Reparameterized Policy Learning for Multimodal Trajectory Optimization

Zhiao Huang [1] **Litian Liang** [1] **Zhan Ling** [1] **Xuanlin Li** [1] **Chuang Gan** [2 3] **Hao Su** [1]

Abstract

We investigate the challenge of parametrizing policies for reinforcement learning (RL) in high-dimensional continuous action spaces. Our objective is to develop a multimodal policy that overcomes limitations inherent in the commonly-used Gaussian parameterization. To achieve this, we propose a principled framework that models the continuous RL policy as a generative model of optimal trajectories. By conditioning the policy on a latent variable, we derive a novel variational bound as the optimization objective, which promotes exploration of the environment. We then present a practical model-based RL method, called Reparameterized Policy Gradient (RPG), which leverages the multimodal policy parameterization and learned world model to achieve strong exploration capabilities and high data efficiency. Empirical results demonstrate that our method can help agents evade local optima in tasks with dense rewards and solve challenging sparse-reward environments by incorporating an object-centric intrinsic reward. Our method consistently outperforms previous approaches across a range of tasks. Code and supplementary materials are available on the project page https://haosulab.github.io/RPG/

1. Introduction

Reinforcement learning (RL) with *high-dimensional continuous action space* is notoriously hard despite its fundamental importance for many application problems such as robotic manipulation (OpenAI et al., 2019; Mu et al., 2021). In practice, popular frameworks (Silver et al., 2014; Haarnoja et al., 2018; Schulman et al., 2017) of deep RL formulate the continuous policy as a neural network that outputs a single-modal density function over the action space (e.g., a Gaussian distribution over actions). This formulation, however, breaks the promise of RL being a global optimizer of the return function because the single-modality policy parameterization introduces local minima that are hard to escape using gradients w.r.t. distribution parameters. Besides, a single-modality policy will significantly weaken the exploration ability of RL algorithms because the sampled actions are usually concentrated around the modality.

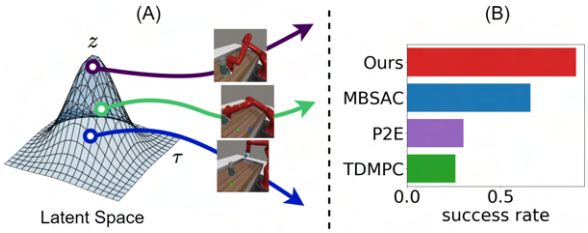


Figure 1. (A) Our method reparameterizes latent variables into multimodal policy to facilitate exploitation and exploration in continuous policy learning; (B) Average performance on 6 hard exploration tasks. Our method outperforms previous methods.

Although there are other candidates beyond the Gaussian distribution for policy parameterization, they often have limitations when used for continuous policy modeling. For example, Gaussian mixture models can only accommodate a limited number of modes; normalizing flow methods (Rezende & Mohamed, 2015) can compute density values, but they may not be as numerically robust due to their dependency on the determinant of the network Jacobian; furthermore, normalizing flows must apply continuous transformations onto a continuously connected distribution, making it difficult to model disconnected modes (Rasul et al., 2021). Option-critic (Bacon et al., 2017) represents policies with options and temporal structure, but it often requires specially designed option spaces for efficient learning, which motivates research on hierarchical imitation learning that uses demonstrations to avoid exploration problems (Peng et al., 2022; Fang et al., 2019). Skill discovery methods learn a population of skills without demonstrations or rewards by optimizing for diversity (Eysenbach et al., 2018). However, the separation of optimization and skill learning can be non-efficient as it expends effort on learning task-irrelevant skills and may ignore more important ones that would benefit a

[1]UC San Diego [2]MIT-IBM Watson AI Lab [3]UMass Amherst. Correspondence to: Zhiao Huang <z2huang@ucsd.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

specific task.

This paper presents a principled framework for learning the continuous RL policy as a multimodal density function through multimodal action parameterization. We adopt a sequence modeling perspective (Chen et al., 2021) and view the policy as a density function over the entire trajectory space (instead of the action space)(Ziebart, 2010; Levine, 2018). This allows us to sample a population of trajectories that cover multiple modalities, enabling concurrent exploration of distant regions in the solution space. Additionally, we use a generative model to parameterize the multimodal policies, drawing inspiration from their success in modeling highly complex distributions such as natural images(Goodfellow et al., 2016; Zhu et al., 2017; Rombach et al., 2022; Ramesh et al., 2021). We condition the policy on a latent variable z and use a powerful function approximator to "reparameterize" the random distribution z into the multimodal trajectory distribution (Kingma & Welling, 2013), from which we can sample trajectories τ. This policy parameterization leads us to adopt the variational method (Kingma & Welling, 2013; Haarnoja et al., 2018; Moon, 1996) to derive a novel framework for modeling the posterior of the optimal trajectory using variational inference, which enables us to model multimodal trajectories and maximize the reward with a single objective.

This framework allows us to build Reparameterized Policy Gradient (RPG), a model-based RL method for multimodal trajectory optimization. The framework has two notable features: First, RPG combines the multimodal policy parameterization with a learned world model, enjoying the sample efficiency of the learned model and gradient-based optimization while providing the additional ability to jump out of the local optima; Second, we equip RPG with a novel density estimator to help the multimodal policy explore in the environments by maximizing the state entropy (Hazan et al., 2019). We verify the effectiveness of our methods on several robot manipulation tasks. These environments only provide sparse rewards when the agent successfully fully finishes the task, which is challenging for single-modal policies even when they are guided by intrinsic motivations. In comparison, our method is able to explore different modalities, improve the exploration efficiency, and outperform single-modal policies, as shown in Fig. 1. Notably, our method is more robust than single-modal policies and consistently outperforms previous approaches across different tasks.

Our contributions are multifold: 1. We propose a variational policy learning framework that models the posterior of multimodal optimal trajectories for reward optimization. 2. We demonstrate that multimodal parameterization can help the policy escape local optima and accelerate exploration in continuous policy optimization. 3. When combined with a learned world model and a delicate density estimator, our method, RPG, is able to solve these challenging sparse-reward tasks more efficiently and reliably.

2. Related Work

Policy as Sequential Generative Model. Maximum entropy reinforcement learning (Todorov, 2006; 2008; Toussaint, 2009; Ziebart, 2010; Kappen et al., 2012) can be viewed as variational inference in probabilistic graphical models (Levine, 2018) with optimality as an observed variable and sampled trajectories as latent variables. When the demonstration or a fixed dataset is provided in the *offline* RL setting (Chen et al., 2021; Reed et al., 2022), policy learning is simplified as a sequence modeling task (Chen et al., 2021; Zheng et al.; Reed et al., 2022). They use autoregressive models to learn the distribution of the whole trajectory, including actions, states, and rewards, and use the action prediction as policy. In our work, we learn a sequential generative model of policy for *online* RL via the variational method.

Variational Skill Discovery Under additional assumptions of rewards, our method degenerates to skill discovery methods. However, previous skill discovery methods focus on unsupervised reinforcement learning (Eysenbach et al., 2018; Achiam et al., 2018; Campos et al., 2020) or diverse skill learning (Kumar et al., 2020; Osa et al., 2022). These methods build latent variable policy and encourage the policy to reach states that are consistent with the sampled latent variables through a mutual information term as a reward. These methods do not consider reward maximization or exploration when learning the skills, making them differ from our method vastly. For example, Eysenbach et al. (2018); Achiam et al. (2018) does not optimize the learned skill for the environment rewards; Osa et al. (2022) does not optimize the mutual information along trajectories; Kumar et al. (2020) needs to solve the optimization problem first before finding a diverse set of solutions. Moreover, these methods fix the latent distributions, limiting their ability to achieve optimality when rewards are given. Mazzaglia et al. (2022) also learns skills within a learned world model. However, it decouples the exploration and skill learning and needs offline data or data generated from other exploration policies to train the model. In contrast, we are motivated by the parameterization problems in *online* RL and jointly optimize the latent representation to model *optimal* trajectories. We show that learning a latent variable model benefits optimization and exploration and they can be considered together.

Hierarchical Methods The hierarchical methods, e.g., option-critic (Bacon et al., 2017), can be regarded as a special way of policy parameterization by conditioning

the lower-level policy over a sequence of latent variables $z = (z_1, \cdots, z_T)$. Usually, most hierarchical RL methods need special designs for the latent space, e.g., state-based subgoals (Kulkarni et al., 2016; Nachum et al., 2018b;a) or predefined skills (Li et al., 2020) to avoid mode-collapse. Osa et al. (2019) regularized options to maximize the mutual information between the action and the options, which are very relevant to ours. However, it does not model temporal structures as ours to ensure consistency along the trajectories. Goal-conditioned RL (Andrychowicz et al., 2017; Mendonca et al., 2021; Nachum et al., 2018b) can also be considered a special hierarchical method that uses states or goals to help parameterize the policy and has been proven efficient in exploration, but designing the goal space, sampling and generating goals in high-dimensional space is non-trivial. The specific reward design of goal-reaching tasks also makes extending goal-conditioned policies to general reward functions not easy.

Hierarchical imitation learning (Gupta et al., 2019; Pertsch et al., 2021; Shankar & Gupta, 2020; Jiang et al., 2022; Lynch et al., 2020; Fang et al., 2020) extracts temporal abstractions from demonstrations using generative models. For example, InfoGAN (Li et al., 2017) and ASE (Peng et al., 2022) use adversarial training (Goodfellow et al., 2020; Ho & Ermon, 2016) to imitate demonstrations. These works all rely on demonstrations rather than rewards to learn abstractions. Co-Reyes et al. (2018) learns representation on the collected dataset with variational inference and then utilizes the trained model for planning or policy learning. The separation of the representation learning and reward maximization makes it differ from our methods: first, it requires a state reconstruction module to supervise the generative model, which is challenging for high-dimensional observations; second, it optimizes neither the latent distribution nor the actions for the reward directly, thus requires additional planning procedure during the execution to find suitable actions.

3. Preliminary

Markov decision process A Markov decision process (MDP) is a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. $p(s'|s,a)$ is the transition probability that transits state s to another state s' after taking action a. The function $R(s,a,s')$ computes a reward per transition. A policy $\pi(a|s)$ outputs an action distribution according to the state s. Executing a policy π starting from the initial state s_1 with density $p(s_1)$ will result in a *trajectory* τ, which is a sequence of states and actions $\{s_1, a_1, s_2, \ldots, s_t, a_t, \ldots\}$ where $a_t \sim \pi(a|s = s_t), s_{t+1} \sim p(s|s = s_t, a = a_t)$. We also use the terminology *environment* to refer to an MDP in an RL problem. The discounted reward of a trajectory is $R_\gamma(\tau) = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$ where $0 < \gamma < 1$ is the discount factor to ensure the series converges. The goal of reinforcement learning (RL) is to find a parameterized policy π_θ that maximizes the expected reward $E_{s_1 \sim p(s_1)}[V^{\pi_\theta}(s_1)] = E_{\tau \sim \pi_\theta, s_1 \sim p(s_1)}[R_\gamma(\tau)]$, where V^{π_θ} is the value function. Many environments have an observation space $\mathcal{O}$ that is not the same to the state space. In this case the agent may need to identify the state s_t from the observation o_t.

RL as probabilistic inference The RL as inference framework (Todorov, 2006; 2008; Toussaint, 2009; Ziebart, 2010; Kappen et al., 2012; Levine, 2018) defines optimality $p(O|\tau) \propto e^{R(\tau)/\mathcal{T}}$, where $\mathcal{T}$ is a temperature scalar and $R(\tau)$ is the total rewards of the trajectory τ. It further defines a prior distribution of the trajectory $p(\tau) = p(s_1) \prod_{t=1}^{T} p(a_t|s_t) p(s_{t+1}|s_t, a_t)$, where $p(a_t|s_t)$ is a known prior action distribution, e.g., a Gaussian distribution. Thus, it can compute the density of optimality $p(O) = \int p(O|\tau) p(\tau) d\tau$. The goal of the framework is to approximate the posterior distribution of optimal trajectories $p(\tau|O) = \frac{p(O|\tau)p(\tau)}{\int p(O|\tau)p(\tau)d\tau}$. In the maximum entropy framework (Haarnoja et al., 2017), one can apply evidence lower bound (Kingma & Welling, 2013) $\log p(O) \geq \mathbb{E}_{\tau \sim \pi} [\log p(O|\tau) + \log p(\tau) - \log \pi(\tau)]$ to train the model.

4. Method

To overcome the limitations of single modality policies, we propose to use latent variables to parameterize multimodal policies in Sec. 4.1. We then propose a novel variational bound as the optimization objective to approximate the posterior of optimal trajectories in Sec. 4.2. The variational bound naturally combines maximum entropy RL and includes a term to encourage consistency (Zhu et al., 2017) between the latent distribution and the sampled trajectories, preventing the policy from mode collapse. To optimize this objective in hard continuous control problems, we propose to learn a world model and build the Reparameterized Policy Gradient, a model-based latent variable policy learning framework in Sec. 4.3.1. We design intrinsic rewards in Sec. 4.3.2 to facilitate exploration. Figure 3 illustrates the whole pipeline.

4.1. Reparameterize Latent Variables for Multimodal Policy Learning

Policy parameterization matters. In continuous RL, it is popular to model action distribution with a unimodal Gaussian distribution. However, theoretically, to make sure that the optimal policy will be captured by RL, the function class of continuous RL policies has to include density functions of arbitrary probabilistic distributions (Sutton & Barto,

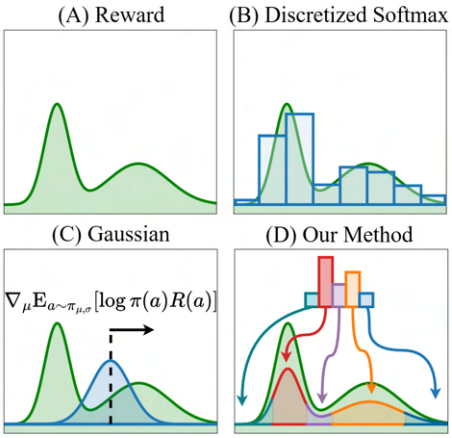


Figure 2. (A) rewards; (B); soft max policy over discrete action space; (C) single-modality Gaussian policy; (D) our methods reparameterize a random variable into multimodal distributions with neural networks.

2018). Consider maximizing a continuous reward function with two modalities as shown in Figure 2(A). When the action space is properly discretized, a SoftMax policy can model the multimodal distribution and find the global optimum after sampling over the entire action space as shown in Figure 2(B). However, discretization can lead to a loss of accuracy and efficiency. If we instead use a Gaussian policy $\mathcal{N}(\mu, \sigma^2)$ by the common practice in literature, we will have trouble – as shown in Figure 2(C), even if its standard deviation is so large to well cover both modalities, the policy gradient can push it towards the local optimum on the right side, causing it to fail to converge to the global optimum. To address the issue, a more flexible policy parameterization is needed for continuous RL problems, one that is simple to sample and optimize.

Multimodal policy by reparameterizing latent variables Motivated by recent developments in generative models that have shown superiority in modeling complex distributions (Kingma & Welling, 2013; Ho et al., 2020; Rombach et al., 2022; Ramesh et al., 2021), we propose to parameterize policies using latent variables, as illustrated in Figure 2(D). Instead of adding random noise to perturb network outputs to generate an action distribution, we build a generative model of policy distribution by taking random noise as input and relying on powerful neural networks to transform it into actions of various modalities.

Formally, let $z \in \mathcal{Z}$ be a random variable, which can be either continuous or categorical. We design our "policy" as a joint distribution $\pi_\theta(z, \tau)$ of the latent z and the trajectory τ. This paper considers a particular factorization of $\pi_\theta(z, \tau)$ that samples z in the beginning of each episode and then sample trajectory τ conditioning on z:

$$\pi_\theta(z, \tau) = p(s_1)\pi_\theta(z|s_1) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|z, s_t) \quad (1)$$

where T is the length of the sampled trajectory.

One can use the policy gradient theorem (Sutton & Barto, 2018), i.e., $\nabla J(\pi) = \mathbb{E}_\tau[R(\tau)\nabla \log p(\tau)]$ to optimize the generative model policy. However, computing $p(\tau)$ needs to marginalize over z, i.e., computing $\int_z p(z, \tau)\, \mathrm{d}z$, which is often intractable when z is continuous. Besides, optimizing the marginal distribution $\log\, p(\tau)$ by gradient descent suffers from local optimality issues (e.g., using gradient descent to optimize Gaussian mixture models which have latent variables is not effective, so EM is often used instead (Ng, 2000)).

4.2. Variational Inference for Optimal Trajectory Modeling

To overcome these obstacles, following Todorov (2006; 2008); Toussaint (2009); Ziebart (2010); Kappen et al. (2012); Levine (2018); Haarnoja et al. (2018), we adopt variational method (maximum entropy RL) to directly optimize the joint distribution of the optimal policy without hassles of integrating over z.

The evidence lower bound We learn $\pi_\theta(z, \tau)$ using variational inference (Kingma & Welling, 2013; Haarnoja et al., 2018; Moon, 1996). Like an EM algorithm, we define an auxiliary distribution $p_\phi(z|\tau)$ to approximate the posterior distribution of z conditioning on τ using function approximators. This auxiliary distribution $p_\phi(z|\tau)$ helps to factorize the joint distribution of optimality O, latent z, and the trajectory τ as $p_\phi(O, z, \tau) = p(O|\tau)p_\phi(z|\tau)p(\tau)$. Treating $\pi_\theta(z, \tau)$ as the variational distribution, we can write the Evidence Lower Bound (ELBO) for the optimality O:

$$\begin{aligned}
&\log p(O) \\
&= \underbrace{E_{z,\tau\sim\pi_\theta}\left[\log p_\phi(O, z, \tau) - \log \pi_\theta(z, \tau)\right]}_{\text{ELBO}} \\
&+ D_{KL}(\pi_\theta(z, \tau)||p_\phi(z, \tau|O)) \\
&\geq E_{z,\tau\sim\pi_\theta}\left[\log p_\phi(O, \tau, z) - \log \pi_\theta(z, \tau)\right] \\
&= E_{z,\tau\sim\pi_\theta}\left[\log p(O, \tau) + \log p_\phi(z|\tau) - \log \pi_\theta(z, \tau)\right] \\
&= E_{z,\tau}\left[\underbrace{\log p(O|\tau)}_{\text{reward}} + \underbrace{\log p(\tau)}_{\text{prior}} + \underbrace{\log p_\phi(z|\tau)}_{\text{cross entropy}} - \underbrace{\log \pi_\theta(z, \tau)}_{\text{entropy}}\right]
\end{aligned} \quad (2)$$

If we optimize $\pi_\theta(z, \tau)$ and $p_\phi(z|\tau)$ using the gradient of the variational bound, the variational distribution $\pi_\theta(z, \tau)$ learns to model the optimal trajectory distribution $p(\tau|O)$.

How it works ELBO contains four parts that can all be computed directly given the sampled z and τ (the environment probability $p(s_{t+1}|s_t, a_t)$ is canceled as in (Levine, 2018)). The first two parts are the predefined reward $\log p(O|\tau) = R(\tau)/\mathcal{T} + c$, where $\mathcal{T}$ is the temperature scalar, and c is the normalizing constant that can be ignored in optimization. The prior distribution $p(\tau)$ is assumed to be known. The third part is the log-likelihood of z, defined by our auxiliary distribution $p_\phi(z|\tau)$. It is easy to see that if we fix π_θ, maximize p_ϕ alone will minimize the cross-entropy $E_{z,\tau\sim\pi_\theta}[-\log p_\phi(z|\tau)]$, similar to the supervised learning of predicting z given τ. This achieves optimality when $p_\phi(z|\tau) = p_\theta(z|\tau) = \frac{\pi_\theta(z,\tau)}{\int_z \pi_\theta(z,\tau)dz}$, modeling the posterior of z for τ sampled from π_θ. On the other hand, by fixing ϕ, the policy π_θ is encouraged to generate trajectories that are easy to identify or classify; this helps to increase diversity and enforce consistency to avoid mode collapse, letting the network not ignore the latent variables. The fourth part is the policy entropy that enables maximum entropy exploration. Maximizing all terms together for the parameters θ and ϕ will minimize $D_{KL}(\pi_\theta(z,\tau)||p_\phi(z,\tau|O)) = D_{KL}(\pi_\theta(z,\tau)||p_\phi(z|\tau)p(\tau|O))$. The optimality can be achieved when $p_\phi(z|\tau)$ equals to $p(z|\tau)$, the true posterior of z. Then, $p_\theta(\tau) = p_\phi(z|\tau)p(\tau|O)/p(z|\tau) = p(\tau|O)$ where $p_\theta(\tau) = \int \pi_\theta(\tau, z)dz$ is the marginal distribution of τ sampled from π_θ.

Relationship with other methods Our method is closely related to skill discovery methods (Eysenbach et al., 2018; Mazzaglia et al., 2022). A skill discovery method usually uses mutual information $I(\tau, z) = H(\tau) - H(\tau|z)$ or $H(z) - H(z|\tau) \geq E_{z,\tau}[\log p_\phi(z|\tau) - \log p(z)]$ to encourage diversity. For example, DIYAN (Eysenbach et al., 2018) directly optimizes mutual information to learn various skills without reward. Dropping out the reward term in Eq. 2 shows that the skill learning objective can be seamlessly embedded into the "RL as inference" framework with external reward, and there is no need to introduce the mutual information term manually. Furthermore, the framework suggests we can model the posterior of the optimal trajectories, which enables us to unify generative modeling and trajectory optimization in a single framework. As for the relationship of our method with other generative models, we refer readers to a more thorough discussion in Appendix F.

4.3. Reparameterized Policy Gradient for Model-based Exploration

We now describe Reparameterized Policy Gradient (RPG), a model-based RL method with intrinsic motivation for sample efficient exploration in continuous control environments. We first simplify the right side of Eq. 2 using the factorization in Eq. 1 and assuming $\log p_\phi(z|\tau) = \sum_{t>0} \log p(z|s_t, a_t)$. Thus, the ELBO becomes $-\log \pi_\theta(z|s_1) + \sum_{t=1}^{\infty} R(s_t, a_t)/\mathcal{T} - \log \pi_\theta(a_t|s_t, z) + \log p_\phi(z|s_t, a_t)$, which can be optimized with an RL algorithm by maximizing the reward

$$\underbrace{R(s_t, a_t)/\mathcal{T}}_{r_t} \underbrace{-\alpha \log \pi_\theta(a_t|s_t, z) + \beta \log p_\phi(z|s_t, a_t)}_{r'_t},$$

where scalars α, β control the exploration and consistency. We use neural networks to model $\log p_\phi(z|s_t, a_t)$ and $\pi_\theta(a_t|s_t, z)$.

4.3.1. Model-based RL with Latent Variables

In our method Reparameterized Policy Gradient (RPG), we train a differentiable world model (Hafner et al., 2019; Schrittwieser et al., 2020; Ye et al., 2021; Hansen et al., 2022) to improve data efficiency. The world model contains the following components: observation encoder $s_t = f_\psi(o_t)$, reward predictor $r_t = R_\psi(s_t, a_t)$, Q value $Q_t = Q_\psi(s_t, a_t, z)$ and dynamics $s_{t+1} = h_\psi(s_t, a_t)$.

Given any z and latent state $s_{t_0} = f_\psi(o_{t_0})$ at time step t_0, the learned dynamics network can generate an imaginary trajectory for any action sequence. If we sample actions from the policy $\pi_\theta(a_t|s_t, z)$ for $t \geq t_0$ and execute them in the latent model, it will produce a Monte-Carlo estimate for the value of s_{t_0} for optimizing the policy π_θ:

$$V_{\text{est}}(o_{t_0}, z) \approx \gamma^K (Q_{t_0+K} + r'_{t_0+K}) + \sum_{t=t_0}^{t_0+K-1} \gamma^{t-t_0}(r_t + r'_t) \tag{3}$$

We self-supervise the dynamics network to ensure state consistency without reconstructing observations as in (Ye et al., 2021; Hansen et al., 2022). For any latent variable z and trajectory segments of length $K+1$ $\tau_{t_0:t_0+K} = \{o_{t_0}, a^{gt}_{t_0}, r^{gt}_{t_0}, o_{t_0+1}, \ldots, o_{t_0+K}\}$ sampled from the replay buffer, we execute actions $\{a^{gt}_t\}$ in the world model and use the following loss function to train the world model, as well as the Q function:

$$L_\psi(\tau) = \sum_{t=t_0}^{t_0+K-1} L_1 \|s_{t+1} - \mathbf{ng}(f_\psi(o_{t+1}))\|^2 + L_2(r_t - r^{gt}_t)^2 + L_3(Q_t - \mathbf{ng}(r^{gt}_t + \gamma V_{\text{est}}(o_{t+1}, z)))^2 \tag{4}$$

where $\mathbf{ng}(x)$ means stopping gradient and $L_1 = 1000, L_2 = L_3 = 0.5$ are constants to balance the loss.

4.3.2. Maximize State Entropy with Object-centric Randomized Network Distillation

For challenging continuous control tasks with sparse rewards, policies that maximize the action entropy of $\pi_\theta(a|s, z)$ usually have trouble obtaining a meaningful reward, making its exploration inefficient. We follow (Hazan et al., 2019) to let the policy additionally maximize

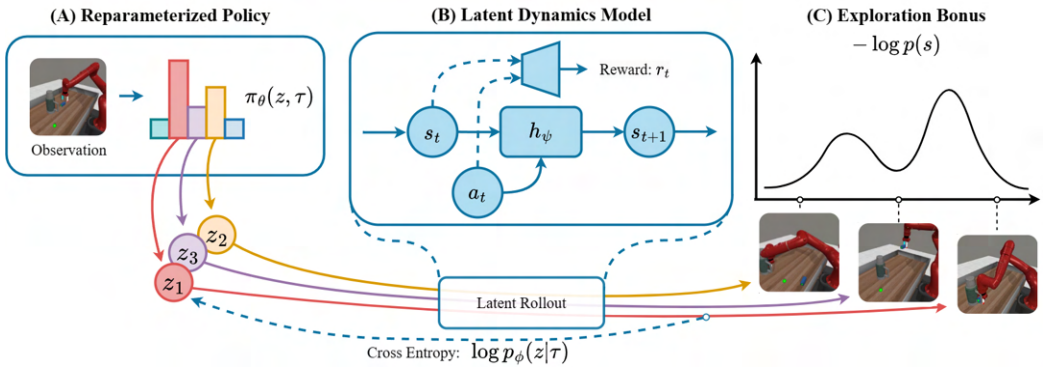


Figure 3. An overview of our model pipeline: A) a reparameterized policy from which we can sample latent variable z and action a given the latent state s; B) a latent dynamics model which can be used to forward simulate the dynamic process when a sequence of actions is known. C) an exploration bonus provided by a density estimator. Our Reparameterized Policy Gradient do multimodal exploration with the help of the latent world model and the exploration bonus.

the entropy of the discounted stationary state distribution $d_\pi(s) = (1-\gamma)\sum_{t=1}^{\infty}\gamma^t P(s_t = s|\pi)$.

We use the *object-centric* Randomized Network Distillation (RND) (Burda et al., 2018) as a simple and effective method to approximate the state density in continuous control tasks. RND uses a network $g_\theta(o_t)$ to distill the output of a random network $g'(o_t)$ by minimizing the difference $\|g_\theta(o_t) - g'(o_t)\|^2$ over states sampled by the current agent and treat the difference as the negative density of each observation o_t.

We make several modifications to the vallina RND to improve its performance for state vector observations in control problems. First, we inject object-prior to the RND estimator to make the policy sensitive to regions that include objects' position change. Specifically, before feeding objects' coordinates into the network, we apply positional encoding (Vaswani et al., 2017; Mildenhall et al., 2021) to turn all scalars x to a vector of $\{\sin(2^i x), \cos(2^i x)\}_{i=1,2,...}$ for objects of interest (e.g., in robot manipulation, the end effector of the robot and the object). Second, we use a large replay buffer to store past states to avoid catastrophic forgetting (Zhang et al., 2021). We verified that it is necessary to normalize the RND's output to stabilize the training and make it an approximated density estimator. Lastly, to account for the latent world model, we relabel trajectories' rewards sampled from the replay buffer instead of estimating them directly in the latent model by reconstructing the observation.

An implicit benefit of a latent variable policy model is its ability to maximize the state entropy better, as will be shown in the experiments of Sec. 5.1. When combined with our RND method, RPG achieves much better state coverage while single modality policy cannot stabilize. The combination of multimodal policy learning and state entropy maximization accelerates the exploration of continuous control tasks with sparse rewards. We describe the whole algorithm in Alg. 1 and implementation details in Appendix A.

5. Experiments

In this section, we first illustrate the potential of RPG in optimization and exploration through two example tasks. We then show that our method can help solve hard continuous control problems, even with only sparse rewards. We ablate essential design choices and provide additional experiments in section 5.3.

5.1. Illustrative Experiments

Can multimodal policies help escape local optima? We study the effects of our method on a 1D bandit problem as shown in Fig. 4. It has a 1d action space and a non-convex reward landscape with an additional discontinuous point.

Fig. 4 compares the performance of our method with a single modality Gaussian policy optimized by REINFORCE. *Notice that we do not add the intrinsic reward for dense reward maximization tasks*. The Gaussian policy, initialized at 0 with a large standard deviation, can cover the whole solution space. However, the gradient w.r.t μ is positive, which means the action probability density will be pushed towards the right, as the expected return on the right side is larger than the left side, although the left side contains a higher extreme value. As a result, the policy will move right and get stuck at the local optimum with a low chance of jumping out. In contrast, under the entropy maximization formulation, our method maximizes the reward while seeking to increase diversity, providing more chances for the policy to explore the whole solution space. Furthermore, by turning the latent variables into action distribution, our method can build a multimodal policy distribution that fits the multimodal rewards, explore both modalities simultaneously, and

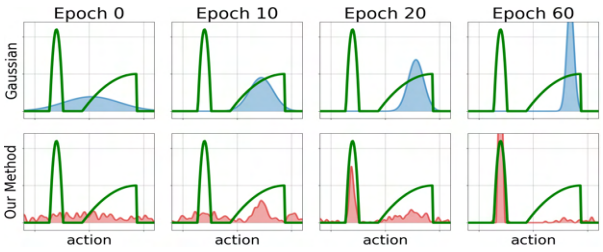


Figure 4. Illustrative experiment on continuous bandit

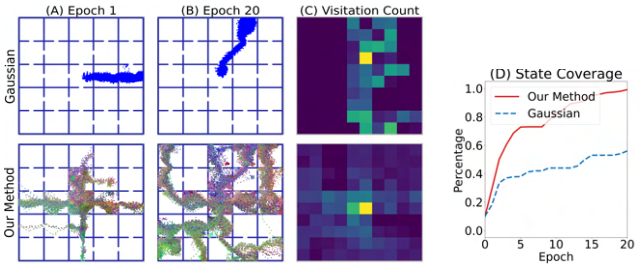


Figure 5. Illustrative experiment on 2D maze navigation problem

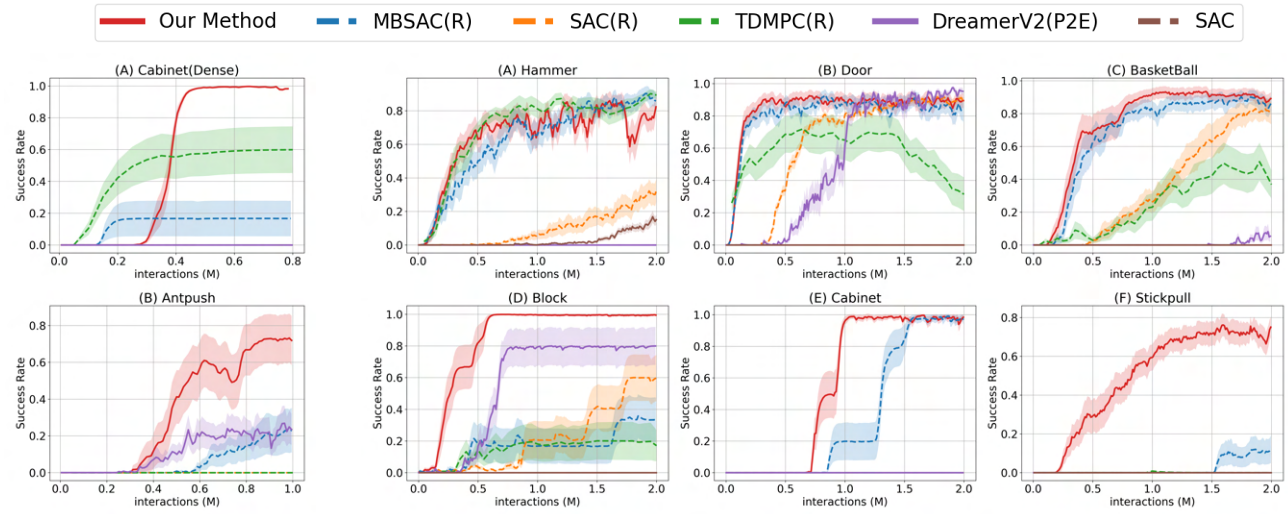


Figure 6. Results on dense reward tasks with local optima (exploration disabled)

Figure 7. Results on sparse reward tasks

eventually stabilize at the global optimum. This experiment suggests that a multimodal policy is necessary for reward maximization, and our method can help the policy better handle local optima.

Can multimodal policies accelerate exploration? We argue that maintaining a multimodal policy is beneficial even in the existence of an intrinsic reward to guide the exploration. We illustrate it in a 2D maze navigation task shown in Figure 5. The maze consists of 5×5 grids. Each of them is connected with neighbors with a narrow passage. The agent starts in the center grid and can move in four directions. The action space is its position change in two directions $(\Delta x, \Delta y)$.

We apply RPG and single-modality model-based SAC (Haarnoja et al., 2018) on this environment to maximize the intrinsic reward described in Sec. 4.3.2. We count the areas covered by the two policies during exploration with respect to the number of samples in Fig. 5(D). The curve suggests that our method explores the domain much faster, quickly reaching most grids, while the Gaussian agent only covers the right part of the maze within a limited sample budget.

To understand their differences, we visualize states sampled at different training steps of the two policies in Fig. 5 (A-B). Our policy below quickly finds four directions to move and gradually expands the state distribution until it fully occupies all grids. Fig. 5(C) shows the historic state visitation count. It is easy to see that our multimodal policy induces a more uniform distribution over the whole state space, generating a higher state distribution entropy. The optimization procedure of single-modality policy, as shown in the first row of Fig. 5, suffers from its policy parameterization. It can only explore one modality every time and has to switch modalities one by one, where modalities refer to different regions of the state space. It is hard to predict when it switches modality, making algorithms behave vastly differently in different environments with different random seeds. Sometimes it moves slowly from one direction to another because it has to wait for samples for density estimators to generate enough momentum. As a result, it never explores the left side in Fig. 5(C). While sometimes, it switches too fast due to the fast updates of the network and does not exploit some modalities enough, missing far-end

grids of certain directions that it has explored once. This also causes issues when maximizing external rewards. Even if a single-modal policy finds the optimal solution, it may switch to another modality to continue exploration and it is hard to guarantee that it would come back in the end. In contrast, our method is more like Monte-Carlo sampling, which samples all candidates while converging to solutions of high rewards with high probability.

5.2. Continuous Control Problems

We now verify if our method can scale up and help solve challenging continuous control problems. We take 8 representative environments from standard RL benchmarks, including 2 table-top environments from MetaWorld (Yu et al., 2020), 2 dexterous hand manipulation tasks from Rajeswaran et al. (2017), 1 navigation problems from Nachum et al. (2018b), and 2 articulated object manipulation from ManiSkill (Mu et al., 2021). We show environment examples and provide a detailed environment description in Appendix C. Only **Cabinet (Dense)** and **AntPush** contain dense rewards that lead to local optima. *The remaining* 6 *environments all only provide sparse rewards*, which means the agents receive a reward 1 when it succeeds to finish the task and 0 otherwise. This change dramatically increases the difficulty of these environments and disastrously hurts the performance of classical RL methods like SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017).

We evaluate our methods against the following baselines: DreamerV2 + Plan2Explore (Sekar et al., 2020), abbreviated as **DreamerV2 (P2E)**, a model-based exploration method based on the disagreement of learned models' prediction. We also consider 3 baselines, **TDMPC**, **MBSAC**, and **SAC** using the same intrinsic rewards as ours. The suffix **(R)** means that when we apply these methods to a sparse-reward environment, we will add RND intrinsic rewards that are the same as in our method. For all results evaluated on dense-reward environments in Figure 6, the exploration method of the corresponding algorithm is disabled. The standard SAC without intrinsic rewards validates the difficulty of our tasks. Details of the baseline implementations are in Appendix D.

Fig. 6 and 7 plots the learning progress of each algorithm in all environments (x-axis: number of environment interaction steps in million, y-axis: task success rate). For all environments, we run each algorithm for at least five trials. The curve and the shaded region shows the average and the standard deviation of performance over trials. ***MBSAC*** *shares almost the same implementation as our method, except that it does not condition its policy on latent variables.*.

We first observe that, for dense reward tasks, our method largely improves the success rate on tasks with local optima (Fig. 6). We can see that in both **AntPush** and **Cabinet (Dense)** tasks, our method outperforms all baselines. Our method consistently finds solutions, regardless of the local optima in the environments. For example, in the task of opening the cabinets' two doors and going to the two sides of the block, our method usually explores the two directions simultaneously and converges at the global optima. In contrast, other methods' performance highly depends on their initialization. If the algorithm starts by opening the wrong doors or pushing the block in the wrong direction, it will not escape from the local minimums; thus, its success rates are low.

Our methods successfully solve the 6 sparse reward tasks as shown in Fig. 7. Especially, it consistently outperforms the **MBSAC(R)** baseline, which is a method that only differs from ours by the existence of latent variables to parameterize the policy. Our method reliably discovers solutions in environments that are extremely challenging for other methods (e.g., the **StickPull** environment), clearly demonstrating the advantages of our method in exploration. Notably, we find that **MBSAC(R)**, which is equipped with our object-centric RND, is a strong baseline that can solve **AdroitHammer** and **AdroitDoor** faster than **DreamerV2(P2E)**, proving the effectiveness of our intrinsic reward design. **TDMPC(R)** has a comparable performance with **MBSAC(R)** on several environments. We validate that it has a faster exploration speed in Adroit Environments thanks to latent planning. We find that the **Dreamer(P2E)** does not perform well except for the **BlockPush** environment without the object prior and is unable to explore the state space well. We visualize modalities explored by our method in Appendix E.

5.3. Additional Experiments

Ablation study We analyze various factors influencing the performance of our method in the Maze navigation task in Section 5.1. More detailed discussion and experiment results are in Appendix B. Experimental comparisons between different latent spaces show that a Gaussian distribution of dimension 12 outperforms the categorical latent space, both surpassing a baseline that does not use latent variables. A moderate latent space size ≥ 6 is found to be sufficient, with performance declining if the latent dimensions are too small. In terms of reward maximization, the weight of the cross-entropy term (β) is crucial, with results indicating an ideal range between 0.001 and 0.01 for the RND design. Furthermore, the performance from RND is tied to maintaining a large replay buffer and using positional embedding, with a lack of either resulting in degraded exploration. A comparative analysis of policy parameterization methods shows the superiority of the vanilla Gaussian policy over the Gaussian Mixture Models (GMM) and CEM-based policy. The latter two display several optimization issues; GMM struggles with log-likelihood maximization, and CEM, despite its proficiency at finding local optima, tends to sacrifice its explorative capabilities. Finally, normalizing flow showed

initial promise but soon encountered numerical instabilities, highlighting the need for further investigation.

Evaluation on locomotion environments We modified the HalfCheetah-v3 environment in OpenAI Gym (Brockman et al., 2016) to study the performance of our methods in locomotion tasks, shown in Figure 11 in the appendix. The cheetah robot moves backward for a certain distance to receive a sparse reward of 1 to succeed. Our exploration method was able to effectively aid the exploration of the Cheetah robot and solve the task easily while removing the exploration term that led to the agent getting stuck. However, in this particular task, modeling multi-modal exploration did not increase the sampling efficiency, as there were only two modalities (moving forward and backward), and model-based SAC could exploit the two modes one by -one and solve the task. This made the advantage of our method negligible in this case. We also evaluated our method compared to SAC (Haarnoja et al., 2018) on the standard Mujoco environments. Results are shown in Fig. 12.

Vision-based RL As a proof of concept, we illustrate, in Fig. 13, the potential of our method for image observations in a single-block pushing environment: the observation consists of two consecutive 64x64 RGB images; the agent needs to control the red block to push the purple box into the target region. We use 4-layer convolutional networks as the encoder for both the policy network and RND estimator. We compare our method with model-based SAC (RND), which has an intrinsic reward to guide exploration but only models single modality policies, and model-based SAC without RND. The result validates our method's effectiveness.

6. Limitation and Future Work

Our approach capitalizes on the advantages offered by multiple components, effectively addressing complex exploration issues in continuous spaces. However, it also introduces certain hurdles and constraints. For instance, our intrinsic reward is predicated on assumptions regarding the recognition of objects and their spatial positioning. This approach may be unsuitable in environments with unidentified objects or where observations don't plainly reveal object-related information, akin to scenarios in vision-based RL; Learning the world model typically results in a slower pace of gradient updates; Incorporating a cross-entropy network adds an extra layer of complexity to the network design and training. Therefore, it is worth discussing potential future directions that might address these limitations.

Object-centric learning for vision-based RL While the Random Network Distillation (RND) is initially tailored for image observations, integrating object-centric design to accelerate exploration in vision-based RL will be an interesting direction. This suggests two typical strategies to apply our method to tasks with vision observations: (1) The first involves directly encoding observations without considering object information. It proves effective in scenarios with no occlusion and a static background, wherein objects emerge as the sole salient feature of the input. We provide a proof-of-concept experiment in Section 5.3. (2) The second approach harnesses computer vision techniques to identify objects for object-centric exploration. This includes applying recent large-scale vision foundation models, which possess zero-shot object detection capabilities as outlined in (Zhang et al., 2022) or leveraging slot-attention for object discovery as described in (Locatello et al., 2020).

Combining with previous model-based control and planning methods Instead of learning the world model from on-policy data, we can pre-train a physical world model (Li et al., 2019) or use analytical models (Posa et al., 2014; Huang et al., 2021) to gain generalizability and efficiency. Moreover, we drew inspiration from RRT-like motion planners (Karaman & Frazzoli, 2011) to derive our policy to sample over the configuration space and bias the exploration towards significant kinematics changes. Thus, an exciting direction is incorporating structures in model-based control into RL algorithms, including temporal structures like dynamics motion primitives (Stulp & Sigaud, 2013) and semantic information from TAMP (Garrett et al., 2021).

Extending to other probabilistic models Our method can be viewed as variational inference (Ranganath et al., 2014) over a particular stochastic computation graph (Weber et al., 2019). The computation graph contains hidden variables, and we use the Bellman equation and a learned model to estimate its gradient. This provides a new perspective that bridges online Reinforcement Learning (RL) with generative models and sequence modeling. In the future, we are interested in exploring how sequence-modeling techniques, such as transformers and hierarchical methods, can be used to model the policy in our framework.

7. Conclusion

We derive a framework that models the policy of continuous RL by a multimodal distribution in the variational inference framework. The method reparameterizes latent variables into trajectories like generative models. Under this framework, we learn a world model to help learn multimodal policy data efficiently. Incorporating an object-centric intrinsic reward, our method can solve challenging continuous control problems with little to no reward signal.

Acknowledgement

This work is in part supported by Qualcomm AI and AI Institute for Learning-Enabled Optimization at Scale (TILOS).

References

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

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

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Campos, V., Trott, A., Xiong, C., Socher, R., Giró-i Nieto, X., and Torres, J. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pp. 1317–1327. PMLR, 2020.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Co-Reyes, J., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International conference on machine learning*, pp. 1009–1018. PMLR, 2018.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: https://doi.org/10.1111/j.2517-6161.1977.tb01600.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Fang, K., Zhu, Y., Garg, A., Savarese, S., and Fei-Fei, L. Dynamics learning with cascaded variational inference for multi-step manipulation. *arXiv preprint arXiv:1910.13395*, 2019.

Fang, K., Zhu, Y., Garg, A., Savarese, S., and Fei-Fei, L. Dynamics learning with cascaded variational inference for multi-step manipulation. In *Conference on Robot Learning*, pp. 42–52. PMLR, 2020.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., and Lozano-Pérez, T. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT Press, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., and Su, H. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained

variational framework. In *International conference on learning representations*, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J. B., and Gan, C. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.

Kappen, H. J., Gómez, V., and Opper, M. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Karaman, S. and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Kumar, S., Kumar, A., Levine, S., and Finn, C. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Li, C., Xia, F., Martin-Martin, R., and Savarese, S. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pp. 603–616. PMLR, 2020.

Li, Y., Song, J., and Ermon, S. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*, 30, 2017.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. PMLR, 2020.

Mazzaglia, P., Verbelen, T., Dhoedt, B., Lacoste, A., and Rajeswar, S. Choreographer: Learning and adapting skills in imagination. *arXiv preprint arXiv:2211.13350*, 2022.

Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., and Pathak, D. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Moon, T. K. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

Mu, T., Ling, Z., Xiang, F., Yang, D., Li, X., Tao, S., Huang, Z., Jia, Z., and Su, H. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

Nachum, O., Gu, S., Lee, H., and Levine, S. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.

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

Ng, A. Cs229 lecture notes. *CS229 Lecture notes*, 1(1):1–3, 2000.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019. URL http://arxiv.org/abs/1910.07113.

Osa, T., Tangkaratt, V., and Sugiyama, M. Hierarchical reinforcement learning via advantage-weighted information maximization. *arXiv preprint arXiv:1901.01365*, 2019.

Osa, T., Tangkaratt, V., and Sugiyama, M. Discovering diverse solutions in deep reinforcement learning by maximizing state–action-based mutual information. *Neural Networks*, 152:90–104, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Peng, X. B., Guo, Y., Halper, L., Levine, S., and Fidler, S. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *arXiv preprint arXiv:2205.01906*, 2022.

Pertsch, K., Lee, Y., and Lim, J. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Posa, M., Cantu, C., and Tedrake, R. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1): 69–81, 2014.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial intelligence and statistics*, pp. 814–822. PMLR, 2014.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pp. 8857–8868. PMLR, 2021.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sekar20a.html.

Shankar, T. and Gupta, A. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pp. 8624–8633. PMLR, 2020.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. PMLR, 2014.

Stulp, F. and Sigaud, O. Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics*, 4(1):49–61, 2013.

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

Todorov, E. Linearly-solvable markov decision problems. *Advances in neural information processing systems*, 19, 2006.

Todorov, E. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pp. 4286–4292. IEEE, 2008.

Toussaint, M. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1049–1056, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553508. URL https://doi.org/10.1145/1553374.1553508.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Weber, T., Heess, N., Buesing, L., and Silver, D. Credit assignment techniques in stochastic computation graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2650–2660. PMLR, 2019.

Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020.

Ye, W., Liu, S., Kurutach, T., Abbeel, P., and Gao, Y. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Zhang, H., Zhang, P., Hu, X., Chen, Y.-C., Li, L., Dai, X., Wang, L., Yuan, L., Hwang, J.-N., and Gao, J. Glipv2: Unifying localization and vision-language understanding. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 36067–36080. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ea370419760b421ce12e3082eb2ae1a8-Paper-Conference.pdf.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. Noveld: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34:25217–25230, 2021.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. *arXiv preprint arXiv:2202.05607*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A. Implementation Details

Network architecture We use the following two-layer MLP to model policy π_θ, value Q_ψ, state encoder f_ψ, and the encoder $p_\phi(z|s)$. The network structures are shown in the pytorch's convention (Paszke et al., 2019).

```
Sequential(
  (0): Linear(in_features=inp_dim, out_features=256, bias=True)
  (1): ELU(alpha=1.0)
  (2): Linear(in_features=256, out_features=256, bias=True)
  (3): ELU(alpha=1.0)
  (4): Linear(in_features=256, out_features=out_dim, bias=True)
)
```

The dynamics network is a single-layer GRU with a hidden dimension 256. The RND network g_θ we use is a 3 layer MLP network with hidden dimension 512 and leaky ReLU as its activation function.

We maintain target networks like the standard double Q learning. The hyperparameters for training the network are listed in Table 1.

Hyperparameter	Value
Discount factor (γ)	0.99
Seed step	1000
Replay buffer size	800000
Model rollout horizon (H)	3
Action distribution	Tanh Normal
Entropy target	$-\lvert\mathcal{A}\rvert$
Initial entropy coefficient α	0.01
Cross-entropy coefficient β	0.005
RND coefficient β	0.1
Environment steps per gradient update	5
Temperature	$\mathcal{T}$
Learning rate	3×10^{-4}
Batch size	512
Target network update ratio	0.005
Actor update freq	2
State embedding dimension	100
grad norm clip	1.0
Positional encoding dimension	6
Latent distribution $\mathcal{Z}$	Normal
$\mathcal{Z}$ dimension	12
$p_\phi(z\|s,a)$ distribution	Normal distribution with std 0.38
$\pi_\theta(z\|s_1)$	$\mathcal{N}(0,1)$ for sparse reward tasks

Table 1. RPG hyperparameters. We here list the hyper-parameters used in the experiments. The hyper-parameters keep the same for our **MBSAC** baseline except that **MBSAC** has no latent space. Notice that for dense reward tasks, the entropy of $\pi_\theta(z|s_1)$ is linearly decayed starting from 3×10^5 environment steps to $1M$ steps to ensure optimality.

B. Ablation Study

We study and compare various factors in our methods in Fig. 8 on the Maze navigation task described in Sec. 5.1. Fig. 8(A) compares different latent spaces to use. The continuous latent space modeled by a Gaussian distribution of dimension 12 outperforms the categorical latent space, while both are better than the one without latent variables, i.e., the **MBSAC** baselines. Fig. 8(B) shows the effects of our method when using a Gaussian distribution as the latent space with different β values. The β controls the scale of the cross entropy term $\log p_\phi(z|s,a)$ in reward maximization, as mentioned in Sec. 4.3.1.

Algorithm 1 Model-based Reparameterized Policy Gradient

Input: $p_\phi, \pi_\theta, h_\psi, R_\psi, f_\psi, Q_\psi$ and an optional density estimator g_θ
Initialize p_ϕ, π_θ, construct the replay buffer $\mathcal{B}$.
while time remains **do**
Sample start state o_1 and encode it as $s_1 = f_\psi(o_1)$. Select z from $\pi_\theta(z|s_1)$.
Execute the policy $\pi_\theta(a|s, z)$ and store transitions into the replay buffer $\mathcal{B}$.
Sample a batch of trajectory segment of length K $\{\tau^i_{t:t+K}, z\}$ from the buffer $\mathcal{B}$.
Optional: update and estimate the density estimator g_θ and relabel transitions with the negative density as the intrinsic reward.
Optimize ψ using Equation 4.
Optimize $\pi_\theta(a|s, z)$ with gradient descent to maximize the value estimate in Equation 3 for s, z sampled from the buffer.
Optimize $\pi_\theta(z|s_1)$ with policy gradient to maximize $V_{\text{estimate}}(s_1, z) - \alpha \log \pi_\theta(z|s_1)$ for s_1 sampled from the buffer.
Optimize α, β if necessary .
end while

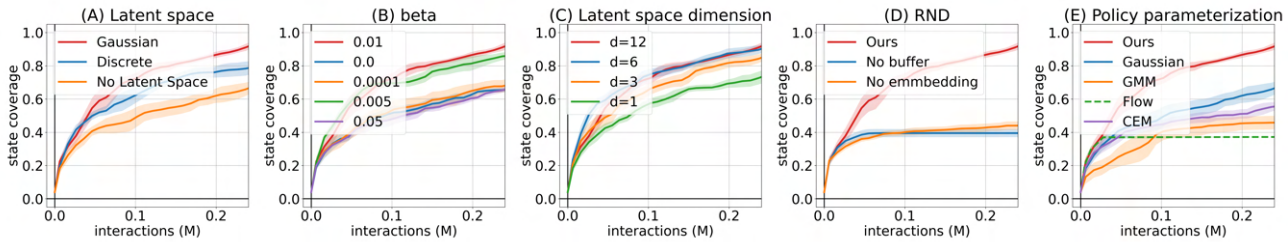
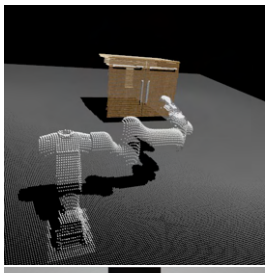


Figure 8. Comparing different factors in our methods.

The policy will ignore the latent variable if the β is too small, e.g., $0., 1e-4$. But if the β is too large, though the policy generates diverse solutions, it may explore too much without exploiting past experiences. This β plays a similar role as β in $\beta-$VAE (Higgins et al., 2017). In experiments, we find that β from 0.001 to 0.01 works well in the case of our RND design. Fig. 8(C) shows the effects of the latent dimensions. For tasks like $2D$ maze, a moderate latent space size $d \geq 6$ is sufficient. But the performance will degrade when it is too small. Fig. 8(D) ablates our design for the RND. When the RND estimator does not maintain a large replay buffer or does not use the positional embedding, the exploration will suffer a lot. We further compare various policy parameterization methods in Fig. 8(E). We find that in our implementation, Gaussian mixture models (GMM) and CEM-based policy do not perform as well as the vanilla Gaussian policy. GMM may have trouble in log-likelihood maximization. We noticed several numerical issues in optimizing GMM and Flow when we applied them with RND in sparse reward tasks. Specifically, we have encountered some instability when optimizing the log prob for GMM due to its non-convex nature and the need for sampling to estimate entropy. Similarly, our experiments with Flow have revealed significant parameter divergence and instabilities, warranting further investigation to pinpoint the root cause. CEM has a stronger ability to find local optima and generates actions with less randomness, which may sacrifice its ability to do exploration. Besides, we find the policy parameterized by a normalizing flow distribution behaves well initially but soon meets numerical instabilities and fails to proceed with optimization, suggesting more investigations are needed in this direction.

C. Environment Details

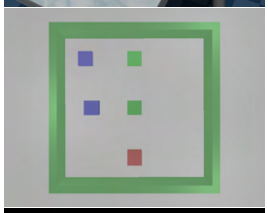


Cabinet (Dense) (Gu et al., 2023). The agent controls the movement of a 12 dof mobile robot arm and gripper robot to open both cabinet doors. The agent receives a dense reward for reaching its nearest door's handle. Besides, it receives a higher reward when it opens the right door than the left door. The agent succeeds when it fully opens the right door while the dense reward will typically drive the agent close to the handle of the left door. The episode length is 60.

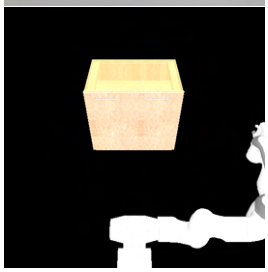
AntPush (Nachum et al., 2018b). The agent controls an ant robot with action dimension 8 to go to the upper room. The reward is the l_2 distance between the agent and a point in the upper room. The optimal path is to go to the left of the red block and push it to the right and go to the upper room. However, agents often get stuck at the local optima, which pushes the block forward or moves to go to the right side. The episode length is 400.

Door (Rajeswaran et al., 2017). The agent controls a dexterous hand with action dimension 26 to open a door. The agent only receives a reward of 1 when it successfully undoes the latch and opens the door. The episode length is 100 with an action repeat 2. Objects of interest include the hand's palm, the latch, and the door.

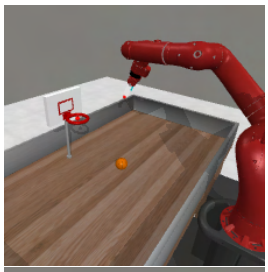
Hammer (Rajeswaran et al., 2017). The agent controls a dexterous hand with action dimension 26 to force drive a nail into the board. The agent only receives a reward of 1 when it has driven the nail all the way in. Action repeat is 2. The episode length is 125. We encode the position of the hand's palm, the hammer, and the nail.



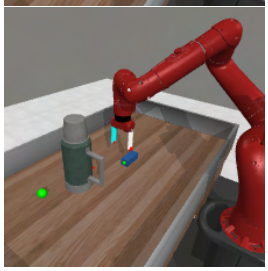
BlockPush (Xiang et al., 2020). The agent controls the movement of the red block with action dimension 2 to push the green block (middle) to the green destination (above) and the blue block (middle) to the blue destination (above). The agent only receives a reward of 1 when it has successfully pushed both blocks to the exact destination with a small tolerance. The objects of interest contain the location of the three blocks. The environment horizon is 60.



Cabinet (Sparse) (Gu et al., 2023). The agent controls the movement of a 9 dof robot arm and gripper robot to open both doors of the cabinet. The agent only receives a reward of 1 when both cabinet doors are fully opened. We encode the position of the robot's end effector and the location of the cabinet's door. Its episode length is 60.



Meta-World BaseketBall (Yu et al., 2020). The agent controls the movement of a gripper with a 4 dof controller to move the ball into the basket. The agent only receives a reward of 1 when the ball is sufficiently close to the basket. The locations of the ball and the location of robots' fingertips are what we are concerned about. The episode length is 100, including 2 action repeats.



Meta-World StickPull (Yu et al., 2020). The agent controls the movement of a gripper with a 4 dof controller to pull the container with a blue stick. The agent receives a reward of 1 only when the stick is inserted inside the handle, and the container is already pulled sufficiently close to the green dot. We encode the positions of the fingertips, the stick, and the handle of the cup for computing intrinsic rewards. The remaining setup is the same as **BasketBall**.

D. Baseline

TDMPC (Hansen et al., 2022), we used the publically available official implementation and default hyperparameters provided by the authors at https://github.com/nicklashansen/tdmpc.

SAC (Haarnoja et al., 2018), we implemented according to the original paper and used the default hyperparameter provided by the authors.

We use the abbreviation **TDMPC(R)**, **SAC(R)** to represent that we add an intrinsic reward with scale 0.1 for exploration in environments with only sparse rewards.

DreamerV2 (Hafner et al., 2020), we used the publically available official implementation and default hyperparameters provided by the authors at https://github.com/danijar/dreamerv2.

Plan2Explore (Sekar et al., 2020), we run DreamerV2 according to the instructions provided by https://github.com/ramanans1/plan2explore with hyperparameters provided by the authors of the paper.

For all baseline algorithms, we only change model update frequency to once every 5 environment steps.

E. Visualization of the Multimodal Exploration

We plot the trajectory of the agent in *AntPush* environment, evaluated at different numbers of training stages in Fig. 9. The agent learned to move forward and explored all directions that would decrease the l_2 distance. It found the left side was easier for moving up in the beginning, but at episode 360, it learned to explore all directions. Ultimately, it explored the left path to the upper room and converged on it.

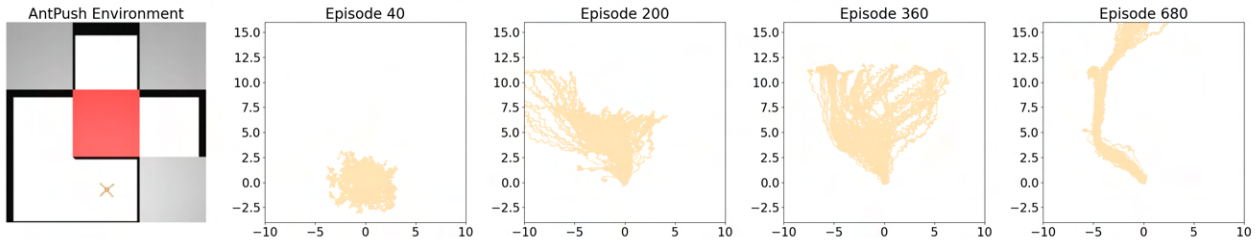


Figure 9. Exploration of AntPush, which has the dense reward to guide the agent to move forward.

We also plot the sampled states during exploration for Block, Cabinet, and Stickpull Envs in Fig. 10.

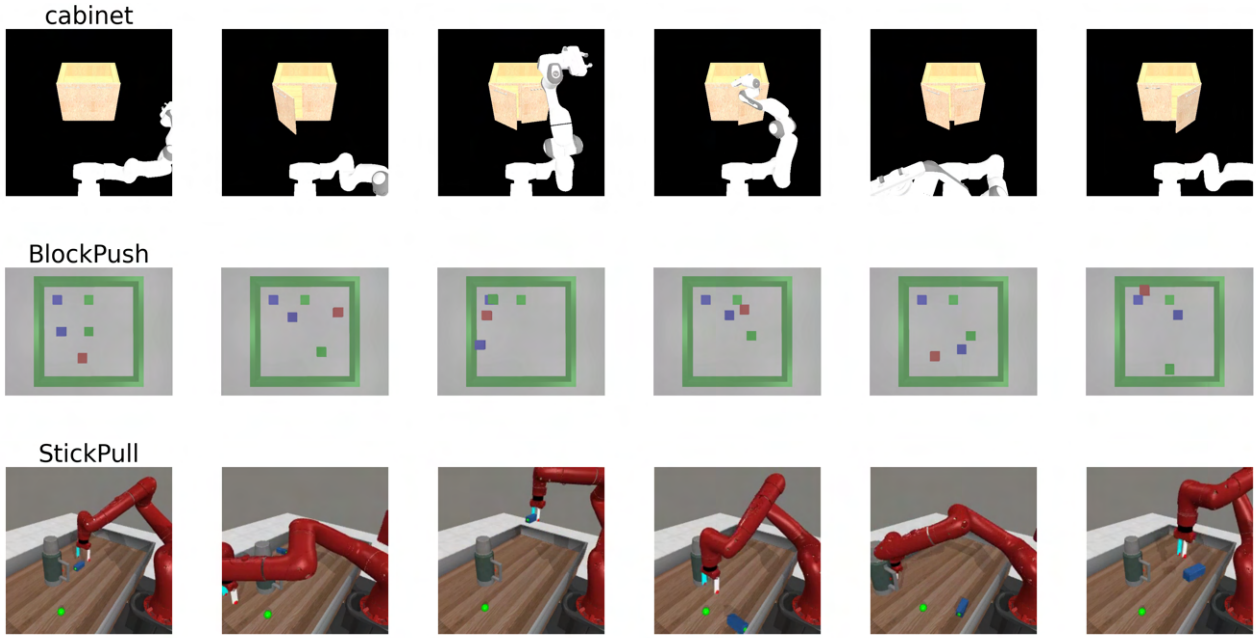


Figure 10. Exploration on several environments; The first column shows the initial state. The right 5 figures of the same row plot states sampled from a single agent.

F. Connection with Other Generative Models

Our method is based on the same variational bound shared with many other generative models

$$\log p(x) = E_{z\sim q(z)}\left[\log p(x,z) - \log q(z)\right] + KL(q(z)\|p(z|x)).$$

By different choices of latent space, posterior $q(z|x)$, joint distribution $p(x,z)$, we can obtain different generative models. For example, VAE models $p_\theta(x,z) = p_\theta(x|z)p(z)$ and $q(z) = q_\phi(z|x)$ using neural networks and then optimize θ, ϕ jointly to maximize the ELBO bound. By doing so, $q_\phi(z|x)$ will align with the true posterior of $p_\theta(z|x)$. Thus

$$\log p(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)]$$

The Expectation–maximization algorithm (EM) (Dempster et al., 1977) for learning Gaussian mixture models assumes that we have $p_\theta(x,z) = p_\theta(x|z)p_\theta(z)$ where z is a categorical representation. E-step: finding $q_\phi(z|x)$ by solving $\max_\phi \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ where $p_\theta(z|x) = p_\theta(x,z)/\int p_\theta(x,z)dz$. M-step: fixing ϕ, find $\max_\theta E_{q_\phi}[\log p_\theta(x,z)] - E_{q_\phi}[\log q_\phi(z|x)]$ which is exactly maximizing the ELBO.

In Maximum Entropy RL (Levine, 2018), we have optimality $p(O,\tau) = p(O|\tau)p(\tau)$ defined by the reward, and we optimize $\pi_\theta(\tau|O)$ only. The ELBO bound becomes a maximum entropy term $\mathbb{E}_{\tau\sim\pi}\left[\log p(O|\tau) + \log p(\tau) - \log \pi(\tau)\right]$. Our method differs from it by introducing an additional variable z. Table 2 compares various generative models.

	Latent	Encoder $q(z\|x)$	Joint $p(x,z)$	MLE objective
VAE	z	$p_\phi(z\|x)$	$p_\theta(x\|z)p(z)$	$p(x)$
EM	z	$\max_\phi \log p_\theta(x) - D_{KL}(q_\phi(z\|x)\|\|p_\theta(z\|x))$	$p_\theta(x\|z)p_\theta(z)$	$p(x)$
Diffusion	$\{x_t\}_{t\geq 1}$	$\prod_{i=1}^T \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$	$p(x_T)\prod_{t>1} p_\theta(x_{t-1}\|x_t)$	$p(x_0)$
MaxEntRL	τ	$\pi_\theta(\tau)$	$p(O\|\tau)p(\tau)$	$p(O)$
RPG	τ, z	$\pi_\theta(z,\tau)$	$p(O\|\tau)p_\phi(z\|\tau)p(\tau)$	$p(O)$

Table 2. Comparison of different algorithms that optimize ELBO bounds for inference

G. Environments and Results in Additional Experiments

Cheetah Back

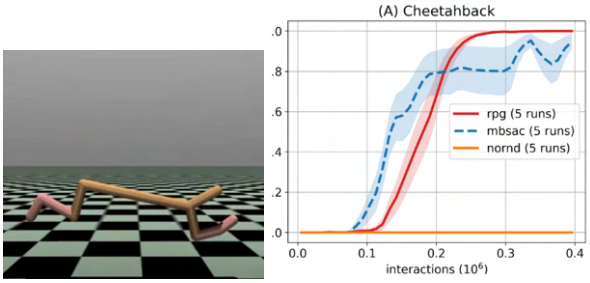


Figure 11. Cheetah Back Task (left), success rate (right)

Standard Mujoco-v2 Environments

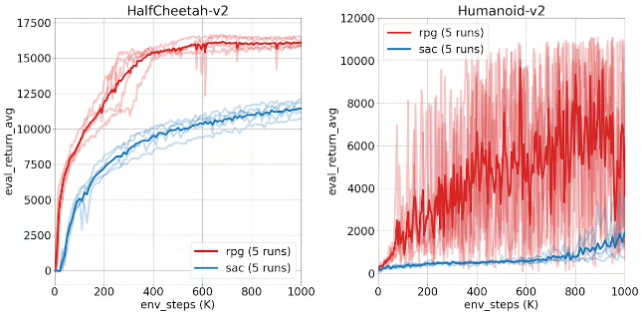


Figure 12. Results on Mujoco-v2 Environments

Vision-based RL

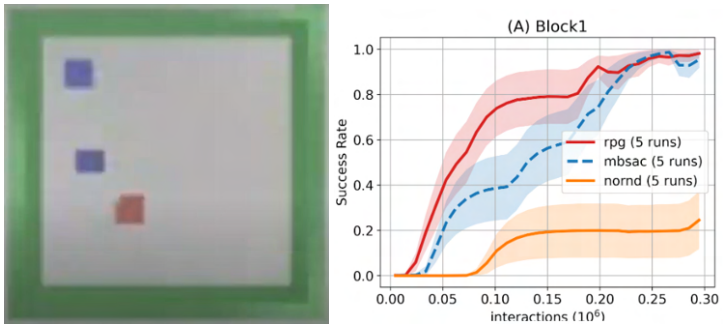


Figure 13. Visual Block Push Task (left), success rate (right)