Offline Reinforcement Learning from Images with Latent Space Models

Rafael Rafailov* [1] RAFAILOV@STANFORD.EDU
Tianhe Yu* [1] TIANHEYU@CS.STANFORD.EDU
Aravind Rajeswaran [2] ARAVRAJ@CS.WASHINGTON.EDU
Chelsea Finn [1] CBFINN@CS.STANFORD.EDU
[1] *Department of Computer Science, Stanford University*
[2] *School of Computer Science & Engineering, University of Washington*

Abstract

Offline reinforcement learning (RL) refers to the problem of learning policies from a static dataset of environment interactions. Offline RL enables extensive use and re-use of historical datasets, while also alleviating safety concerns associated with online exploration, thereby expanding the real-world applicability of RL. Most prior work in offline RL has focused on tasks with compact state representations. However, the ability to learn directly from rich observation spaces like images is critical for real-world applications such as robotics. In this work, we build on recent advances in model-based algorithms for offline RL, and extend them to high-dimensional visual observation spaces. Model-based offline RL algorithms have achieved state of the art results in state based tasks and have strong theoretical guarantees. However, they rely crucially on the ability to quantify uncertainty in the model predictions, which is particularly challenging with image observations. To overcome this challenge, we propose to learn a latent-state dynamics model, and represent the uncertainty in the latent space. Our approach is both tractable in practice and corresponds to maximizing a lower bound of the ELBO in the unknown POMDP. In experiments on a range of challenging image-based locomotion and manipulation tasks, we find that our algorithm significantly outperforms previous offline model-free RL methods as well as state-of-the-art online visual model-based RL methods. Moreover, we also find that our approach excels on an image-based drawer closing task on a real robot using a pre-existing dataset. All results including videos can be found online at https://sites.google.com/view/lompo/.

1. Introduction

For robots and artificial agents to be competent in a wide variety of dynamic and uncertain environments, they require the ability to perceive the world and act based on rich sensory observations like vision. In most real-world scenarios like homes or disaster management, it is difficult to hand-design state representations or simulators, let alone instrument the world to estimate the states. This suggests the need for an end-to-end integration of sensing and control.

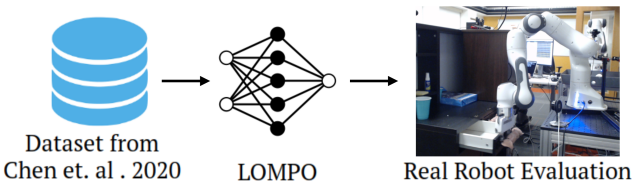

Figure 1: LOMPO learns vision-based policies from offline datasets, without any interaction in the environment.

* denotes equal contribution.

Despite recent advances (Hafner et al., 2020; Kostrikov et al., 2020; Laskin et al., 2020), the interactive (or online) sample complexity for learning control policies from vision is prohibitively high. Furthermore, interactive reinforcement learning (RL) with physical systems in the real world is fraught with safety challenges that limit widespread applicability. Our goal in this work is to develop approaches for overcoming these challenges by utilizing offline datasets.

Offline RL (Lange et al., 2012) involves the learning of control policies from static pre-collected data. By using previously collected data, we alleviate the safety challenges associated with online exploration. Large offline datasets are also already available in domains like autonomous driving (Caesar et al., 2020), recommendation systems (Harper and Konstan, 2015), and robotic manipulation (Finn and Levine, 2017; Sharma et al., 2018; Dasari et al., 2019; Mandlekar et al., 2019), typically in image (or video) format. Prior work in offline RL (see Section 2) has typically focused on environments with compact state representations, which are not representative of challenges faced in real-world applications. In this work, we focus on control from pixels using offline datasets. We believe the ability to train vision based policies using offline data can reduce interactive sample complexity, enhance safety, and greatly expand the applicability of RL.

Our work builds on recent advances in model-based offline RL (Kidambi et al., 2020; Yu et al., 2020b). Model-based RL algorithms have demonstrated impressive sample efficiency in interactive RL (Janner et al., 2019; Rajeswaran et al., 2020; Hafner et al., 2020). For offline RL, model-based algorithms (Kidambi et al., 2020; Yu et al., 2020b) have been shown to be mimimax optimal, obtain state of the art results in a variety of benchmark tasks, as well as generalize to new out-of-distribution tasks. Uncertainty quantification and pessimism have emerged as key principles and requirements for successful offline RL, with both model-based and model-free approaches. This represents a unique challenge for control from pixels, since directly translating successful approaches in state based domains (e.g. ensembles of models) to image spaces can be computationally prohibitive.

Our Contributions. The main contribution of our work is an algorithm, latent offline model-based policy optimization (LOMPO), which enables learning of visuomotor policies using offline datasets. LOMPO can be summarized as follows. (i) Using the available offline data, we learn a variational model with an image encoder, an image decoder, and an ensemble of latent dynamics models. (ii) We construct an uncertainty penalized MDP in the latent state space (which induces a corresponding uncertainty-penalized POMDP in observation space), where we quantify uncertainty based on disagreement between forward models in the latent state space. (iii) We learn a control policy in the learned latent space using the offline dataset by optimizing an uncertainty-penalized objective. The learned uncertainty penalized MDP provides a pessimistic regularizing effect for policy learning and guards against major challenges like distributional shift and model exploitation. We evaluate our algorithm on four simulated visuomotor control tasks and one real-world robotic manipulation task. We find that LOMPO outperforms or matches prior model-based and model-free methods across the board on these challenging tasks.

2. Related Work

Our work is at the intersection of offline RL and control from high-dimensional inputs (i.e. images). We review related work from these fields below.

Offline RL. Offline RL has recently emerged as a prominent paradigm for learning control policies (Lange et al., 2012; Levine et al., 2020). Most offline RL algorithms augment well known RL

algorithms with various forms of regularization. These include regularized variants of importance sampling based algorithms (Liu et al., 2019; Swaminathan and Joachims, 2015; Nachum et al., 2019; Zhang et al.), actor-critic algorithms (Wu et al., 2019; Jaques et al., 2019; Siegel et al., 2020; Peng et al., 2019), approximate dynamic programming algorithms (Fujimoto et al., 2018; Kumar et al., 2019, 2020; Liu et al., 2020; Agarwal et al., 2019), and model-based RL algorithms (Kidambi et al., 2020; Yu et al., 2020b; Argenson and Dulac-Arnold, 2020; Matsushima et al., 2020; Swazinna et al., 2020). However, most of these prior works focus on problems with low-dimensional compact state information, except that a few that learn to play Atari games (Fujimoto et al., 2018; Agarwal et al., 2019; Kumar et al., 2020) or propose benchmarks that include simulated locomotion tasks with pixel inputs (Gulcehre et al., 2020); our focus in this work is on continuous robot control from high-dimensional perceptual inputs in both simulation and the real world.

Control from Pixels. Control from high-dimensional observation inputs has become an important problem within control and robotics as it makes real world applications more practical. Prior works have tackled this problem with model-free RL methods by learning policies either from pixel inputs end-to-end (Haarnoja et al., 2018; Gelada et al., 2019; Singh et al., 2019; Kostrikov et al., 2020; Laskin et al., 2020; Han et al., 2020) or on top of unsupervised visual representations Lange and Riedmiller (2010); Ghadirzadeh et al. (2017); Nair et al. (2018). Alternatively (Lee et al., 2020; Wayne et al., 2018) train a variational latent space model, but use it only as a filter and train a separate policy on top of the learned latent representation. Model-based RL learns a dynamics model either in the pixel space (Finn and Levine, 2017; Ebert et al., 2018) or in a latent space (Levine et al., 2016; Finn et al., 2016; Watter et al., 2015; Banijamali et al., 2018; Zhang et al., 2019; Hafner et al., 2019, 2020; Ha and Schmidhuber, 2018; Kipf et al., 2019; Chen et al., 2020) and can either learn a policy within the model or deploy shooting-based planning methods. However, most of those prior works rely critically on online data collection to be successful. Visual foresight algorithms (Finn and Levine, 2017; Ebert et al., 2018; Suh and Tedrake, 2020; Yen-Chen et al., 2019; Chen et al., 2020) handle control from pixels in a fully offline setting, but do not explicitly tackle the distributional shift issue that arises; meanwhile, our method is designed to specifically address this. As a result, we find in Section 5.2 that our approach significantly outperforms visual foresight.

3. Preliminaries

POMDPs. We consider a class of partially observable Markov decision processes (POMDPs), whose transition dynamics can be described using a compact MDP, and observations using an emmision model. Concretely, we consider POMDPs of the form $M = (\mathcal{X}, \mathcal{S}, \mathcal{A}, T, D, r, \mu_0, \gamma)$ where $\mathcal{X}$ denotes the the visual observation space, $\mathcal{S}$ the unobserved state space, $\mathcal{A}$ the action space, $T(s'|s,a)$ the latent transition distribution, $D(x|s)$ the observation model, $r(s,a)$ the reward function, $\mu_0(s_0)$ the initial latent state distribution, and $\gamma \in (0,1)$ the discount factor. The goal is to learn a policy $\pi(a_t|x_t)$ that maximizes the discounted expected return $\eta_M := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$.

Control as inference. For MDPs with directly observable states s_t, actions a_t, rewards r_t, initial state distribution $\mu_0(s_1)$, and the stochastic transition dynamics $T(s_{t+1}|s_t, a_t)$, the control problem is equivalent to an inference problem in the graphical model with a binary random variable $\mathcal{O}_t$, which indicates if the agent is optimal at time step t. When $p(\mathcal{O}_t = 1|s_t, a_t) = \exp(r(s_t, a_t))$, maximizing $p(\mathcal{O}_{1:H})$ for some finite horizon H via approximate inference is equivalent to optimizing the maximum entropy RL objective, $\mathbb{E}[\sum_{t=1}^{H}(r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)))]$ (Levine, 2018).

Lee et al. (2020) extend the framework to POMDPs by factorizing the variational distribution $q(s_{1:H}, a_{t+1:H}|x_{1:t+1}, a_{1:t})$ into a product of inference terms $q(s_{t+1}|x_{t+1}, s_t, a_t)$, latent dynamics terms $T(s_{t+1}|s_t, a_t)$, and policy terms $\pi(a_t|x_{1:t}, a_{1:t-1})$. The evidence lower bound (ELBO) of $p(x_{1:t+1}, \mathcal{O}_{t+1:H}|a_{1:t})$, the likelihood of observed data and future optimality of the agent is:

$$\begin{aligned}
&\log p(x_{1:t+1}, \mathcal{O}_{t+1:H}|a_{1:t}) \\
&\quad \geq \mathbb{E}_{(s_{1:H}, a_{t+1:H})\sim q}\Bigg[\sum_{\tau=0}^{t}\left(\log D(x_{\tau+1}|s_{\tau+1}) - D_{KL}\left(q(s_{\tau+1}|x_{\tau+1}, s_\tau, a_\tau)\|T(s_{\tau+1}|s_\tau, a_\tau)\right)\right) \\
&\qquad + \sum_{\tau=t+1}^{H}\left(r(s_\tau, a_\tau) + \log p(a_\tau) - \log \pi(a_\tau|x_{1:\tau}, a_{1:\tau-1})\right)\Bigg] := \mathcal{L}_{\text{ELBO}} \qquad (1)
\end{aligned}$$

where $r(s_\tau, a_\tau) = \log p(\mathcal{O}_\tau = 1|s_\tau, a_\tau)$ and $p(a_\tau)$ is the prior action distribution.

Offline RL. In the offline RL problem, the agent must learn a policy using only a fixed dataset of interactions. In our work, we focus on offline RL in high-dimensional POMDPs, where the agent has access to the fixed dataset $\mathcal{D}_{\text{env}}$ of trajectories, consisting of high-dimensional observations, actions and rewards. The dataset is collected by a behavior policy π^{B}, which may correspond to a mixture of policies. No additional environment interaction is possible. We call the distribution induced by $\mathcal{D}_{\text{env}}$ as the *behavioral distribution*.

Model-based offline RL. Model-based RL in conjunction with the key idea of pessimism or conservatism, has emerged as a promising paradigm for offline RL (Kidambi et al., 2020; Yu et al., 2020b; Matsushima et al., 2020; Argenson and Dulac-Arnold, 2020). In this work, we build on the MOPO framework (Yu et al., 2020b). Given a dataset $\mathcal{D}_{\text{env}}$ from an MDP M, MOPO learns a dynamics model $\widehat{T}(\cdot|s, a)$ and reward model $\widehat{r}(s, a)$, and uses these models to construct an uncertainty-penalized MDP $\widetilde{M}$, with dynamics $\widehat{T}$ and a modified reward function $\widetilde{r}(s, a) = \widehat{r}(s, a) - u(s, a)$, where $u(s, a)$ is an estimate of model uncertainty. To account for distribution shift, the policy is optimized in this uncertainty penalized MDP. With an admissible uncertainty estimator such that $u(s, a) \geq \frac{r_{\max}}{1-\gamma} D_{TV}(T(s, a), \widehat{T}(s, a))$, (Yu et al., 2020b) theoretically show that optimizing a policy under the *uncertainty-penalized* MDP $\widetilde{M} = (\mathcal{S}, \mathcal{A}, \widehat{T}, \widetilde{r}, \mu_0, \gamma)$ is equivalent to optimizing a lower bound of the return under the learned policy in the true MDP M. While MOPO achieves impressive results in tasks with low-dimensional observation spaces, it is hard to scale to realistic environments with image observations. In the next section, we present LOMPO, which aims to solve the offline RL problem in a model-based way using high-dimensional observation spaces.

4. LOMPO: Latent Offline Model-Based Policy Optimization

Our goal is to design an offline model-based RL method that handles high-dimensional observations. Since we need to learn a model from a fixed dataset without further interaction with the environment, the model predictions will become less trustworthy as the policy rollouts move further from the behavioral distribution. Such inaccurate model predictions would generate observations that could negatively impact the policy optimization (the model exploitation phenomenon). Therefore, quantifying the uncertainty of the observations generated by the learned model is important for offline model-based RL to avoid large extrapolation error on out-of-distribution observations.

However, estimating model uncertainty in high-dimensional spaces is challenging: the common approach to uncertainty quantification of learning an ensemble of models is notably memory-intensive and computationally-expensive when applied to visual dynamics models.

In this section, we present our offline visual model-based RL algorithm that address the above challenges by learning a latent dynamics model and estimating the model uncertainty in the compact latent space. Specifically, under the assumption that the control problem in the latent space is an MDP, we construct an uncertainty-penalized latent MDP with the reward penalized by the uncertainty of the latent dynamics model (Section 4.1). Next, we construct the corresponding uncertainty-penalized POMDP and optimize the policy and the latent dynamics model in the control as inference framework by maximizing the ELBO in the uncertainty-penalized POMDP, which is a lower bound of the ELBO in the true POMDP (Section 4.2). Finally, we discuss our overall practical algorithm LOMPO (Section 4.3).

4.1. Quantifying Model Uncertainty in the Latent Space

In order to quantify the uncertainty in the latent state space, we first make the assumption that the latent state space $\mathcal{S}$ forms an MDP, which we define as the latent MDP $M_{\mathcal{S}} = (\mathcal{S}, \mathcal{A}, T, \mu_0, \mu_0, \gamma)$ with the same notation from Section 3. Similarly, we define the estimated latent MDP $\widehat{M}_{\mathcal{S}} = (\mathcal{S}, \mathcal{A}, \widehat{T}, r, \mu_0, \gamma)$ where $\widehat{T}(s'|s,a)$ denotes the learned latent dynamics model. The objective of our algorithm is to learn an optimal policy $\pi(a|s)$ in $\widehat{M}_{\mathcal{S}}$ that also maximizes expected return in $M_{\mathcal{S}}$. With the above definitions of $M_{\mathcal{S}}$ and $\widehat{M}_{\mathcal{S}}$, we can construct the *uncertainty-penalized* latent MDP $\widetilde{M}_{\mathcal{S}} = (\mathcal{S}, \mathcal{A}, \widehat{T}, \widetilde{r}, \mu_0, \gamma)$ where $\widetilde{r}(s,a) = r(s,a) - \lambda u(s,a)$ where $u(s,a)$ is an admissible uncertainty estimator as defined in Section 3. Under the uncertainty-penalized latent MDP, we first define $\epsilon_u(\pi) = \mathop{\bar{\mathbb{E}}}_{(s,a)\sim\rho^{\pi}_{\widehat{T},\mu_0}}[u(s,a)]$ where $\rho^{\pi}_{T,\mu_0}(s,a) := \sum_{t=0}^{\infty}\gamma^t \mathbb{P}^{\pi}_{T,\mu,t}(s)\pi(a \mid s)$ denotes the discounted state-action distribution. Then we proceed to show that the total return of a policy π under the uncertainty-penalized latent MDP is a lower bound of the the total return under the true latent state MDP and the gap between the learned policy under the uncertainty-penalized latent MDP and the optimal policy π^* depends on the latent dynamics model error $\epsilon_u(\pi^*)$. Directly following Theorem 4.4 in (Yu et al., 2020b), we can show the above claims as following:

$$\eta_{M_{\mathcal{S}}}(\pi) \geq \eta_{\widetilde{M}_{\mathcal{S}}}(\pi), \tag{2}$$

$$\eta_{M_{\mathcal{S}}}(\hat{\pi}) \geq \sup_{\pi}\{\eta_{M_{\mathcal{S}}}(\pi) - 2\lambda\epsilon_u(\pi)\} \tag{3}$$

where $\hat{\pi}$ is the policy learned via maximizing the return under $\widetilde{M}_{\mathcal{S}}$. In practice, we do not have access to the uncertainty quantification oracle $u(s,a)$ that upper bounds the latent model error, and we estimate the uncertainty of the latent model using heuristics as discussed in Section 4.3.

4.2. Latent Model Training and Policy Optimization with Uncertainty-Penalized ELBO

With the uncertainty-penalized latent MDP defined, we can construct the corresponding uncertainty-penalized POMDP $\widetilde{M} = (\mathcal{X}, \mathcal{S}, \mathcal{A}, \widehat{T}, D, \widetilde{r}, \mu_0, \gamma)$. In the following subsection, we will derive the objectives of the policy learning and latent model learning under the uncertainty-penalized POMDP.

We define the learned variational distribution $\widehat{q}(s_{1:H}, a_{t+1:H}|x_{1:t+1}, a_{1:t})$ as a product of inference terms $q(s_{t+1}|x_{t+1}, s_t, a_t)$, learned latent dynamics terms $\widehat{T}(s_{t+1}|s_t, a_t)$ and policy terms

$\pi(a_t|x_{1:t}, a_{1:t-1})$ as follows

$$\widehat{q}(s_{1:H}, a_{t+1:H}|x_{1:t+1}, a_{1:t}) = \prod_{\tau=0}^{t} q(s_{\tau+1}|x_{\tau+1}, s_\tau, a_\tau) \prod_{\tau=t+1}^{H-1} \widehat{T}(s_{\tau+1}|s_\tau, a_\tau) \prod_{\tau=t+1}^{H} \pi(a_\tau|x_{1:\tau}, a_{1:\tau-1}).$$

With $\widehat{q}$ and Eq. 1, we can bound the expected return term in Eq. 1 from below as follows:

$$\mathbb{E}_{s_{t+1:H}, a_{t+1:H} \sim q} \left[\sum_{\tau=t+1}^{H} r(s_\tau, a_\tau) \right] = \bar{\mathbb{E}}_{s_{t+1:H}, a_{t+1:H} \sim \rho^{\pi}_{T, q(s_{t+1}|x_{t+1}, s_t, a_t)}} [r(s_\tau, a_\tau)] \tag{4}$$

$$\geq \bar{\mathbb{E}}_{s_{t+1:H}, a_{t+1:H} \sim \rho^{\pi}_{\widehat{T}, q(s_{t+1}|x_{t+1}, s_t, a_t)}} [\widetilde{r}(s_\tau, a_\tau)] = \mathbb{E}_{s_{t+1:H}, a_{t+1:H} \sim \widehat{q}} \left[\sum_{\tau=t+1}^{H} \widetilde{r}(s_\tau, a_\tau) \right] \tag{5}$$

where Eq. 4 follows from the definition of $\widehat{q}$ and the discounted state-action distribution with the initial state distribution being $q(s_{t+1}|x_{t+1}, s_t, a_t)$ and the inequality in Eq. 5 follows from Eq. 2 and the latent MDP assumption. Now we can derive the ELBO in the uncertainty-penalized POMDP from the ELBO in the original POMDP defined in Equation 1 as follows:

$$\mathcal{L}_{\text{ELBO}} \geq \mathbb{E}_{s_{1:t}, a_{t+1:H} \sim q} \left[\sum_{\tau=0}^{t} \left(\underbrace{\log D(x_{\tau+1}|s_{\tau+1})}_{\text{reconstruction}} - \underbrace{D_{KL}\left(q(s_{\tau+1}|x_{\tau+1}, s_\tau, a_\tau) \| T(s_{\tau+1}|s_\tau, a_\tau)\right)}_{\text{consistency}} \right) \right] \tag{6}$$

$$+ \mathbb{E}_{s_{t+1:H}, a_{t+1:H} \sim \widehat{q}} \left[\sum_{\tau=t+1}^{H} \left(\widetilde{r}(s_\tau, a_\tau) + \log p(a_\tau) - \log \pi(a_\tau|x_{1:\tau}, a_{1:\tau-1}) \right) \right] \coloneqq \widetilde{\mathcal{L}}_{\text{ELBO}} \tag{7}$$

where $\tilde{\mathcal{L}}_{\text{ELBO}}$ denotes the ELBO in the uncertainty-penalized POMDP, which turns out to be a lower bound of $\mathcal{L}_{\text{ELBO}}$, the ELBO in the original POMDP. With the uncertainty-penalized ELBO, we optimize the latent dynamics model and the inference model using Eq. 6 (the reconstruction and consistency terms), which can be viewed as offline latent model training, and optimize the policy with Eq. 7, which uses an uncertainty penalized reward similar to MOPO. Next, we will discuss the practical implementation of our model training and policy optimization in Section 4.3.

4.3. Practical Implementation of LOMPO

We now present our practical LOMPO algorithm, outlined in Algorithm 1 in Appendix D, and visualize the training framework in Figure 2.

Variational Model Training. To estimate the uncertainty term in the uncertainty-penalized POMDP, we train an ensemble of latent transition models and use model-disagreement as a proxy. In designing the optimization approach, we have two main considerations: (1) we need all of the members of the ensemble to be grounded within the same latent space, (2) we should minimize additional model complexity and training overhead. Considering these, we optimize the following objective:

$$\sum_{\tau=0}^{H-1} \left[\mathbb{E}_q[\log D(x_{\tau+1}|s_{\tau+1})] - \mathbb{E}_q D_{KL}(q(s_{\tau+1}|x_{\tau+1}, s_\tau, a_\tau) \| \widehat{T}_\tau(s_{\tau+1}|s_\tau, a_\tau)) \right] \tag{8}$$

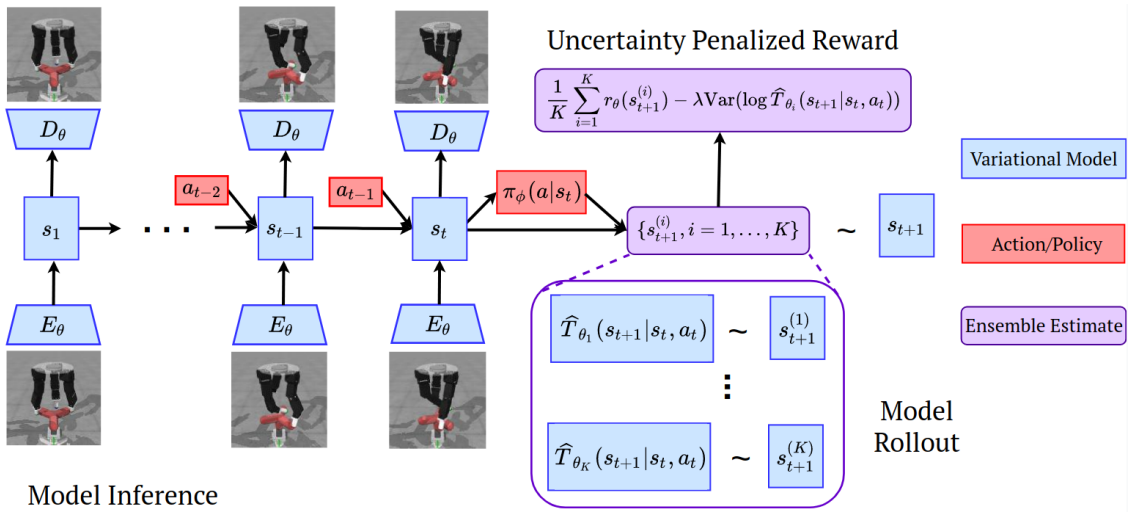

Figure 2: Images are passed through a convolutional encoder E_θ to form a compact representation which are then used along with previous state to infer the current state s_t. The model is trained by reconstructing the images from the latent states through the decoder network D_θ. Latent rollouts are carried by choosing a random learned transition model $\widehat{T}_{\theta_j}(s_{t+1}|s_t, a_t)$ and rewards are penalized based on ensemble disagreement.

Here at each step we sample a random learned forward transition model $\widehat{T}_\tau$ from a fixed set of K models $\{\widehat{T}_1, \ldots, \widehat{T}_K\}$. The inference distribution q is modeled via the standard mean-field approximation as a uni-modal Gaussian distribution and is shared across all time steps. This explicitly grounds all forward models within the same latent space state representation induced by the inference distribution addressing concern (1) above. Moreover since we use a single forward model at each time step this procedure has the same computational overhead as the regular training of a single variational model, which addresses concern (2).

Model-based Policy Optimization. We train a policy and a critic with components $\pi_\phi(a_t|s_t)$ and $Q_\phi(s_t, a_t)$ on top of the latent space representation similar to (Lee et al., 2020), however we deploy model inference for the policy as well as the critic. We made this design choice as it's faster and more efficient to carry out rollouts in the latent space, rather than sampling from the observation model. We maintain two replay buffers $\mathcal{B}_{real}$, $\mathcal{B}_{sample}$. The real data replay buffer contains transition tuples s_t, a_t, r_t, s_{t+1} from the latent MDP, where states are sampled from the inference distribution $s_{1:H} \sim q(s_{1:H}|x_{1:H}, a_{1:H-1})$ over trajectories from the real dataset $x_{1:H}, r_{1:H}, a_{1:H-1} \sim \mathcal{D}_{env}$. The latent data buffer contains transitions from rolling-out the policy in the model latent space utilizing the ensemble of learned forward models. During rollouts transitions are carried out at each step by picking a a random forward model from the ensemble. As discussed in Section 4.1, the final rewards for the rollout use ensemble estimates and the uncertainty penalty term and are computed as:

$$\widetilde{r}_t(s_t, a_t) = \frac{1}{K}\sum_{i=1}^{K} r_\theta(s_t^{(i)}, a_t) - \lambda u(s_t, a_t) \tag{9}$$

where $s_t^{(i)} \sim \widehat{T}_{\theta_i}(s_{t-1}, a_{i-1})$ are sampled from each forward model and s_t is sampled from $\{s_t^{(i)}, i = 1, \ldots, K\}$. Here $u(s_t, a_t)$ is an estimate of model uncertainty and λ is a penalty parameter. In particular, we pick $u(s_t, a_t)$ as disagreement of the latent model predictions in the ensemble, i.e. the variance of log-likelihoods under the ensemble $u(s_t, a_t) = \mathrm{Var}(\{\log \widehat{T}_{\theta_i}(s_t|s_{t-1}, a_{t-1}), i =$

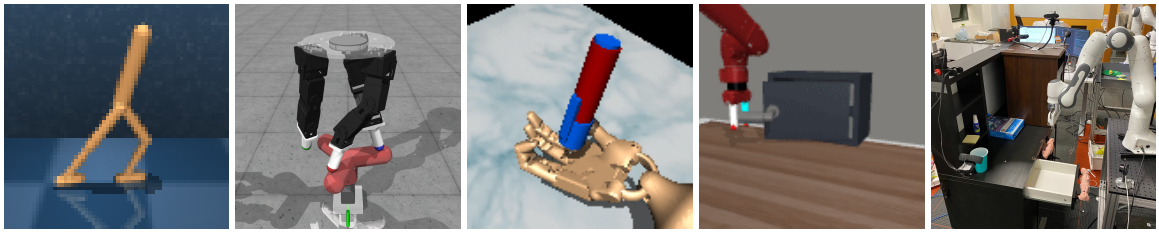

Figure 3: Test environments: DeepMind Control Walker task - the observations are raw 64×64 images. Robel D'Claw Screw and Adroit Pen tasks observations are raw 128×128 images and robot proprioception. Sawyer Door open environment - the observation space is raw 128×128 images. The observations for the real robot environment are raw 64×64 images from the overhead camera.

$1, \ldots, K\}$), since ensembles have been shown to capture the epistemic uncertainty (Bickel and Freedman, 1981) and also work well for model-based RL in practice (Yu et al., 2020b; Kidambi et al., 2020). We used this heuristic to estimate uncertainty as it estimates disagreement across the means of the forward models, as well as the variances. Finally the actor and critic $\pi_\phi(a_t|s_t), Q_\phi(s_t, a_t)$ are trained using standard off-policy training algorithm using batches of equal data mixed from the real and sampled replay buffers, as we find that maintaining a fixed sampling proportion is important in preventing distributional shift in the actor-critic training.

5. Experiments

The goal of our experimentation evaluation is to answer the following questions. (1) Can offline RL reliably scale to realistic robot environments with complex dynamics and interactions? (2) How does LOMPO compare to prior offline model-free RL algorithms and online model-based RL algorithms when learning vision-based control tasks from offline data? (3) How does the quality and size of the dataset affect performance? (4) Can LOMPO be applied to an offline RL task on a real robot with raw camera image observations? We answer questions (1), (2) and (3) in Section 5.1 and address question (4) in Section 5.2. All implementation details such as model architectures and hyperparameter choices are included in Appendix C.

5.1. Simulated Experiments

Previous offline RL benchmarks are not well-suited for answering questions (1), (2) and (3) above as they largely lack image-based robot control problems. Thus, to answer those three questions, we design a suite of four simulated image-based offline RL problems, focusing on robotics applications, described in Appendix A and visualized in the four pictures on the left in Figure 3. We also include the visualization of the samples generated from our learned variational model on the four environments in Appendix E. All of the environments and datasets will be open-sourced to allow future work to also study this problem and make direct comparisons.

Comparisons. We compare our proposed method to both model-free and model-based learning algorithms. Our first comparison is direct behavior cloning (BC) from raw image observations, which has proved to be a strong baseline in the past (Fu et al. (2020)). We also benchmark to Conservative Q-Learning (Kumar et al. (2020)), which is a state of the art offline learning algorithm in the low-dimensional case (Fu et al. (2020)); however, we again train it from raw image observations. We evaluate an MBPO based model (Janner et al. (2019)), which also carries out policy rollouts in latent space similar to LOMPO, but does not apply an uncertainty penalty. The performance of this

Environment	Dataset	LOMPO (ours)	LMBRL	Offline SLAC	CQL	BC
Walker Walk	medium-replay	**74.9**	44.7	-0.1	14.7	5.3
Walker Walk	medium-expert	**91.7**	76.3	32.8	45.1	15.6
Walker Walk	expert	**75.8**	24.5	11.3	40.3	11.8
D'Claw Screw	medium-replay	**71.8**	**72.4**	65.9	26.3	11.7
D'Claw Screw	medium-expert	**100.4**	**96.2**	76.3	30.3	27.6
D'Claw Screw	expert	**99.2**	90.8	63.4	24.2	25.2
Adroit Pen	medium-replay	**82.8**	5.2	5.4	25.8	46.7
Adroit Pen	medium-expert	**94.6**	0.0	-1.7	43.5	41.8
Adroit Pen	expert	**96.1**	0.2	-0.4	51.4	45.4
Door Open	medium-expert	**95.7**	0.0	0.0	0.0	72.2
Door Open	expert	0.0	0.0	0.0	0.0	**97.4**

Table 1: Results for the DeepMind Control Walker task, the Robel D'Claw Screw task, Adroit Pen task and the Sawyer Door Open task. The scores are undiscounted average returns normalized to roughly lie between 0 and 100, where a score of 0 corresponds to a random policy, and 100 corresponds to an expert. LOMPO consistently outperforms LMBRL, offline SLAC, CQL, and behavioral cloning in almost all settings.

method is indicative of online model-based methods (Hafner et al., 2019, 2020). We also train the Stochastic Latent Actor Critic (SLAC) model (Lee et al. (2020)), a state of the art online learning algorithm from images, however we train fully online. The goal of this benchmark is to evaluate the need for representation learning in offline RL.

Results. Results are reported in Table 1. We see that LOMPO achieves high-scores across most high-fidelity simulation environments, using raw observations. Moreover, our proposed model outperforms other model-based learning algorithms across the board and is the only model-based learning algorithm that achieves any success on several environments. Comparing to model-free algorithms, LOMPO still outperforms CQL and behaviour cloning across most environments, with the exception of learning on the expert dataset on the Door Open task. This is a well-known phenomenon when learning dynamics models from narrow expert data. On the other hand, given the thin data distribution and relatively simple dynamics of the task, direct behavior cloning from images performs well on both the medium-expert and expert dataset. We hypothesize that LMOPO performs well on the D'Claw and Adroit expert datasets, as these environments are relatively stationary, as compared to a robot arm manipulation task, and even actions from a stochastic expert cover a wide range of the environment dynamics. The question of dataset size (question (3)) has not been extensively studied in offline RL, however we found that this almost as important as the data distribution itself. We believe this is an important question as collecting large robot dataset can still be complex and expensive task. We carried additional ablation experiments (Appendix B) and discover that performance for regular model-based and model-free methods can decline drastically as data size decrease, while LOMPO performance remains relatively stable.

5.2. Real Robot Experiments

To answer question (4), we deploy LOMPO on a real Franka Emika Panda robot arm.

Task. The environment consists of a Panda arm mounted in front of an Ikea desk cluttered with random distractor objects. The robot arm is initialized randomly above the desk and the drawer is initialized randomly in a open position. The goal of the robot is to navigate to the handle, hook it, and close the drawer. Observations are raw RGB images from a single overhead camera. The complete setup is shown in the rightmost picture in Figure 3.

Dataset. We use a pre-existing dataset of 1000 trajectories that was collected using a semi-supervised batch exploration algorithm (Chen et al., 2020). A small balanced dataset of 200 images (0.2% of the full dataset) is manually labeled with whether the drawer is open or closed. Following the set up by (Chen et al., 2020), we use this dataset to train a classifier to predict whether the drawer is open or closed. We use the classifier probability as a reward for RL, which leads to a sparse, noisy, and unstable reward signal, which is reflective of one challenge of real-world RL. Since the dataset was collected and labeled in the context of a different paper (Chen et al., 2020), this experiment evaluates the ability to reuse existing offline datasets, which further exemplifies real-world problems.

Comparisons. We compare LOMPO, LMBRL, Offline SLAC, and visual foresight (Finn and Levine, 2017) using an SV2P model (Babaeizadeh et al., 2018) and a CEM planner. As CQL did not achieve competitive performance on the simulated environments, we did not deploy it on the real robot. Moreover the offline dataset has high variance and consists of mostly non-task centric exploration, which is not suitable for imitation; hence we also did not evaluate behavioral cloning.

Results. We carry out 25 evaluation rollouts on the real robot[1] and summarize the results in Table 2. Overall, 24/25 of the LOMPO agent rollouts successfully navigate to the drawer handle, hook it, and push the drawer in; however the agent fully closes the drawer in only 19 of the rollouts for a final success rate of **76%**. We hypothesize that the agent does not always close the door as the classifier reward incorrectly predicts the drawer as closed when the drawer is slightly open. In contrast, the LMBRL, Offline SLAC, and visual foresight agents do not manage to successfully navigate to the correct handle location, hence achieving a success rate of **0%**. These experiments suggest that LOMPO's uncertainty estimation and pessimism are critical for good offline RL performance. Finally, we note that (Chen et al., 2020) evaluate visual foresight in the same environment but on an easier version of this task, where the robot arm is initialized near the drawer handle. In this shorter-horizon problem, visual foresight achieves a success rate of **65%** (Figure 8 of Chen et al. (2020)), which is still lower than LOMPO's success rate in the more difficult setting. Hence, this suggests that LOMPO is better able to solve problems with a longer time horizons by incorporating pessimism and a learned value function.

Table 2: Results for the Franka desk drawer-closing task.

Method	Success
LOMPO (ours)	**76.0%**
LMBRL	0.0%
Offline SLAC	0.0%
Visual Foresight	0.0%

6. Conclusion

We present an offline model-based RL algorithm that handles high-dimensional observations with latent dynamics models and uncertainty quantification. We noted that learning a visual dynamics model is challenging and quantifying uncertainty in the pixel space is extremely costly. We address such challenges by learning a latent space and quantify the model uncertainty in the latent space. Our algorithm, LOMPO, penalizes latent states with latent model uncertainty implemented as the latent model ensemble disagreement. LOMPO empirically outperforms previous latent model-based and model-free models in the offline setting on four simulated locomotion and manipulation tasks and one real-world robotic manipulation task. A potential direction for future work is to apply LOMPO to the multi-task setting. Having a single shared model trained with data from multiple tasks can help learn more accurate vision and dynamics models, thus improving performance, sample efficiency, and generalization.

1. Evaluation videos are available at https://sites.google.com/view/lompo/.

Acknowledgments

We want to thank Suraj Nair for sharing the BEE dataset with us and his help with setting up the Panda drawer environment. This work was supported in part by ONR grant N00014-20-1-2675 and Intel Corporation. CF is a CIFAR Fellow in the Learning in Machines and Brains program. Aravind Rajeswaran was supported by a JP Morgan PhD Fellowship (2020).

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2019.

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. Robel: Robotics benchmarks for learning with low-cost robots, 2019.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *ICML*, 2018.

Ershad Banijamali, Rui Shu, Hung Bui, Ali Ghodsi, et al. Robust locally-linear controllable embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 1751–1759. PMLR, 2018.

Peter J Bickel and David A Freedman. Some asymptotic theory for the bootstrap. *The annals of statistics*, pages 1196–1217, 1981.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

Annie S. Chen, HyunJi Nam, Suraj Nair, and Chelsea Finn. Batch exploration with examples for scalable robotic reinforcement learning, 2020.

Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, pages 885–897, 2019.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *arXiv preprint arXiv:1906.02736*, 2019.

Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2351–2358. IEEE, 2017.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *International Conference on Learning Representations*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *International Conference on Learning Representations*, 2020.

Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. *International Conference on Learning Representations*, 2020.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12498–12509, 2019.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, volume 12. Springer, 2012.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arxiv:1907.00953.pdf*, 2020.

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

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *CoRR*, abs/1904.08473, 2019.

Yao Liu, A. Swaminathan, A. Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *ArXiv*, abs/2007.08202, 2020.

Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1048–1055. IEEE, 2019.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. In *ICML*, 2020.

Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *arXiv preprint arXiv:1810.07121*, 2018.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

HJ Suh and Russ Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. *arXiv preprint arXiv:2002.09093*, 2020.

Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *J. Mach. Learn. Res*, 16:1731–1755, 2015.

Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *arXiv preprint arXiv:2008.05533*, 2020.

Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. dm control: Software and tasks for continuous control, 2020.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja1, Agnieszka, Grabska-Barwinska, Jack Rae1, Piotr Mirowski, Joel Z. Leibo, Adam Santoro, Mevlana Gemici, Malcolm Reynolds, Tim Harley, Josh Abramson, Shakir Mohamed, Danilo Rezende, David Saxton, Adam

Cain, Chloe Hillier, David Silver, Koray Kavukcuoglu, Matt Botvinick, Demis Hassabis, and Timothy Lillicrap. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arxiv:1803.10760*, 2018.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Lin Yen-Chen, Maria Bauza, and Phillip Isola. Experience-embedded visual foresight. In *Conference on Robot Learning*, pages 1015–1024, 2019.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020a.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020b.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, pages 7444–7453. PMLR, 2019.

Ruiyi Zhang, Bo Dai, Li Lihong, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values, 2020. *Preprint*.

Appendix A. Environments and Datasets

We describe the simulated environments and the dataset collection in more detail below.
Environments and Tasks. We evaluate our method in four simulated environments.

1. The first task is the standard walker task from the DeepMind Control suite (Tassa et al., 2020). The observations consist of raw 64×64 images, similiar to previous works. We apply an action repeat of 2 over the base environment. This is a standard practice (Lee et al., 2020; Hafner et al., 2020) as timestep in these environments is relatively short and leaves low visual footprint, which makes model learning hard.

2. The second experiment consists of a modified version of the D'Claw screw task from the Robel benchmark (Ahn et al., 2019), where the goal of the robot is to continuously turn the valve as fast as possible. The agent receives a dense negative penalty for positioning the fingers and a sparse reward whenever it turns the valve. The observation space consist of robot proprioception and 128×128 raw images. We apply an action repeat of 2 over the base environment.

3. The third environment is based on the Adroit pen task (Rajeswaran et al., 2018) with a fixed goal, which requires the agent to flip the pen around and catch it at certain angle. The observation space consist of robot proprioception and 128×128 raw images. We apply an action repeat of 4 over the base environment.

4. The final simulation experiment is based on a Sawyer manipulation task (Yu et al., 2020a), which requires opening a door. The agent received a sparse reward when the door is fully opened and no reward otherwise. The goal of this environment is to test learning in a realistic multi-stage robot arm environment with a sparse reward. This is a hard environment, not solvable with online RL. The observation space consists of 128×128 images without access to the robot state. We apply an action repeat of 4 over the base environment.

We provide visualizations of all simulated tasks in Figure 3. Each of these environments presents different challenges. In the Walker task the agent needs to learn a forward dynamics model completely from images, which are generated by a non-stationary camera. The D'Claw and Adroit environments are both quite challenging as the model needs to merge proprioception and visual information in order to estimate a hard contact model with realistic physics, as well as forward dynamics under a high-dimensional action space (9 and 26 respectively) and a sparse reward function (for the calw environment). The Sawyer arm environment requires learning a 3D dynamics model without access to depth estimation and a sparse reward function, both of which are common in real world RL.
Datasets. We construct new sets of offline datasets with image observations. Similar to the protocol by Fu et al. (2020), we create three types of datasets for each task, which are obtained by training an agent using soft-actor critic (Haarnoja et al., 2018) from the ground-truth state and recording the corresponding image observations, actions, and rewards. The *medium-replay* datasets consist of data from the training replay buffer up to the point where the policy reaches performance of about half the expert level performance. The goal of these datasets is to test learning on incomplete training data. The *medium-expert* datasets consist of the second half of the replay buffer after the agent reaches medium-level performance. The goal of these datasets is to test learning on a mixture

of data from sub-optimal policies. The *expert* dataset consist of data sampled from the stochastic SAC expert policy. The goal of these datasets is to test learning on a thin data distribution. The dataset sizes for Walker, D'Claw, Adroit and the Sawyer environments are 100K, 180K, 1M and 50K transitions respectively.

Appendix B. Ablation Studies on Varying Offline Dataset Size

We test the effect of dataset size, given that the data is sampled from the same distribution. We create a medium-expert dataset of size 1M on the D'Claw Screw environment by mixing data from 3 separate policy training runs. We then created two more datasets by sub-sampling by a factor of 5 and 25 respectively. We observe that LMOPO still performs well in the low-data regime, as compared to regular latent model-based RL and Offline SLAC.

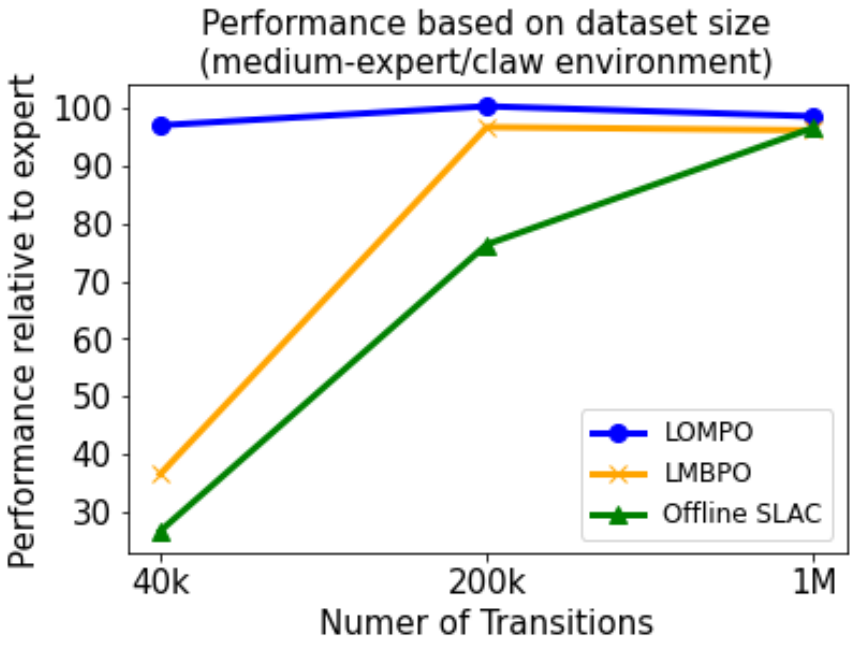

Figure 4: Agent performance based on dataset size

Appendix C. Implementation details

The latent dynamics model and the observation model consist of the following components as in (Hafner et al., 2020):

$$\begin{aligned}
&\text{Image encoder:} && h_t = E_\theta(x_t) \\
&\text{Inference model:} && s_t \sim q_\theta(s_t|h_t, s_{t-1}, a_{t-1}) \\
&\text{Latent transition model:} && s_t \sim \widehat{T}_\theta(s_t|s_{t-1}, a_{t-1}) \\
&\text{Reward predictor:} && r_t \sim p_\theta(r_t|s_t) \\
&\text{Image decoder:} && x_t \sim D_\theta(x_t|s_t).
\end{aligned} \tag{10}$$

where $x_t \in \mathcal{X}, s_t \in \mathcal{S}, a_t \in \mathcal{A}$ are the environment observation, latent state, action at time t respectively. We use θ to denote the concatenation of all the parameters involved in the latent space dynamics model. The latent dynamics model is represented by a RSSM (Hafner et al. (2019)). Specifically, we adopt the latent space representation $s_t = [d_t, z_t]$, which consists of a deterministic d_t and a sampled stochastic representation z_t. With such a latent space representation, we use the following components:

$$\begin{aligned} &\text{Deterministic State Model:} && d_t = f_\theta(d_{t-1}, z_{t-1}, a_{t-1}) \\ &\text{Stochastic Inference Model:} && z_t \sim q_\theta(z_t | h_t, d_t) \\ &\text{Ensemble of Transition Models:} && z_t \sim p_{\theta_k}(z_t | d_t, z_{t-1}, a_{t-1}) \end{aligned} \tag{11}$$

where $h_t = E_\theta(x_t)$ are observation features as defined in Eq. 10. The deterministic representation f_θ is implemented as a single GRU cell and is shared between the forward and inference models. All the learned forward models $\widehat{T}_{\theta_k}, k = 1, \ldots, K$ in the ensemble share the same deterministic model f_θ but separate stochastic transition models $p_{\theta_k}, k = 1, \ldots, K$, which are implemented as MLPs. Finally the stochastic inference model is also implemented as an MLP network.

The encoder network E_θ is modeled as a convolutional neural network. For the DeepMind Control Walker task the network has 4 layers with [32, 64, 128, 256] channels respectively. For the D'Claw Screw, Adroit Pen and Door Open environments the convolutional model has 5 layers with [32, 64, 128, 256, 256] channels. All kernels have size 4 and stride 2. The reconstruction model D_θ for DeepMind Control walker task has 4 layers with [128, 64, 32, 3] channels with kernel size [5, 5, 6, 6] and stride 2. For the other environments the reconstruction network has 5 layers with [128, 64, 32, 32, 3] channels with kernel size [5,5,5,4,4] and stride 2. The reward reconstruction network is a two-layer fully connected network. The deterministic path f_θ of the RSSM is modeled as a GRU cell with 256 units for all environments, except the Adroit Pen task, which uses 512 units. All the forward models $T_{\theta_i}, i = 1, \ldots, K$ and inference q_θ models are 3-layer fully-connected networks with 256 units, except for the Adroit Pen task, which uses 512. The variational model is trained with the Adam optimizer with $lr = 6e-4$.

Both the actor and the critic are modeled as fully-connected networks with 3 layers and 256 units. We use the Adam optimizer with $lr = 3e-4$.

Appendix D. Main Algorithm

We present the full LOMPO algorithm in Algorithm 1.

Appendix E. Variational Latent Model Samples

For samples generated by our variational latent models, see Figure 5.

Algorithm 1 LOMPO

Input: model train steps, initial real steps, initial latent steps, number of epochs, epoch real batch, epoch latent batch, number of models K

for *model train steps* **do**
  Sample a batch of sequences of raw observations $(o_{1:H}, a_{1:H-1}, r_{1:H-1})$ from batch dataset $\mathcal{D}_{\text{env}}$ and train variational ensemble model using equation 8 with K models
end
while *size* $\mathcal{B}_{real} <$ *initial real steps* **do**
  Sample a batch of sequences $(o_{1:H}, a_{1:H-1}, r_{1:H-1})$ from batch dataset D_{env}
  Sample latent states $s_{1:H} \sim q_\theta(s_{1:H}|o_{1:H}, a_{1:H-1})$ from the trained inference model
  Add batches s_t, a_t, s_{t+1}, r_t to real replay buffer $\mathcal{B}_{real}$
end
while *size* $\mathcal{B}_{latent} <$ *initial latent steps* **do**
  Sample a batch of sequences $(o_{1:H}, a_{1:H-1}, r_{1:H-1})$ from batch dataset $\mathcal{D}_{\text{env}}$
  Sample a set of latent states $\mathbf{S} \sim q_\theta(s_{1:H}|o_{1:H}, a_{1:H-1})$ from the trained inference model **for** $s_0 \in \mathbf{S}$ **do**
    for $h \in \{1 : H\}$ **do**
      Sample a latent transition model from p_{θ_j} from $i = 1, \ldots, K$
      Sample a random action a_{h-1} and next state $s_h \sim \widehat{T}_{\theta_j}(s|s_{h-1}, a_{h-1})$
      Compute reward $\tilde{r}_{h-1}$ using equation 9 and add $(s_{h-1}, a_{h-1}, s_h, \tilde{r}_{h-1})$ to $\mathcal{B}_{latent}$
    end
  end
end
for *number of epochs* **do**
  for *number actor-critic steps* **do**
    Equally sample batch $(\mathbf{s_t}, \mathbf{a_t}, \mathbf{r_t}, \mathbf{s_{t+1}})$ from $\mathcal{B}_{real} \cup \mathcal{B}_{latent}$
    Update $Q_\phi(s, a), \pi_\phi(a|s)$ using any off-policy algorithm
  end
  for *epoch real batch* **do**
    Sample a batch of sequences $(o_{1:H}, a_{1:H-1}, r_{1:H-1})$ from batch dataset D_{env}
    Sample latent states $s_{1:H} \sim q_\theta(s_{1:H}|o_{1:H}, a_{1:H-1})$ from the trained inference model
    Add batches s_t, a_t, s_{t+1}, r_t to real replay buffer $\mathcal{B}_{real}$
  end
  for *epoch latent batch* **do**
    Sample a batch of sequences $(o_{1:H}, a_{1:H-1}, r_{1:H-1})$ from batch dataset $\mathcal{D}$
    Sample a set of latent states $\mathbf{S} \sim q_\theta(s_{1:H}|o_{1:H}, a_{1:H-1})$ from the trained inference model
    for $s_0 \in \mathbf{S}$ **do**
      for $h \in \{1 : H\}$ **do**
        Sample a latent transition model from $\widehat{T}_{\theta_j}$ from $i = 1, \ldots, K$
        Sample an action $a_{h-1} \sim \pi_\phi(a|s_{h-1})$ and next state $s_h \sim \widehat{T}_{\theta_j}(s|s_{h-1}, a_{h-1})$
        Compute reward $\tilde{r}_{h-1}$ using equation 9 and add transition to $\mathcal{B}_{latent}$
      end
    end
  end
end

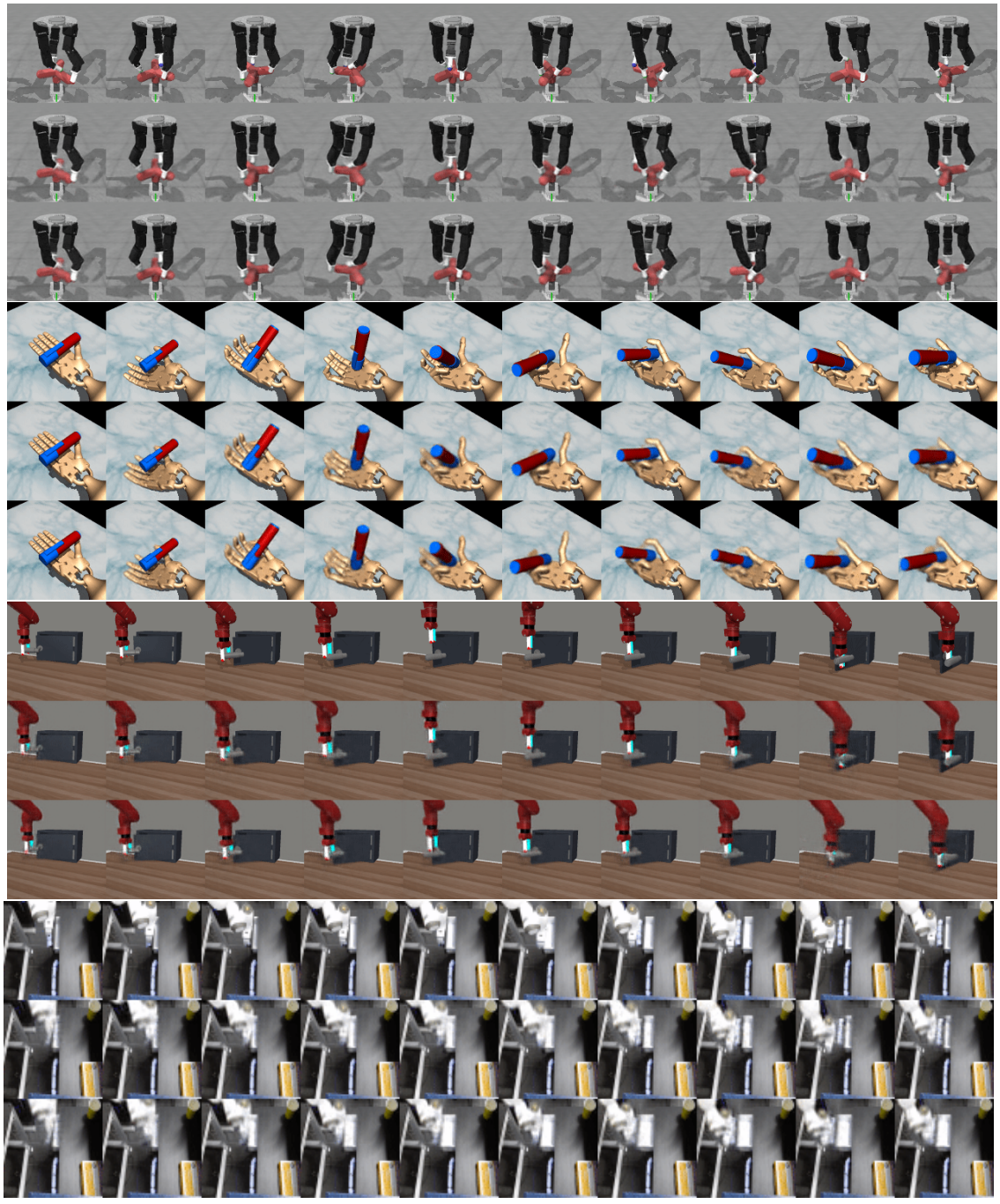

Figure 5: Samples from the learned variational model. Fist row: ground truth sequence; second row: posterior model samples; third row: ensemble latent model rollout conditioned on the action sequence.