Offline Transition Modeling via Contrastive Energy Learning

Ruifeng Chen [*1 2] **Chengxing Jia** [*1 2] **Zefang Huang** [1] **Tian-Shuo Liu** [1 2] **Xu-Hui Liu** [1] **Yang Yu** [1 2]

Abstract

Learning a high-quality transition model is of great importance for sequential decision-making tasks, especially in offline settings. Nevertheless, the complex behaviors of transition dynamics in real-world environments pose challenges for the standard forward models because of their inductive bias towards smooth regressors, conflicting with the inherent nature of transitions such as discontinuity or large curvature. In this work, we propose to model the transition probability implicitly through a scalar-value energy function, which enables not only flexible distribution prediction but also capturing complex transition behaviors. The Energy-based Transition Models (ETM) are shown to accurately fit the discontinuous transition functions and better generalize to out-of-distribution transition data. Furthermore, we demonstrate that energy-based transition models improve the evaluation accuracy and significantly outperform other off-policy evaluation methods in DOPE benchmark. Finally, we show that energy-based transition models also benefit reinforcement learning and outperform prior offline RL algorithms in D4RL Gym-Mujoco tasks.

1. Introduction

Learning a transition model provides a promising approach for enhancing efficient decision-making (Sutton & Barto, 2018), known as model-based methods (Janner et al., 2019; Luo et al., 2024b; Moerland et al., 2023). Model-based methods involve encapsulating the transition dynamics of the real environment to predict the subsequent state given a specific state-action pair. By leveraging collected transition data to model these dynamics, the learned model extends beyond the initial dataset, effectively facilitating subsequent decision-making and evaluation. This is particularly advantageous in offline settings (Levine et al., 2020) where additional samples cannot be obtained by online interaction. In such scenarios, policies can be executed within the learned model instead of directly interacting with the real environment, leading to improved and more robust offline policy evaluation (Thomas & Brunskill, 2016; Fu et al., 2021) and optimization (Yu et al., 2020; 2021; Chen et al., 2021; Rigter et al., 2022; Sun et al., 2023). Therefore, model-based methods have the potential to significantly enhance real-world decision-making applications.

As the core of model-based methods, learning the transition function requires the ability to accurately capture the characteristics of the environment. In sequential tasks, as the trajectory length increases and the policy distribution shifts, model errors have a greater impact on decision-making (Asadi et al., 2019; Xu et al., 2020). Previous works primarily address the detrimental impact of model errors in two ways. One approach is to avoid exploring areas with high prediction uncertainty (Yu et al., 2020; Sun et al., 2023). The second approach aims to introduce new architectures for transition modeling. For example, Zhang et al. suggests using autoregressive dynamics models for continuous control. Additionally, Trajectory Transformer (TT) (Janner et al., 2021) adopts a transformer architecture to model the trajectory distribution. However, these works simply adopt new model architectures and larger capacities, without changing models' optimization process and then, the generalization ability, fundamentally.

The current transition models typically explicitly predict the next state, which we refer to as a Forward Transition Model (FTM). It inputs the current state-action pair and outputs the next-state prediction with variance estimate. Theoretically, FTM, given a sufficient model capacity, should have the ability to fit various scenarios. However, we have found that in some complex situations, especially for environmental transitions with discontinuities or sharp changes that are ubiquitous in real-world tasks with frictional contact (Todorov, 2014; Coumans, 2015; Pfrommer et al., 2021), FTMs are difficult to perform well, which must explicitly match the large gradient of the steep function that may cause generalization issues especially for the areas with low data density, and requires sufficiently dense data and thorough optimization to compensate this effect. We demonstrate that

[*]Equal contribution [1]National Key Laboratory for Novel Software Technology, Nanjing University, China & School of Artificial Intelligence, Nanjing University, China [2]Polixir Technologies. Correspondence to: Yang Yu <yuy@nju.edu.cn>.

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

even a simple case could lead to a poor generalization.

In this paper, we turn to energy-based transition modeling (ETM) as an alternative to FTM, which can be more consistent with the complex nature of real-world transition dynamics. Energy-based models (Song & Kingma, 2021) introduce an energy function to implicitly represent the target distribution and have demonstrated enhanced generalization capabilities compared to forward models when confronted with non-smoothness, extrapolations, and multi-modality (Florence et al., 2021). They can approximate steep or discontinuous functions without large gradients in the function approximator, distinctly superior to forward models. Corresponding to a robust variant of the maximum likelihood method, training the energy models contrastively by minimizing InfoNCE loss also optimizes the energy prediction at the contrastive points, balancing and shaping the energy surface. These desirable properties enable our energy-based transition models (ETM) to effectively capture irregular transition behaviors and perform better in complex scenarios.

We showcase the efficacy of our ETM for irregular transition modeling in a didactic environment with discontinuous transition dynamics. We also find the energy-based transition models trained on offline data with limited coverage have a smaller absolute error when tested on out-of-distribution transitions. This property is beneficial to offline policy evaluation tasks, where we can simulate trajectories for the policies and directly estimate their expected values. Besides, we also conduct experiments on D4RL benchmarks (Fu et al., 2020) , where the improvement of model accuracy boosts the performance of policy optimization. Concretely, the key contributions of this work are summarized as follows:

1. This work is the first to introduce energy-based models to learn the transition dynamics.
2. We reveal the relationship between a contrastive learning objective InfoNCE and the maximum likelihood principle for energy-based models.
3. Our ETMs significantly outperform existing OPE methods on a set of DOPE tasks (Fu et al., 2021).
4. Our ETMs also boost the offline policy optimization over D4RL MuJoCo tasks (Fu et al., 2020), surpassing or matching the previous state-of-the-art performance.

2. Preliminaries

2.1. Reinforcement Learning

The objective of reinforcement learning is to learn a policy that maximizes the expected return in a Markov Decision Process (MDP) (Sutton & Barto, 2018). An MDP can be described by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ is the transition probability, $r(s,a)$ is the reward function, $\gamma \in (0,1)$ is the discount factor and ρ_0 is the initial state distribution.

For a given policy π, the value function $V^\pi(s) = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)|s_0 = s\big]$ is the expected discounted cumulative rewards for the trajectories starting from s_0 and following policy π. The action-value function (or Q function) is similarly defined as $Q^\pi(s,a) = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)|s_0 = s, a_0 = a\big]$. The expected value $v^\pi = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\big] = \mathbb{E}_{s\sim\rho_0}[V^\pi(s)]$, then the evaluation of policy π is to estimate v^π and reinforcement learning is to find a policy π^* whose expected value $v^{\pi^*} \geq v^{\pi'}$ for any other policy π'.

2.2. Offline RL and Off-Policy Evaluation

In the online setting, the agent can interact with the environment to collect experiences to estimate the policy value and improve the policy performance. However, in the offline setting (Levine et al., 2020), the agent is provided a static dataset, and further interaction with the environment is not allowed. Here the offline dataset $\mathcal{D} = \{(s,a,r,s')\}$ consists of transitions from trajectories collected by some behavior policies, which normally differ from the current policy to be evaluated or optimized. This distribution discrepancy leads to a primary obstacle faced in offline reinforcement learning, extrapolation errors, which result in severe value overestimation due to the bootstrapped Bellman value update. To mitigate these errors, offline RL algorithms adopt conservatism in different ways (Fujimoto et al., 2019; Yu et al., 2020; Kumar et al., 2020; Fujimoto & Gu, 2021; Kostrikov et al., 2021; Sun et al., 2023), where the agent is discouraged from exploiting extrapolation.

Off-policy evaluation (OPE) aims to evaluate the performance of target policies based on static off-policy experiences (Fu et al., 2021). The Monte Carlo estimate of true policy values requires on-policy experiences, normally obtained by online interaction with the environment, and the static offline setting presents a substantial challenge to accurate evaluation, usually leading to significant value gaps.

2.3. Model-based Reinforcement Learning

Model-based methods attempt to learn a parametric model P_θ to recover the underlying transition dynamics P of the environment from the available experiences, allowing the agent to synthesize imagined experiences to facilitate policy evaluation and optimization. This paradigm is believed to have the better potential to fully leverage the limited experiences, which reduces the demand for the interaction data and improves the sample efficiency for the online RL tasks (Janner et al., 2019). In the offline scenarios, the learned transition model plays a more important role because it

serves as a surrogate for the environment dynamics to interact with the agent, potentially enhancing adaptability to out-of-distribution states and actions (Yu et al., 2020; Kidambi et al., 2020). However, the learned transition models inevitably suffer from errors for the limited dataset, which will be further amplified during long-horizon rollout. The model errors not only result in poor value estimation but also tend to be erroneously exploited by the reinforcement learning algorithms, leading to significant performance degradation. To mitigate the impact of model errors, recent approaches incorporate uncertainty estimation to penalize the rewards at the unpredictable region (Yu et al., 2020; 2021; Sun et al., 2023), which can be regarded as a model-based version of conservatism.

The standard transition model learning is to use a forward model that takes the current state and action as input and predict the distribution of the next state as a multivariate Gaussian with a diagonal covariance, which is trained to fit the true transition P via log-likelihood maximization:

$$\max_{\theta} L(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \log[(P_\theta(s'|s,a))]. \tag{1}$$

2.4. Inductive bias of NN-based forward models

Forward models based on neural networks trained via SGD are known to be biased towards smooth regressors of the data (Belkin et al., 2019). Although this effect is often seen as a form of regularization and beneficial in many problems with smoothness nature, it will cause negative generalizations in some complex scenarios where the groundtruth mapping has discontinuity or extreme curvature (Pfrommer et al., 2021; Florence et al., 2021). Conflicting with the problems' irregularity property, training NN-based forward models requires significant effort (sufficient data density and thorough optimization) to overcome the inductive bias and perform well therein. This negative generalization effect caused jointly by problem irregularities and the conflicting inductive bias of smooth function approximator is explained by the concept of *interference* (Bengio et al., 2020), which can be characterized in the first order by the inner product of the objectives' gradients:

$$\rho_{1,2} = \nabla_\theta L^\top(x_1) \nabla_\theta L(x_2), \tag{2}$$

where x_1, x_2 correspond to two different data points and L is a point-wise loss function. If $\rho_{1,2} > 0$, minimizing the loss at one point x_1 by a gradient step also decreases the loss at the other point x_2, resulting in a constructive generalization. However, if $\rho_{1,2} < 0$, gradient descent at x_1 will increase the loss $L(x_2)$, which is the destructive interference. In Appendix A we show that a simplistic case with sharp value changes in groundtruth function can lead to such negative interference. In more complex cases, similar effects often occur to a greater extent.

Energy-based implicit models (Florence et al., 2021) turn to predict an energy function to score various possible target predictions instead of predicting the target directly. Therefore, the interference between different sample points brought by the function approximator does not directly affect the predictions but affects the energy estimate. Training energy function in a contrastive manner also balances the influence of different sample points and thereby weakens the negative interference. These features enable energy-based models to approximate steep or discontinuous functions without large gradients in the function approximator that may cause generalization issues.

3. Method

Inspired by the characteristics difference between forward models and energy-based models in Section 2.4, we choose the energy formulation to learn the environment transitions.

3.1. Energy-based Transition Models

The transition model can be defined implicitly by a parameterized scalar function $E_\theta(s, a, s')$, and the reduced transition probability distribution is

$$p_\theta(s'|s,a) = \frac{\exp\left(-E_\theta(s,a,s')\right)}{Z_\theta(s,a)}, \tag{3}$$

where $Z_\theta(s,a) = \int \exp\left(-E_\theta(s,a,s')\right) \mathrm{d}s'$ is the normalizing constant. This formulation is known as energy-based models (EBM) (Song & Kingma, 2021) and the scalar function $E_\theta(s,a,s')$ is named energy function. Notice that only the energy function is directly parameterized (usually by a neural network), while $Z_\theta(s,a)$ is intractable due to the integration and hence distribution $p_\theta(s'|s,a)$ is implicitly defined and direct sampling from it is also intractable. We name it **Energy-based Transition Model (ETM)**.

The salient feature that using a scalar function to specify a probability distribution first results in a flexible distribution representation that is able to model complex dependency of the state dimensions in contrast to the diagonal multivariate Gaussian predicted by standard forward models. The cost of this flexibility is the difficulty of the exact likelihood computation and exact sampling. The common method to approximate samples is to use efficient MCMC sampling method (Liu & Liu, 2001; Neal et al., 2011), e.g., Langevin MCMC (Welling & Teh, 2011), which iteratively refines samples with step size $\epsilon > 0$ to simulate the real sampling:

$$\hat{s}'_{k+1} \leftarrow \hat{s}'_k - \frac{\epsilon^2}{2} \nabla_{s'} E_\theta(s,a,\hat{s}'_k) + \epsilon z_k, \tag{4}$$

where z_k are i.i.d standard Gaussian noises. In practice, the noise scale can be modulated to control sample stochasticity. If the noise scale reduces to zero, the iterative sampling reduces to finding the lowest energy point:

$\hat{s}' = \arg\min_{s'} E_\theta(s, a, s')$, corresponding to the maximum probability point of distribution (3), where the composition of the $\arg\min$ operator and the continuous energy function enables the ability to capture complex functional behaviours, especially for the discontinuity (Florence et al., 2021).

We train the energy function E_θ by optimizing the infoNCE loss function (Oord et al., 2018) in a contrastive fashion:

$$l(s, a, s', \{s'_j\}_{j=1}^K; \theta) = -\log \frac{\exp\left(-E_\theta(s, a, s')\right)}{\sum_{j=1}^K \exp\left(-E_\theta(s, a, s'_j)\right)}, \tag{5}$$

where (s, a, s') is a real transition experience and $\{s'_j\}_{j=1}^K$ are generated negative samples for the next state. In practice, one of the K negative samples is chosen to be the positive sample s' so that the optimization can be seen as a classification problem to tell the positive sample s' from all the K samples $\{s'_j\}_{j=1}^K$. Other $K-1$ negative samples are generated independently from a Langevin MCMC process (4) as an approximation to the real samples from the intractable exact distribution $p(s'|s, a; \theta) \propto \exp\left(-E_\theta(s, a, s')\right)$.

We also use a gradient penalty (Jolicoeur-Martineau & Mitliagkas, 2019) to regularize the energy function during training as in (Florence et al., 2021), because if the energy function overfits the training data points, the landscape will raise a challenge to the gradient-based Langevin sampling, leading to low-quality negative sample generation and therefore degenerated contrastive training. This effect resembles the generative adversarial network (GAN) (Goodfellow et al., 2014; Arjovsky et al., 2017) training, where an overly aggressive discriminator might hinder generator training. The gradient penalty pushes down the Lipschitz constant of energy function $E_\theta(s, a, s')$ w.r.t. s', in favor of the gradient-based Langevin MCMC, and stabilizes training.

3.2. Connections between InfoNCE loss and Maximum Likelihood Principle

Learning the energy function by minimizing InfoNCE loss (5) seems intuitively reasonable; however, it still requires some theoretical justification for its rationale. In contrastive representation learning, InfoNCE is a widely used loss function due to the mutual information argument (Oord et al., 2018), while probabilistic models are normally trained by maximum likelihood principle, and there is no clear relationship between the two methods previously.

In fact, we find that the InfoNCE loss can be interpreted as a robust variant of the maximal likelihood training for the energy-based model. For ease of elaboration, we use x to denote features and y to denote labels.

To minimize the NLL (negative log likelihood) of $p(y|x; \theta)$, we have a expression of its gradient:

$$\nabla_\theta - \log p(y_i|x_i; \theta) = \nabla_\theta E_\theta(x_i, y_i) - \mathbb{E}_{p(y|x_i;\theta)} \nabla_\theta E_\theta(x_i, y),$$

which means gradient descent according to the likelihood is equivalent to minimizing the following objective

$$l(x_i, y_i; \theta) = E_\theta(x_i, y_i) - \mathbb{E}_{\mathrm{sg}(p(y|x_i;\theta))} E_\theta(x_i, y), \tag{6}$$

where sg denotes stop gradient operator. However, it is impractical to directly sample from the energy-induced distribution $p(y|x_i, \theta)$, and many previous works use Markov Chain Monte Carlo (MCMC) method to approximate the real samples, which amounts to minimizing the surrogate objective known as Contrastive Divergence (Hinton, 2002):

$$l_{CD}(x_i, y_i; \theta) = E_\theta(x_i, y_i) - \mathbb{E}_{\hat{p}(y|x_i)} E_\theta(x_i, y),$$

where $\hat{p}$ is the data distribution of the samples obtained via MCMC. However, this will introduce approximation errors unless using long enough Markov chains, which hurts the model training as noticed in previous literature on energy-based models. If we take into account this sampling error and tolerate an ϵ KL divergence $D_{\mathrm{KL}}(p(\cdot|x_i; \theta), \hat{p}(\cdot|x_i)) \leq \epsilon$ between the sampling distribution $\hat{p}(y|x_i)$ and the real distribution $p(y|x_i; \theta)$, we can instead minimize an upper bound

$$\max_{\bar{p}: D_{\mathrm{KL}}(\bar{p}, \hat{p}) \leq \epsilon} E_\theta(x_i, y_i) - \hat{\mathbb{E}}_{\bar{p}(y|x_i)} E_\theta(x_i, y), \tag{7}$$

so that the original objective (6) is no larger than this surrogate. This constrained optimization is equivalent to minimizing the Lagrange dual function for some $\lambda > 0$ according to KKT conditions:

$$\max_{\bar{p}} E_\theta(x_i, y_i) - \hat{\mathbb{E}}_{\bar{p}(y|x_i)} E_\theta(x_i, y) - \lambda D_{\mathrm{KL}}(\bar{p}, \hat{p}). \tag{8}$$

Solving the maximization w.r.t. $\bar{p}$ obtains the final objective:

$$\begin{aligned} & E_\theta(x_i, y_i) + \lambda \log \mathbb{E}_{\hat{p}(y|x_i;\theta)} \exp\Big(-\frac{1}{\lambda} E_\theta(x_i, y)\Big) \\ = & -\lambda \log \frac{\exp\Big(-\frac{1}{\lambda} E_\theta(x_i, y_i)\Big)}{\mathbb{E}_{\hat{p}} \exp\Big(-\frac{1}{\lambda} E_\theta(x_i, y)\Big)}. \end{aligned} \tag{9}$$

Now we can see the InfoNCE loss (5) is an approximately unbiased estimate of Equation (9) if the negative sample number K is large enough and constants are ignored. Therefore we conclude that InfoNCE loss is a robust variant of the maximum likelihood energy learning objective that is more tolerant to the error of negative sample distribution.

3.3. Overall Model Learning Framework

The learning process of energy-based transition models is summarized in Algorithm 1. We use fully-connected networks with the same hidden layers and widths as in forward transition models to represent the energy function in later experiments for fair comparisons. The reward function $r(s, a)$ can either be jointly learned by adding an extra dimension to s', or separately incorporates a standard reward model. The detailed hyperparameter setting is listed in Appendix C.

Algorithm 1 Energy-based Transition Model Learning

Require: Offline transitions $\mathcal{D} = \{(s, a, s')\}$, initialized energy network E_θ, batch size B, number of negative samples K, iteration number N.
for $i = 1$ **to** N **do**
  Sample transition batch $\{(s, a, s')\}_B \sim \mathcal{D}$
  Generate negative samples $\{s'_j\}_{j=1}^K$ via Langevin (4)
  Compute energy values for both positive transitions $E_\theta(s, a, s')$ and negative transitions $\{E_\theta(s, a, s'_j)\}_{j=1}^K$
  Optimize the energy function parameter θ according to $l_{\text{InfoNCE}} + l_{\text{GradPen}}$ shown in (5) and (22).
end for
return Energy network E_θ

4. Experiments

In this section, we conduct a series of experiments to answer the following questions: (1). Does ETM better recover the discontinuous transition behaviors than standard FTMs? (2). Does ETM have a smaller transition error on out-of-distribution transitions? (3). Can ETM facilitate sequential decision-making tasks like off-policy evaluation and offline RL? [1].

4.1. Model Analysis

We answer the first question by a didactic example in Section 4.1.1 and the second question in Section 4.1.2.

4.1.1. Didactic example of discontinuous transition

To investigate the capabilities of forward and energy-based transition models to deal with discontinuous transition behaviors, we craft a didactic example featuring sharp transition changes. We construct a jumping transition prediction task, where the observation contains two elements: position and height, and agent utilize an one-dimension action as force to jump. Different actions and current positions can lead to distinct or even discontinuous next height. Figure 1(a) depicts the actual jump height within a certain action range from a height of 0, revealing a highly non-smooth transition. See more details of the task in Appendix C.4.

After training the forward transition model and the energy-based model using a dataset where the action is randomly sampled from 0.7 to 0.9 for jumping from any position on the track with an equal number of samples, we proceeded to evaluate both models across the data where the action ranges from 0.65 to 0.95 at all positions on the track. The relative model errors of in-distribution actions (from 0.65 to 0.7 and from 0.9 to 0.95) and that of out-of-distribution actions (from 0.7 to 0.9) are shown in Figure 4.1.1. The predicted height is depicted in Figure 1(b,c), and we also present the results for specific actions, including action values of 0.74 in Figure 2(b) and 0.68 (representing out-of-distribution and extrapolation actions) in Figure 2(c). When viewed from a three-dimensional mesh in Figure 1 (b) and (c), our method demonstrates superior generalization. However, forward models struggle to predict such non-smooth scenarios, and we have achieved better results on extrapolated data points. Notably, the forward model even generated negative predictions while all the predicted targets are positive. Our findings highlight that our method excels at generalizing on non-smooth and extrapolation-reliant data, indicating that our approach adeptly captures the data patterns while circumventing the negative interference caused by the smooth approximator.

4.1.2. Errors on out-of-distribution transitions

We report the absolute error of FTMs and ETMs on the holdout samples from their training dataset in mujoco tasks in Table 8, where FTMs have smaller model errors in 16/20 tasks, demonstrating the flexible distribution modeling and in-distribution generalization. However, transition models are often faced with out-of-distribution data, requiring OOD generalization. We train the ETMs and standard FTMs respectively on the random and medium level dataset for the hopper and walker2d tasks in D4RL, then test their accuracy on datasets of all five levels. The random datasets are collected by random policies, and therefore significantly differ from the other four level datasets, which are collected by the policies trained via Soft Actor Critic (SAC) algorithms ranging from medium to expert performance. Therefore such out-of-distribution generalization is of great challenge. The results in Figure 3 show that though achieving similarly small errors on the training datasets, the standard FTM has a significantly larger transition error on the other four unseen datasets than our ETM. This result showcases the energy-based transition model's advantage of out-of-distribution generalization over forward transition models.

A typical situation to utilize the transition models is to roll out the policies within the model to simulate trajectories, in which case the transition models are faced with out-of-distribution transitions due to the autoregressive prediction. We visualize the simulated trajectories by energy-based and forward transition models in Appendix H, comparing them with the real trajectories. The off-policy evaluation experiment in the next subsection also demonstrates the OOD generalization ability of our ETMs.

4.2. Off-policy Evaluation

Off-policy evaluation task provides a good test scenario for the dynamics learning, where an accurate transition model

[1] code: https://github.com/Ruifeng-Chen/Energy-Transition-Models.git

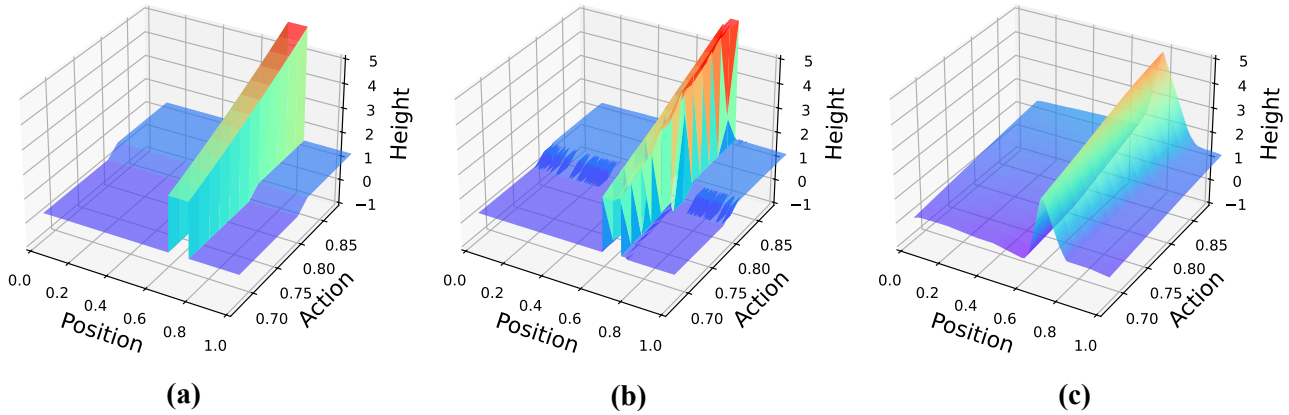
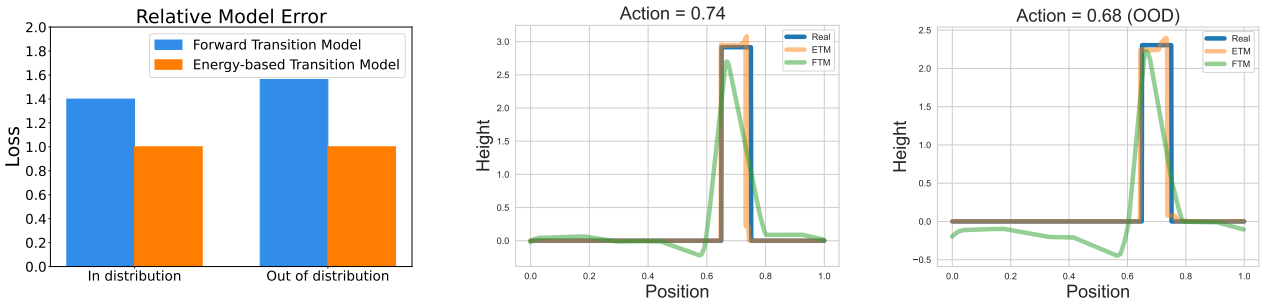


Figure 1. The visualization of the height of the next observation calculated from the current position and action in real transition (left), ETM (middle), and FTM (right).



(a) Relative model error. (b) Prediction at in-dist action 0.74 (c) Prediction at out-of-dist action 0.68

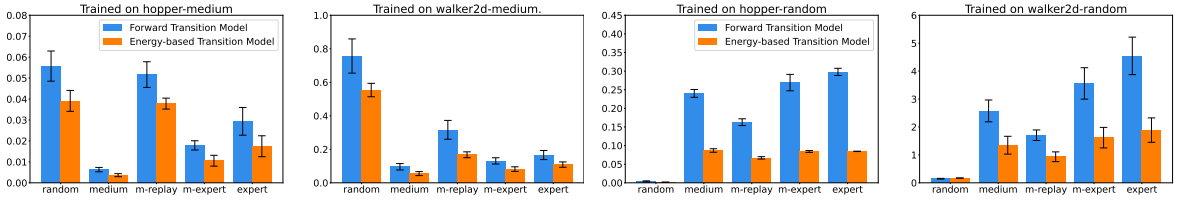
Figure 2. The model error comparison between ETM and FTM in different action distributions (a) and the predicted height of real, FTM, and ETM transitions at in-distribution action 0.74 (b) and out-of-distribution action 0.68 (c).

allows an accurate value estimation by rolling out the target policies therein. We follow the DOPE benchmark (Fu et al., 2021) to evaluate policies over various D4RL environments. The benchmark contains challenging OPE tasks where the training dataset include varying levels of coverage of the state-action space, and target policies are designed toward resulting in state-action distributions different from the ones induced by behavioral policies, which poses a great challenge to the model's generalization ability.

Enviroments and Tasks. We use 4 Gym-Mujoco environments (hopper, walker2d, halfcheetah, ant) with training datasets of 5 levels and 4 Adroit environments with training dataset of 3 levels, resulting in a total of 32 sets of tasks. DOPE provides 11 target policies for each environment to be evaluated by the OPE methods.

Baselines and Evaluation Metrics. We compare the direct method using ETMs (Algo.2) with five model-free OPE baselines reported by DOPE, i.e. **Fitted Q-Evaluation (FQE)** (Le et al., 2019), that estimates the policy value via iteratively performing Bellman update, **Doubly Robust (DR)** (Jiang & Li, 2016), that combines the importance sampling technique with a value estimator for variance reduction, **Importance Sampling (IS)** (Kostrikov & Nachum, 2020), that performs importance sampling with a learned behavior policy, **DICE** (Yang et al., 2020), that uses a saddle-point objective to estimate marginalized importance weights, **Variational Power Method (VPM)** (Wen et al., 2020), that runs a variational power iteration algorithm to estimate the importance weights without the knowledge of the behavior policy, as well as the model-based method using FTMs. Following the DOPE benchmark, our evaluation metrics include mean absolute error (MAE), rank correlation, and regret@1, as detailed in Appendix D.1. The overall results are reported in Figure 4, where all results are averaged over three seeds to keep aligned with the DOPE benchmark.

Results. Figure 4 shows the mean overall performance of our ETM and baselines over 32 Gym-Mujoco and Adroit tasks. We find that ETM outperforms FTM and other baselines significantly. In general ETM attains the lowest absolute error, which demonstrates the better model accuracy and therefore the reduced value gaps. Besides, the higher rank correlation and smaller regrets show that ETM can also help select good policies, which is a desirable property in realistic applications. The tabular results (including the



(a) hopper-medium (b) walker2d-medium (c) hopper-random (d) walker2d-random

Figure 3. The mean absolute errors of standard forward models and energy-based transition models **trained** on **random** or **medium** level datasets of hopper and walker2d tasks in D4RL, **tested** under **all five level** datasets. The results are averaged over 3 seeds.

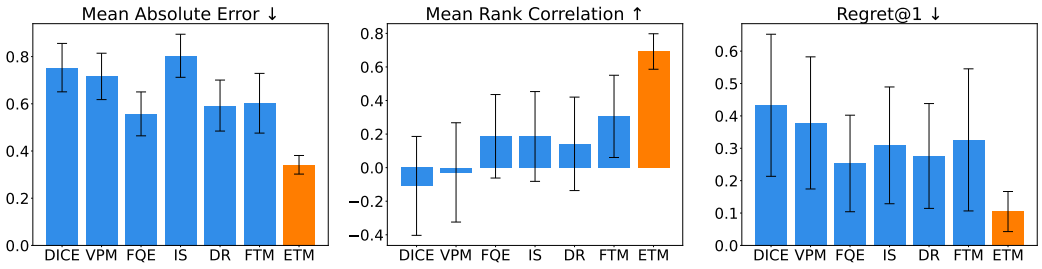


Figure 4. The overall Off-Policy Evaluation results across 32 tasks, averaged over 3 seeds.

raw absolute error) for each environment and dataset are reported in Appendix D.2.

4.3. Offline RL

Model-based methods for offline RL are believed to benefit from the better generalization ability of transition models compared to that of value functions for the model-free methods. We apply our ETM to offline RL algorithm and evaluate the effectiveness on 12 D4RL Gym-Mujoco tasks.

Implementation. Similar to previous model-based RL algorithms (Chua et al., 2018; Janner et al., 2019; Yu et al., 2020; 2021), we adopt an ensemble of five energy-based transition models for policy optimization. For the policy optimization part, we use Soft Actor Critic (Haarnoja et al., 2018) as the base algorithm with the reward penalized by the std norm of the next-state predictions of the model ensembles:

$$\mathcal{U}(s,a;\{P_{\theta_i}\}_{i=1\sim5}) = \|\text{std}(\{\hat{s}' \sim P_{\theta_i}(\cdot|s,a)\})\|. \quad (10)$$

This penalty form resembles the ensemble-std version (Lu et al., 2022) of MOPO (Yu et al., 2020), where the difference is that ETMs do not provide a direct variance estimate and we simply use the empirical std of samples. We name the method **Energy-Model-based offline Policy Optimization (EMPO)**. It is possible to design a more advanced penalty to further improve the performance, which is beyond the scope of this work. Our implementation is based on OfflineRL-Kit (Sun, 2023).

Baselines and Tasks. We compare our method with several offline RL algorithms, including model-free methods: **CQL** (Kumar et al., 2020), **TD3+BC** (Fujimoto & Gu, 2021), **EDAC** (An et al., 2021); and model-based methods: **MOPO** (Yu et al., 2020) , **COMBO** (Yu et al., 2021), **Trajectory Transformer (TT)** (Janner et al., 2021), **RAMBO** (Rigter et al., 2022), and **MOBILE** (Sun et al., 2023). These approaches are evaluated on a total of twelve datasets involving three environments (hopper, walker2d, halfcheetah) and four dataset types (random, medium, medium-replay, medium-expert) per environment. The baseline results are obtained from (Sun et al., 2023).

Results. Table 1 reports the normalized score for each dataset with standard derivation among five seeds, the average performance over all datasets, and the number of solved tasks whose score ≥ 95.0. We find that our method EMPO outperforms or on par with previous best methods on 11 out of 12 tasks and achieves the highest average score among all methods. Besides, our method solves 7 out of 12 tasks to achieve scores greater than 95.0, while previous methods at most solve 5 (EDAC and MOBILE). These improvements demonstrate the efficacy of ETM for policy optimization.

4.4. Ablation Study

In previous main experiments, we use 16 negative samples in the ETM training for all tasks. To evaluate the impact of the number of negative samples, we also train ETMs using 8 or 24 negative samples for model learning and report the OPE results (absolute error, rank correlation, and regret) for

Table 1. Normalized average returns in 12 D4RL tasks, averaged over 5 seeds. *Solved tasks* denotes the number of the tasks whose scores $\geq$ 95.0. The best results are **bolded** and the previously best results are underlined. The letters m, r and e in some task names represent medium, replay and expert respectively to save space.

Task Name	CQL	TD3+BC	EDAC	MOPO	COMBO	TT	RAMBO	MOBILE	EMPO (Ours)
halfcheetah-r	31.3	11.0	28.4	38.5	38.8	6.1	39.5	39.3	**42.6 ± 2.1**
hopper-random	5.3	8.5	25.3	31.7	17.9	6.9	25.4	**31.9**	**32.2 ± 0.3**
walker-random	5.4	1.6	16.6	7.4	7.0	5.9	0.0	17.9	**20.6 ± 2.9**
halfcheetah-m	46.9	48.3	65.9	73.0	54.2	46.9	**77.9**	74.6	**77.4 ± 0.6**
hopper-medium	61.9	59.3	101.6	62.8	97.2	67.4	87.0	**106.6**	**106.2 ± 1.2**
walker-medium	79.5	83.7	92.5	84.1	81.9	81.3	84.9	87.7	**97.2 ± 1.3**
halfcheetah-m-r	45.3	44.6	61.3	72.1	55.1	44.1	68.7	71.7	**73.8 ± 1.5**
hopper-m-replay	86.3	60.9	101.0	103.5	89.5	99.4	99.5	103.9	**105.1 ± 0.7**
walker-m-replay	76.8	81.8	87.1	85.6	56.0	82.6	89.2	89.9	**95.2 ± 0.3**
halfcheetah-m-e	95.0	90.7	106.3	90.8	90.0	95.0	95.4	**108.2**	103.8 ± 2.3
hopper-m-expert	96.9	98.0	110.7	81.6	111.1	110.0	88.2	112.6	**113.7 ± 0.6**
walker-m-expert	109.1	110.1	114.7	112.9	103.3	101.9	56.7	**115.2**	**115.4 ± 0.8**
Average	61.6	58.2	76.0	70.3	66.8	62.3	67.7	80.0	**82.0**
Solved tasks	3/12	2/12	5/12	2/12	3/12	4/12	2/12	5/12	**7/12**

four tasks in Table 9, 10 and 11 in Appendix. It shows that the results are relatively robust to the number of negative samples within a reasonable range.

5. Related Works

Transition Modeling. The standard approach to represent transition models is to use probabilistic feedforward models to output the state prediction mean and variance (Chua et al., 2018), modeled as a diagonal Gaussian distribution. Later Zhang et al. proposed to use autoregressive models to represent transition models. Janner et al. use a transformer architecture to model the trajectory distribution, where the transition information is encapsulated coupled with policy. For the learning principle, most methods directly train the model by likelihood maximization. Besides, Xu et al. shows that adversarial model learning address the compounding error issue. Luo et al. proposes to learn a generalizable dynamics reward by inverse reinforcement learning method. Chen et al. proposes to train the transition models conditioned on policies.

Replay buffers are also regarded as non-parametric transition models (Fedus et al., 2020), which remember all the transition experiences available to the agents. Recent researches mainly focus on how to replay the experiences (Schaul et al., 2016; Sinha et al., 2022; Liu et al., 2021b; Chen et al., 2024b) to boost the policy learning.

Offline Model-based RL. Offline model-based reinforcement learning algorithms (Luo et al., 2024b; Moerland et al., 2023) leverage transition dynamics models learned from offline experiences to generate rollout data, facilitating policy optimization. In order to mitigate the impact of model errors, many recent works (Yu et al., 2020; 2021; Sun et al., 2023; Rigter et al., 2022) incorporate conservatism into learning algorithms. Some algorithms (Yu et al., 2020; Sun et al., 2023) utilize uncertainty estimation to trust states with low uncertainty, while some methods (Yu et al., 2021) try to limit the policy to acting surrounding the dataset. (Chen et al., 2021) introduced contextual meta-policy learning in models to enable generalization to unseen situations.

Off-policy Evaluation. The previous works on OPE include methods based on fitted q-evaluation (Le et al., 2019), importance sampling (Kostrikov & Nachum, 2020; Yang et al., 2020), doubly robust method (Jiang & Li, 2016), and model-based rollout (direct method) (Fu et al., 2021; Zhang et al., 2021). Some other model-based methods consider combining the model-based rollout and fitted value estimate (Thomas & Brunskill, 2016; Hanna et al., 2017; Jin et al., 2022), which may be combined with our ETMs.

Energy-based Models. Energy learning for distribution modeling (LeCun et al., 2006; Song & Kingma, 2021) has a rich history in machine learning. Langevin MCMC (Welling & Teh, 2011; Neal et al., 2011) sampling is often used for training and implicit inference (Du & Mordatch, 2019). Energy-based models have drawn much attention in computer vision, especially for generative image modeling (Xie et al., 2016; Gao et al., 2018; Du & Mordatch, 2019; Grathwohl et al., 2019). Some works also explore conditional energy-based models as a formulation for probabilistic regression (Gustafsson et al., 2020b), demonstrating particularly impressive performance on vision tasks (Bhat et al., 2019; Gustafsson et al., 2020a). Recently, energy-based modeling has also been applied to imitation learning (Liu et al., 2021a; Florence et al., 2021; Qin et al., 2023) and

planning (Du et al., 2020; Xu et al., 2022). Our work is an attempt to apply the energy modeling method to transition dynamics learning, showcasing flexible transition modeling ability and better generalization.

6. Conclusion and Limitation

This paper shows the promise of energy-based transition models (ETM) in learning transition dynamics for offline control tasks, particularly their ability to capture the prevalent discontinuous transition behaviors found in real-world environments. Empirical results demonstrate that ETMs can better generalize to out-of-distribution data and achieve great improvement in off-policy evaluation tasks. We also showcase the efficacy of ETM to improve model-based policy optimization in offline reinforcement learning tasks.

One limitation of our method is the requirement for iterative gradient-based sampling or search during inference, which leads to more time cost than standard feedforward models. For instance, ETMs require approximately five times more inference time than standard FTMs, and employing ETMs in EMPO results in an approximately 40% increase in the overall learning time compared to MOPO. This limitation can be relieved by advanced sampling techniques or by using parameterized models to avoid the iterative search.

Acknowledgements

This work is supported by National Science Foundation of China (61921006). We thank the anonymous reviewers for their helpful suggestions on improving the paper.

Impact Statement

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

References

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations*, 2023.

An, G., Moon, S., Kim, J., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Advances in Neural Information Processing Systems*, 2021.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Asadi, K., Misra, D., Kim, S., and Littman, M. L. Combating the compounding-error problem with a multi-step model. *CoRR*, abs/1905.13320, 2019.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Bengio, E., Pineau, J., and Precup, D. Interference and generalization in temporal difference learning. In *Proceedings of 37th International Conference on Machine Learning*, 2020.

Bhat, G., Danelljan, M., Gool, L. V., and Timofte, R. Learning discriminative model prediction for tracking. In *International Conference on Computer Vision*, 2019.

Chen, R., Chen, X.-H., Sun, Y., Xiao, S., Li, M., and Yu, Y. Policy-conditioned environment models are more generalizable. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=g9mYBdooPA.

Chen, R., Liu, X.-H., Liu, T.-S., Jiang, S., Xu, F., and Yu, Y. Foresight distribution adjustment for off-policy reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2024b.

Chen, X., Yu, Y., Li, Q., Luo, F., Qin, Z. T., Shang, W., and Ye, J. Offline model-based adaptable policy learning. In *Advances in Neural Information Processing Systems*, 2021.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 2018.

Coumans, E. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, pp. 1. 2015.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, 2019.

Du, Y., Lin, T., and Mordatch, I. Model-based planning with energy-based models. In *Conference on Robot Learning*, 2020.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. In *Conference on Robot Learning*, 2021.

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

Fu, J., Norouzi, M., Nachum, O., Tucker, G., Wang, Z., Novikov, A., Yang, M., Zhang, M. R., Chen, Y., Kumar, A., et al. Benchmarks for deep off-policy evaluation. *CoRR*, abs/2103.16596, 2021.

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

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Gao, R., Lu, Y., Zhou, J., Zhu, S.-C., and Wu, Y. N. Learning generative convnets via multi-grid modeling and sampling. In *the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2019.

Gustafsson, F. K., Danelljan, M., Bhat, G., and Schön, T. B. Energy-based models for deep probabilistic regression. In *Computer Vision - ECCV - European Conference*, 2020a.

Gustafsson, F. K., Danelljan, M., Timofte, R., and Schön, T. B. How to train your energy-based model for regression. *CoRR*, abs/2005.01698, 2020b.

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

Hanna, J., Stone, P., and Niekum, S. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *AAAI Conference on Artificial Intelligence*, 2017.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Jin, X.-K., Liu, X.-H., Jiang, S., and Yu, Y. Hybrid value estimation for off-policy evaluation and offline reinforcement learning. *CoRR*, abs/2206.02000, 2022.

Jolicoeur-Martineau, A. and Mitliagkas, I. Gradient penalty from a maximum margin perspective. *CoRR*, abs/1910.06922, 2019.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Kostrikov, I. and Nachum, O. Statistical bootstrapping for uncertainty estimation in off-policy evaluation. *CoRR*, abs/2007.13609, 2020.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2021.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing*, 2020.

Le, H., Voloshin, C., and Yue, Y. Batch policy learning under constraints. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

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

Liu, J. S. and Liu, J. S. *Monte Carlo strategies in scientific computing*, volume 75. Springer, 2001.

Liu, M., He, T., Xu, M., and Zhang, W. Energy-based imitation learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021a.

Liu, X.-H., Xue, Z., Pang, J., Jiang, S., Xu, F., and Yu, Y. Regret minimization experience replay in off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021b.

Liu, X.-H., Xu, F., Zhang, X., Liu, T., Jiang, S., Chen, R., Zhang, Z., and Yu, Y. How to guide your learner: Imitation learning with active adaptive expert involvement. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*, 2023.

Lu, C., Ball, P., Parker-Holder, J., Osborne, M., and Roberts, S. J. Revisiting design choices in offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2022.

Luo, F.-M., Xu, T., Cao, X., and Yu, Y. Reward-consistent dynamics models are strongly generalizable for offline reinforcement learning. In *International Conference on Learning Representations*, 2024a.

Luo, F.-M., Xu, T., Lai, H., Chen, X.-H., Zhang, W., and Yu, Y. A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2):121101, 2024b.

Moerland, T. M., Broekens, J., Plaat, A., Jonker, C. M., et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1): 1–118, 2023.

Neal, R. M. et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

Pfrommer, S., Halm, M., and Posa, M. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. In *Conference on Robot Learning*, 2021.

Qin, A., Gao, F., Li, Q., Zhu, S.-C., and Xie, S. Learning non-markovian decision-making from state-only sequences. In *Advances in Neural Information Processing Systems*, 2023.

Qin, R.-J., Zhang, X., Gao, S., Chen, X.-H., Li, Z., Zhang, W., and Yu, Y. Neorl: A near real-world benchmark for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Rigter, M., Lacerda, B., and Hawes, N. RAMBO-RL: robust adversarial model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

Sinha, S., Song, J., Garg, A., and Ermon, S. Experience replay with likelihood-free importance weights. In *Learning for Dynamics and Control Conference*, 2022.

Song, Y. and Kingma, D. P. How to train your energy-based models. *CoRR*, abs/2101.03288, 2021.

Sun, Y. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. https://github.com/yihaosun1124/OfflineRL-Kit, 2023.

Sun, Y., Zhang, J., Jia, C., Lin, H., Ye, J., and Yu, Y. Model-bellman inconsistency for model-based offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

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

Thomas, P. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Todorov, E. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *IEEE International Conference on Robotics and Automation*, 2014.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Wen, J., Dai, B., Li, L., and Schuurmans, D. Batch stationary distribution estimation. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Xie, J., Lu, Y., Zhu, S.-C., and Wu, Y. A theory of generative convnet. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments. In *Advances in Neural Information Processing Systems*, 2020.

Xu, Y., Xie, J., Zhao, T., Baker, C., Zhao, Y., and Wu, Y. N. Energy-based continuous inverse optimal control. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Yang, M., Nachum, O., Dai, B., Li, L., and Schuurmans, D. Off-policy evaluation via the regularized lagrangian. In *Advances in Neural Information Processing Systems*, 2020.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. COMBO: conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2021.

Zhang, M. R., Paine, T., Nachum, O., Paduraru, C., Tucker, G., Norouzi, M., et al. Autoregressive dynamics models for offline policy evaluation and optimization. In *International Conference on Learning Representations*, 2021.

A. A simplistic case of Interference

Interference (Bengio et al., 2020) can be characterized in the first order by the inner product of the objectives' gradients:

$$\rho_{1,2} = \nabla_\theta L^\top(x_1)\nabla_\theta L(x_2), \tag{11}$$

where x_1, x_2 correspond to two different data points and L is a point-wise loss function. If $\rho_{1,2} > 0$, minimizing the loss at one point x_1 by a gradient step also decreases the loss at the other point x_2, resulting in a constructive generalization. However, if $\rho_{1,2} < 0$, gradient descent at x_1 will increase the loss $L(x_2)$, which is the destructive interference.

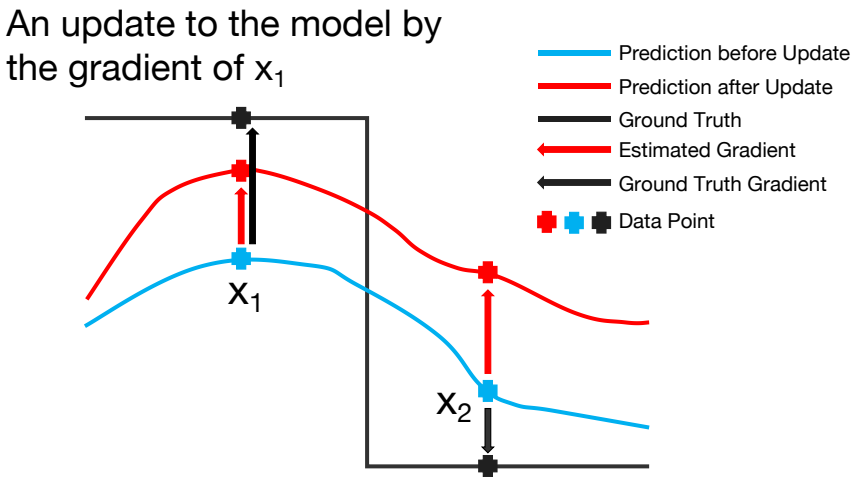


Figure 5. The interference in a simple value jump case.

A simplistic case is the sharp value jump in the groundtruth function, where x_1 and x_2 are located on opposite sides of this jump as illustrated in Figure 5. Let f_θ be the forward model and y_i be the true target of x_i. Using mean squared error loss, we have

$$\rho_{1,2} = (f_\theta(x_1) - y_1)(f_\theta(x_2) - y_2)\nabla_\theta f_\theta(x_1)^\top \nabla_\theta f_\theta(x_2). \tag{12}$$

It is often the case that the model prediction lies between the two jump values, exhibiting an averaging effect during training, and therefore the product $(f_\theta(x_1) - y_1)(f_\theta(x_2) - y_2) < 0$. Considering forward models f_θ biased towards smooth regressors, we assume that f_θ has Lipschitz gradients for some Lipschitz constant $L > 0$, yielding that

$$2\nabla_\theta f_\theta(x_1)^\top \nabla_\theta f_\theta(x_2) \geq \|\nabla_\theta f_\theta(x_1)\|^2 + \|\nabla_\theta f_\theta(x_2)\|^2 - L^2\|x_1 - x_2\|^2. \tag{13}$$

For x_1, x_2 very close, the negative term tends to be small relative to the gradient norm, and therefore the inner product of the model output gradients at x_1 and x_2 is greater than a positive. Thereby $\rho_{1,2} < 0$ in this case and results in the negative interference. Similar effects also occur in more complex cases, often to a greater extent.

B. Derivation details of the connection between InfoNCE and Maximum Likelihood.

For the energy function $E_\theta(x, y)$, its induced implicit probability distribution

$$p(y|x;\theta) = \frac{\exp(-E_\theta(x,y))}{Z_\theta(x)}, \tag{14}$$

where the normalizing factor $Z_\theta(x) = \int \exp(-E_\theta(x,y))\mathrm{d}y$ is intractable and thereby the probability likelihood cannot be directly obtained. Fortunately, we can instead compute the gradient of the negative log-likelihood:

$$\nabla_\theta - \log p(y|x;\theta) = \nabla_\theta E_\theta(x,y) + \nabla_\theta \log Z_\theta(x) \tag{15}$$

$$= \nabla_\theta E_\theta(x,y) - \frac{1}{Z_\theta(x)} \int \exp(-E_\theta(x,y)) \nabla_\theta E_\theta(x,y) \mathrm{d}y \tag{16}$$

$$= \nabla_\theta E_\theta(x,y) - \mathbb{E}_{p(y|x;\theta)} \nabla_\theta E_\theta(x,y), \tag{17}$$

where the intractable $Z_\theta(x)$ is hidden in the expectation w.r.t. $p(y|x;\theta)$, which requires sampling techniques to estimate the gradient. Therefore, gradient descent according to the negative log-likelihood is equivalent to minimizing the objective:

$$l(x_i, y_i; \theta) = E_\theta(x_i, y_i) - \mathbb{E}_{\mathrm{sg}[p(y|x_i;\theta)]} E_\theta(x_i, y), \tag{18}$$

where sg denotes the stop gradient operator, meaning that we use $p(y|x,;\theta)$ to compute the expectation but do not consider its dependence on θ. Efficient MCMC methods can be utilized for approximate sampling, and direct substitution gives Contrastive Divergence objective (Hinton, 2002). However, there are usually approximation errors between the MCMC sampling distribution $\hat{p}(y|x;\theta)$ and the real $p(y|x;\theta)$ for finite simulation steps. Here we explicitly take into account this approximation error. If $D_{\mathrm{KL}}(\hat{p}(\cdot|x;\theta), p(\cdot|x;\theta)) \leq \epsilon$, then the objective 18 can be upper bounded by

$$\max_{\bar{p}: D_{\mathrm{KL}}(\bar{p},\hat{p}) \leq \epsilon} E_\theta(x_i, y_i) - \hat{\mathbb{E}}_{\bar{p}(y|x_i)} E_\theta(x_i, y), \tag{19}$$

which therefore serves as a surrogate objective to be minimized. According to KKT conditions in convex optimization, this constraint optimization with respect to distribution $\bar{p}$ is equivalent to minimizing the Lagrange dual function for some Lagrange multiplier $\lambda > 0$:

$$\max_{\bar{p}} E_\theta(x_i, y_i) - \hat{\mathbb{E}}_{\bar{p}(y|x_i)} E_\theta(x_i, y) - \lambda D_{\mathrm{KL}}(\bar{p}, \hat{p}), \tag{20}$$

which can be solved in a closed form:

$$\bar{p}(y|x) \propto \hat{p}(y|x) \exp(-\frac{1}{\lambda} E_\theta(x,y)). \tag{21}$$

Substitute this solution back into 20, we obtain

$$E_\theta(x_i, y_i) + \lambda \log \mathbb{E}_{\hat{p}(y|x_i;\theta)} \exp\Big(-\frac{1}{\lambda} E_\theta(x_i, y)\Big) = -\lambda \log \frac{\exp\Big(-\frac{1}{\lambda} E_\theta(x_i, y_i)\Big)}{\mathbb{E}_{\hat{p}} \exp\Big(-\frac{1}{\lambda} E_\theta(x_i, y)\Big)}.$$

C. Implementation Details

C.1. Implementation Details of Energy-based Transition Model Learning

We use 4-layer MLP for both the energy function of ETMs and the forward transition models. The loss function is composed of the InfoNCE loss and gradient penalty term:

$$l_{\mathrm{InfoNCE}} + l_{\mathrm{GradPen}} = -\log \frac{\exp\left(-E_\theta(s,a,s')\right)}{\sum_{j=1}^K \exp\left(-E_\theta(s,a,s'_j)\right)} + \sum_{j=1}^K \max\left(0, (\|\nabla_{s'} E_\theta(s,a,s'_j)\| - M)\right)^2, \tag{22}$$

where M controls the scale of the gradient, and we use $M = 5$ across all experiments. One of the negative samples $\{s'_j\}_{j=1}^K$ is chosen to be s', and other $K-1$ are generated by Langevin sampling:

$$\hat{s}'_{k+1} \leftarrow \hat{s}'_k - \frac{\epsilon^2}{2} \nabla_{s'} E_\theta(s,a,\hat{s}'_k) + \epsilon z_k. \tag{23}$$

where the ϵ is Langevin stepsize and z_k is Gaussian noises whose scale can be controlled in practice. We use 1e-3 stepsize and 0.5 Gaussian variance during training. An abused notation is that we in fact predict the state change δs in our implementation for continuous control instead of the next state s', so the true next state prediction is $s' = s + \delta s$. The hyperparameters are listed in Table 2. The model errors tested on holdout data are reported in Table 8 for D4RL mujoco environments.

Hyperparamter	Value
Training iterations	300
Batch size	1024
Learning rate	1e-3
Energy network hidden	[200, 200, 200, 200]
Energy network activation	ReLU
Optimizer	Adam
Negative sample number	16
Langevin steps	50
Langevin noise	0.5
Langevin stepsize	1e-3
Langevin delta clip	0.5
Gradient penalty margin	5

Table 2. Hyperparameters for energy-based transition model learning.

C.2. Implementation Details of Off-policy Evaluation with Direct Model Rollout

Model-based off-policy evaluation methods contain direct methods, that simply simulate the policy trajectories within the dynamics model (Fu et al., 2021; Zhang et al., 2021), and hybrid methods, that combine fitted value function and simulation experience(Thomas & Brunskill, 2016; Hanna et al., 2017; Jin et al., 2022). In this work, we mainly focus on the transition model learning and evaluate the learned models in OPE tasks. Therefore we simply use the direct method to roll out the target policies to estimate the policy values, as summarized in algorithm 2. It is possible to combine the hybrid methods with our energy-based transition model to obtain even better performance.

In practical evaluation, we use $\gamma = 0.995$ and $N = 10$. Max horizon length H is 1000 for Gym-Mujoco and 200 for Adroit. During energy model inference, we use 100 Langevin steps and 0.1 Langevin noise for accurate prediction. For Mujoco tasks, the reward function is learned along with the state transition by adding an extra dimension in the ETM. For Adroit tasks, we find incorporating a separately learned reward model is more beneficial, and clipping the model prediction according to the numerical range of the offline training dataset also helps for both FTMs and ETMs.

Algorithm 2 Off-policy Evaluation with Direct Model Rollout

Require: Transition model P_θ learned on offline dataset $\mathcal{D}$ (maybe standard FTMs or ETMs), policy π to be evaluated, number of trajectories N, initial state distribution $\mathcal{S}_0$, discount factor γ, horizon length H.
for $i = 1$ **to** N **do**
  $R_i = 0$
  Sample initial state $s_0 \sim \mathcal{S}_0$
  for $t = 0$ **to** $H - 1$ do **do**
    $a_t \sim \pi(\cdot|s_t)$
    $s_{t+1}, r_t \sim P_\theta(\cdot|s_t, a_t)$
    $R_i = R_i + \gamma^t r_t$
  end for
end for
return $\frac{1}{N}\sum_{i=1}^{N} R_i$

C.3. Implementation Details of Offline RL with Energy-based Transition Model

We use an ensemble of five ETMs for policy optimization, and each step we randomly pick one of the five models to generate transitions. The proposed EMPO uses Soft-Actor-Critic (SAC) (Haarnoja et al., 2018) as the base policy optimization algorithm and adopts an uncertainty estimate of the model predicted next-state as the reward penalty to introduce conservatism:

$$\mathcal{U}(s, a; \{P_{\theta_i}\}_{i=1\sim5}) = \mathrm{std}(\{\hat{s}' \sim P_{\theta_i}(\cdot|s, a)\}). \tag{24}$$

This form of penalty can be seen as the ensemble-std version (Lu et al., 2022) of MOPO (Yu et al., 2020), adapting to the energy-based models where the variance is not directly available but estimated by samples.

Most hyperparameters of SAC follow its standard implementations. For each update, we sample a batch size of 256 transitions where 5% of them is from the real dataset $\mathcal{D}$ and another 95% is from the synthetic dataset $\mathcal{D}_{model}$. We use the base hyperparameter settings in Table 3 for all the Gym-Mujoco tasks. Two hyperparameters, penalty coefficient β and rollout length h, are tuned for each task and we list them in Table 4.

We use the results of all other baselines reported in (Sun et al., 2023).

C.4. Details of the transition in didactic example

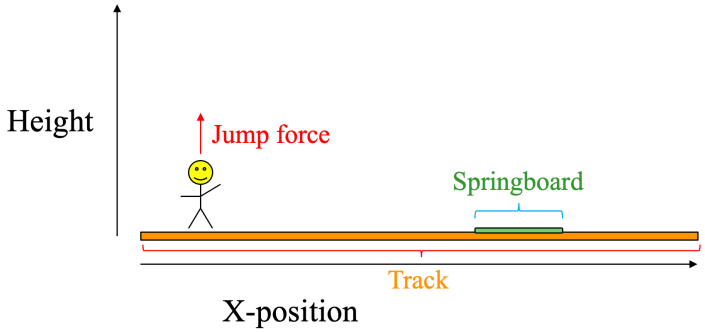


Figure 6. A schematic diagram of the didactic environment.

Our setup involves creating a track with a normalized length ranging from 0 to 1, which is shown in Figure 6. Positioned between 0.65 and 0.75 along the track, there exists a springboard. The action space extends from 0 to 1, representing the force applied for jumping. Beyond the springboard area, a force greater than 0.8 can achieve a specific height, while within the springboard area, a force of 0.2 or greater is sufficient for jumping. When the force is within the threshold of 0.2, the jump height remains constant. However, beyond this threshold, it follows a quadratic function. We conclude the relationship between the current position x, action a and the next height h:

$$h = \begin{cases} \max(10(a-0.2)^2, 0.4) \times \mathbb{I}(a \geq 0.2), & 0.65 \leq x \leq 0.75 \\ \max(10(a-0.8)^2, 0.4) \times \mathbb{I}(a \geq 0.8), & \text{others} \end{cases}$$

D. Details of OPE

D.1. Metrics

The metrics we use in OPE experiments follow DOPE benchmark (Fu et al., 2021):

Absolute Error The absolute error is defined as the difference between the value and estimated value of a policy:

$$\text{AbsErr} = |V^\pi - \hat{V}^\pi|, \tag{25}$$

where V^π is the true value of the policy and $\hat{V}^\pi$ is the estimated value of the policy.

Rank correlation Rank correlation measures the correlation between the ordinal rankings of the value estimates and the true values, which can be written as:

$$\text{RankCorr} = \frac{\text{Cov}(V^\pi_{1:N}, \hat{V}^\pi_{1:N})}{\sigma(V^\pi_{1:N})\sigma(\hat{V}^\pi_{1:N})}, \tag{26}$$

where $1:N$ denotes the indices of the evaluated policies.

Hyperparamter	Value
Total gradient steps	3M
Model ensemble number	5
Critic number	2
Q network hidden	[256, 256]
policy network hidden	[256, 256]
Target Q smoothing coefficient	5e-3
Discount factor	0.99
Batch size	256
Q learning rate	3e-4
Actor learning rate	1e-4
Optimizer	Adam
Ratio of real experiences	0.05
Langevin inference step	30
Langevin inference noise	0.5

Table 3. Hyperparameters for policy optimization.

Task	Penalty coefficient	Rollout length
hopper-random	15	5
hopper-medium	10	5
hopper-medium-replay	10	5
hopper-medium-expert	15	5
walker-random	2.5	5
walker-medium	2.5	5
walker-medium-replay	1.0	5
walker-medium-expert	5.0	1
halfcheetah-random	0.5	5
halfcheetah-medium	2.5	5
halfcheetah-medium-replay	1.5	10
halfcheetah-medium-expert	2.5	5

Table 4. Tuned Hyperparameters for EMPO.

Regret@k Regret@k is the difference between the value of the best policy in the entire set, and the value of the best policy in the top-k set (where the top-k set is chosen by estimated values). It can be defined as:

$$\text{Regret @k} = \max_{i \in 1:N} V_i^\pi - \max_{j \in \text{topk}(1:N)} V_j^\pi, \tag{27}$$

where topk$(1:N)$ denotes the indices of the top K policies as measured by estimated values $\hat{V}^\pi$. We use regret@1 in both Gym-Mujoco and Adroit environments.

D.2. Detailed Results

We report the raw absolute error, rank correlation and regret@1 for each OPE method and each OPE task in Table 5, Table 6 and Table 7, respectively. The results of FQE (Le et al., 2019), DR (Jiang & Li, 2016), IS (Kostrikov & Nachum, 2020), DICE (Yang et al., 2020), and VPM (Wen et al., 2020) come from DOPE benchmark (Fu et al., 2021).

Table 5. Raw absolute error for each OPE method, averaged over 3 seeds.

Env.	Level	FQE	DR	IS	DICE	VPM	FTM	ETM
ant	expert	**583 ± 122**	**584 ± 114**	**605 ± 104**	**558 ± 108**	**607 ± 108**	626 ± 12	**533 ± 23**
	m-expert	**319 ± 67**	**326 ± 66**	604 ± 102	**471 ± 100**	604 ± 106	522 ± 14	**379 ± 28**
	medium	**345 ± 64**	**345 ± 66**	594 ± 104	**495 ± 90**	570 ± 109	538 ± 38	**421 ± 18**
	m-replay	410 ± 79	421 ± 72	603 ± 101	583 ± 110	612 ± 105	423 ± 51	**175 ± 37**
	random	**398 ± 111**	**404 ± 106**	606 ± 103	530 ± 92	570 ± 99	445 ± 21	**318 ± 53**
hopper	expert	282 ± 76	426 ± 99	**106 ± 29**	259 ± 54	442 ± 43	295 ± 128	**71 ± 16**
	m-expert	252 ± 28	234 ± 77	360 ± 47	266 ± 40	-	313 ± 130	**32 ± 4**
	medium	283 ± 73	307 ± 85	405 ± 48	215 ± 41	433 ± 44	303 ± 22	**47 ± 21**
	m-replay	295 ± 7	298 ± 14	438 ± 11	398 ± 2	-	76 ± 22	**29 ± 8**
	random	261 ± 42	289 ± 50	412 ± 45	**122 ± 16**	438 ± 44	324 ± 19	236 ± 15
walker2d	expert	**453 ± 142**	519 ± 79	**405 ± 62**	437 ± 60	**367 ± 68**	458 ± 43	**364 ± 7**
	m-expert	233 ± 42	217 ± 46	436 ± 62	322 ± 60	425 ± 61	446 ± 59	**152 ± 9**
	medium	350 ± 79	368 ± 74	428 ± 60	273 ± 31	426 ± 60	393 ± 108	**159 ± 13**
	m-replay	313 ± 73	296 ± 54	427 ± 60	374 ± 51	424 ± 64	358 ± 85	**132 ± 31**
	random	**354 ± 73**	**347 ± 74**	430 ± 61	419 ± 57	440 ± 58	466 ± 27	**339 ± 10**
halfcheetah	expert	1031 ± 95	1025 ± 95	1404 ± 152	944 ± 161	945 ± 161	1087 ± 197	**758 ± 116**
	m-expert	1014 ± 101	1015 ± 103	1400 ± 146	1078 ± 132	1427 ± 111	**1184 ± 421**	**689 ± 203**
	medium	1211 ± 130	1222 ± 134	1217 ± 123	1382 ± 130	1374 ± 153	969 ± 66	**655 ± 114**
	m-replay	1003 ± 132	1001 ± 129	1409 ± 154	1440 ± 158	1384 ± 148	1009 ± 76	**727 ± 119**
	random	**938 ± 125**	**949 ± 126**	1405 ± 155	1446 ± 156	1411 ± 154	1001 ± 105	**842 ± 42**
door	human	**389 ± 60**	**379 ± 65**	870 ± 173	1108 ± 199	862 ± 163	628 ± 106	**325 ± 79**
	cloned	**438 ± 81**	**424 ± 73**	891 ± 188	697 ± 79	1040 ± 188	951 ± 26	**575 ± 77**
	expert	1343 ± 84	1353 ± 218	648 ± 122	856 ± 134	879 ± 182	949 ± 22	**297 ± 101**
pen	human	2872 ± 170	2846 ± 200	3926 ± 128	4193 ± 244	1569 ± 215	2482 ± 173	**695 ± 31**
	cloned	**1232 ± 105**	**1323 ± 98**	1707 ± 128	**1454 ± 219**	2324 ± 129	**1493 ± 382**	**1113 ± 241**
	expert	**1057 ± 281**	2013 ± 564	4547 ± 222	2963 ± 279	2325 ± 136	2535 ± 174	**1007 ± 169**
hammer	human	**6000 ± 612**	**5768 ± 751**	7352 ± 1118	**5677 ± 936**	7105 ± 1107	7599 ± 117	**5905 ± 40**
	cloned	**5415 ± 558**	6101 ± 679	7403 ± 1126	**4169 ± 839**	7459 ± 1114	7545 ± 131	6405 ± 698
	expert	**2950 ± 728**	**3485 ± 590**	**3052 ± 608**	**3963 ± 758**	7312 ± 1117	7570 ± 126	4763 ± 80
relocate	human	**593 ± 113**	**606 ± 116**	**638 ± 217**	4526 ± 474	806 ± 166	681 ± 48	**529 ± 29**
	cloned	**439 ± 125**	**412 ± 124**	**632 ± 215**	1347 ± 485	**586 ± 135**	648 ± 42	**552 ± 20**
	expert	1351 ± 393	1193 ± 350	2731 ± 147	1095 ± 221	**620 ± 214**	**639 ± 63**	**594 ± 28**

Table 6. Rank correlation for each OPE method, averaged over 3 seeds.

Env.	Level	FQE	DR	IS	DICE	VPM	FTM	ETM
ant	expert	-0.13±0.32	-0.28±0.32	**0.14±0.41**	-0.13±0.37	-0.42±0.38	**0.14±0.27**	**0.43±0.13**
	m-expert	0.37±0.35	0.35±0.35	-0.21±0.35	-0.33±0.4	-0.28±0.28	-0.10±0.21	**0.80±0.07**
	medium	**0.65±0.25**	**0.66±0.26**	-0.17±0.32	-0.36±0.28	-0.2±0.31	0.29±0.09	**0.75±0.04**
	m-replay	0.57±0.28	0.45±0.32	0.07±0.39	-0.24±0.39	-0.26±0.29	0.74±0.06	**0.94±0.03**
	random	0.04±0.33	0.01±0.33	0.26±0.34	-0.21±0.35	0.24±0.31	**0.52±0.23**	**0.76±0.03**
hopper	expert	-0.33±0.30	-0.41±0.27	0.37±0.27	-0.08±0.32	0.21±0.32	0.46±0.11	**0.85±0.05**
	m-expert	0.01±0.08	-0.08±0.30	0.35±0.26	0.08±0.14	-	0.54±0.3	**0.95±0.01**
	medium	-0.29±0.33	-0.31±0.34	-0.55±0.26	0.19±0.33	0.13±0.37	0.58±0.21	**0.94±0.04**
	m-replay	0.45±0.13	0.05±0.17	-0.16±0.03	0.27±0.28	-0.16±0.03	0.91±0.03	**0.97±0.02**
	random	-0.11±0.36	-0.19±0.36	**0.23±0.34**	-0.13±0.39	-0.46±0.20	0.35±0.06	**0.61±0.15**
walker2d	expert	**0.35±0.33**	**0.26±0.34**	**0.22±0.37**	-0.37±0.27	**0.17±0.32**	**-0.05±0.51**	**0.54±0.11**
	m-expert	**0.25±0.32**	0.19±0.33	**0.24±0.33**	-0.34±0.34	**0.49±0.37**	0.21±0.13	**0.67±0.14**
	medium	-0.09±0.36	0.02±0.37	-0.25±0.35	0.12±0.38	0.44±0.21	**0.37±0.30**	**0.78±0.12**
	m-replay	-0.19±0.36	-0.37±0.39	**0.65±0.24**	**0.55±0.23**	-0.52±0.25	0.14±0.42	**0.77±0.10**
	random	**0.21±0.31**	**0.16±0.29**	**-0.05±0.38**	**-0.19±0.36**	**-0.42±0.34**	**-0.04±0.35**	**-0.12±0.32**
halfcheetah	expert	**0.78±0.15**	**0.77±0.17**	0.01±0.35	-0.44±0.30	0.18±0.35	**0.51±0.55**	**0.81±0.10**
	m-expert	**0.62±0.27**	**0.62±0.27**	-0.06±0.37	-0.08±0.35	-0.47±0.29	0.21±0.13	**0.91±0.03**
	medium	0.34±0.17	0.32±0.32	**0.80±0.11**	-0.26±0.07	-	0.37±0.30	**0.78±0.12**
	m-replay	0.26±0.37	**0.32±0.37**	**0.59±0.26**	-0.15±0.41	-0.07±0.36	**0.71±0.13**	**0.77±0.10**
	random	-0.11±0.41	-0.02±0.38	-0.24±0.36	-0.70±0.22	0.27±0.36	**0.75±0.10**	**0.76±0.10**
door	human	0.07±0.09	0.01±0.18	-0.12±0.35	-0.02±0.20	-	**0.90±0.03**	**0.89±0.02**
	cloned	0.55±0.27	0.60±0.28	0.66±0.22	0.18±0.31	-0.29±0.36	0.67±0.21	**0.90±0.01**
	expert	**0.89±0.09**	**0.76±0.13**	**0.76±0.17**	-0.06±0.32	**0.65±0.23**	0.06±0.59	**0.91±0.03**
pen	human	-0.31±0.21	-0.36±0.29	**0.28±0.28**	**0.17±0.33**	-	**0.35±0.14**	**0.59±0.19**
	cloned	0.06±0.42	**0.39±0.25**	**0.71±0.08**	-0.07±0.26	-	0.27±0.16	**0.49±0.17**
	expert	**-0.01±0.33**	**0.52±0.28**	-0.45±0.31	-0.53±0.30	**0.08±0.33**	-0.04±0.12	**0.48±0.29**
hammer	human	0.14±0.10	-0.04±0.25	**0.39±0.07**	**0.11±0.18**	-	**-0.20±0.68**	**0.50±0.21**
	cloned	-0.15±0.33	-0.70±0.20	**0.58±0.27**	**0.35±0.38**	-0.77±0.22	**0.31±0.36**	**0.45±0.09**
	expert	**0.29±0.34**	**0.49±0.31**	**0.64±0.24**	-0.42±0.31	**0.39±0.31**	**0.41±0.14**	**0.58±0.12**
relocate	human	**0.62±0.11**	**0.65±0.19**	-0.23±0.07	-0.23±0.16	-	-0.67±0.11	**0.58±0.13**
	cloned	0.15±0.17	0.10±0.16	-0.22±0.18	0.22±0.16	-	0.33±0.26	**0.64±0.23**
	expert	-0.57±0.28	-0.40±0.24	**0.52±0.23**	-0.27±0.34	**0.39±0.31**	**0.16±0.31**	**0.47±0.28**

E. Ablation Experiments

In the results of our main experiments, we use 16 negative samples in the ETM training for tasks. To ablate the influence of the number of negative samples, we also report the OPE results of the learned ETM using 8 or 24 negative samples during training for four tasks in Table 9, 10 and 11, where we see that the results are relatively robust to the number of negative samples within a reasonable range.

F. Additional Experiment Results on NeoRL benchmark

We also conduct the EMPO on several tasks in NeoRL benchmark (Qin et al., 2022) and report the results in Table 13. The datasets in NeoRL tasks are all collected by conservative behavior policies and therefore have relatively narrow coverage. This fact would be especially unfavorable for model-based methods because the out-of-distribution ability of the learned transition models is limited by the narrow data coverage, detrimental to policy optimization. Consequently, we see that MOPO achieves significantly inferior performance compared to model-free methods, including naive behavior cloning. Our EMPO learns decent policies comparable to those model-free SOTA methods.

Table 7. Regret@1 for each OPE method, averaged over 3 seeds.

Env.	Level	FQE	DR	IS	DICE	VPM	FTM	ETM
ant	expert	**0.43 ± 0.22**	**0.43 ± 0.22**	**0.47 ± 0.32**	0.62 ± 0.15	0.88 ± 0.22	**0.34 ± 0.32**	**0.14 ± 0.12**
	m-expert	**0.36 ± 0.14**	**0.37 ± 0.13**	0.46 ± 0.18	0.60 ± 0.16	**0.32 ± 0.24**	**0.59 ± 0.23**	**0.17 ± 0.10**
	medium	**0.12 ± 0.18**	**0.12 ± 0.18**	0.61 ± 0.18	0.43 ± 0.1	0.4 ± 0.21	0.37 ± 0.12	**0.08 ± 0.03**
	m-replay	**0.05 ± 0.09**	**0.05 ± 0.09**	**0.16 ± 0.23**	0.64 ± 0.13	0.72 ± 0.43	0.21 ± 0.10	**0.05 ± 0.05**
	random	**0.28 ± 0.15**	**0.28 ± 0.15**	0.56 ± 0.22	**0.50 ± 0.29**	**0.15 ± 0.24**	**0.39 ± 0.21**	**0.18 ± 0.12**
hopper	expert	0.41 ± 0.20	**0.34 ± 0.35**	**0.06 ± 0.03**	0.20 ± 0.08	**0.13 ± 0.10**	**0.07 ± 0.11**	**0.08 ± 0.02**
	m-expert	0.42 ± 0.08	**0.34 ± 0.39**	**0.10 ± 0.12**	**0.16 ± 0.08**	-	**0.14 ± 0.14**	**0.08 ± 0.07**
	medium	**0.32 ± 0.32**	**0.32 ± 0.32**	0.38 ± 0.28	**0.18 ± 0.19**	**0.10 ± 0.14**	0.42 ± 0.15	**0.05 ± 0.04**
	m-replay	**0.18 ± 0.23**	0.34 ± 0.24	0.88 ± 0.	0.16 ± 0.13	-	0.16 ± 0.12	**0.0 ± 0.0**
	random	0.36 ± 0.22	0.41 ± 0.17	**0.05 ± 0.05**	0.30 ± 0.15	0.26 ± 0.10	0.40 ± 0.20	**0.20 ± 0.10**
walker2d	expert	**0.06 ± 0.07**	**0.06 ± 0.07**	0.43 ± 0.26	**0.35 ± 0.36**	**0.09 ± 0.19**	0.42 ± 0.35	**0.05 ± 0.05**
	m-expert	0.22 ± 0.14	0.30 ± 0.12	0.13 ± 0.07	0.78 ± 0.27	**0.24 ± 0.42**	**0.43 ± 0.46**	**0.03 ± 0.02**
	medium	0.31 ± 0.10	0.25 ± 0.09	0.70 ± 0.39	**0.27 ± 0.43**	0.08 ± 0.06	**0.35 ± 0.41**	**0.0 ± 0.0**
	m-replay	0.24 ± 0.20	0.68 ± 0.23	**0.02 ± 0.05**	0.18 ± 0.12	0.46 ± 0.31	**0.21 ± 0.22**	**0.02 ± 0.01**
	random	**0.15 ± 0.21**	**0.15 ± 0.20**	0.74 ± 0.33	**0.39 ± 0.33**	0.88 ± 0.20	**0.57 ± 0.40**	**0.16 ± 0.09**
halfcheetah	expert	**0.12 ± 0.07**	**0.11 ± 0.08**	**0.15 ± 0.08**	**0.32 ± 0.40**	**0.14 ± 0.09**	**0.30 ± 0.36**	**0.12 ± 0.07**
	m-expert	**0.14 ± 0.07**	**0.14 ± 0.07**	0.73 ± 0.42	**0.38 ± 0.37**	0.80 ± 0.34	**0.36 ± 0.45**	**0.11 ± 0.10**
	medium	0.38 ± 0.13	0.37 ± 0.15	**0.05 ± 0.05**	0.82 ± 0.29	0.33 ± 0.19	**0.17 ± 0.10**	**0.11 ± 0.08**
	m-replay	**0.36 ± 0.16**	**0.33 ± 0.18**	**0.13 ± 0.10**	**0.30 ± 0.07**	**0.25 ± 0.09**	**0.23 ± 0.12**	**0.16 ± 0.12**
	random	0.37 ± 0.08	0.31 ± 0.10	0.31 ± 0.11	0.81 ± 0.30	**0.12 ± 0.07**	0.41 ± 0.08	**0.21 ± 0.12**
door	human	**0.05 ± 0.08**	**0.05 ± 0.09**	0.45 ± 0.40	**0.10 ± 0.27**	0.69 ± 0.24	**0.10 ± 0.10**	**0.03 ± 0.01**
	cloned	0.11 ± 0.06	**0.11 ± 0.08**	**0.02 ± 0.07**	0.65 ± 0.45	0.81 ± 0.33	**0.02 ± 0.03**	**0.02 ± 0.01**
	expert	**0.03 ± 0.03**	**0.05 ± 0.07**	**0.01 ± 0.04**	0.37 ± 0.27	**0.03 ± 0.03**	**0.33 ± 0.47**	**0.07 ± 0.03**
pen	human	**0.07 ± 0.05**	**0.09 ± 0.08**	**0.17 ± 0.15**	**0.04 ± 0.09**	0.28 ± 0.12	**0.30 ± 0.34**	**0.16 ± 0.11**
	cloned	**0.12 ± 0.07**	**0.13 ± 0.06**	**0.14 ± 0.09**	**0.12 ± 0.08**	**0.36 ± 0.18**	**0.19 ± 0.15**	**0.14 ± 0.09**
	expert	**0.11 ± 0.14**	**0.05 ± 0.07**	0.31 ± 0.10	0.33 ± 0.20	**0.25 ± 0.13**	**0.24 ± 0.27**	**0.13 ± 0.06**
hammer	human	0.46 ± 0.23	0.46 ± 0.23	**0.19 ± 0.30**	**0.04 ± 0.08**	**0.18 ± 0.29**	**0.38 ± 0.43**	**0.09 ± 0.03**
	cloned	**0.36 ± 0.39**	0.78 ± 0.38	**0.03 ± 0.15**	0.67 ± 0.48	0.72 ± 0.39	**0.39 ± 0.44**	**0.34 ± 0.23**
	expert	**0.05 ± 0.04**	**0.09 ± 0.09**	**0.01 ± 0.04**	**0.24 ± 0.34**	**0.04 ± 0.07**	**0.19 ± 0.23**	**0.04 ± 0.01**
relocate	human	**0.17 ± 0.14**	**0.17 ± 0.15**	0.63 ± 0.41	0.97 ± 0.11	0.77 ± 0.18	0.71 ± 0.21	**0.08 ± 0.06**
	cloned	**0.29 ± 0.42**	**0.18 ± 0.27**	**0.63 ± 0.41**	0.96 ± 0.18	**0.11 ± 0.29**	**0.71 ± 0.35**	**0.13 ± 0.06**
	expert	1.00 ± 0.06	0.98 ± 0.08	**0.18 ± 0.14**	0.97 ± 0.07	0.76 ± 0.23	**0.33 ± 0.47**	**0.12 ± 0.05**

G. Performance Comparison to Diffusion-based Methods

Following the reviewer's suggestion, we compare our EMPO with Diffuser (Janner et al., 2022) and Decision Diffuser (DD, Ajay et al. (2023)) in Table 12.

H. Simulated Trajectory Visualization

We visualize the simulated trajectories within the learned energy-based transition models (ETM) and forward transition models (FTM), and compare them with the real trajectories in the environment, shown in Figure 7. In each subplot, the three trajectories are generated by rolling out the same policy in different dynamics, i.e. ETM (top), real (middle), and FTM (bottom). We can see that in most cases the trajectories simulated by ETMs can remain visually aligned with the real trajectories for a relatively long horizon, while FTMs often lead to erroneous rollout in a few steps.

Table 8. Mean Absolute Error of the transition predictions of FTM and ETM for each task on the holdout dataset, averaged over 5 seeds.

Task	FTM	ETM
ant-expert	0.04851 ± 0.00152	**0.03538 ± 0.00286**
ant-medium-expert	0.05751 ± 0.00486	**0.03478 ± 0.00175**
ant-medium	0.07077 ± 0.00256	**0.03547 ± 0.00075**
ant-medium-replay	**0.04845 ± 0.00058**	0.05366 ± 0.00189
ant-random	**0.08610 ± 0.00189**	0.12960 ± 0.00265
hopper-expert	0.00505 ± 0.00057	**0.00041 ± 0.00006**
hopper-medium-expert	0.00597 ± 0.00035	**0.00341 ± 0.00021**
hopper-medium	0.00612 ± 0.00023	**0.00380 ± 0.00019**
hopper-medium-replay	0.00955 ± 0.00062	**0.00709 ± 0.00039**
hopper-random	0.00378 ± 0.00015	**0.00161 ± 0.00007**
walker2d-expert	0.03704 ± 0.00102	**0.03416 ± 0.00323**
walker2d-medium-expert	0.06744 ± 0.00422	**0.04528 ± 0.00187**
walker2d-medium	0.09093 ± 0.00587	**0.05315 ± 0.00124**
walker2d-medium-replay	0.14247 ± 0.00374	**0.11626 ± 0.00899**
walker2d-random	**0.15076 ± 0.00201**	0.16107 ± 0.00277
halfcheetah-expert	0.08520 ± 0.00404	**0.05936 ± 0.00773**
halfcheetah-medium-expert	0.08739 ± 0.00170	**0.08248 ± 0.00329**
halfcheetah-medium	0.09841 ± 0.00201	**0.08634 ± 0.00472**
halfcheetah-medium-replay	0.18054 ± 0.00289	**0.12594 ± 0.01701**
halfcheetah-random	**0.08946 ± 0.00139**	0.10963 ± 0.00532

Table 9. Mean Absolute Error for ETMs using different numbers of negative samples for training, averaged over 3 seeds.

Task	8	16	24
hopper-medium-replay	0.047 ± 0.006	0.053 ± 0.015	0.050 ± 0.005
hopper-medium	0.179 ± 0.027	0.086 ± 0.038	0.127 ± 0.013
walker2d-medium-replay	0.295 ± 0.028	0.243 ± 0.056	0.248 ± 0.032
walker2d-medium	0.330 ± 0.039	0.291 ± 0.023	0.312 ± 0.018

Table 10. Rank Correlation for ETMs using different numbers of negative samples for training, averaged over 3 seeds.

Task	8	16	24
hopper-medium-replay	0.977 ± 0.008	0.969 ± 0.020	0.965 ± 0.004
hopper-medium	0.834 ± 0.044	0.940 ± 0.038	0.925 ± 0.012
walker2d-medium-replay	0.867 ± 0.042	0.750 ± 0.120	0.714 ± 0.131
walker2d-medium	0.686 ± 0.111	0.778 ± 0.116	0.667 ± 0.144

Table 11. Regret@1 for ETMs using different numbers of negative samples for training, averaged over 3 seeds.

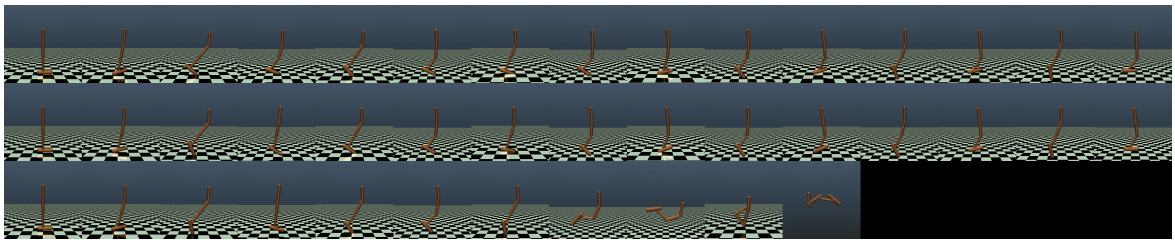
Task	8	16	24
hopper-medium-replay	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
hopper-medium	0.125 ± 0.071	0.080 ± 0.060	0.104 ± 0.098
walker2d-medium-replay	0.030 ± 0.010	0.030 ± 0.010	0.010 ± 0.010
walker2d-medium	0.011 ± 0.016	0.000 ± 0.000	0.011 ± 0.016

Table 12. Performance comparison across different tasks, including results of Diffuser (Janner et al., 2022) and Decision Diffuser (DD, Ajay et al. (2023)).

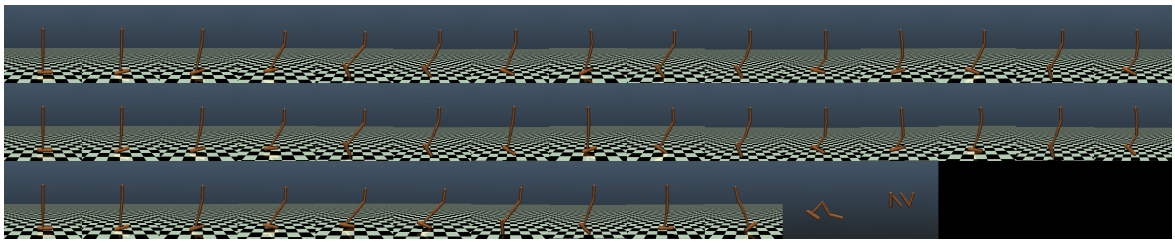
Task Name	**DT**	**TT**	**Diffuser**	**DD**	**CQL**	**EDAC**	**MOPO**	**MOBILE**	**EMPO (Ours)**
halfcheetah-m-e	86.8	95.0	79.8	90.6	91.6	106.3	90.8	108.2	103.8±2.3
hopper-m-expert	107.6	110.0	107.2	111.8	105.4	110.7	81.6	112.6	113.7±0.6
walker-m-expert	108.1	101.9	108.4	108.8	108.8	114.7	112.9	115.2	115.4±0.8
halfcheetah-m	42.6	46.9	44.2	49.1	44.0	65.9	73.0	74.6	77.4±0.6
hopper-medium	67.6	61.1	58.5	79.3	58.5	101.6	62.8	106.6	106.2±1.2
walker-medium	74.0	79.0	79.7	82.5	72.5	92.5	84.1	87.7	97.2±1.3
halfcheetah-m-r	36.6	41.9	42.2	39.3	45.5	61.3	73.0	71.7	73.8±1.5
hopper-m-replay	82.7	91.5	96.8	100.0	95.0	101.0	62.8	103.9	105.1±0.7
walker-m-replay	66.6	82.6	61.2	75.0	77.2	87.1	84.1	89.9	95.2±0.3
Average	74.7	78.9	75.3	81.8	77.6	93.5	80.6	96.7	98.6

Table 13. Normalized average returns on NeoRL tasks, averaged over 4 random seeds.

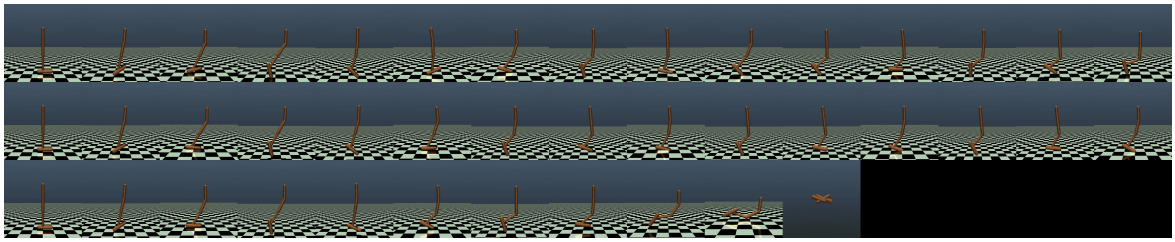
Task Name	**BC**	**CQL**	**TD3+BC**	**EDAC**	**MOPO**	**EMPO (Ours)**
HalfCheetah-L	29.1	38.2	30.0	31.3	40.1	35.5 $\pm$ 3.8
Hopper-L	15.1	16.0	15.8	18.3	6.2	18.5 $\pm$ 4.2
Walker2d-L	28.5	44.7	43.0	40.2	11.6	41.4 $\pm$ 5.6
HalfCheetah-M	49.0	54.6	52.3	54.9	62.3	54.6 $\pm$ 3.1
Hopper-M	51.3	64.5	70.3	44.9	1.0	66.7$\pm$ 12.5
Walker2d-M	48.7	57.3	58.5	57.6	39.9	58.0 $\pm$ 1.4
HalfCheetah-H	71.3	77.4	75.3	81.4	65.9	77.1 $\pm$ 4.7
Hopper-H	43.1	76.6	75.3	52.5	11.5	77.8 $\pm$ 17.4
Walker2d-H	72.6	75.3	69.6	75.5	18.0	74.0 $\pm$ 3.2
Average	45.4	56.1	54.5	50.7	28.5	56.0



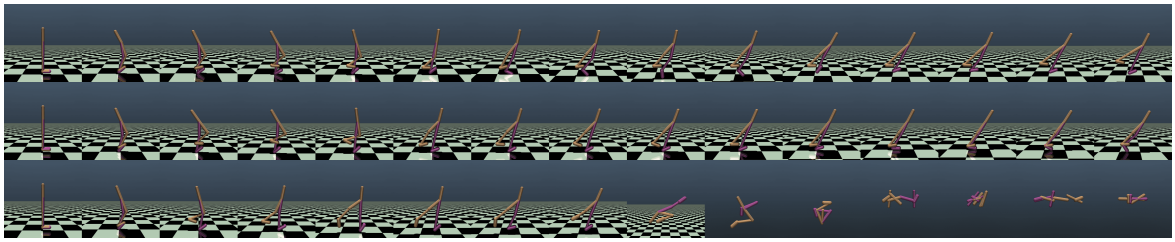
(a) hopper-medium-replay



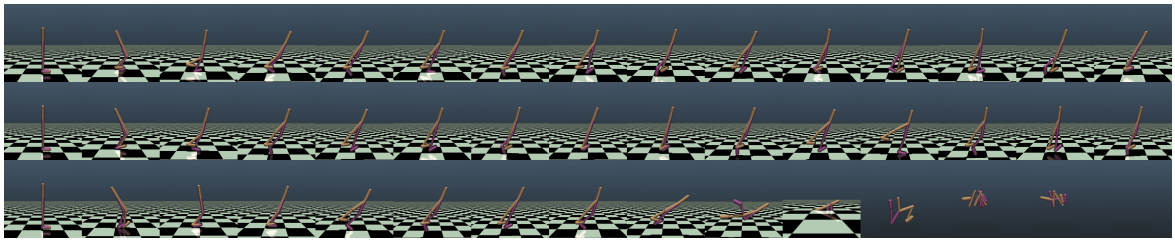
(b) hopper-medium



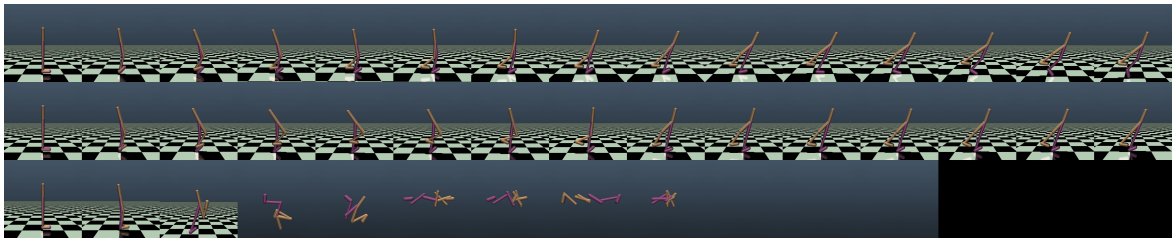
(c) hopper-medium-expert



(d) walker2d-medium-replay



(e) walker2d-medium



(f) walker2d-medium-expert

Figure 7. Comparison of simulated trajectories within ETM (top), real dynamics (middle), and FTM (bottom) in different tasks.