List of Content in Appendix

A ALGORITHM BOX

Algorithm 1 Temporally-Correlated Episodic RL (TCE)

Initialize policy parameters θ and value function parameters ϕ
for iteration = 1, 2, ... **do**
  Get the initial state $\boldsymbol{s}_0$
  Predict the mean $\boldsymbol{\mu}_{\boldsymbol{w}}$, covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}$, and sample $\boldsymbol{w}*$
  Generate the trajectory $\boldsymbol{y}*$ using Eq. (2) and execute it in the environment
  Collect step-based information through the execution
  Update ϕ, use true return or GAE style return (Schulman et al., 2015b)

  Select K time-pairs, e.g. choose every 10 steps along the trajectory
  Compute the segment-wise likelihood $\{p_k^{\text{old}}\}_{k=1:K}$ using Eq. (6) and 7 under π^{old}
  for update epoch = 1, 2, ... **do**
    Make prediction of the mean $\boldsymbol{\mu}_{\boldsymbol{w}}^{\text{new}}$, covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}^{\text{new}}$ under the latest policy π^{new}
    Enforce Trust Region by projecting $\boldsymbol{\mu}_{\boldsymbol{w}}^{\text{new}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{w}}^{\text{new}}$ through TRPL using Eq. (5)
    Get projected policy $\tilde{\pi}^{\text{new}}$, represented by $\tilde{\boldsymbol{\mu}}_{\boldsymbol{w}}^{\text{new}}$ and $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{w}}^{\text{new}}$
    Compute the segment-wise likelihood $\{p_k^{\text{new}}\}_{k=1:K}$ using Eq. (6) and 7 under $\tilde{\pi}^{\text{new}}$
    Update θ by taking a gradient step w.r.t. $J(\theta)$ in Eq. (9)
  end for
end for

B MATHEMATICAL FORMULATIONS OF MOVEMENT PRIMITIVES.

In this section, we provide a concise overview of the mathematical formulations of movement primitives utilized in this paper. We begin with the fundamentals of DMPs and ProMPs, followed by a detailed presentation of ProDMPs. This includes a focus on trajectory computation and the mapping between parameter distributions and trajectory distributions. For clarity, we begin with a single DoF system and then present the full trajectory distribution using a multi-DoF systems.

B.1 DYNAMIC MOVEMENT PRIMITIVES (DMPS)

Schaal (2006); Ijspeert et al. (2013) describe a single movement as a trajectory $[y_t]_{t=0:T}$, which is governed by a second-order linear dynamical system with a non-linear forcing function f. The mathematical representation is given by

$$\tau^2 \ddot{y} = \alpha(\beta(g-y) - \tau\dot{y}) + f(x), \quad f(x) = x\frac{\sum \varphi_i(x) w_i}{\sum \varphi_i(x)} = x\boldsymbol{\varphi}_x^\intercal \boldsymbol{w}, \tag{10}$$

where $y = y(t)$, $\dot{y} = \mathrm{d}y/\mathrm{d}t$, $\ddot{y} = \mathrm{d}^2 y/\mathrm{d}t^2$ denote the position, velocity, and acceleration of the system at a specific time t, respectively. Constants α and β are spring-damper parameters, g signifies a goal attractor, and τ is a time constant that modulates the speed of trajectory execution. To ensure convergence towards the goal, DMPs employ a forcing function governed by an exponentially decaying phase variable $x(t) = \exp(-\alpha_x/\tau; t)$. Here, $\varphi_i(x)$ represents the basis functions for the forcing term. The trajectory's shape as it approaches the goal is determined by the weight parameters $w_i \in \boldsymbol{w}$, for $i = 1, ..., N$. The trajectory $[y_t]_{t=0:T}$ is typically computed by numerically integrating the dynamical system from the start to the end point (Pahič et al., 2020; Bahl et al., 2020). However, this numerical process is computationally intensive, and complicates a directly translation between a parameter distribution $p(w)$ to its corresponding trajectory distribution $p(y)$ (Amor et al., 2014; Meier & Schaal, 2016). Previous method, such as GMM/GMR (Calinon et al., 2012; Calinon, 2016; Yang et al., 2018) used Gaussian Mixture Models to cover the trajectories' domain. However, this neither captures temporal correlation nor provide a generative model for the trajectories.

B.2 PROBABILISTIC MOVEMENT PRIMITIVES (PROMPS)

Paraschos et al. (2013) introduced a framework for modeling MPs using trajectory distributions, capturing both temporal and inter-dimensional correlations. Unlike DMPs that use a forcing term, ProMPs directly model the intended trajectory. The probability of observing a 1-DoF trajectory $[y_t]_{t=0:T}$ given a specific weight vector distribution $p(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}})$ is represented as a linear basis function model:

$$\text{Linear basis function:} \quad [y_t]_{t=0:T} = \boldsymbol{\Phi}_{0:T}^\intercal \boldsymbol{w} + \epsilon_y, \tag{11}$$

$$\text{Mapping distribution:} \quad p([y_t]_{t=0:T};\ \boldsymbol{\mu}_{\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{y}}) = \mathcal{N}(\boldsymbol{\Phi}_{0:T}^\intercal \boldsymbol{\mu}_{\boldsymbol{w}},\ \boldsymbol{\Phi}_{0:T}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_{0:T}\ + \sigma_y^2 \boldsymbol{I}). \tag{12}$$

Here, ϵ_y is zero-mean white noise with variance σ_y^2. The matrix $\boldsymbol{\Phi}_{0:T}$ houses the basis functions for each time step t. Similar to DMPs, these basis functions can be defined in terms of a phase variable instead of time. ProMPs allows for flexible manipulation of MP trajectories through probabilistic operators applied to $p(\boldsymbol{w})$, such as conditioning, combination, and blending (Maeda et al., 2014; Gomez-Gonzalez et al., 2016; Shyam et al., 2019; Rozo & Dave, 2022; Zhou et al., 2019). However, ProMPs lack an intrinsic dynamic system, which means they cannot guarantee a smooth transition from the robot's initial state or between different generated trajectories.

B.3 PROBABILISTIC DYNAMIC MOVEMENT PRIMITIVES (PRODMPS)

Solving the ODE underlying DMPs Li et al. (2023) noted that the governing equation of DMPs, as specified in Eq. (10), admits an analytical solution. This is because it is a second-order linear non-homogeneous ODE with constant coefficients. The original ODE and its homogeneous counterpart can be expressed in standard form as follows:

$$\text{Non-homo. ODE:} \quad \ddot{y} + \frac{\alpha}{\tau}\dot{y} + \frac{\alpha\beta}{\tau^2} y = \frac{f(x)}{\tau^2} + \frac{\alpha\beta}{\tau^2} g \equiv F(x, g), \tag{13}$$

$$\text{Homo. ODE:} \quad \ddot{y} + \frac{\alpha}{\tau}\dot{y} + \frac{\alpha\beta}{\tau^2} y = 0. \tag{14}$$

The solution to this ODE is essentially the position trajectory, and its time derivative yields the velocity trajectory. These are formulated as:

$$y = [y_2 \boldsymbol{p_2} - y_1 \boldsymbol{p_1} \quad y_2 q_2 - y_1 q_1] \begin{bmatrix} \boldsymbol{w} \\ g \end{bmatrix} + c_1 y_1 + c_2 y_2 \tag{15}$$

$$\dot{y} = [\dot{y}_2 \boldsymbol{p_2} - \dot{y}_1 \boldsymbol{p_1} \quad \dot{y}_2 q_2 - \dot{y}_1 q_1] \begin{bmatrix} \boldsymbol{w} \\ g \end{bmatrix} + c_1 \dot{y}_1 + c_2 \dot{y}_2. \tag{16}$$

Here, the learnable parameters $\boldsymbol{w}$ and g which control the shape of the trajectory, are separable from the remaining terms. Time-dependent functions $y_1, y_2, \boldsymbol{p}_1, \boldsymbol{p}_2, q_1, q_2$ in the remaining terms offer the basic support to generate the trajectory. The functions y_1, y_2 are the complementary solutions to the homogeneous ODE presented in Eq. (14), and $\dot{y}_1, \dot{y}_2$ their time derivatives respectively. These time-dependent functions take the form as:

$$y_1(t) = \exp\left(-\frac{\alpha}{2\tau}t\right), \qquad y_2(t) = t\exp\left(-\frac{\alpha}{2\tau}t\right), \tag{17}$$

$$\boldsymbol{p}_1(t) = \frac{1}{\tau^2}\int_0^t t' \exp\left(\frac{\alpha}{2\tau}t'\right) x(t')\boldsymbol{\varphi}_x^\intercal \mathrm{d}t', \qquad \boldsymbol{p}_2(t) = \frac{1}{\tau^2}\int_0^t \exp\left(\frac{\alpha}{2\tau}t'\right) x(t')\boldsymbol{\varphi}_x^\intercal \mathrm{d}t', \tag{18}$$

$$q_1(t) = \left(\frac{\alpha}{2\tau}t - 1\right)\exp\left(\frac{\alpha}{2\tau}t\right) + 1, \qquad q_2(t) = \frac{\alpha}{2\tau}\left[\exp\left(\frac{\alpha}{2\tau}t\right) - 1\right]. \tag{19}$$

It's worth noting that the $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ cannot be analytically derived due to the complex nature of the forcing basis terms $\boldsymbol{\varphi}_x$. As a result, they need to be computed numerically. Despite this, isolating the learnable parameters, namely $\boldsymbol{w}$ and g, allows for the reuse of the remaining terms across all generated trajectories. These residual terms can be more specifically identified as the position and velocity basis functions, denoted as $\boldsymbol{\Phi}(t)$ and $\dot{\boldsymbol{\Phi}}(t)$, respectively. When both $\boldsymbol{w}$ and g are included in a concatenated vector, represented as $\boldsymbol{w}_g$, the expressions for position and velocity trajectories can be formulated in a manner akin to that employed by ProMPs:

$$\textbf{Position:} \quad y(t) = \boldsymbol{\Phi}(t)^\intercal \boldsymbol{w}_g + c_1 y_1(t) + c_2 y_2(t), \tag{20}$$

$$\textbf{Velocity:} \quad \dot{y}(t) = \dot{\boldsymbol{\Phi}}(t)^\intercal \boldsymbol{w}_g + c_1 \dot{y}_1(t) + c_2 \dot{y}_2(t). \tag{21}$$

In the main paper, for simplicity and notation convenience, we use $\boldsymbol{w}$ instead of $\boldsymbol{w}_g$ to describe the parameters and goal of ProDMPs.

Smooth trajectory transition The coefficients c_1 and c_2 serve as solutions to the initial value problem delineated by the Eq.(20)(21). Li et al. propose utilizing the robot's initial state or the replanning state, characterized by the robot's position and velocity $(y_b, \dot{y}_b)$ to ensure a smooth commencement or transition from a previously generated trajectory. Denote the values of the complementary functions and their derivatives at the condition time t_b as $y_{1_b}, y_{2_b}, \dot{y}_{1_b}$ and $\dot{y}_{2_b}$. Similarly, denote the values of the position and velocity basis functions at this time as $\boldsymbol{\Phi}_b$ and $\dot{\boldsymbol{\Phi}}_b$ respectively. Using these notations, c_1 and c_2 can be calculated as follows:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \frac{\dot{y}_{2_b} y_b - y_{2_b}\dot{y}_b}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}} + \frac{y_{2_b}\dot{\boldsymbol{\Phi}}_b^\intercal - \dot{y}_{2_b}\boldsymbol{\Phi}_b^\intercal}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}}\boldsymbol{w}_g \\ \frac{y_{1_b}\dot{y}_b - \dot{y}_{1_b} y_b}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}} + \frac{\dot{y}_{1_b}\boldsymbol{\Phi}_b^\intercal - y_{1_b}\dot{\boldsymbol{\Phi}}_b^\intercal}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}}\boldsymbol{w}_g \end{bmatrix}. \tag{22}$$

Substituting Eq. (22) into Eq. (20) and Eq. (21), the position and velocity trajectories take the form as

$$y = \xi_1 y_b + \xi_2 \dot{y}_b + [\xi_3 \boldsymbol{\Phi}_b + \xi_4 \dot{\boldsymbol{\Phi}}_b + \boldsymbol{\Phi}]^\intercal \boldsymbol{w}_g, \tag{23}$$

$$\dot{y} = \dot{\xi}_1 y_b + \dot{\xi}_2 \dot{y}_b + [\dot{\xi}_3 \boldsymbol{\Phi}_b + \dot{\xi}_4 \dot{\boldsymbol{\Phi}}_b + \dot{\boldsymbol{\Phi}}]^\intercal \boldsymbol{w}_g \tag{24}$$

Here, ξ_k for $k \in \{1, 2, 3, 4\}$ serve as intermediate terms that are derived from the complementary functions and the initial conditions. The formations of these terms are elaborated below. To find their derivatives $\dot{\xi}_k$, one can simply replace y_1, y_2 with their time derivatives $\dot{y}_1, \dot{y}_2$ in the equations.

$$\xi_1(t) = \frac{\dot{y}_{2_b} y_1 - \dot{y}_{1_b} y_2}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}}, \qquad \xi_2(t) = \frac{y_{1_b} y_2 - y_{2_b} y_1}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}},$$

$$\xi_3(t) = \frac{\dot{y}_{1_b} y_2 - \dot{y}_{2_b} y_1}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}}, \qquad \xi_4(t) = \frac{y_{2_b} y_1 - y_{1_b} y_2}{y_{1_b}\dot{y}_{2_b} - y_{2_b}\dot{y}_{1_b}}.$$

Extend to a High DoF system Both ProMPs and ProDMPs can be generalized to accommodate high-DoF systems. This allows for the capture of both temporal correlations and interactions among various DoF. Such generalization is implemented through modifications to matrix structures and vector concatenations, as illustrated in Paraschos et al. (2013); Li et al. (2023). To be specific, the basis functions $\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}$, along with their values at the condition time $\boldsymbol{\Phi}_b, \dot{\boldsymbol{\Phi}}_b$, are extended to block-diagonal matrices $\boldsymbol{\Psi}, \dot{\boldsymbol{\Psi}}$, $\boldsymbol{\Psi}_b$ and $\dot{\boldsymbol{\Psi}}_b$ respectively. This extension is executed by tiling the existing basis function matrices D times along their diagonal, where D is the number of DoF. Additionally, the robot initial conditions for each DoF are concatenated into one vectors. For instance, the initial positions $y_b^1,\ ...,\ y_b^D$ are unified into a single vector $\boldsymbol{y}_b = [y_b^1,\ ...,\ y_b^D]^\intercal$. In this way, the position and velocity trajectories are extended as

$$\boldsymbol{y} = \xi_1 \boldsymbol{y}_b + \xi_2 \dot{\boldsymbol{y}}_b + [\xi_3 \boldsymbol{\Psi}_b + \xi_4 \dot{\boldsymbol{\Psi}}_b + \boldsymbol{\Psi}]^\intercal \boldsymbol{w}_g, \tag{25}$$

$$\dot{\boldsymbol{y}} = \dot{\xi}_1 \boldsymbol{y}_b + \dot{\xi}_2 \dot{\boldsymbol{y}}_b + [\dot{\xi}_3 \boldsymbol{\Psi}_b + \dot{\xi}_4 \dot{\boldsymbol{\Psi}}_b + \dot{\boldsymbol{\Psi}}]^\intercal \boldsymbol{w}_g. \tag{26}$$

Parameter distribution to trajectory distribution In a manner analogous to the description provided for ProMPs from Equation Eq. (2) to Equation Eq. (3), ProDMPs also exhibits a comparable architecture framework. This similarity is particularly evident in the structure of the learnable parameters, denoted as $\boldsymbol{w}_g$, which follow a linear basis function format. Consequently, it is reasonable to delineate the trajectory distribution for ProDMPs in fashion akin to that of ProMPs. Given that the parameter distribution $\boldsymbol{w}_g$ follows a Gaussian distribution $\boldsymbol{w}_g \sim \mathcal{N}(\boldsymbol{w}_g | \boldsymbol{\mu}_{\boldsymbol{w}_g}, \boldsymbol{\Sigma}_{\boldsymbol{w}_g})$ and adhering to the probabilistic formulation analogous to ProMPs as indicated in Eq. (3), the trajectory distribution for ProDMPs can be expressed as:

$$p([y_t]_{t=0:T};\ \boldsymbol{\mu}_{\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{y}}) = \mathcal{N}([\boldsymbol{y}_t]_{t=0:T} |\ \boldsymbol{\mu}_{\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{y}}), \tag{27}$$

where

$$\boldsymbol{\mu}_{\boldsymbol{y}} = \boldsymbol{\xi}_1 \boldsymbol{y}_b + \boldsymbol{\xi}_2 \dot{\boldsymbol{y}}_b + \boldsymbol{H}_{0:T}^\intercal \boldsymbol{\mu}_{w_g}, \qquad \boldsymbol{\Sigma}_{\boldsymbol{y}} = \boldsymbol{H}_{0:T}^\intercal \boldsymbol{\Sigma}_{w_g} \boldsymbol{H}_{0:T} + \sigma_n^2 \mathbf{I},$$
$$\boldsymbol{H}_{0:T} = \boldsymbol{\xi}_3 \boldsymbol{\Psi}_b + \boldsymbol{\xi}_4 \dot{\boldsymbol{\Psi}}_b + \boldsymbol{\Psi}_{0:T}, \qquad \boldsymbol{\xi}_k = [\xi_k(t)]_{t=0:T}.$$

In this context, the trajectory mean, denoted as $\boldsymbol{\mu}_{\boldsymbol{y}}$ constitutes a vector of dimension DT, whereas the trajectory covariance, represented by $\boldsymbol{\Sigma}_{\boldsymbol{y}}$ is a $DT \times DT$ dimensional matrix. These quantities serve to integrate the trajectory values across all degrees of freedom (DoF) and temporal steps, encapsulating them within a single distribution. This multi-DoF ProDMPs representation can be seen as an enhancement of the ProMPs framework, augmented by the inclusion of initial condition terms. This ensures that the trajectories sampled under this distribution start from the specified initial state. **Additionally, the time range $t = 0 : T$ is flexible and can be replaced by any set of discrete time points. For instance, in the TCE method, a pair of time points t_k and t_k' can define a trajectory segment, allowing for down-sampling of the trajectory distribution to specific trajectroy segment.**

C EXPERIMENT DETAILS

C.1 DETAILS OF METHODS IMPLEMENTATION

PPO Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a prominent on-policy step-based RL algorithm that refines the policy gradient objective, ensuring policy updates remain close to the behavior policy. PPO branches into two main variants: PPO-Penalty, which incorporates a KL-divergence term into the objective for regularization, and PPO-Clip, which employs a clipped surrogate objective. In this study, we focus our comparisons on PPO-Clip due to its prevalent use in the field. Our implementation of PPO is based on the implementation of Raffin et al. (2021).

SAC Soft Actor-Critic (SAC) (Haarnoja et al., 2018a;b) employs a stochastic step-based policy in an off-policy setting and utilizes double Q-networks to mitigate the overestimation of Q-values for stable updates. By integrating entropy regularization into the learning objective, SAC balances between expected returns and policy entropy, preventing the policy from premature convergence. Our implementation of SAC is based on the implementation of Raffin et al. (2021).

TRPL Trust Region Projection Layers (TRPL) (Otto et al., 2021), akin to PPO, addresses the problem of stabilizing the on-policy policy gradient by constraining the learning policy staying close to the behavior policy. TRPL formulates the constrained optimization problem as a projection problem, providing a mathematically rigorous and scalable technique that precisely enforces trust regions on each state, leading to stable and efficient on-policy updates. We evaluated its performance based on the implementation of the original work.

gSDE Generalized State Dependent Exploration (gSDE) (Raffin et al., 2022; Rückstieß et al., 2008; Rückstiess et al., 2010) is an exploration method designed to address issues with traditional step-based exploration techniques and aims to provide smoother and more efficient exploration in the context of robotic reinforcement learning, reducing jerky motion patterns and potential damage to robot motors while maintaining competitive performance in learning tasks.

To achieve this, gSDE replaces the traditional approach of independently sampling from a Gaussian noise at each time step with a more structured exploration strategy, that samples in a state-dependent manner. The generated samples not only depend on parameter of the Gaussian distribution μ & Σ, but also on the activations of the policy network's last hidden layer (s). We generate disturbances ϵ_t using the equation

$$\epsilon_t = \theta_\epsilon s, \text{ where } \theta_\epsilon \sim \mathcal{N}^d(0, \Sigma).$$

The exploration matrix θ_ϵ is composed of vectors of length $\text{Dim}(a)$ that were drawn from the Gaussian distribution we want gSDE to follow. The vector s describes how this set of pre-computed exploration vectors are mixed. The exploration matrix is resampled at regular intervals, as guided by the 'sde sampling frequency' (ssf), occurring every n-th step if n is our ssf.

gSDE is versatile, applicable as a substitute for the Gaussian Noise source in numerous on- and off-policy algorithms. We evaluated its performance in an on-policy setting using PPO by utilizing the reference implementation for gSDE from Raffin et al. (2022). In order for training with gSDE to remain stable and reach high performance the usage of a linear schedule over the clip range had to be used for some environments.

PINK We utilize SAC to evaluate the effectiveness of pink noise for efficient exploration. Eberhard et al. (2022) propose to replace the independent action noise ϵ_t of

$$a_t = \mu_t + \sigma_t \cdot \epsilon_t$$

with correlated noise from particular random processes, whose power spectral density follow a power law. In particular, the use of pink noise, with the exponent $\beta = 1$ in $S(f) = |\mathcal{F}[\epsilon](f)|^2 \propto f^{-\beta}$, should be considered (Eberhard et al., 2022).

We follow the reference implementation and sample chunks of Gaussian pink noise using the inverse Fast Fourier Transform method proposed by Timmer & Koenig (1995). These noise variables are used for SAC's exploration but the the actor and critic updates sample the independent action distribution without pink noise. Each action dimension uses an independent noise process which

causes temporal correlation within each dimension but not across dimensions. Furthermore, we fix the chunk size and maximum period to 10000 which avoids frequent jumps of chunk borders and increases relative power of low frequencies.

BBRL-Cov/Std Black-Box Reinforcement Learning (BBRL) (Otto et al., 2022; 2023) is a recent developed episodic reinforcement learning method. By utilizing ProMPs (Paraschos et al., 2013) as the trajectory generator, BBRL learns a policy that explores at the trajectory level. The method can effectively handle sparse and non-Markovian rewards by perceiving an entire trajectory as a unified data point, neglecting the temporal structure within sampled trajectories. However, on the other hand, BBRL suffers from relatively low sample efficiency due to its black-box nature. Moreover, the original BBRL employs a degenerate Gaussian policy with diagonal covariance. In this study, we extend BBRL to learn Gaussian policy with full covariance to build a more competitive baseline. For clarity, we refer to the original method as BBRL-Std and the full covariance version as BBRL-Cov. We integrate BBRL with ProDMPs (Li et al., 2023), aiming to isolate the effects attributable to different MP approaches.

C.2 METAWORLD PERFORMANCE PROFILE ANALYSIS

The distribution of success rates, reported in the performance profile in Fig. 5b, may seem to contradict the nearly perfect IQM of TCE but in reality provide insight into the consistency of TCE. Nearly two thirds of runs exceed 99% success rate and are therefore able to perfectly solve the task with this seed. The individual performances reported in Appendix C.3 show that only very few tasks, e.g., Disassemble and Hammer, have a significant fraction of unsuccessful seeds. This consistency per task is also visible in the profile, as only two percent of runs fall between zero and sixty percent success rate which is visible by the near zero slope in this range. All methods are entirely unable to solve a small set of tasks and therefore show a gap in the profile. This does not contradict the very high IQM success rate of TCE, as the IQM trims the upper and lower 25% of results. The commonly reported median effectively trims 50% and would result in even higher values. Due to the smaller and later decline in the profile compared to the other methods, the intersection between 75% of runs and the profile is located at a success rate of 90%. Therefore, only a small fraction of runs, roughly ten percent, fall within the 25% trim but only slightly decrease the value of the IQM due their high success rate. Other methods have a larger fraction of imperfect runs with lower success rate within the quartiles.

C.3 PERFORMANCE ON INDIVIDUAL METAWORLD TASKS

We report the Interquartile Mean (IQM) of success rates for each Metaworld task. The plots clearly illustrate the varying levels of difficulty across different tasks.

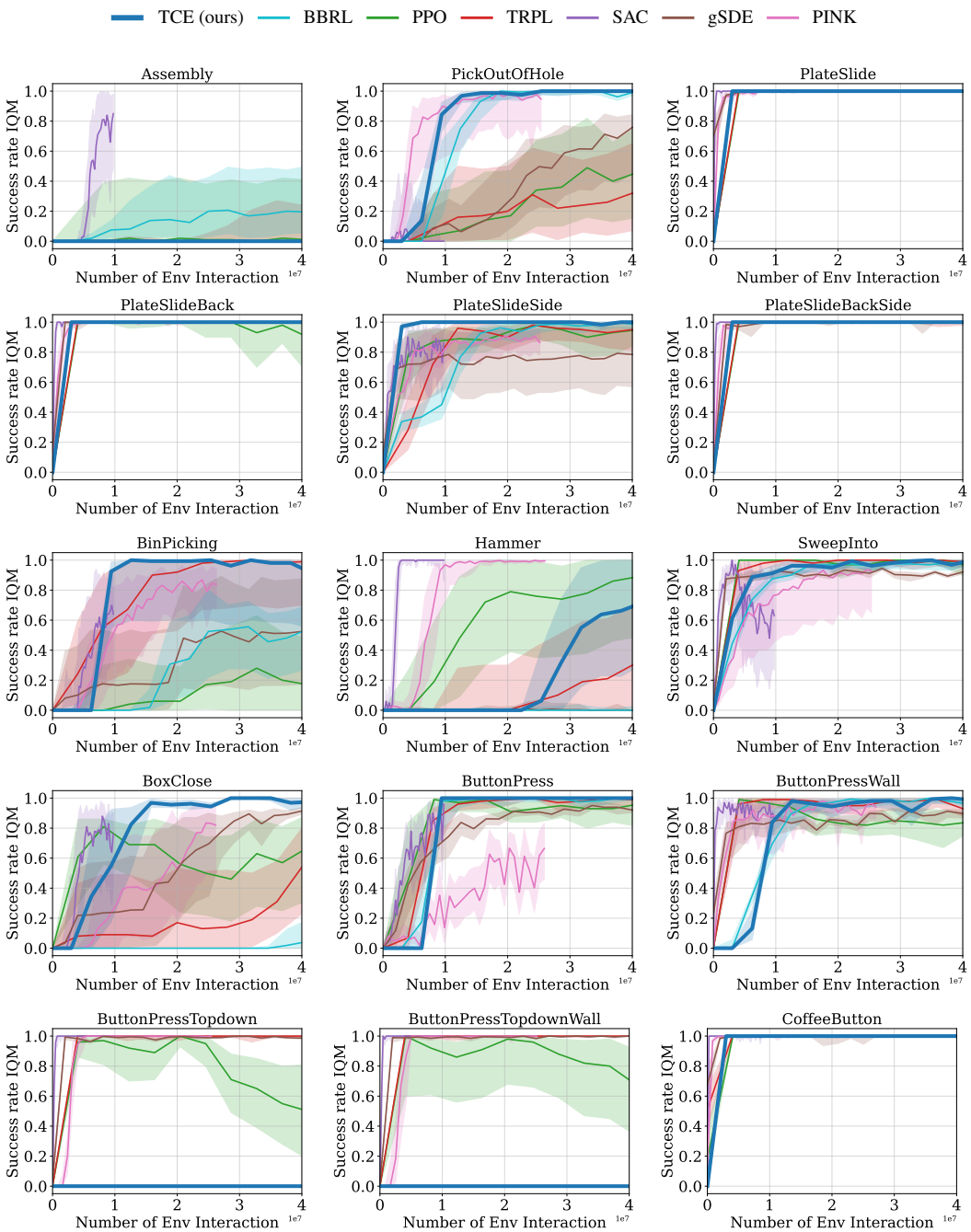


Figure 8: Success Rate IQM of each individual Metaworld tasks.

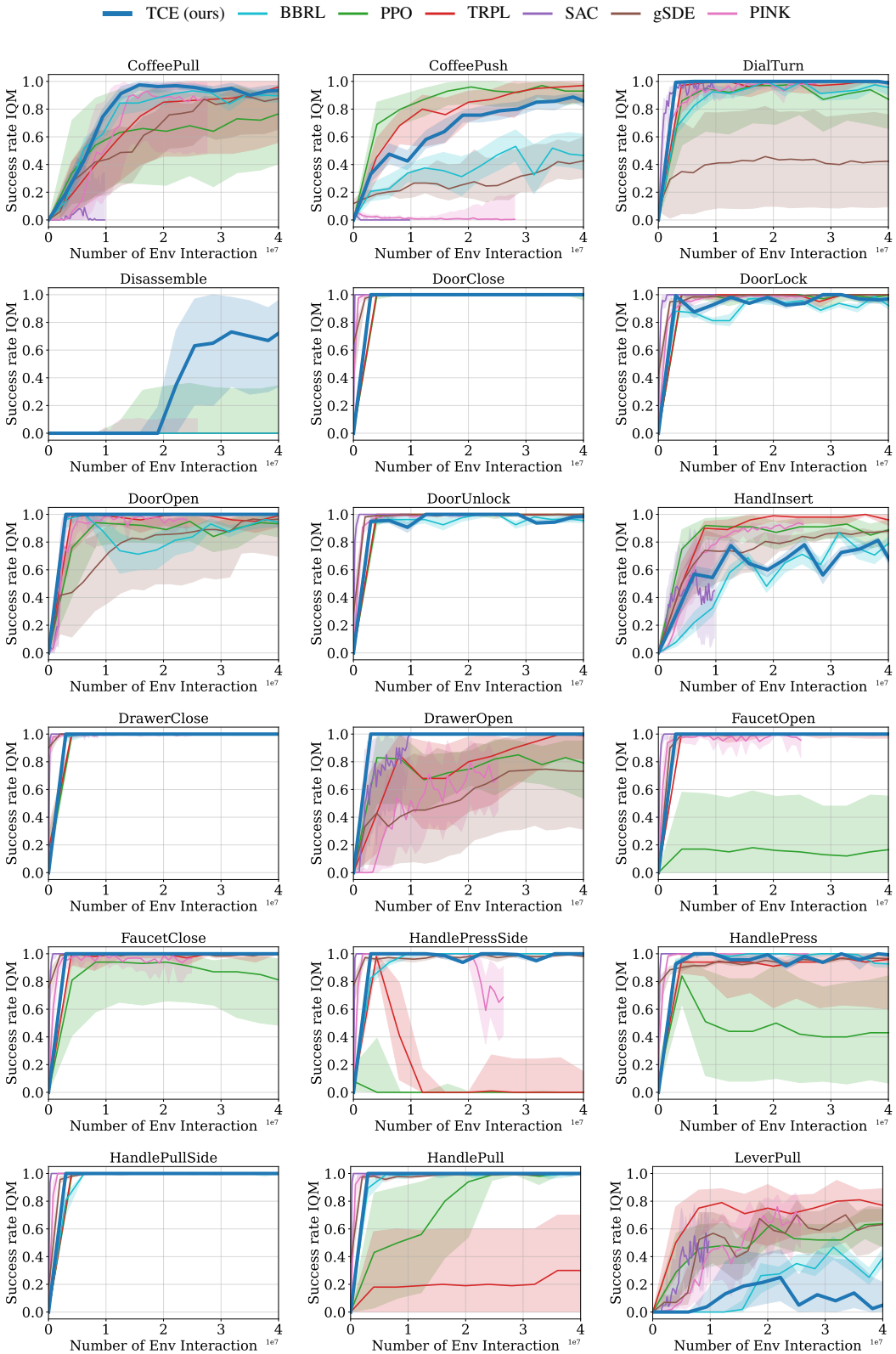


Figure 9: Success Rate IQM of each individual Metaworld tasks.

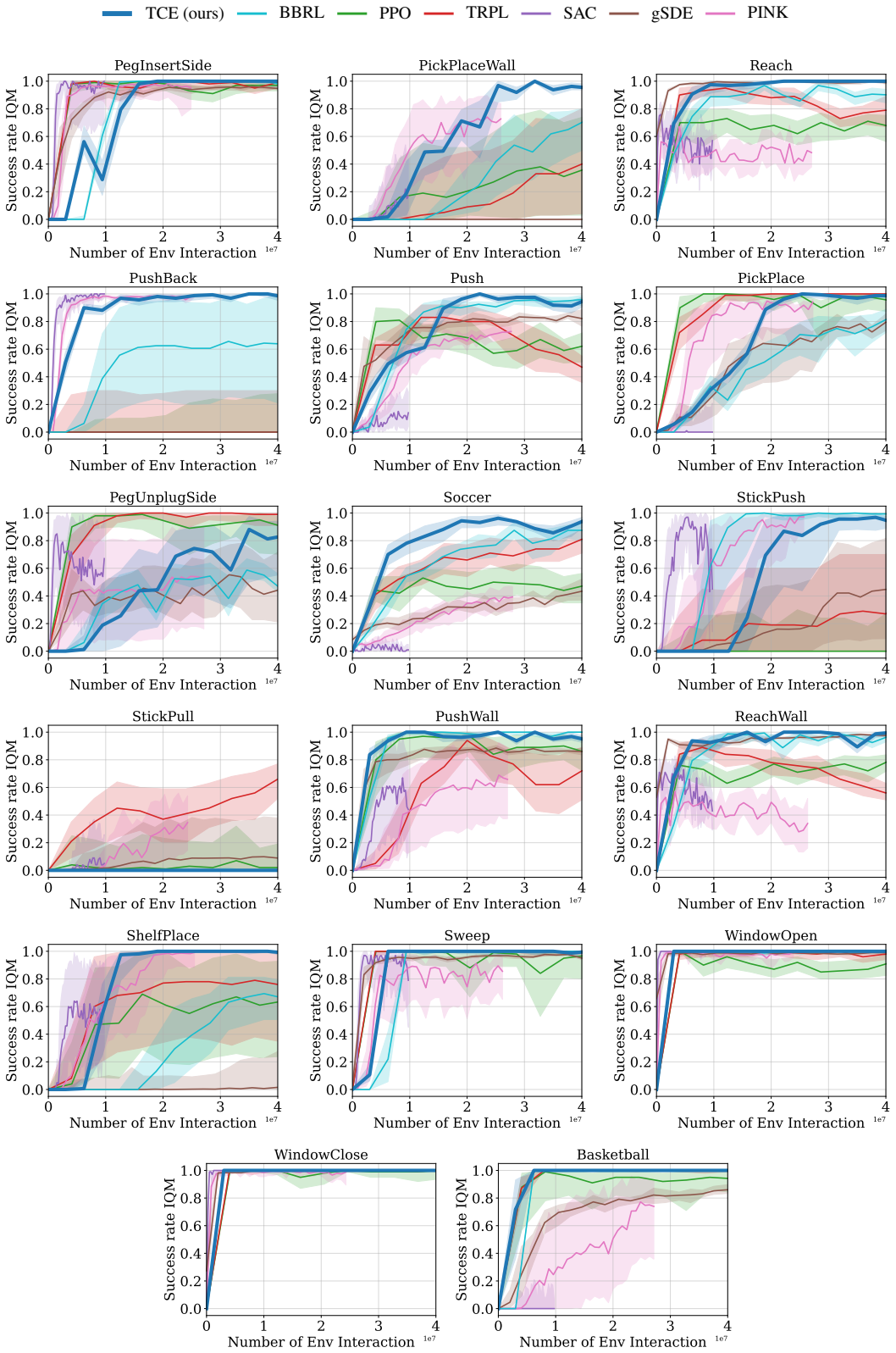
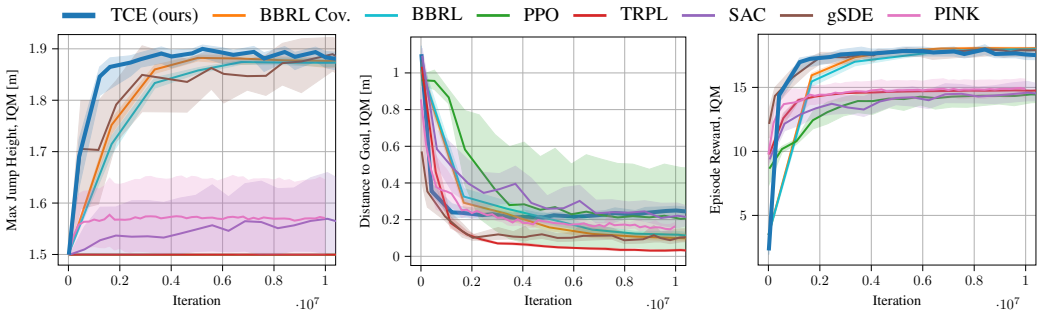


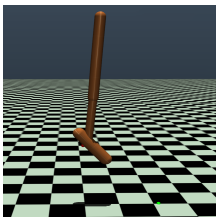
Figure 10: Success Rate IQM of each individual Metaworld tasks.

C.4 HOPPER JUMP



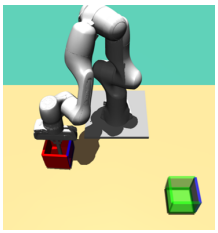
(a) Max Jump Height (b) Distance to Goal (c) Overall, Height + Dist.

Figure 11: Hopper Jump



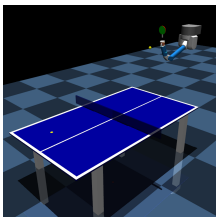
As an addition to the main paper, we provide more details on the Hopper Jump task. We look at both the main goal of maximizing jump height and the secondary goal of landing on a desired position. These are shown along with the overall episode reward in Fig. 11. Our method shows quick learning and does well in achieving high jump height, consistent with what we reported earlier. While it's not as strong in landing accuracy, it still ranks high in overall performance. Both versions of BBRL have similar results. However, they train more slowly compared to TCE, highlighting the speed advantage of our method due to the use of intermediate states for policy updates. Looking at other methods, step-based ones like PPO and TRPL focus too much on landing distance and miss out on jump height, leading to less effective policies. On the other hand, gSDE performs well but is sensitive to the initial setup, as shown by the wide confidence ranges in the results. Lastly, SAC and PINK shows inconsistent results in jump height, indicating the limitations of using pink noise for exploration, especially when compared to gSDE.

C.5 BOX PUSHING

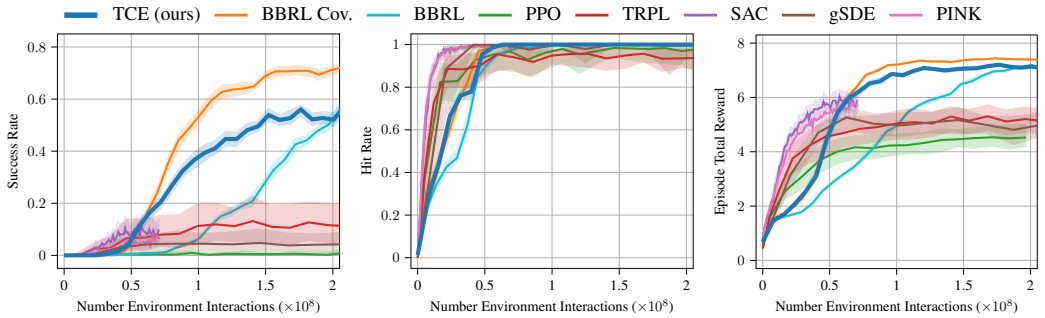


The goal of the box-pushing task is to move a box to a specified goal location and orientation using the 7-DoFs Franka Emika Panda (Otto et al., 2022). To make the environment more challenging, we extend the environment from a fixed initial box position and orientation to a randomized initial position and orientation. The range of both initial and target box pose varies from $x \in [0.3, 0.6], y \in [-0.45, 0.45], \theta_z \in [0, 2\pi]$. Success is defined as a positional distance error of less than 5 cm and a z-axis orientation error of less than 0.5 rad. We refer to the original paper for the observation and action spaces definition and the reward function.

C.6 TABLE TENNIS



The goal of table tennis environment to use the 7-DoFs robotic arm to hit the incoming ball and return it as close as possible to the specified goal location. We adapt the table tennis environment from Celik et al. (2022); Otto et al. (2022) and extend it to a randomized initial robot joint configuration. As context space we consider the initial ball position $x \in [-1, -0.2]$, $y \in [-0.65, 0.65]$ and the goal position $x \in [-1.2, -0.2]$, $y \in [-0.6, 0.6]$. The task is considered successful if the returned ball lands on the opponent's side of the table and within $\leq$ 0.2m to the goal location. We refer to the original paper for the observation and action spaces definition and the reward function.



(a) Success Rate (b) Hit Rate (c) Episode Reward

Figure 14: Robot Table Tennis Rand2Rand

D Additional Evaluation and Ablation Study

D.1 Trajectory Smoothness Metric

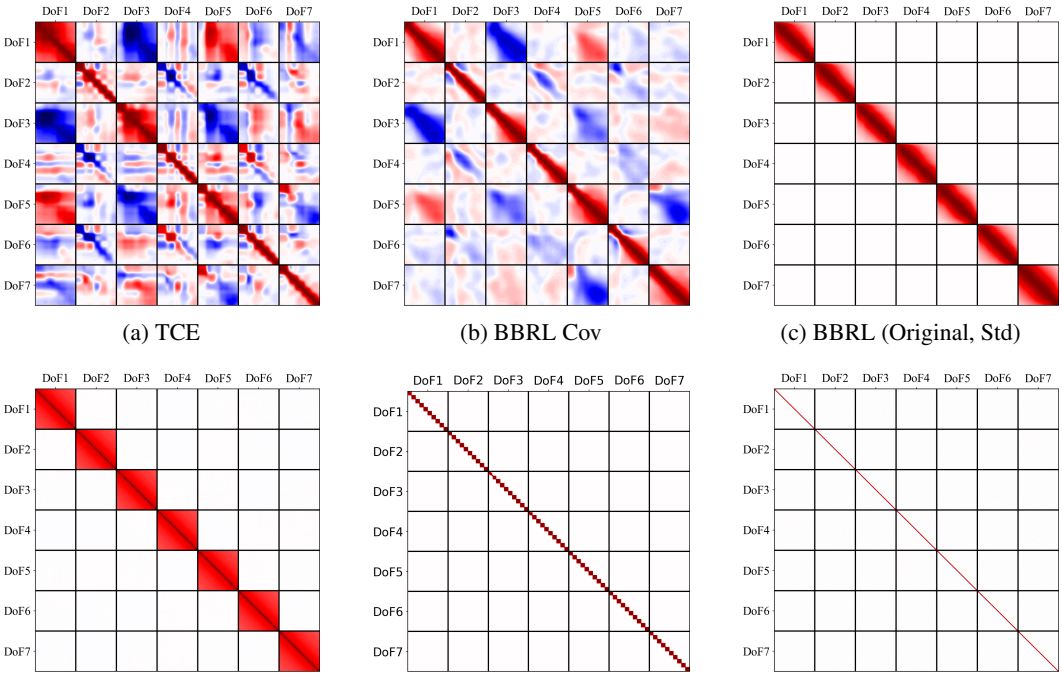
We compared the trajectory smoothness of all methods in Table 1. To ensure a fair comparison, all methods were trained using the fixed start box pushing dense reward setting as originally reported in Otto et al. (2022), where each method achieved a minimum 50% success rate. Trajectories for evaluation were generated using the mean prediction of the policy. The smoothness was assessed using three metrics: *maximum jerk*, *mean squared jerk* (Wininger et al., 2009), and *dimensionless jerk* (Hogan & Sternad, 2009). The first two metrics are standard in robot trajectory generation (Berscheid & Kröger, 2021; Lange & Suppa, 2015), while the last is proposed as a more equitable measure of smoothness, eliminating the effects of motion magnitude and time scaling. TCE and BBRL Cov outperformed all other methods in smoothness, followed by the original BBRL. This performance disparity likely stems from the original BBRL's inability to model inter-DoF movement correlations. In contrast, all step-based methods exhibited lower smoothness, attributable to their inherent per-step action selection approach.

Table 1: Trajectory Smoothness, Mean (Std) of Three Metrics over 400 Trajectories.

Metric	TCE	BBRL Cov	BBRL	PPO	TRPL	gSDE	SAC	PINK
Maximum Jerk, $\times 10^3 \text{rad/s}^3$	**3.4 (1.9)**	3.5 (1.5)	4.3 (1.4)	9.1 (3.3)	12.9 (4.8)	6.9 (2.2)	9.3 (4.2)	6.5 (1.7)
Mean Sq. Jerk, $\times 10^6 \text{rad}^2/\text{s}^6$	**0.2 (0.2)**	**0.2 (0.3)**	0.6 (0.6)	1.3 (0.9)	5.5 (8.6)	0.8 (0.6)	3.9 (1.1)	1.7 (0.7)
Dimensionless Jerk, $\times 10^6$	**61 (73)**	64 (56)	128 (49)	141 (67)	555 (472)	122 (83)	506 (262)	311 (127)

D.2 Action Correlations Predicted by Trained Policies

We plot the action correlation coupling DoF and time steps in Fig. 16. All policies were trained under the box-pushing task with a dense reward setting. The action outputs for TCE, BBRL, and BBRL Cov are the positions of the robot joints, while step-based methods, such as PPO, predict actions in the torque space. TCE and BBRL Cov demonstrate the ability to predict actions correlated both temporally and across DoF, as indicated by the non-zero off-diagonal elements in their correlation matrices. In contrast, the original BBRL translates a factorized weight distribution into a block-diagonal action correlation matrix, capturing variance within individual DoF but not between them. Similarly, PINK is constrained to modeling intra-DoF correlations, which depend solely on the time difference. This limitation arises from the wide-sense stationarity of the noise, resulting in a constant value along each diagonal. gSDE, however, models temporal correlation but only over a few consecutive time steps, observable along the diagonal elements. Actions predicted by PPO, TRPL, and SAC lack both temporal and DoF correlation, resulting in correlation matrices resembling identity matrices. Interestingly, for methods that only capture intra-DoF correlations, these correlations are uniformly positive. This trend may relate to the control cost in the reward function, promoting consistent movement within each DoF over time. On the other hand, TCE and BBRL Cov are unique in their ability to capture negative correlations, both between and within DoF, enhancing their flexibility in trajectory sampling.



(a) TCE (b) BBRL Cov (c) BBRL (Original, Std)

(d) PINK (e) gSDE (f) PPO / TRPL / SAC

Figure 16: This figure presents predicted actions' correlation across 7 DoF and 100 time steps, visualized in a 700x700 correlation matrix. Each 100 $\times$ 100 square tile demonstrates the movement correlation between two DoF during these steps. Correlation values range from -1 (negative correlation, depicted in blue) to 1 (positive correlation, depicted in red), with white areas indicating no correlation. The action outputs for TCE, BBRL, and BBRL Cov are the positions of the robot joints, whereas step-based methods predict actions in the torque space. TCE and BBRL Cov exhibit to a higher capacity of movement correlations. The original BBRL and PINK only model correlations within each DoF. gSDE models correlations over a few consecutive time steps. We show only one representative matrix for PPO, TRPL, and SAC, as their results are visually identical, typically resulting in matrices resembling the identity matrix.

D.3 Ablation: SAC + Motion Primitives-based Method

Training movement primitive-based methods using standard RL techniques, such as PPO and SAC, generally poses challenges due to the complex, higher-dimensional trajectory parameter space. In the study by Otto et al. (2022), an ablation study employing a PPO-style trust region (likelihood clipping) for training BBRL demonstrated inferior performance compared to the use of a differentiable trust region projection layer.

In Figure 17, we present an additional ablation study where SAC is used to learn the trajectory parameters of movement primitives. This study compares the performance of SAC with that of the original BBRL and BBRL Cov, leading to relatively poorer performance. The SAC model selected for reporting was the best performer among 40 different combinations of hyperparameters. The hyperparameters adjusted include the output action scaling factor (necessary because the SAC action space is bounded by $[-1, 1]$), policy/critic learning rate, batch size, and the size of the policy/critic network. The relatively shorter training curve of SAC can be attributed to its higher computational cost in policy update (Haarnoja et al., 2018b).

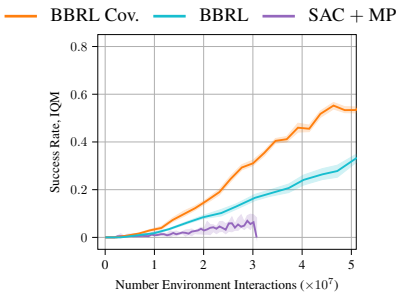


Figure 17: By employing the standard SAC method to learn trajectory parameters $\boldsymbol{w}$, we compared its performance with that of the original BBRL and BBRL Cov methods under the box pushing dense reward setting. The ablated method, using SAC, showed relatively poorer performance.

D.4 Ablation: Using PPO Style Trust Regions for TCE Method

We developed an ablated version of our method, incorporating the PPO-style trust region via likelihood clipping. We tuned the clipping ratio ϵ between 0.05 and 0.2. As illustrated in Figure 18, this version's performance falls between the original TCE and the standard PPO. The movement primitives' high-dimensional parameter space limits the effectiveness of likelihood clipping in precisely maintaining the trust region during policy updates. This limitation likely accounts for the performance gap between TCE and its PPO variant. Nonetheless, the ablated model still demonstrates a notable advantage over standard PPO, further substantiating our model's effectiveness in temporally correlated trajectory prediction.

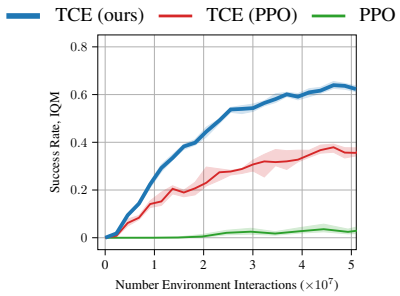
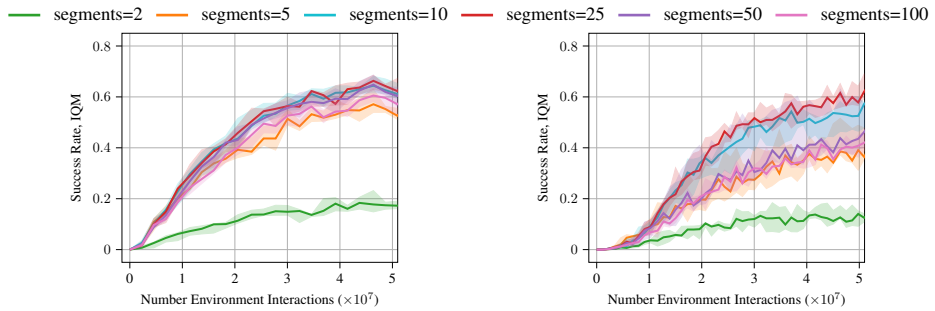


Figure 18: Training TCE with a PPO-style trust region, employing likelihood clipping with $\epsilon = 0.1$, yields suboptimal performance in the box pushing dense reward setting. Nevertheless, its superior performance compared to standard PPO underscores our method's effectiveness in episodic trajectory modeling.

D.5 Ablation: Selection of the Amount of Segments K

We conducted an ablation study to evaluate the effect of varying the number of segments (k) on model performance. The number of segments tested ranged from 2 to 100. Our experiments involved training in both dense and sparse box-pushing environments. The results revealed a greater sensitivity to the number of segments in the sparse reward environment compared to the dense environment. We attribute this to the challenges associated with the value function approximation under sparse reward settings. However, within an optimal range, such as 10-25 segments, this parameter is not overly sensitive compared to other hyper-parameters. Consequently, we have adopted k=25 for all experiments in this paper.



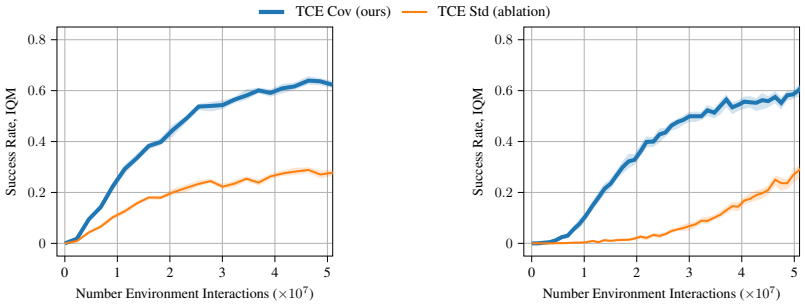
(a) Box Pushing, Dense (b) Box Pushing, Sparse

Figure 19: Study of the number of segment per trajectory.

D.6 TCE Cov vs. STD

We conducted an ablation study to assess the impact of employing a full covariance policy in the TCE framework. This involved comparing the standard TCE with its variant, TCE Std, which utilizes a factorized Gaussian policy $\mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\sigma}^2_{\boldsymbol{w}})$. The comparison was conducted in scenarios involving both dense and sparse reward settings in box pushing tasks. The findings revealed that the ablated version, TCE Std, consistently underperformed compared to the full covariance version. This underperformance is attributed to the limited correlation capacity of the factorized Gaussian policy.

Furthermore, it is important to note that while the factorized Gaussian distribution results in a relatively lower computational load in the parameter space, it does not offer a marked advantage when translated into trajectory space. As illustrated in Fig. 16(c) of Section D.2, a factorized parameter distribution ultimately converts into a blocked diagonal trajectory distribution. Although this format is visually simpler compared to a full trajectory covariance matrix, both share same time complexity in terms of likelihood computation. This computational process is significantly more resource-intensive than that for a purely diagonal matrix. Therefore, we utilize the techniques in Li et al. (2023) to apply a likelihood estimation and reduce the computational cost.



(a) Box Pushing, Dense (b) Box Pushing, Sparse

Figure 20: Study of the TCE Cov vs. TCE Std.

E HYPER PARAMETERS

We executed a large-scale grid search to fine-tune key hyperparameters for each baseline method. For other hyperparameters, we relied on the values specified in their respective original papers. Below is a list summarizing the parameters we swept through during this process.

BBRL: Policy net size, critic net size, policy learning rate, critic learning rate, samples per iteration, trust region dissimilarity bounds, number of parameters per movement DoF.

TCE: Same types of hyper-parameters listed in BBRL, plus the number of segments per trajectory. A learning rate decaying scheduler is applied to stabilize the training in the end.

PPO: Policy network size, critic network size, policy learning rate, critic learning rate, batch size, samples per iteration.

TRPL: Policy network size, critic network size, policy learning rate, critic learning rate, batch size, samples per iteration, trust region dissimilarity bounds.

gSDE: Same types of hyper-parameters listed in PPO, together with the state dependent exploration sampling frequency (Raffin et al., 2022).

SAC: Policy network size, critic network size, policy learning rate, critic learning rate, alpha learning rate, batch size, Update-To-Data (UTD) ratio.

PINK: Same types of hyper-parameters listed in SAC.

The detailed hyper parameters used are listed in the following tables. Unless stated otherwise, the notation lin_x refers to a linear schedule. It interpolates linearly from x to 0 during training. The ERL methods (TCE, BBRL) take an entire trajectory as a sample where the SRL methods take one time step as a sample. In this way, one sample in ERL is equivlent to T sample of SRL, where T is the length of one task episode.

Table 2: Hyperparameters for the Meta-World experiments. Episode Length $T = 500$

	PPO	gSDE	TRPL	SAC	PINK	TCE	BBRL
number samples	16000	16000	16000	1000	4	16	16
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.
discount factor	0.99	0.99	0.99	0.99	0.99	1	1
ϵ_μ	n.a.	n.a.	0.005	n.a.	n.a.	0.005	0.005
ϵ_Σ	n.a.	n.a.	0.0005	n.a.	n.a.	0.0005	0.0005
trust region loss coef.	n.a.	n.a.	10	n.a.	n.a.	1	10
optimizer	adam	adam	adam	adam	adam	adam	adam
epochs	10	10	20	1000	1	50	100
learning rate	3e-4	1e-3	5e-5	3e-4	3e-4	3e-4	3e-4
use critic	True	True	True	True	True	True	True
epochs critic	10	10	10	1000	1	50	100
learning rate critic	3e-4	1e-3	3e-4	3e-4	3e-4	3e-4	3e-4
number minibatches	32	n.a.	64	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	500	n.a.	256	512	n.a.	n.a.
buffer size	n.a.	n.a.	n.a.	1e6	2e6	n.a.	n.a.
learning starts	0	0	0	10000	1e5	0	0
polyak_weight	n.a.	n.a.	n.a.	5e-3	5e-3	n.a.	n.a.
SDE sampling frequency	n.a.	4	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0	0	auto	auto	0	0
normalized observations	True	True	True	False	False	True	False
normalized rewards	True	True	False	False	False	False	False
observation clip	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	lin_0.3[1]	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	lin_0.3[1]	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[128, 128]	[128, 128]	[128, 128]	[256, 256]	[256, 256]	[128, 128]	[32, 32]
hidden layers critic	[128, 128]	[128, 128]	[128, 128]	[256, 256]	[256, 256]	[128, 128]	[32, 32]
hidden activation	tanh	tanh	tanh	relu	relu	relu	relu
orthogonal initialization	Yes	No	Yes	fanin	fanin	Yes	Yes
initial std	1.0	0.5	1.0	1.0	1.0	1.0	1.0
Movement Primitive (MP) type	n.a.	n.a.	n.a.	n.a.	n.a.	ProDMPs	ProDMPs
number basis functions	n.a.	n.a.	n.a.	n.a.	n.a.	8	5
weight scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.1	0.1
goal scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.1	0.1

[1]Linear Schedule from 0.3 to 0.01 during the first 25% of the training. Then continued with 0.01.

Table 3: Hyperparameters for the Box Pushing Dense, Episode Length $T = 100$

	PPO	gSDE	TRPL	SAC	PINK	TCE	BBRL	BBRL Cov.
number samples	48000	80000	48000	8	8	152	152	152
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.	n.a.
discount factor	1.0	1.0	1.0	0.99	0.99	1.0	1.0	1.0
ϵ_μ	n.a.	n.a.	0.005	n.a.	n.a.	0.05	0.1	0.05
ϵ_Σ	n.a.	n.a.	0.00005	n.a.	n.a.	0.0005	0.00025	0.0005
trust region loss coef.	n.a.	n.a.	10	n.a.	n.a.	1	10	10
optimizer	adam	adam	adam	adam	adam	adam	adam	adam
epochs	10	10	20	1	1	50	20	20
learning rate	5e-5	1e-4	5e-5	3e-4	3e-4	3e-4	3e-4	3e-4
use critic	True	True	True	True	True	True	True	True
epochs critic	10	10	10	1	1	50	10	10
learning rate critic	1e-4	1e-4	1e-4	3e-4	3e-4	1e-3	3e-4	3e-4
number minibatches	40	n.a.	40	n.a.	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	2000	n.a.	512	512	n.a.	n.a.	n.a.
buffer size	n.a.	n.a.	n.a.	2e6	2e6	n.a.	n.a.	n.a.
learning starts	0	0	0	1e5	1e5	0	0	0
polyak_weight	n.a.	n.a.	n.a.	5e-3	5e-3	n.a.	n.a.	n.a.
SDE sampling frequency	n.a.	4	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0.01	0	auto	auto	0	0	0
normalized observations	True	True	True	False	False	True	False	False
normalized rewards	True	True	False	False	False	False	False	False
observation clip	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	0.2	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	0.2	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[512, 512]	[256, 256]	[512, 512]	[256, 256]	[256, 256]	[128, 128]	[128, 128]	[128, 128]
hidden layers critic	[512, 512]	[256, 256]	[512, 512]	[256, 256]	[256, 256]	[256, 256]	[256, 256]	[256, 256]
hidden activation	tanh	tanh	tanh	tanh	tanh	leaky_relu	leaky_relu	leaky_relu
orthogonal initialization	Yes	No	Yes	fanin	fanin	Yes	Yes	Yes
initial std	1.0	0.05	1.0	1.0	1.0	1.0	1.0	1.0
MP type	n.a.	n.a.	n.a.	n.a.	n.a.	ProDMPs	ProDMPs	ProDMPs
number basis functions	n.a.	n.a.	n.a.	n.a.	n.a.	8	8	8
weight scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.3	0.3	0.3
goal scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.3	0.3	0.3

Table 4: Hyperparameters for the Box Pushing Sparse, Episode Length $T = 100$

	PPO	gSDE	TRPL	SAC	PINK	TCE	BBRL	BBRL Cov.
number samples	48000	80000	48000	8	8	76	76	76
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.	n.a.
discount factor	1.0	1.0	1.0	0.99	0.99	1.0	1.0	1.0
ϵ_μ	n.a.	n.a.	0.005	n.a.	n.a.	0.05	0.1	0.05
ϵ_Σ	n.a.	n.a.	0.00005	n.a.	n.a.	0.0005	0.00025	0.001
trust region loss coef.	n.a.	n.a.	10	n.a.	n.a.	1	10	10
optimizer	adam	adam	adam	adam	adam	adam	adam	adam
epochs	10	10	20	1	1	50	20	20
learning rate	5e-4	1e-4	5e-5	3e-4	3e-4	3e-4	3e-4	3e-4
use critic	True	True	True	True	True	True	True	True
epochs critic	10	10	10	1	1	50	10	10
learning rate critic	1e-4	1e-4	1e-4	3e-4	3e-4	3e-4	3e-4	3e-4
number minibatches	40	n.a.	40	n.a.	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	2000	n.a.	512	512	n.a.	n.a.	n.a.
buffer size	n.a.	n.a.	n.a.	2e6	2e6	n.a.	n.a.	n.a.
learning starts	0	0	0	1e5	1e5	0	0	0
polyak_weight	n.a.	n.a.	n.a.	5e-3	5e-3	n.a.	n.a.	n.a.
SDE sampling frequency	n.a.	4	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0.01	0	auto	auto	0	0	0
normalized observations	True	True	True	False	False	True	False	False
normalized rewards	True	True	False	False	False	False	False	False
observation clip	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	0.2	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	0.2	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[512, 512]	[256, 256]	[512, 512]	[256, 256]	[256, 256]	[128, 128]	[128, 128]	[128, 128]
hidden layers critic	[512, 512]	[256, 256]	[512, 512]	[256, 256]	[256, 256]	[256, 256]	[256, 256]	[256, 256]
hidden activation	tanh	tanh	tanh	tanh	tanh	leaky_relu	leaky_relu	leaky_relu
orthogonal initialization	Yes	No	Yes	fanin	fanin	Yes	Yes	Yes
initial std	1.0	0.05	1.0	1.0	1.0	1.0	1.0	1.0
MP type	n.a.	n.a.	n.a.	n.a.	n.a.	ProDMPs	ProDMPs	ProDMPs
number basis functions	n.a.	n.a.	n.a.	n.a.	n.a.	8	8	8
weight scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.3	0.3	0.3
goal scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.3	0.3	0.3

Table 5: Hyperparameters for the Hopper Jump, Episode Length $T = 250$

	PPO	gSDE	TRPL	SAC	PINK	TCE	BBRL	BBRL Cov.
number samples	8000	8192	8000	1000	1	64	64	64
GAE λ	0.95	0.99	0.95	n.a.	n.a.	0.95	n.a.	n.a.
discount factor	1.0	0.999	1.0	0.99	0.99	1.0	1.0	1.0
ϵ_μ	n.a.	n.a.	0.05	n.a.	n.a.	0.1	n.a.	0.005
ϵ_Σ	n.a.	n.a.	0.0005	n.a.	n.a.	0.02	n.a.	0.00005
trust region loss coef.	n.a.	n.a.	10	n.a.	n.a.	1	n.a.	10
optimizer	adam	adam	adam	adam	adam	adam	adam	adam
epochs	10	10	20	1000	1	50	100	100
learning rate	3e-4	9.5e-5	3e-4	1e-4	2e-4	1e-4	1e-4	1e-4
use critic	True	True	True	True	True	True	True	True
epochs critic	10	10	10	1000	1	50	100	100
learning rate critic	3e-4	9.5e-5	3e-4	1e-4	2e-4	1e-4	1e-4	1e-4
number minibatches	40	n.a.	40	n.a.	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	128	n.a.	256	256	n.a.	n.a.	n.a.
buffer size	n.a.	n.a.	n.a.	1e6	1e6	n.a.	n.a.	n.a.
learning starts	0	0	0	10000	1e5	0	0	0
polyak_weight	n.a.	n.a.	n.a.	5e-3	5e-3	n.a.	n.a.	n.a.
SDE sampling frequency	n.a.	8	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0.0025	0	auto	auto	0	0	0
normalized observations	True	False	True	False	False	True	False	False
normalized rewards	True	False	False	False	False	False	False	False
observation clip	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	lin_0.4	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	lin_0.4	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[32, 32]	[256, 256]	[32, 32]	[256, 256]	[32, 32]	[128, 128]	[32, 32]	[32, 32]
hidden layers critic	[32, 32]	[256, 256]	[32, 32]	[256, 256]	[32, 32]	[128, 128]	[32, 32]	[32, 32]
hidden activation	tanh	tanh	tanh	relu	relu	leaky_relu	tanh	tanh
orthogonal initialization	Yes	No	Yes	fanin	fanin	Yes	Yes	Yes
initial std	1.0	0.1	1.0	1.0	1.0	1.0	1.0	1.0
MP type	n.a.	n.a.	n.a.	n.a.	n.a.	ProDMPs	ProDMPs	ProDMPs
number basis functions	n.a.	n.a.	n.a.	n.a.	n.a.	3	3	3
weight scale	n.a.	n.a.	n.a.	n.a.	n.a.	1	1	1
goal scale	n.a.	n.a.	n.a.	n.a.	n.a.	1	1	1

Table 6: Hyperparameters for the Table Tennis, Episode Length $T = 300$

	PPO	gSDE	TRPL	SAC	PINK	TCE	BBRL	BBRL Cov.
number samples	48000	24000	48000	8	8	76	76	76
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.	n.a.
discount factor	1.0	1.0	1.0	0.99	0.99	1.0	1.0	1.0
ϵ_μ	n.a.	n.a.	0.005	n.a.	n.a.	0.005	0.004	0.005
ϵ_Σ	n.a.	n.a.	0.0005	n.a.	n.a.	0.00025	0.000025	0.001
trust region loss coef.	n.a.	n.a.	10	n.a.	n.a.	1	25	25
optimizer	adam	adam	adam	adam	adam	adam	adam	adam
epochs	10	10	20	1	1	50	100	100
learning rate	5e-5	1e-4	5e-5	3e-4	3e-4	3e-4	3e-4	3e-4
use critic	True	True	True	True	True	True	True	True
epochs critic	10	10	10	1	1	50	100	100
learning rate critic	1e-4	1e-4	1e-4	3e-4	3e-4	3e-4	3e-4	3e-4
number minibatches	40	n.a.	40	n.a.	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	4000	n.a.	512	512	n.a.	n.a.	n.a.
buffer size	n.a.	n.a.	n.a.	4e6	4e6	n.a.	n.a.	n.a.
learning starts	0	0	0	1e5	1e5	0	0	0
polyak_weight	n.a.	n.a.	n.a.	5e-3	5e-3	n.a.	n.a.	n.a.
SDE sampling frequency	n.a.	8	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0	0	auto	auto	0	0	0
normalized observations	True	True	True	False	False	True	False	False
normalized rewards	True	True	False	False	False	False	False	False
observation clip	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	0.2	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	0.2	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[512, 512]	[256, 256]	[256, 256]	[256, 256]	[256, 256]	[256]	[256]	[256]
hidden layers critic	[512, 512]	[256, 256]	[512, 512]	[256, 256]	[256, 256]	[256, 256]	[256]	[256]
hidden activation	tanh	tanh	tanh	tanh	tanh	tanh	tanh	tanh
orthogonal initialization	Yes	Yes	Yes	fanin	fanin	Yes	Yes	Yes
initial std	1.0	0.1	1.0	1.0	1.0	1.0	1.0	1.0
MP type	n.a.	n.a.	n.a.	n.a.	n.a.	ProDMPs	ProDMPs	ProDMPs
number basis functions	n.a.	n.a.	n.a.	n.a.	n.a.	3	3	3
weight scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.7	0.7	0.7
goal scale	n.a.	n.a.	n.a.	n.a.	n.a.	0.1	0.1	0.1