks. As evaluation results presented in Table 5, the naïve autoencoder-based inverse dynamics model "$\phi/\psi + h_{inv}$" fails to form a reasonable latent space, yielding 0 average normalized scores across all Antmaze environments. The introduction of latent dynamics models "h_{fwd}" and "h_{rvs}" provides marginal improvements by capturing partial system dynamics yet remains insufficient for effective policy learning. Notably, enforcing ODE properties on decoders and utilizing T-symmetry consistency proves to be the strongest performance improvement factor, which substantially enhances representation reliability for the downstream guide-policy optimization process.

Impact of T-symmetry regularizer term in stochastic policy optimization. As shown in Figure 10 (left), we conduct ablation experiments to evaluate the impact of incorporating the T-symmetry consistency regularization term $\ell_{\text{T-sym}}$, during the guide-policy optimization process of TELS. The results demonstrate that the regularization term plays a critical role in ensuring the learned policy respects the underlying physical symmetries of the system, which becomes especially crucial in environments with limited data coverage. By penalizing deviations from T-symmetry, the guide-policy is encouraged to generate state transitions that are consistent with the system's dynamics, even in OOD regions.

Effectiveness of learned representations for stochastic policy optimization. As shown in Figure 10 (right), we conduct ablation studies to assess TELS's performance with various representation learning models. The results reveal that all baseline models fail to construct informative latent spaces as the task becomes more complex and the dataset expands. In contrast, TS-IDM uniquely learns a well-structured representation that preserves system dynamics. This empirical evidence emphasizes the importance of learning compact and generalizable representations for effective policy optimization in complex environments with sparse data distributions.

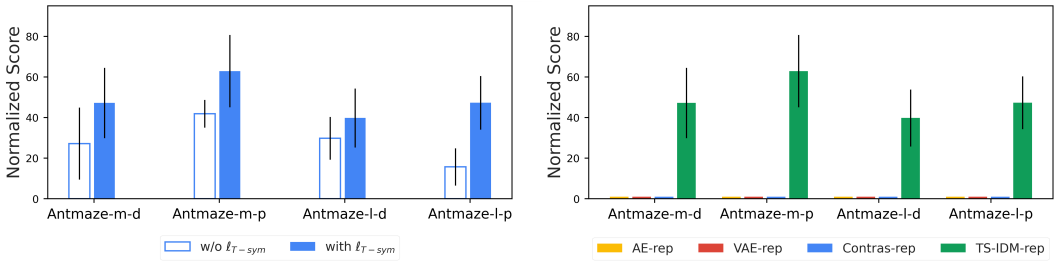


Figure 10: **Left:** Impact of $\ell_{\text{T-sym}}$ on policy optimization. **Right:** Performance of TELS with different representation models on 10k datasets, error bars indicate the normalized scores over 5 random seeds.

Table 5: Ablations on the components of TS-IDM in Antmaze tasks.

	$\phi/\psi + h_{inv}$	$+ h_{fwd}, h_{rvs} \uparrow$	$+ \ell_{\text{ode}} \uparrow$	$+ \ell_{\text{T-sym}} \uparrow$
Antmaze-m-d	0	23.6 ± 18.4	34.1 ± 15.7	$\mathbf{47.2 \pm 17.3}$
Antmaze-m-p	0	30.4 ± 9.3	48.7 ± 13.3	$\mathbf{62.9 \pm 17.8}$
Antmaze-l-d	0	14.4 ± 5.6	20.1 ± 8.9	$\mathbf{39.8 \pm 14.1}$
Antmaze-l-p	0	7.8 ± 3.4	22.6 ± 16.7	$\mathbf{47.3 \pm 13.1}$

C Implementation Details

C.1 Implementation Details for TS-IDM

- **Network Structure.** For all MuJoCo locomotion and Antmaze tasks, we deployed 3-layer feed-forward neural networks for the state encoder ϕ_s, latent inverse dynamics model h_{inv}, forward and reverse dynamics models h_{fwd} and h_{rvs}, and decoder models ψ_s and ψ_a for the latent states and actions. The activation function is ReLU and uses Adam optimizer to update the parameters. We present the hyperparameters details of training TS-IDM in Table 6, including the details of the structure we have implemented as well as the hyperparameters we used during the training process.
- **ODE Property Enforcement on ϕ_s and ψ_s.** We adopt a similar approach to TSRL (Cheng et al., 2023) to train the ODE enforced forward and reverse dynamic models. Specifically, we compute the time-derivative of the state encoder $\phi_s(s)$ by calculating its jacobian matrix through `vmap()` function in Functorch [2]. This allows us to derive the supervision values $\frac{d\phi_s(s)}{ds} \cdot \dot{s}$ and $\frac{d\phi_s(s')}{ds'} \cdot (-\dot{s})$ for the forward dynamics model and reverse dynamics model respectively as in Eq. (4). This approach implicitly enforces the ODE property on the state encoder ϕ_s as the encoder is required to produce state representations that satisfy the ODE constraints. Unlike TSRL, which enforces ODE properties only on the encoders and not on the decoders, our method further regularizes the state decoder ψ_s. Specifically, ψ_s is trained to decode the predicted latent state variables generated by $h_{fwd}(z_s, z_a) = \dot{z}_s$ and $h_{rvs}(z_{s'}, z_a) = -\dot{z}_s$ ensuring that it also satisfies the ODE constraints in Eq. (5). To achieve this, we apply the same approach to compute $\frac{d\psi_s(z_s)}{dt}$ and train the state decoder accordingly.

C.2 Implementation Details for T-symmetry Regularized Policy Optimization

- **Network Structure.** For all MuJoCo locomotion and Antmaze tasks, we deployed 2-layer feed-forward neural networks for the guide-policy π_g and the value function V. The activation function is ReLU and uses Adam optimizer to update the parameters. We list the parameter details in Table 7.
- **Hyperparameters for Policy Optimization.** Under both small-sample and full datasets settings, we employ a deterministic policy update strategy for MuJoCo locomotion tasks, as defined in Eq. (9), with learning rates of 1e-4 for both value and policy functions. The normalization term λ is computed as $\lambda_\alpha = \alpha/[\sum_{s_i} |V(\phi_s(s_i))|/N]$, where α controls the trade-off between value maximization and policy regularization and N denotes the

[2] https://pytorch.org/functorch/stable/functorch.html

number of samples in the training batch. For Antmaze tasks, we utilize a stochastic policy optimization strategy, as outlined in Eq. (10), with learning rates of 1e-3 for value and policy functions.

Full dataset setting: We set $(\tau, \alpha, \eta) = (0.7, 0.01, 10)$ for all MuJoCo locomotion tasks and $(\tau, \alpha) = (0.9, 10)$ for all Antmaze tasks.

Small-sample setting: For Halfcheetah and Walker2d tasks, we set $(\tau, \alpha, \eta) = (0.5, 0.01, 5)$ and incorporate policy dropout to mitigate overfitting. These tasks share identical state and action dimensions (17 states and 6 actions), enabling the use of the same parameter set for guide-policy training. In contrast, Hopper tasks with a smaller state-action space (11 states and 3 actions) are comparatively simpler given the same amount of training data (e.g., 10k samples). Consequently, we adopt a more aggressive learning strategy for Hopper, setting $(\tau, \alpha, \eta) = (0.7, 0.1, 10)$ to prioritize value maximization. For Antmaze tasks, we use an identical set of parameters $(\tau, \alpha) = (0.9, 10)$ as in the full dataset setting to train the guide-policy.

Training resources. To train a TS-IDM, we utilize one NVIDIA GeForce RTX 4090 with an AMD Ryzen 9 7950X 16-Core Processor and 16GB of memory for approximately 30 minutes, running on Ubuntu 22.04.2 LTS 64-bit. We employ the same resource configurations for approximately 6 hours for the guide-policy training.

Table 6: Hyperparameters of TS-IDM.

	Hyperparameters	**Value**
TS-IDM Architecture	State encoder hidden units	512×256
	State encoder activation function	ReLU
	Latent forward model hidden units	256×256
	Latent forward model activation function	ReLU
	Latent reverse model hidden units	256×256
	Latent reverse model activation function	ReLU
	latent inverse model hidden units	1024×1024
	Latent inverse model activation function	ReLU
	Latent inverse model dropout	True
	Latent inverse model dropout rate	0.1
	State decoder hidden units	256×512
	State decoder activation function	ReLU
	Action decoder hidden units	512×512
	Action decoder activation function	ReLU
Hyperparameters	Optimizer type	Adam
	β	1 (locomotion tasks); 0.1 (antmaze tasks)
	Weight of ℓ_{rec}	1
	Learning rate	3e-4
	Batch size	256
	Training epoch	1000
	State normalize	True
	Weight decay	0 (locomotion tasks); 1e-5 (antmaze tasks and full dataset setting)

Table 7: Structure and training parameters of guide-policy optimization

	Hyperparameters	**Value**
Guide-policy structure	Value network hidden units	1024×1024
	Value network activation function	ReLU
	Policy network hidden units	1024×1024
	Policy network hidden units	ReLU
Training Perparameters	Optimizer type	Adam
	Target Value network moving average	0.05
	Batch size	256
	Training steps	100000
	State normalize	True

D Detailed Experiment Setups

Reduced-size dataset generation. To create reasonable reduced-size D4RL datasets for a fair comparison, we use the identical small-sample as in TSRL paper (Cheng et al., 2023) for the locomotion tasks training. For Antmaze tasks, we adopt a similar approach by randomly sub-sampling trajectories from the original dataset to construct smaller training datasets. Specifically, for the "Antmaze-umaze" tasks, we randomly sample 10k data points for training. However, for the "Antmaze-medium" and "Antmaze-large" tasks, we use larger datasets to account for the significantly larger map scales.

The rationale behind this adjustment is that the "medium" and "large" environments are significantly more expansive than the "umaze" environment. Sampling only 10k data points would likely result in trajectories that lack the fundamental information necessary to describe the task. As a result, we relax the small-sample constraints for these environments to ensure that the reduced datasets at least contain enough successful trajectories for effective training.

Experiment setups for various representation learning. To assess the effectiveness of the representations learned by TS-IDM, we integrate them as the representation model with IQL and TD3+BC, verifying the usability of the learned latent space as illustrated in Figure 4. Specifically, we process the original states s and next states s' from the dataset using the pre-trained state encoder ϕ_s to derive their latent representations: $\phi_s(s) \to z_s$ and $\phi_s(s') \to z_{s'}$. The algorithms are then trained entirely within this latent state space. The implementation details of these representation models are as follows:

- **"AE-rep"**: We implement a naïve autoencoder-based inverse dynamics framework, consisting of a state encoder and decoders ϕ_s and ψ_s to construct the latent state space, The inverse dynamics model h_{inv} is then built within this latent space, serving as the execute-policy, as in TELS. For a fair comparison, we use the same network parameters for the encoder, decoder, and inverse dynamics model as in TS-IDM. The "AE-rep" model is trained with a reconstruction loss to capture the essential features of the input, while the inverse dynamics model is simultaneously trained on the latent representations to predict actions.

- **"VAE-rep"**: The variational autoencoder (VAE) (Kingma and Welling, 2014) is built based on the "AE-rep" model by introducing additional KL divergence loss terms. Specifically, the encoder outputs parameters of a Gaussian distribution in the latent space, and the latent representations are sampled using the reparameterization trick. The VAE is trained using a combined loss function that includes both the reconstruction loss and the KL divergence loss, which regularizes the latent space to follow a prior distribution. The inverse dynamic model is trained simultaneously with the VAE, sharing the latent space and optimizing for both the reconstruction of the input data and the prediction of actions.

- **"Contras-rep"**: We utilize the NT-Xent loss (Normalized Temperature-Scaled Cross Entropy Loss) used in SimCLR (Chen et al., 2020) within the latent representation space on top of the "AE-rep" model. The overall loss function combines the contrastive loss with the reconstruction loss, ensuring that the latent space not only captures the structure of the data but also learns semantically meaningful representations that are robust to variations. The inverse dynamic model is trained simultaneously within the latent space to predict actions.

Experiment setups for OOD generalization tasks in Antmaze. In Section 4.2, we conduct a more challenging scenario to verify the OOD generalizability of the algorithm. Specifically, based on 100k "Antmaze-medium-diverse-v2" dataset, we manually selected five critical intervals and erased the data points within these intervals by randomly deleting them. The selection of intervals was determined based on the XY-axis coordinates. In this dataset, the first two dimensions of the state represent the vertical and horizontal coordinates, respectively. Based on this information, we randomly deleted 70% and 100% of the data in the chosen intervals. We then trained IQL (Kostrikov et al., 2022), DOGE (Li et al., 2022), IDQL (Hansen-Estruch et al., 2023), POR (Xu et al., 2022a), TSRL (Cheng et al., 2023), and TELS using this modified dataset to evaluate their generalizability.

E LEARNING CURVES

The following are the learning curves of TS-IDM and the T-symmetry regularized guide-policy optimization in TELS on the reduced-size D4RL MuJoCo and Antmaze datasets. We evaluate the policy with 10 episodes over 5 random seeds.

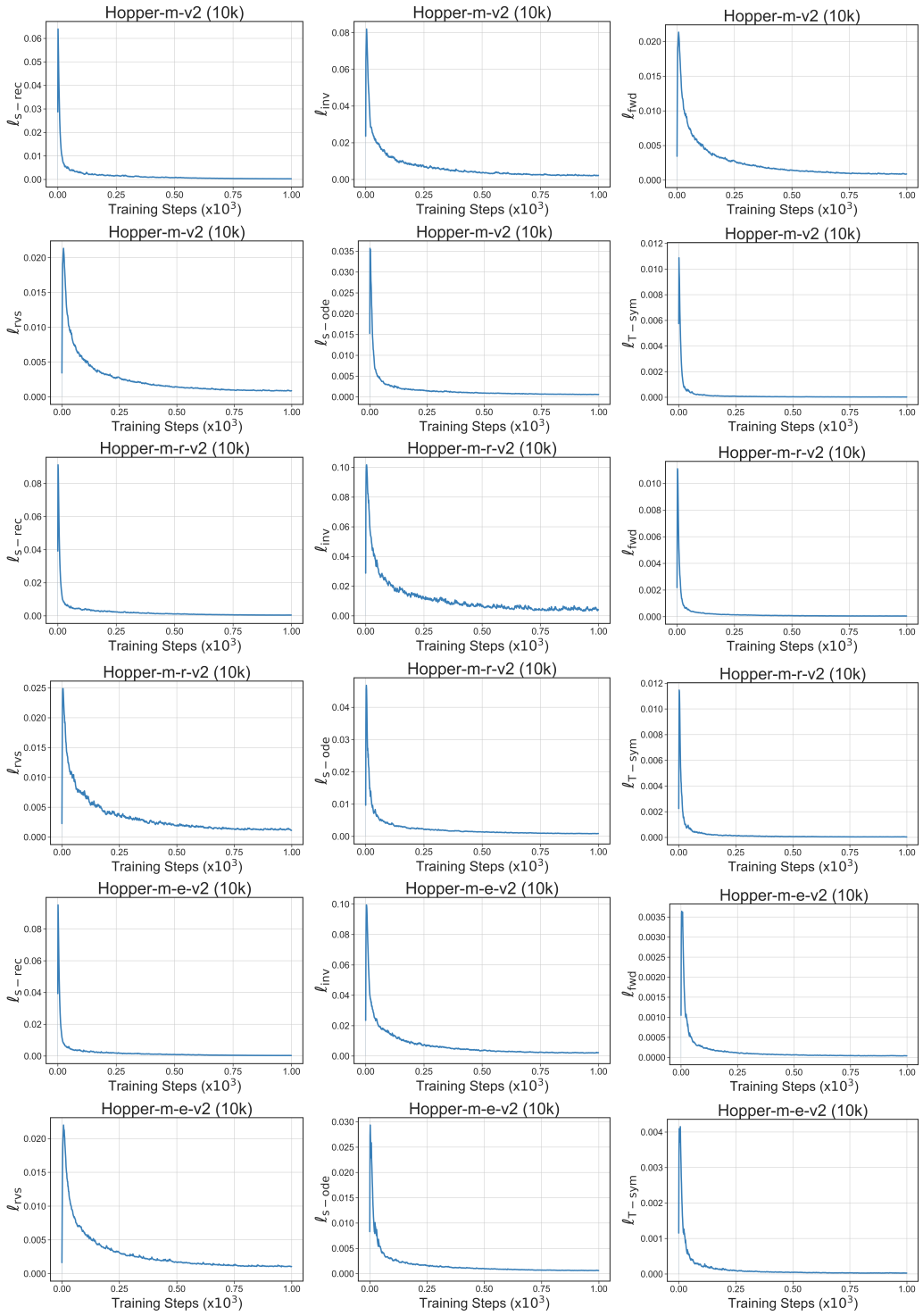


Figure 11: Learning curves of the overall and each individual loss terms in TS-IDM for Hopper tasks.

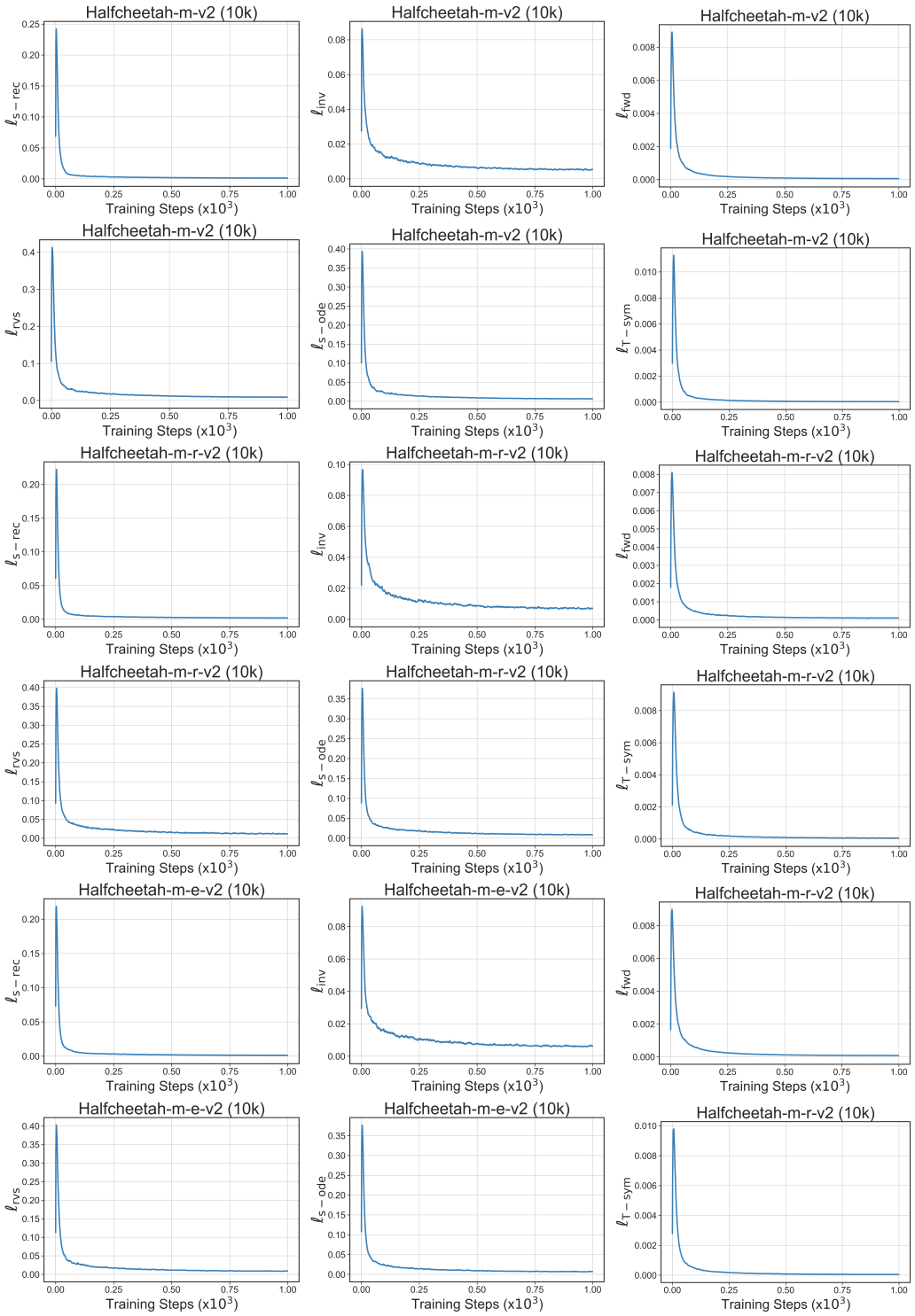


Figure 12: Learning curves of the overall and each individual loss terms in TS-IDM for Halfcheetah tasks.

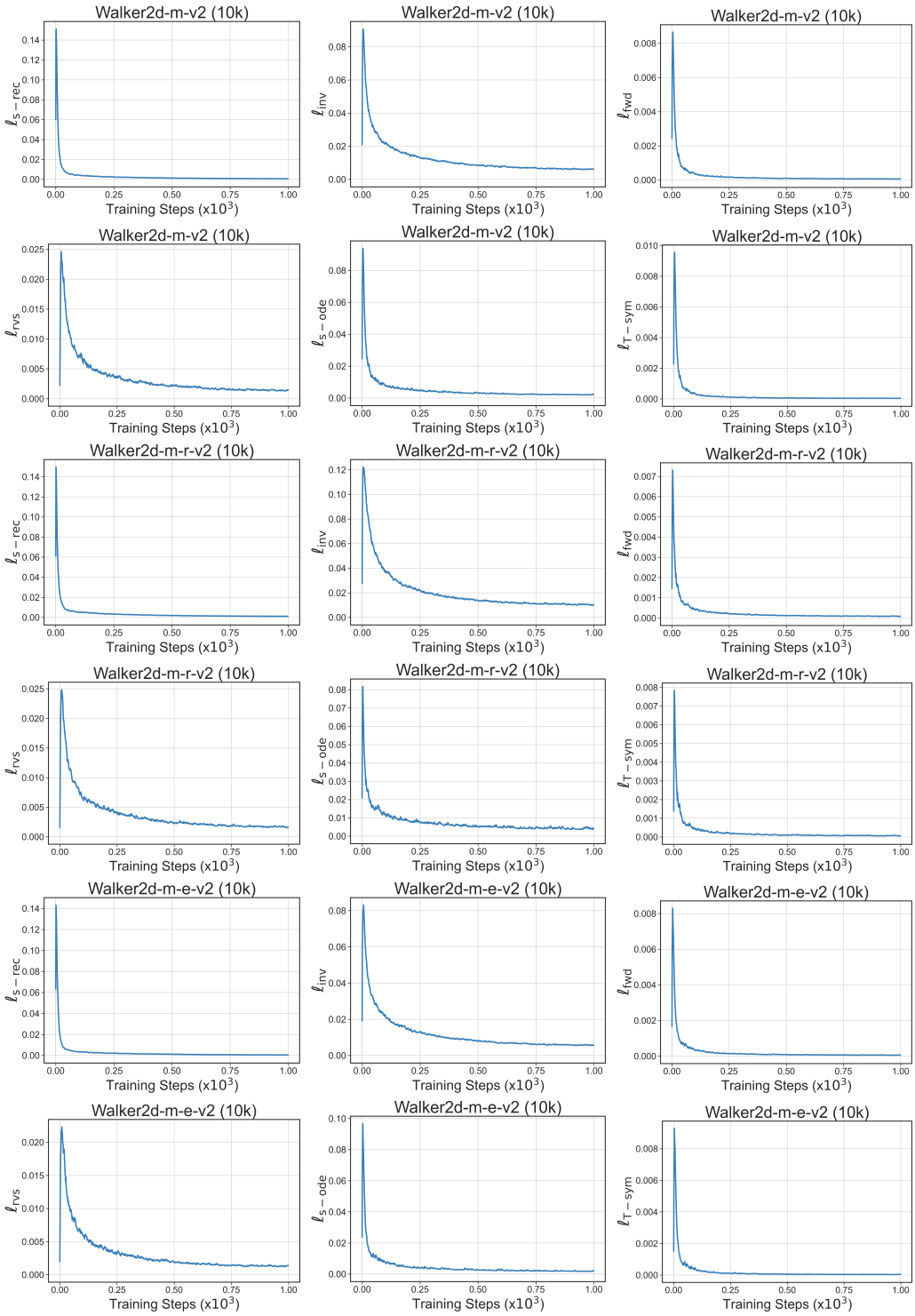


Figure 13: Learning curves of the overall and each individual loss terms in TS-IDM for Walker2d tasks.

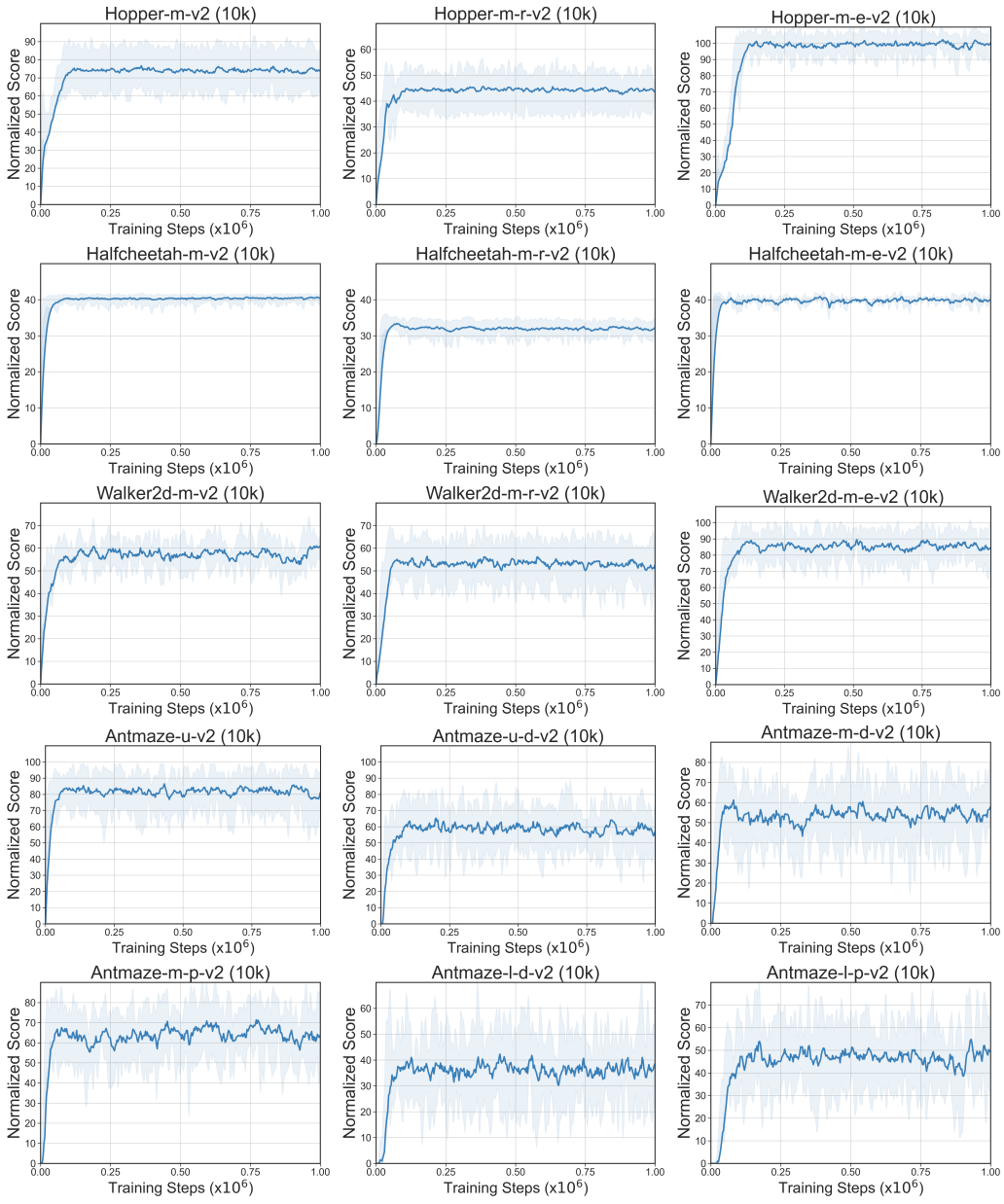


Figure 14: Learning curves of policy optimization in TELS for D4RL MuJoCo and Antmaze tasks with reduced-size datasets. We evaluate the policy within 10 episodes over 5 random seeds.