A Implementation Details

Implementation details for TDM. As discussed in Section 3 and the illustrative Figure 2, TDM is modeled as a physics-informed reconstruction model with embedded ODE latent dynamics and T-symmetry preserving design.To ensure optimal training performance of TDM, we have included some additional implementation details below. More hyperparameters details of TDM are discussed in the next section.

- **Network structure:** In all our D4RL experiments, we implement the encoder, decoders, latent forward and reverse dynamics as 4-layer feed-forward neural networks with ReLU activation, and optimized using Adam optimizer. For the state decoder $\psi(\cdot, \delta_s)$, we concatenate an extra indicator δ_s in the input to help the state decoder to decide the target output. More specifically, to decode $z_s \to s$, we concatenate $\delta_s = 0$ with z_s as input; and for $\dot{z}_s \to \dot{s}$, we concatenate $\delta_s = 1$ with $\dot{z}_s$.
- **Computing second derivative of $\phi(\cdot)$:** As TDM involves a pair of latent ODE forward and reverse dynamics models, whose training losses Eq. (4) and (6) involve regressing on $\frac{\partial\phi(s,a)}{\partial s}\dot{s}$ and $\frac{\partial\phi(s',a)}{\partial s'}(-\dot{s})$ as target values. This results in a gradient through a gradient of $\phi(\cdot)$. Computationally, we calculate the Jacobian matrix $\frac{\partial\phi(s,a)}{\partial s}$ using the `vmap()` function in Functorch[†] to ensure the second derivative of $\phi(\cdot)$ can be correctly backpropagated during stochastic gradient descent. Similar a treatment can also be implemented with other auto-differentiation frameworks like Jax[‡] that support computing higher-order derivatives.
- **Pre-training the encoder and decoders:** As the final learning objective of TDM Eq. (8) involves several loss terms, we observe that in small datasets, loss terms such as the reconstruction loss (Eq. (3)) for the encoder and decoders converges much slower than other loss terms. When updating all the loss terms with the same number of training steps, some losses suffer from over-fitting while others are still not fully converged. For these cases, we pre-train the encoder and decoders with the reconstruction loss for a given number of training steps, and then use the complete learning objective of TDM (Eq. (4)) for the rest of the training. The numbers of pre-training/training epochs for the experiments in this paper are reported in Table 2.

  As reported in Table 2, we find that the number of pre-training epochs required for TDM to reach the best learning performance is associated with the specific task and the size of training data. For small datasets, TDM generally needs more training and pre-training epochs to avoid overfitting the latent dynamics and T-symmetry losses. For MuJoCo locomotion tasks, we recommend pre-training the encoder and decoders for 10% of the total training epoch. For the more complex adroit tasks, TDM requires more epochs to extract the ODE dynamics and T-symmetry property of the system dynamics. In this case, there is no pre-training necessary for the encoder and decoders.

Table 2: Training epochs of TDM for D4RL tasks with different dataset scales

	Locomotion Tasks			**Adroit Tasks**	
	5k&10k	50k & 100k	Full dataset	5k&10k	Full dataset
Training epoch	2000	1000	200	2000	200
Pre-train epoch	200	100	20	0	0

- **Enhancement on the T-symmetry regularization:** We observe that in some small datasets (mainly in the Halfcheetah environment), the training of the latent reverse dynamics model g might suffer from a certain level of degeneration. This is reflected as the $g(z_s + f(z_s, z_a), z_a)$ produces similar values as $-f(z_s, z_a)$, resulting in small T-symmetry consistency loss values (Eq. (7)), however, the discrepancy between $g(z_{s'}, z_a)$ and $-f(z_s, z_a)$ remains large. To solve this issue and further enforce the T-symmetry, we apply the following enhanced T-symmetry regularization when such a phenomenon is observed:

$$\ell_{Enhanced\text{-}T\text{-}sym}(z_s, z_a) = \|f(z_s, z_a) + g(z_s + f(z_s, z_a), z_a)\|_2^2 + \|f(z_s, z_a) + g(z_{s'}, z_a)\|_2^2 \qquad (12)$$

  We find that applying the above enhanced T-symmetry loss can successfully resolve the degeneration issue of the latent reverse dynamics model and achieve good performance in the downstream offline

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

[‡]https://github.com/google/jax

RL tasks. However, we find in most small datasets, the original T-symmetry consistency loss is sufficient. We advise only to use the above enhanced T-symmetry consistency loss when large discrepancies between $\|f(z_s, z_a) + g(z_s + f(z_s, z_a), z_a)\|_2^2$ and $\|f(z_s, z_a) + g(z_{s'}, z_a)\|_2^2$ are observed.

Hyperparameter details for TDM and TSRL. The architectural parameters of TDM and TSRL, as well as the TSRL hyperparameters are summarized in Table 3. Based on the different scales of the datasets, we basically only use two sets of hyperparameters for all D4RL-MuJoCo locomotion tasks and only one set of hyperparameters for all D4RL-Adroit tasks. Because of the extremely narrow distribution of the reduced-size expert dataset, we apply Dropout [56] with dropout rate of 0.1 to regularize the policy network in all tasks with 10k expert data.

Table 3: Hyperparameter details for TDM and TSRL

	Hyperparameters	**Value**
TDM Architecture	Optimizer type	Adam
	Weight of ℓ_{T-sym} and ℓ_{ds} and ℓ_{rec}	1
	Weight of ℓ_{rvs} and ℓ_{fwd}	0.1
	Learning rate	3×10^{-4}
	State normalization	True
	Hidden units of forward and reverse model	512
	Hidden units of encoder	$512 \times 256 \times 128$
TSRL Architecture	Critic neural network layer width	512
	Actor neural network layer width	512
	State normalization	True
	Actor learning rate	3×10^{-4}
	Critic learning rate	3×10^{-4}
	Policy noise	0.2
	Policy noise clipping	0.5
	Policy update frequency	2
	Discount factor γ	0.99
	Number of iterations	10^6
	Target update rate	0.005
	λ_{L1}	1e-5
TSRL Hyperparameters	α	2.5
	τ	50% for Walker2d and Adroit tasks, 70% for HalfCheetah and Hopper2d
	λ_1	MuJoCo: 5 or 10 for full dataset, 100 or 200 for 10k dataset Adroit: 10,000 for both full and reduced datasets
	λ_2	1 for MuJoCo full & Adroit datasets 100 for MuJoCo 10k dataset

B Detailed Experiment Setups

Reduced-size dataset generation. To create reasonable reduced-size D4RL datasets for a fair comparison, we sub-sample the trajectories in the datasets rather than directly sampling the (s, a, s', r) transitions. For example, there are 2M (s, a, s', r) transitions in the "halfcheetah-medium-expert" dataset, we first split these records into 2,000 trajectories based on the done condition, then randomly draw 10 trajectories (10k transition points) to serve as the reduced-size datasets for model training.

Experiment setups for representation learning evaluation. To evaluate the representation quality and the impact of each design choice of TDM, we compare TDM representation with several baselines on the small dataset settings. We provide the detailed description of the representation learning baselines as follows:

- **"AE-rep" model:** We construct a vanilla auto-encoder without any further constraints during the learning process, which was trained by the reconstruction loss only. The network sizes of the encoder and decoders are the same as the ones used in TDM.
- **"AE-fwd-rep" model:** Similar to the "AE-rep" model but with a latent forward dynamics prediction model f, which is implemented as a 4-layer feed-forward neural network with ReLU activation, and optimized using Adam optimizer (same as TDM). The forward model was trained by minimizing the loss term $\|\dot{z}_s - f(\phi(s,a))\|_2^2$, where we directly regress $f(\phi(s,a))$ with the $\dot{z}_s$ derived from the latent states obtained from the encoder as $\dot{z}_s = z_{s'} - z_s$. Note that in this baseline, no ODE property nor T-symmetry regularization is included. Again we use the decoder to decode $\dot{z}_s \rightarrow \dot{s}$ as in TDM for the next state prediction.
- **"TDM-no-ODE" model:** Holds the same structure with TDM but trained with no ODE property. More specifically, similar with "AE-fwd-rep", the latent forward and reverse dynamics model was trained by $\|\dot{z}_s - f(\phi(s,a))\|_2^2$ and $\|(-\dot{z}_s) - g(\phi(s',a))\|_2^2$, where $-\dot{z}_s$ is directly calculated from the encoded latent states, i.e., $\dot{z}_s = z_{s'} - z_s$. Note that in this baseline, the T-symmetry is also implicitly captured, since both the latent forward and reverse dynamics models are regressing the same $\dot{z}_s$ and its opposite value.
- **"SimSiam" model:** For the self-supervised representation learning baseline, we implement an auto-encoder structure with the optimization objective proposed in the SimSiam paper [44]. For detailed model description and hyperparameters setting, please refer to Chen et al. [44].

Experiment setups for evaluating generalization performance. To evaluate TSRL's generalization capability beyond the offline datasets, we construct two low-speed datasets based on the original D4RL Walker2d medium and medium-expert datasets. In accordance with the Gym documentation, we selected the "x-coordinate velocity of the top" (8th dimension of the states) in the walker environment to perform data filtering. We remove all samples with the x-coordinate velocity of the top greater than $0.2\times$ max-x-velocity recorded in the data. This results in two smaller low-speed datasets (about 200k for the medium dataset and 250k for the medium-expert dataset). We train TDM and TSRL on these low-speed datasets and the results are reported in Figure 7 (main paper).

C Additional Results

Complete results on D4RL Adroit tasks. The complete results of TSRL in Adroit human and cloned tasks with different dataset scales are presented in Table 4. As shown in the results, TSRL achieves much better performance in the pen tasks, both the full datasets and the reduced-size datasets.

Table 4: Complete results on D4RL Adroit tasks

Task	Ratio	Size	BC	TD3+BC	MOPO	CQL	IQL	DOGE	TSRL
Pen-human	1	5k	34.4	8.4	9.7	37.5	71.5	42.6 ± 16.3	**80.1±18.1**
Hammer-human	1	5k	1.5	2.0	0.2	**4.4**	1.4	-1.2 ± 0.2	0.2± 0.3
door-human	1	5k	0.5	0.5	-0.2	**9.9**	4.3	-1.1 ± 0.2	0.5± 0.3
Relocate-human	1	5k	0.0	-0.3	-0.2	**0.2**	0.1	-0.3 ± 0.5	0.1 ± 0.1
Pen-cloned	1	500k	56.9	41.5	-0.1	39.2	37.3	56.9 ± 15.2	**64.9 ± 20.1**
	1/50	10k	37.4 ± 37.6	-0.1 ± 6.9	-0.1 ± 0.1	1.5 ± 4.8	35.6 ± 30.5	30.1 ± 19.7	**41.6 ± 27.5**
Hammer-cloned	1	500k	0.8	0.8	0.2	**2.1**	**2.1**	0.2 ± 0.3	1.7 ± 1.9
	1/50	10k	0.3 ± 0.4	0.2 ± 0.1	0.1 ± 0.1	0.2 ± 0.1	0.4 ± 0.2	0.3 ± 0.1	**0.6 ± 0.3**
Door-cloned	1	500k	-0.1	-0.4	-0.1	0.4	**1.6**	-0.1 ± 0.1	-0.1 ± 0.6
	1/50	10k	-0.1 ± 0.1	-0.3 ± 0.1	-0.2 ± 0.1	-0.3 ± 0.1	**1.5 ± 0.8**	-0.5 ± 0.5	-0.1 ± 0.3
Relocate-cloned	1	500k	-0.1	-0.3	-0.3	**0.1**	-0.2	-0.2 ± 0.1	-0.2 ± 0.1
	1/50	10k	-0.2 ± 0.1	-0.3 ± 0.1	-0.3 ± 0.1	-0.3± 0.1	**-0.1 ± 0.5**	-0.2 ± 0.1	-0.2 ± 0.1

Additional results on Antmaze-umaze tasks. We also conduct experiments on the D4RL Antmaze-umaze tasks with full and reduced-size 10k datasets. The results are presented in Table 5. We use the

Table 5: Results on D4RL Antmaze-umaze tasks with full and reduced-size datasets

Task	**Ratio**	**Size**	**BC**	**TD3+BC**	**CQL**	**IQL**	**DOGE**	**TSRL(ours)**
Antmaze-u	1	1M	54.6	78.6	84.8	85.5	**97.0** ± **1.8**	81.4 ± 19.2
	1/100	10k	44.7 ±42.1	0.7 ± 1.2	0.1 ± 0.0	65.1 ± 19.4	56.3 ± 24.4	**76.1**±**15.6**
Antmaze-u-d	1	1M	45.6	71.4	43.4	66.7	63.5 ± 9.3	**76.5** ± **29.7**
	1/100	10k	24.1±22.2	16.27 ± 16.4	0.5 ± 0.1	34.6 ± 18.5	41.7 ± 18.9	**52.2** ±**22.1**

Table 6: Performance of data augmentation methods with 10k reduced-size D4RL datasets.

Task	**CABI**	**S4RL-N**	**S4RL-U**	**TSRL**
Hopper-m	48.3 ± 3.9	28.3 ± 6.2	23.6 ± 4.7	**62.0** ± **3.7**
Hopper-mr	19.8 ± 3.9	16.6 ± 12.9	12.5 ± 12.2	**21.8** ± **8.2**
Hopper-me	38.3 ± 5.3	12.5 ± 3.8	13.1 ± 4.7	**50.9** ± **8.6**
Hopper-e	34.6 ± 24.4	14.1 ± 12.9	12.2 ± 11.6	**82.7** ± **21.9**
Halfcheetah-m	34.8 ± 1.9	25.1 ± 6.8	23.2 ± 7.1	**38.4** ± **3.1**
Halfcheetah-mr	23.5 ± 3.4	15.1 ± 9.3	14.8 ± 9.5	**28.1** ± **3.5**
Halfcheetah-me	29.9 ± 1.7	27.1 ± 7.1	23.4 ± 8.2	**39.9** ± **21.1**
Halfcheetah-e	4.2 ± 4.1	2.4 ± 3.9	1.8 ± 3.1	**40.6** ± **24.4**
Walker2d-m	42.4 ± 23.3	24.5 ± 4.3	21.9 ± 4.8	**49.7** ± **10.6**
Walker2d-mr	11.7 ± 7.6	1.5 ± 2.1	1.4 ± 2.3	**26.0** ± **11.3**
Walker2d-me	17.4 ± 9.2	21.9 ± 16.4	16.0 ± 13.2	**46.7** ± **17.4**
Walker2d-e	20.2 ± 3.4	56.5 ± 26.7	51.1 ± 29.7	**102.2** ± **11.3**

same hyperparameters as in the D4RL MuJoCo tasks. Again, we find that TSRL achieves comparable performance as other baselines on the full datasets, but is substantially better under small datasets.

Additional comparative results on data augmentation. We conduct additional experiments on the reduced-size 10k MuJoCo datasets to compare TSRL and other offline RL methods using data augmentation. In particular, we compared with model-free method S4RL [42] with Gaussian (S4RL-N) and uniform (S4RL-U) noises, as well as a recent model-based data augmentation method CABI [43]. CABI employs a pair of predictive dynamics models to assess the reliability of the augmented data. The results are presented in Table 6.

The results clearly show that TSRL outperforms all offline RL baselines with data augmentation under small datasets. It is also observed that model-based methods TSRL and CABI generally perform better than model-free data augmentation method S4RL in this setting, due to access to additional dynamics information. Moreover, as CABI does not learn a strongly regularized dynamics model with T-symmetry consistency as in our proposed TDM, it still has a noticeable performance gap as compared to our method.

Ablation on the level of ODE and T-symmetry regularization in TDM. As discussed in the conclusion section of the main paper, TDM adds extra ODE dynamics and symmetry regularizations, which are beneficial to improve model generalization, but will lose some model expressiveness if the regularization is too strong. In this section, we conduct an ablation study on the impact of the regularization strength of the ODE property and the satisfaction with the T-symmetry. Specifically, we vary the loss weights of ℓ_{fwd}, ℓ_{rvs} and ℓ_{T-sym} in the TDM learning objective (Eq. (8)), and train a loosely regularized and a strongly regularized TDM model on the 10k datasets (see Table 7). The loosely regularized model has the maximum reconstruction expressivity but may not produce a well-behaved representation due to weak regularization. Whereas the strongly regularized model sacrifices the expressivity for regularized behaviors. We further evaluate their performance with TSRL, with the results reported in Table 8. The experiment results demonstrated that an overly expressive model could not help the RL algorithm to derive a well-behaved policy with limited data

Table 7: TDM with different regularization strengths

Different versions of TDM	ℓ_{rec}	ℓ_{ds}	ℓ_{fwd}	ℓ_{rvs}	ℓ_{T-sym}	λ_{L1}
Loosely regularized	1	1	0.01	0.01	0.01	1e-5
Paper	1	1	0.1	0.1	0.1	1e-5
Strongly regularized	1	1	1	1	1	1e-5

Table 8: Performance of TSRL with different TDM models on 10k datasets

Task	**TDM (loosely regularized)**	**TDM (paper)**	**TDM (strongly regularized)**
Hopper-m	50.7±13.6	**62.0**±**3.7**	43.6±14.3
Hopper-m-r	15.4±9.7	**21.8**±**8.2**	15.6±9.8
Hopper-m-e	49.7±17.1	**50.9**±**8.6**	30.9±20.5
Halfcheetah-m	**39.1**±**3.6**	38.4±3.1	36.6±30.0
Halfcheetah-m-r	**28.3**±**6.9**	28.1±3.5	22.9±8.4
Halfcheetah-m-e	36.2±5.4	**39.9**±**21.1**	31.0 ± 3.4
Walker2d-m	43.2±27.3	**49.7**±**10.6**	35.6±26.2
Walker2d-m-r	20.2±18.1	**26.0**±**11.3**	21.7±6.1
Walker2d-m-e	25.9±20.7	**46.4**±**17.4**	29.4±24.7

due to potential overfitting and inconsistency with the T-symmetry property. On the other hand, an overly regularized model may also hurt performance. This is consistent with our previous insight that a trade-off exists between model expressiveness and T-symmetry agreement. A proper balance between these two behaviors can be necessary for small-sample learning.

Learning curves for TSRL on D4RL locomotion tasks. The learning curves for reduced-size D4RL MuJoCo datasets with 10k samples are showed in Figure 8. For each evaluation step, the policies are evaluated with 5 episodes over 3 random seeds.

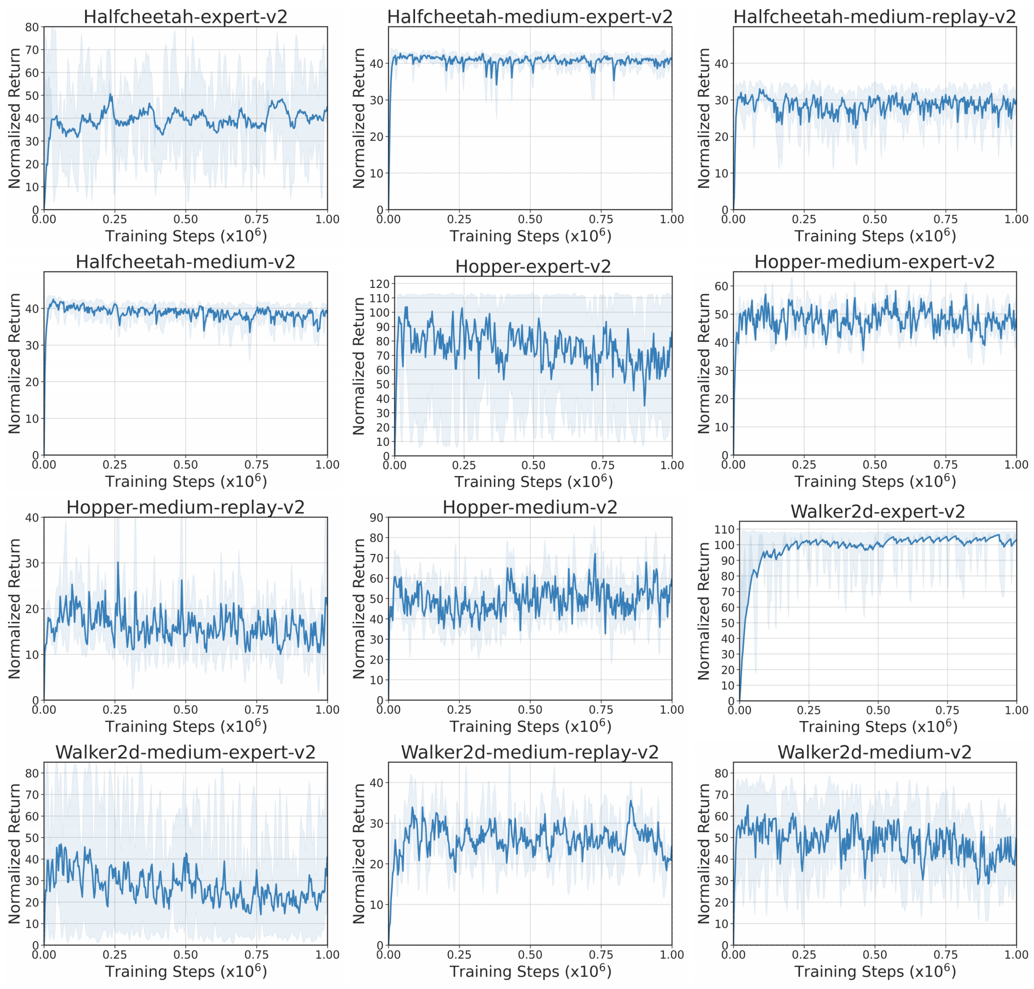

Figure 8: Learning curves for reduced-size D4RL MuJoCo datasets. Error bars indicate min and max values over 3 random seeds.