VARIATIONAL LATENT BRANCHING MODEL FOR OFF-POLICY EVALUATION

Qitong Gao* **Ge Gao†** **Min Chi†** **Miroslav Pajic***

ABSTRACT

Model-based methods have recently shown great potential for off-policy evaluation (OPE); offline trajectories induced by behavioral policies are fitted to transitions of Markov decision processes (MDPs), which are used to rollout simulated trajectories and estimate the performance of policies. Model-based OPE methods face two key challenges. First, as offline trajectories are usually fixed, they tend to cover limited state and action space. Second, the performance of model-based methods can be sensitive to the initialization of their parameters. In this work, we propose the variational latent branching model (VLBM) to learn the transition function of MDPs by formulating the environmental dynamics as a compact latent space, from which the next states and rewards are then sampled. Specifically, VLBM leverages and extends the variational inference framework with the ***recurrent state alignment (RSA)***, which is designed to capture as much information underlying the limited training data, by smoothing out the information flow between the variational (encoding) and generative (decoding) part of VLBM. Moreover, we also introduce the ***branching architecture*** to improve the model's robustness against randomly initialized model weights. The effectiveness of the VLBM is evaluated on the deep OPE (DOPE) benchmark, from which the training trajectories are designed to result in varied coverage of the state-action space. We show that the VLBM outperforms existing state-of-the-art OPE methods in general.

1 INTRODUCTION

Off-policy evaluation (OPE) allows for evaluation of reinforcement learning (RL) policies without online interactions. It is applicable to many domains where on-policy data collection could be prevented due to efficiency and safety concerns, *e.g.*, healthcare (Gao et al., 2022c;a; Tang & Wiens, 2021), recommendation systems (Mehrotra et al., 2018; Li et al., 2011), education (Mandel et al., 2014), social science (Segal et al., 2018) and optimal control (Silver et al., 2016; Vinyals et al., 2019; Gao et al., 2020a; 2019; 2020b). Recently, as reported in the deep OPE (DOPE) benchmark (Fu et al., 2020b), model-based OPE methods, leveraging feed-forward (Fu et al., 2020b) and auto-regressive (AR) (Zhang et al., 2020a) architectures, have shown promising results toward estimating the return of target policies, by fitting transition functions of MDPs. However, model-based OPE methods remain challenged as they can only be trained using offline trajectory data, which often offers limited coverage of state and action space. Thus, they may perform sub-optimally on tasks where parts of the dynamics are not fully explored (Fu et al., 2020b). Moreover, different initialization of the model weights could lead to varied evaluation performance (Hanin & Rolnick, 2018; Rossi et al., 2019), reducing the robustness of downstream OPE estimations. Some approaches in RL policy optimization literature use latent models trained to capture a compact space from which the dynamics underlying MDPs are extrapolated; this allows learning expressive representations over the state-action space. However, such approaches usually require *online* data collections as the focus is on quickly navigating to the high-reward regions (Rybkin et al., 2021), as well as on improving coverage of the explored state and action space (Zhang et al., 2019; Hafner et al., 2019; 2020a) or sample efficiency (Lee et al., 2020).

In this work, we propose the variational latent branching model (VLBM), aiming to learn a compact and disentangled latent representation space from offline trajectories, which can better capture the

*Duke University, USA. Emails: {qitong.gao, miroslav.pajic}@duke.edu.
†North Carolina State University, USA. Emails: {ggao5, mchi}@ncsu.edu
Code available at https://github.com/gaoqitong/vlbm.

dynamics underlying environments. VLBM enriches the architectures and optimization objectives for existing latent modeling frameworks, allowing them to learn from a *fixed* set of *offline* trajectories. Specifically, VLBM considers learning variational (encoding) and generative (decoding) distributions, both represented by long short-term memories (LSTMs) with reparameterization (Kingma & Welling, 2013), to encode the state-action pairs and enforce the transitions over the latent space, respectively. To train such models, we optimize over the evidence lower bound (ELBO) jointly with a *recurrent state alignment* (RSA) term defined over the LSTM states; this ensures that the information encoded into the latent space can be effectively teased out by the decoder. Then, we introduce the *branching architecture* that allows for multiple decoders to jointly infer from the latent space and reach a consensus, from which the next state and reward are generated. This is designed to mitigate the side effects of model-based methods where different weight initializations could lead to varied performance (Fu et al., 2020b; Hanin & Rolnick, 2018; Rossi et al., 2019).

We focus on using the VLBM to facilitate OPE since it allows to better distinguish the improvements made upon learning dynamics underlying the MDP used for estimating policy returns, as opposed to RL training where performance can be affected by multiple factors, *e.g.*, techniques used for exploration and policy optimization. Moreover, model-based OPE methods is helpful for evaluating the safety and efficacy of RL-based controllers before deployments in the real world (Gao et al., 2022b), *e.g.*, how a surgical robot would react to states that are critical to a successful procedure. The key contributions of this paper are summarized as follows: (i) to the best of our knowledge, the VLBM is the first method that leverages variational inference for OPE. It can be trained using offline trajectories and capture environment dynamics over latent space, as well as estimate returns of target (evaluation) policies accurately. (ii) The design of the RSA loss term and branching architecture can effectively smooth the information flow in the latent space shared by the encoder and decoder, increasing the expressiveness and robustness of the model. This is empirically shown in experiments by comparing with ablation baselines. (iii) Our method generally outperforms existing model-based and model-free OPE methods, for evaluating policies over various D4RL environments (Fu et al., 2020a). Specifically, we follow guidelines provided by the DOPE benchmark (Fu et al., 2020b), which contains challenging OPE tasks where the training trajectories include varying levels of coverage of the state-action space, and target policies are designed toward resulting in state-action distributions different from the ones induced by behavioral policies.

2 Variational Latent Branching Model

In this section, we first introduce the objective of OPE and the variational latent model (VLM) we consider. Then, we propose the recurrent state alignment (RSA) term as well as the branching architecture that constitute the variational latent branching model (VLBM).

2.1 OPE Objective

We first introduce the MDP used to characterize the environment. Specifically, an MDP can be defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, s_0, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition distribution usually captured by probabilities $p(s_t|s_{t-1}, a_{t-1})$, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, s_0 is the initial state sampled from the initial state distribution $p(s_0)$, $\gamma \in [0, 1)$ is the discounting factor. Finally, the agent interacts with the MDP following some policy $\pi(a|s)$ which defines the probabilities of taking action a at state s. Then, the goal of OPE can be formulated as follows. Given trajectories collected by a *behavioral* policy β, $\rho^\beta = \{[(s_0, a_0, r_0, s_1), \ldots, (s_{T-1}, a_{T-1}, r_{T-1}, s_T)]^{(0)}, [(s_0, a_0, r_0, s_1), \ldots]^{(1)}, \ldots | a_t \sim \beta(a_t|s_t)\}$[1], estimate the expected total return over the unknown state-action visitation distribution ρ^π of the *target* (evaluation) policy π – *i.e.*, for T being the horizon,

$$\mathbb{E}_{(s,a)\sim\rho^\pi, r\sim R}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right]. \tag{1}$$

2.2 Variational Latent Model

We consider the VLM consisting of a prior $p(z)$ over the latent variables $z \in \mathcal{Z} \subset \mathbb{R}^l$, with $\mathcal{Z}$ representing the latent space and l the dimension, along with a variational encoder $q_\psi(z_t|z_{t-1}, a_{t-1}, s_t)$

[1]We slightly abuse the notation ρ^β, to represent either the trajectories or state-action visitation distribution under the behavioral policy, depending on the context.

and a generative decoder $p_\phi(z_t, s_t, r_{t-1}|z_{t-1}, a_{t-1})$, parameterized by ψ and ϕ respectively. Basics of variational inference are introduced in Appendix F.

Latent Prior $p(z_0)$. The prior specifies the distribution from which the latent variable of the *initial* stage, z_0, is sampled. We configure $p(z_0)$ to follow a Gaussian with zero mean and identity covariance matrix, which is a common choice under the variational inference framework (Kingma & Welling, 2013; Lee et al., 2020).

Variational Encoder for Inference $q_\psi(z_t|z_{t-1}, a_{t-1}, s_t)$. The encoder is used to approximate the intractable posterior, $p(z_t|z_{t-1}, a_{t-1}, s_t) = \frac{p(z_{t-1}, a_{t-1}, z_t, s_t)}{\int_{z_t \in \mathcal{Z}} p(z_{t-1}, a_{t-1}, z_t, s_t) dz_t}$, where the denominator requires integrating over the unknown latent space. Specifically, the encoder can be decomposed into two parts, given that

$$\begin{aligned} & q_\psi(z_{0:T}|s_{0:T}, a_{0:T-1}) \\ =& q_\psi(z_0|s_0) \prod_{t=1}^{T} q_\psi(z_t|z_{t-1}, a_{t-1}, s_t); \end{aligned} \quad (2)$$

here, $q_\psi(z_0|s_0)$ encodes the initial state s_0 in to the corresponding latent variable z_0, then, $q_\psi(z_t|z_{t-1}, a_{t-1}, s_t)$ enforces the transition from z_{t-1} to z_t conditioned on a_{t-1} and s_t. Both distributions are *diagonal* Gaussians[2], with means and diagonal of covariance matrices determined by multi-layered perceptron (MLP) (Bishop, 2006) and long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) respectively. The weights for both neural networks are referred to as ψ in general.

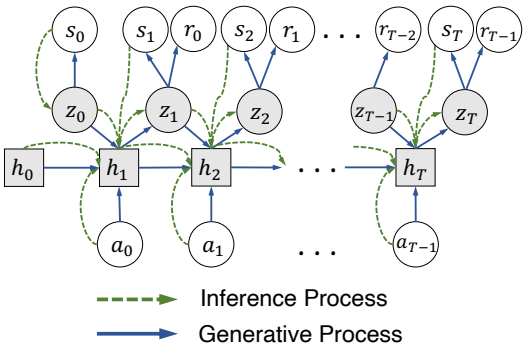

Figure 1: Architecture of variational latent model (VLM) we consider.

Consequently, the *inference* process for z_t can be summarized as

$$z_0^\psi \sim q_\psi(z_0|s_0), \quad h_t^\psi = f_\psi(h_{t-1}^\psi, z_{t-1}^\psi, a_{t-1}, s_t), \quad z_t^\psi \sim q_\psi(z_t|h_t^\psi), \quad (3)$$

where f_ψ represents the LSTM layer and h_t^ψ the LSTM recurrent (hidden) state. Note that we use ψ in superscripts to distinguish the variables involved in this *inference* process, against the *generative* process introduced below. Moreover, reparameterization can be used to sample z_0^ψ and z_t^ψ, such that gradients of sampling can be back-propagated, as introduced in (Kingma & Welling, 2013). Overview of the inference and generative processes are illustrated in Fig. 1.

Generative Decoder for Sampling $p_\phi(z_t, s_t, r_{t-1}|z_{t-1}, a_{t-1})$. The decoder is used to interact with the target policies and acts as a synthetic environment during policy evaluation, from which the expected returns can be estimated as the mean return of simulated trajectories. The decoder can be represented by the multiplication of three diagonal Gaussian distributions, given that

$$p_\phi(z_{1:T}, s_{0:T}, r_{0:T-1}|z_0, \pi) = \prod_{t=0}^{T} p_\phi(s_t|z_t) \prod_{t=1}^{T} p_\phi(z_t|z_{t-1}, a_{t-1}) p_\phi(r_{t-1}|z_t), \quad (4)$$

with $a_t \sim \pi(a_t|s_t)$ at each time step. Specifically, $p_\phi(z_t|z_{t-1}, a_{t-1})$ has its mean and covariance determined by an LSTM, enforcing the transition from z_{t-1} to z_t in the latent space given action a_{t-1}. In what follows, $p_\phi(s_t|z_t)$ and $p_\phi(r_{t-1}|z_t)$ generate the current state s_t and reward r_{t-1} given z_t, whose mean and covariance are determined by MLPs. As a result, the *generative* process starts with sampling the initial latent variable from the latent prior, *i.e.*, $z_0^\phi \sim p(z_0)$. Then, the initial state $s_0^\phi \sim p_\phi(s_0|z_0^\phi)$ and action $a_0 \sim \pi(a_0|s_0^\phi)$ are obtained from p_ϕ and target policy π, respectively; the rest of *generative* process can be summarized as

$$\begin{aligned} & h_t^\phi = f_\phi(h_{t-1}^\phi, z_{t-1}^\phi, a_{t-1}), \quad \tilde{h}_t^\phi = g_\phi(h_t^\phi), \quad z_t^\phi \sim p_\phi(\tilde{h}_t^\phi), \\ & s_t^\phi \sim p_\phi(s_t|z_t^\phi), \quad r_{t-1}^\phi \sim p_\phi(r_{t-1}|z_t^\phi), \quad a_t \sim \pi(a_t|s_t^\phi), \end{aligned} \quad (5)$$

[2]Assume that different dimensions of the states are non-correlated with each other. Otherwise, the states can be projected to orthogonal basis, such that non-diagonal elements of the covariance matrix will be zeros.

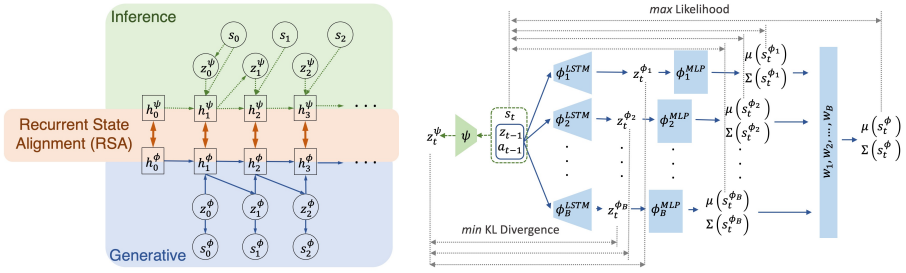

Figure 2: (Left) Recurrent state alignment (RSA) applied over the recurrent hidden states between inference and generative process illustrated separately. (Right) Single-step forward pass of the variational latent branching model (VLBM), the training objectives for each branch and final predictions.

where f_ϕ is the LSTM layer producing recurrent state h_t^ϕ. Then, an MLP g_ϕ is used to generate mapping between h_t^ϕ and $\tilde{h}_t^\phi$ that will be used for recurrent state alignment (RSA) introduced below, to augment the information flow between the inference and generative process.

Furthermore, to train the elements in the encoder (3) and decoder (5), one can maximize the evidence lower bound (ELBO), a lower bound of the joint log-likelihood $p(s_{0:T}, r_{0:T-1})$, following

$$\mathcal{L}_{ELBO}(\psi,\phi) = \mathbb{E}_{q_\psi}\Big[\sum_{t=0}^{T}\log p_\phi(s_t|z_t) + \sum_{t=1}^{T}\log p_\phi(r_{t-1}|z_t) - KL\big(q_\psi(z_0|s_0)||p(z_0)\big) - \sum_{t=1}^{T} KL\big(q_\psi(z_t|z_{t-1},a_{t-1},s_t)||p_\phi(z_t|z_{t-1},a_{t-1})\big)\Big]; \tag{6}$$

here, the first two terms represent the log-likelihood of reconstructing the states and rewards, and the last two terms regularize the approximated posterior. The proof can be found in Appendix E.

2.3 Recurrent State Alignment

The latent model discussed above is somewhat reminiscent of the ones used in model-based RL policy training methods, *e.g.*, recurrent state space model (RSSM) used in PlaNet (Hafner et al., 2019) and Dreamer (Hafner et al., 2020a;b), as well as similar ones in Lee et al. (2020); Lu et al. (2022). Such methods rely on a *growing* experience buffer for training, which is collected *online* by the target policy that is being concurrently updated (with exploration noise added); however, OPE aims to extrapolate returns from a fixed set of *offline* trajectories which may result in limited coverage of the state and action space. Consequently, directly applying VLM for OPE can lead to subpar performance empirically; see results in Sec. 3. Moreover, the encoder above plays a key role of capturing the temporal transitions between latent variables, *i.e.*, $p_\psi(z_t|z_{t-1},a_{t-1},s_t)$ from (2). However, it is *absent* in the generative process, as the decoder leverages a separate network to determine the latent transitions, *i.e.*, $p_\phi(z_t|z_{t-1},a_{t-1})$. Moreover, from the ELBO (6) above it can be seen that only the KL-divergence terms are used to regularize these two parts, which may not be sufficient for OPE as limited offline trajectories are provided. As a result, we introduce the RSA term as part of the training objective, to further regularize $p_\psi(z_t|z_{t-1},a_{t-1},s_t)$ and $p_\phi(z_t|z_{t-1},a_{t-1})$. A graphical illustration of RSA can be found in Fig. 2.[3]

Specifically, RSA is defined as the mean *pairwise* squared error between h_t^ψ from the encoder (3) and $\tilde{h}_t^\phi$ from the decoder (5), *i.e.*,

$$\mathcal{L}_{RSA}(\tilde{h}_t^\phi, h_t^\psi;\psi,\phi) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\frac{M(M-1)}{2}\Big[\sum_{j=1}^{M-1}\sum_{k=j+1}^{M}\big((\tilde{h}_t^\phi[j]-\tilde{h}_t^\phi[k])-(h_t^\psi[j]-h_t^\psi[k])\big)^2\Big]; \tag{7}$$

here, we assume that both LSTM recurrent states have the same dimension $\tilde{h}_t^\phi, h_t^\psi \in \mathbb{R}^M$, with $h_t^{(\cdot)}[j]$ referring to the j-th element of the recurrent state, and N the number of training trajectories.

Here, we choose the pairwise squared loss over the classic mean squared error (MSE), because MSE could be too strong to regularize h_t^ψ and $\tilde{h}_t^\phi$ which support the inference and generative processes respectively and are not supposed to be exactly the same. In contrast, the pairwise loss (7) can

[3]Rewards and actions are omitted for conciseness of the presentation.

promote structural similarity between the LSTM recurrent states of the encoder and decoder, without strictly enforcing them to become the same. Note that this design choice has been justified in Sec. 3 through an ablation study by comparing against models trained with MSE. In general, the pairwise loss has also been adopted in many domains for similar purposes, *e.g.*, object detection (Gould et al., 2009; Rocco et al., 2018), ranking systems (Doughty et al., 2018; Saquil et al., 2021) and contrastive learning (Wang et al., 2021; Chen et al., 2020). Similarly, we apply the pairwise loss over h_t^ψ and $\tilde{h}_t^\phi$, instead of directly over h_t^ψ and h_t^ϕ, as the mapping g_ϕ (from equation 5) could serve as a regularization layer to ensure optimality over $\mathcal{L}_{RSA}$ without changing h_t^ψ, h_t^ϕ significantly.

As a result, the objective for training the VLM, following architectures specified in (3) and (5), can be formulated as

$$\max_{\psi,\phi} \mathcal{L}_{VLM}(\psi,\phi) = \max_{\psi,\phi} \Big(\mathcal{L}_{ELBO}(\psi,\phi) - C \cdot \mathcal{L}_{RSA}(\tilde{h}_t^\phi, h_t^\psi; \psi, \phi) \Big), \tag{8}$$

with $C > 0$ and $C \in \mathbb{R}$ being the constant balancing the scale of the ELBO and RSA terms.

2.4 Branching for Generative Decoder

The performance of model-based methods can vary upon different design factors (Fu et al., 2020b; Hanin & Rolnick, 2018). Specifically, Rossi et al. (2019) has found that the convergence speed and optimality of variational models are sensitive to the choice of weight initialization techniques. Moreover, under the typical variational inference setup followed by the VLM above, the latent transitions reconstructed by the decoder, $p_\phi(z_t|z_{t-1}, a_{t-1})$, are only trained through regularization losses in (6) and (7), but are fully responsible for rolling out trajectories during evaluation. Consequently, in this sub-section we introduce the branching architecture for decoder, with the goal of minimizing the impact brought by random weight initialization of the networks, and allowing the decoder to best reconstruct the latent transitions $p_\phi(z_t|z_{t-1}, a_{t-1})$ as well as s_t's and r_{t-1}'s correctly. Specifically, the branching architecture leverages an ensemble of $B \in \mathbb{Z}^+$ decoders to tease out information from the latent space formulated by the encoder, with final predictions sampled from a mixture of the Gaussian output distributions from (5). Note that the classic setup of ensembles is not considered, *i.e.*, train and average over B VLMs end-to-end; because in this case B different latent space exist, each of which is still associated with a single decoder, leaving the challenges above unresolved. This design choice is justified by ablations studies in Sec. 3, by comparing VLBM against a (classic) ensemble of VLMs.

Branching Architecture. Consider the generative process involving B branches of the decoders parameterized by $\{\phi_1, \dots, \phi_B\}$. The forward architecture over a single step is illustrated in Fig. 2.[4] Specifically, the procedure of sampling $z_t^{\phi_b}$ and $s_t^{\phi_b}$ for each $b \in [1, B]$ follows from (5). Recall that by definition $p_{\phi_b}(s_t|z_t^{\phi_b})$ follows multivariate Gaussian with mean and diagonal of covariance matrix determined by the corresponding MLPs, *i.e.*, $\mu(s_t^{\phi_b}) = \phi_{b,\mu}^{MLP}(z_t^{\phi_b})$ and $\Sigma_{diag}(s_t^{\phi_b}) = \phi_{b,\Sigma}^{MLP}(z_t^{\phi_b})$. In what follows, the final outcome s_t^ϕ can be sampled following diagonal Gaussian with mean and variance determined by weighted averaging across all branches using weights w_b's, *i.e.*,

$$s_t^\phi \sim p_\phi(s_t|z_t^{\phi_1}, \dots, z_t^{\phi_B}) = \mathcal{N}\Big(\boldsymbol{\mu} = \sum_b w_b \cdot \mu(s_t^{\phi_b}), \boldsymbol{\Sigma}_{diag} = \sum_b w_b^2 \cdot \Sigma_{diag}(s_t^{\phi_b})\Big). \tag{9}$$

The objective below can be used to jointly update, w_b's, ψ and ϕ_b's, *i.e.*,

$$\begin{aligned}
&\max_{\psi,\phi,w} \mathcal{L}_{VLBM}(\psi, \phi_1, \dots, \phi_B, w_1, \dots, w_B) \\
&= \max_{\psi,\phi,w} \Big(\sum_{t=0}^{T} \log p_\phi(s_t^\phi|z_t^{\phi_1}, \dots, z_t^{\phi_B}) - C_1 \cdot \sum_b \mathcal{L}_{RSA}(\tilde{h}_t^{\phi_b}, h_t^\psi; \psi, \phi_b) + C_2 \sum_b \mathcal{L}_{ELBO}(\psi, \phi_b) \Big), \\
&\text{s.t.} \quad w_1, \dots, w_B > 0\ , \ \sum_b w_b = 1 \text{ and constants } C_1, C_2 > 0.
\end{aligned} \tag{10}$$

Though the first term above already propagates through all w_b's and ϕ_b's, the third term and constraints over w_b's regularize ϕ_b in each individual branch such that they are all trained toward maximizing

[4] For simplicity, the parts generating rewards are omitted without lost of generality.

the likelihood $p_{\phi_b}(s_t^{\phi_b}|z_t^{\phi_b})$. Pseudo-code for training and evaluating the VLBM can be found in Appendix C. Further, in practice, one can define $w_b = \frac{v_b^2}{\epsilon+\sum_b v_b^2}$, with $v_b \in \mathbb{R}$ the learnable variables and $0 < \epsilon \ll 1$, $\epsilon \in \mathbb{R}$, the constant ensuring denominator to be greater than zero, to convert (10) into unconstrained optimization and solve it using gradient descent. Lastly, note that complementary latent modeling methods, *e.g.*, latent overshooting from Hafner et al. (2019), could be adopted in (10). However, we keep the objective straightforward, so that the source of performance improvements can be isolated.

3 Experiments

To evaluate the VLBM, we follow the guidelines from the deep OPE (DOPE) benchmark (Fu et al., 2020b). Specifically, we follow the D4RL branch in DOPE and use the Gym-Mujoco and Adroit suites as the test base (Fu et al., 2020a). Such environments have long horizons and high-dimensional state and action space, which are usually challenging for model-based methods. The provided offline trajectories for training are collected using behavioral policies at varied scale, including limited exploration, human teleoperation etc., which can result in different levels of coverage over the state-action space. Also, the target (evaluation) policies are generated using online RL training, aiming to reduce the similarity between behavioral and target policies; it introduces another challenge that during evaluation the agent may visit states unseen from training trajectories.

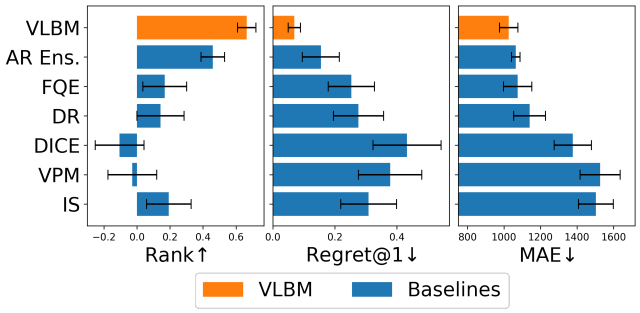

Figure 3: Mean rank correlation, regret@1 and MAE over all the 32 Gym-Mujoco and Adroit tasks, showing VLBM achieves state-of-the-art performance overall.

Environmental and Training Setup. A total of 8 environments are provided by Gym-Mujoco and Adroit suites (Fu et al., 2020b;a). Moreover, each environment is provided with 5 (for Gym-Mujoco) or 3 (for Adroit) training datasets collected using different behavioral policies, resulting in a total of 32 sets of `env-dataset` tasks[5] – a full list can be found in Appendix A. DOPE also provides 11 target policies for each environment, whose performance are to be evaluated by the OPE methods. They in general result in varied scales of returns, as shown in the x-axes of Fig. 7. Moreover, we consider the decoder to have $B = 10$ branches, *i.e.*, $\{p_{\phi_1}, \dots, p_{\phi_{10}}\}$. The dimension of latent space is set to be 16, *i.e.*, $z \in \mathcal{Z} \subset \mathbb{R}^{16}$. Other implementation details can be found in Appendix A.

Baselines and Evaluation Metrics. In addition to the five baselines reported from DOPE, *i.e.*, importance sampling (IS) (Precup, 2000), doubly robust (DR) (Thomas & Brunskill, 2016), variational power method (VPM) (Wen et al., 2020), distribution correction estimation (DICE) (Yang et al., 2020), and fitted Q-evaluation (FQE) (Le et al., 2019), the effectiveness of VLBM is also compared against the state-of-the-art model-based OPE method leveraging the auto-regressive (AR) architecture (Zhang et al., 2020a). Specifically, for each task we train an ensemble of 10 AR models, for fair comparisons against VLBM which leverages the branching architecture; see Appendix A for details of the AR ensemble setup. Following the DOPE benchmark (Fu et al., 2020b), our evaluation metrics includes rank correlation, regret@1, and mean absolute error (MAE). VLBM and all baselines are trained using 3 different random seeds over each task, leading to the results reported below.

Ablation. Four ablation baselines are also considered, *i.e.*, VLM, VLM+RSA, VLM+RSA(MSE) and VLM+RSA Ensemble. Specifically, VLM refers to the model introduced in Sec. 2.2, trained toward maximizing only the ELBO, *i.e.*, (6). Note that, arguably, VLM could be seen as the generalization of directly applying latent-models proposed in existing RL policy optimization literature (Lee et al., 2020; Hafner et al., 2019; 2020a;b; Lu et al., 2022); details can be found in Sec. 4 below. The VLM+RSA ablation baseline follows the same model architecture as VLM, but is trained to optimize over both ELBO and recurrent state alignment (RSA) as introduced in (8), *i.e.*, branching is not used comparing to VLBM. The design of these two baselines can help analyze the effectiveness of the RSA

[5]From now on the dataset names are abbreviated by their initials, *e.g.*, Ant-M-R refers to Ant-Medium-Replay.

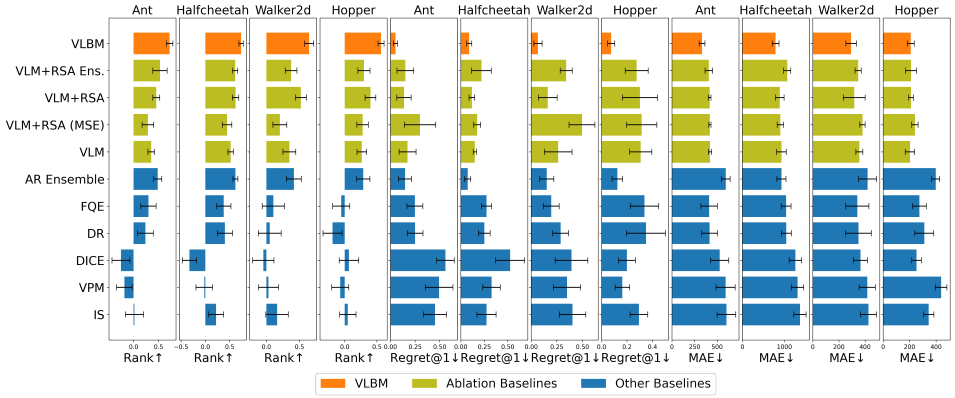

Figure 4: Mean rank correlation, regret@1 and MAE over all datasets, for each Mujoco environment.

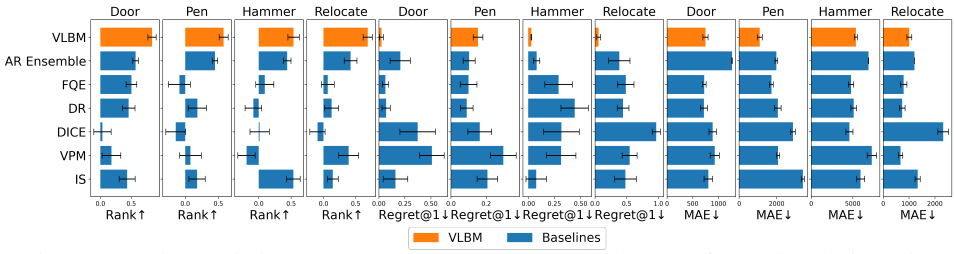

Figure 5: Mean rank correlation, regret@1 and MAE over all datasets, for each Adroit environment.

loss term and branching architecture introduced in Sec. 2.3 and 2.4. Moreover, VLM+RSA(MSE) uses mean squared error to replace the pairwise loss introduced in (7), and the VLM+RSA Ensemble applies classic ensembles by averaging over B VLM+RSA models end-to-end, instead of branching from decoder as in VLBM. These two ablation baselines can help justify the use of pairwise loss for RSA, and the benefit of using branching architecture over classic ensembles.

Results. Fig. 3 shows the mean overall performance attained by VLBM and baselines over all the 32 Gym-Mujoco and Adroit tasks. In general VLBM leads to significantly increased rank correlations and decreased regret@1's over existing methods, with MAEs maintained at the state-of-the-art level. Specifically, VLBM achieves state-of-the-art performance in 31, 29, and 15 (out of 32) tasks in terms of rank correlation, regret@1 and MAE, respectively. Performance for each task can be found in Tables 1- 6 at the end of Appendices. Note that results for IS, VPM, DICE, DR, and FQE are obtained directly from DOPE benchmark (Fu et al., 2020b), since the same experimental setup is considered. Fig. 4 and 5 visualize the mean performance for each Gym-Mujoco and Adroit environment respectively, over all the associated datasets. It can be also observed that the model-based and FQE baselines generally perform better than the other baselines, which is consistent with findings from DOPE.

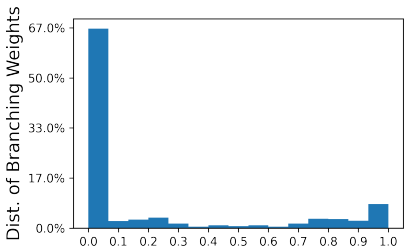

Figure 6: Distribution of all branching weights, w_b's, over all VLBMs trained on the 32 tasks.

The fact that VLM+RSA outperforming the VLM ablation baseline, as shown in Fig. 4, illustrates the need of the RSA loss term to smooth the flow of information between the encoder and decoder, in the latent space. Moreover, one can observe that VLM+RSA(MSE) sometimes performs worse than VLM, and significantly worse than VLM+RSA in general. Specifically, it has be found that, compared to VLM and VLM+RSA respectively, VLM+RSA(MSE) significantly worsen at least two metrics in 7 and 12 (out of 20) Gym-Mujoco tasks; detailed performance over these tasks can be found in Tables 1- 6 at the end of Appendices. Such a finding backs up the design choice of using pairwise loss for RSA instead of MSE, as MSE could be overly strong to regularize the LSTM recurrent states of the encoder and decoder, while pairwise loss only enforces structural similarities. Moreover, VLBM significantly improves rank correlations and regrets greatly compared to VLM+RSA, illustrating the importance of the branching architecture. In the paragraph below, we show empirically the benefits brought in by branching over classic ensembles.

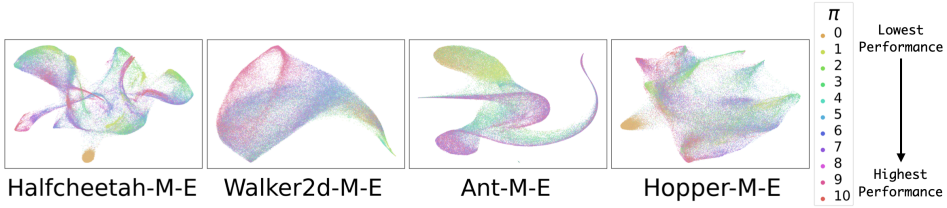

Figure 8: t-SNE visualization over the latent space, capturing encoded state-action visitations induced from all target policies. Each point is colored by the corresponding policy from which it is generated. Policies in the legend are sorted in the order of increasing performance.

Branching versus Classic Ensembles. Fig. 4 shows that the VLM+RSA Ensemble does not improve performance over the VLM+RSA in general, and even leads to worse overall rank correlations and regrets in Walker2d and Hopper environments. This supports the rationale provided in Sec. 2.4 that each decoder still samples from different latent space exclusively, and averaging over the output distributions may not help reduce the disturbance brought in by the modeling artifacts under the variational inference framework, *e.g.*, random weight initializations (Hanin & Rolnick, 2018; Rossi et al., 2019). In contrast, the VLBM leverages the branching architecture, allowing all the branches to sample from the same latent space formulated by the encoder. Empirically, we find that the branching weights, w_b's in (9), allows VLBM to kill branches that are not helpful toward reconstructing the trajectories accurately, to possibly overcome bad initializations etc. Over all the the 32 tasks we consider, most of VLBMs only keep 1-3 branches (out of 10), *i.e.*, $w_b < 10^{-5}$ for all other branches. The distribution of all w_b's, from VLBMs trained on the 32 tasks, are shown in Fig. 6; one can observe that most of the w_b's are close to zero, while the others generally fall in the range of $(0, 0.25]$ and $[0.75, 1)$.

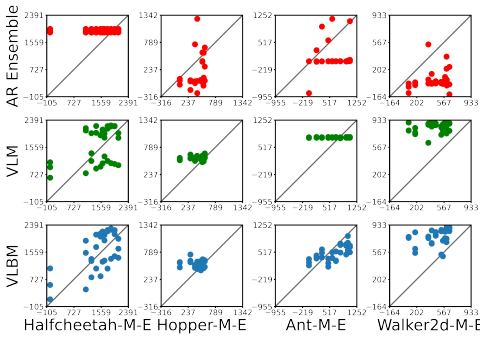

Figure 7: Correlation between the estimated (y-axis) and true returns (x-axis), across different model-based OPE methods and environments.

AR ensembles also lead to compelling rank correlations and regrets, but attains much smaller margins in MAEs over other baselines in general; see Fig. 3. From Fig. 7, one can observe that it tends to significantly under-estimate most of the high-performing policies. Scatter plots for the other tasks can be found in Appendix A, which also show this trend. The reason could be that its model architecture and training objectives are designed to directly learn the transitions of the MDP; thus, may produce biased predictions when the target policies lead to visitation of the states that are not substantially presented in training data, since such data are obtained using behavioral policies that are sub-optimal. In contrast, the VLBM can leverage RSA and branching against such situations, thus outperforming AR ensembles in most of the OPE tasks in terms of all metrics we considered. Interestingly, Fig. 7 also shows that latent models could sometimes over-estimate the returns. For example, in Hopper-M-E and Walker2d-M-E, VLM tends to over-estimate most policies. The VLBM performs consistently well in Hopper-M-E, but is mildly affected by such an effect in Walker2d-M-E, though over fewer policies and smaller margins. It has been found that variational inference may fall short in approximating true distributions that are asymmetric, and produce biased estimations (Yao et al., 2018). So the hypothesis would be that the dynamics used to define certain environments may lead to asymmetry in the true posterior $p(z_t|z_{t-1}, a_{t-1}, s_t)$, which could be hard to be captured by the latent modeling framework we consider. More comprehensive understanding of such behavior can be explored in future work. However, the VLBM still significantly outperforms VLM overall, and achieves top-performing rank correlations and regrets; such results illustrate the VLBM's improved robustness as a result of its architectural design and choices over training objectives.

t-SNE Visualization of the Latent Space. Fig. 8 illustrates t-SNE visualization of the latent space by rolling out trajectories using all target policies respectively, followed by feeding the state-action pairs into the encoder of VLBM which maps them into the latent space. It shows the encoded state-action pairs induced from policies with similar performance are in general swirled and clustered together, illustrating that VLBM can learn expressive and disentangled representations of its inputs.

4 Related Work

Latent Modeling in RL. Though variational inference has rarely been explored to facilitate model-based OPE methods so far, there exist several latent models designed for RL policy optimization that are related to our work, such as SLAC (Lee et al., 2020), SOLAR (Zhang et al., 2019), LatCo (Rybkin et al., 2021), PlaNet (Hafner et al., 2019), Dreamer (Hafner et al., 2020a;b). Below we discuss the connections and distinctions between VLBM and the latent models leveraged by them, with a detailed overview of these methods provided in Appendix G. Specifically, SLAC and SOLAR learn latent representations of the dynamics jointly with optimization of the target policies, using the latent information to improve sample efficiency. Similarly, LatCo performs trajectory optimization over the latent space to allow for temporarily bypassing dynamic constraints. As a result, latent models used in such methods are not designed toward rolling out trajectories independently, as opposed to the use of VLBM in this paper. PlaNet and Dreamer train the recurrent state space model (RSSM) using a *growing* experience dataset collected by the target policy that is being concurrently updated (with exploration noise added), which requires *online* data collection. In contrast, under the OPE setup, VLBM is trained over a *fixed* set of offline trajectories collected over unknown behavioral policies. Moreover, note that the VLM baseline is somewhat reminiscent of the RSSM and similar ones as in Lee et al. (2020); Lu et al. (2022), however, experiments above show that directly using VLM for OPE could lead to subpar performance. On the other hand, though MOPO (Yu et al., 2020), LOMPO (Rafailov et al., 2021) and COMBO (Yu et al., 2021) can learn from offline data, they focus on quantifying the uncertainty of model's predictions toward next states and rewards, followed by incorporating them into policy optimization objectives to penalize for visiting regions where transitions are not fully captured; thus, such works are also orthogonal to the use case of OPE.

OPE. Classic OPE methods adopt IS to estimate expectations over the unknown visitation distribution over the target policy, resulting in weighted IS, step-wise IS and weighted step-wise IS (Precup, 2000). IS can lead to estimations with low (or zero) bias, but with high variance (Kostrikov & Nachum, 2020; Jiang & Li, 2016), which sparks a long line of research to address this challenge. DR methods propose to reduce variance by coupling IS with a value function approximator (Jiang & Li, 2016; Thomas & Brunskill, 2016; Farajtabar et al., 2018). However, the introduction of such approximations may increase bias, so the method proposed in Tang et al. (2019) attempts to balance the scale of bias and variance for DR. Unlike IS and DR methods that require the behavioral policies to be fully known, DICE family of estimators (Zhang et al., 2020c;b; Yang et al., 2021; 2020; Nachum et al., 2019; Dai et al., 2020) and VPM (Wen et al., 2020) can be behavioral-agnostic; they directly capture marginalized IS weights as the ratio between the propensity of the target policy to visit particular state-action pairs, relative to their likelihood of appearing in the logged data. There also exist FQE methods which extrapolate policy returns from approximated Q-functions (Hao et al., 2021; Le et al., 2019; Kostrikov & Nachum, 2020). Existing model-based OPE methods are designed to directly fit MDP transitions using feed-forward (Fu et al., 2020b) or auto-regressive (Zhang et al., 2020a) models, and has shown promising results over model-free methods as reported in a recent benchmark (Fu et al., 2020b). However, such model-based approaches could be sensitive to the initialization of weights (Hanin & Rolnick, 2018; Rossi et al., 2019) and produce biased predictions, due to the limited coverage over state and action space provided by offline trajectories (Fu et al., 2020b). Instead, VLBM mitigates such effects by capturing the dynamics over the latent space, such that states and rewards are evolved from a compact feature space over time. Moreover, RSA and the branching can lead to increased expressiveness and robustness, such that future states and rewards are predicted accurately. There also exist OPE methods proposed toward specific applications (Chen et al., 2022; Saito et al., 2021; Gao et al., 2023; 2022b).

5 Conclusion and Future Work

We have developed the VLBM which can accurately capture the dynamics underlying environments from offline training data that provide limited coverage of the state and action space; this is achieved by using the RSA term to smooth out the information flow from the encoders to decoders in the latent space, as well as the branching architecture which improve VLBM's robustness against random initializations. We have followed evaluation guidelines provided by the DOPE benchmark, and experimental results have shown that the VLBM generally outperforms the state-of-the-art model-based OPE method using AR architectures, as well as other model-free methods. VLBM can also facilitate off-policy optimizations, which can be explored in future works. Specifically, VLBM can serve as a synthetic environment on which optimal controllers (*e.g.*, linear–quadratic regulator) can be deployed. On the other hand, similar to Dreamer and SLAC, policies can be updated jointly with training of VLBM, but without the need of online interactions with the environment during training.

ACKNOWLEDGMENTS

This work is sponsored in part by the AFOSR under award number FA9550-19-1-0169, and by the NSF CNS-1652544, CNS-1837499, DUE-1726550, IIS-1651909 and DUE-2013502 awards, as well as the National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant CNS-2112562.

REFERENCES

Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

Minmin Chen, Can Xu, Vince Gatto, Devanshu Jain, Aviral Kumar, and Ed Chi. Off-policy actor-critic for recommender systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 338–349, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pp. 1597–1607. PMLR, 2020.

Bo Dai, Ofir Nachum, Yinlam Chow, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. Coindice: Off-policy confidence interval estimation. *Advances in Neural Information Processing Systems*, 33:9398–9411, 2020.

Hazel Doughty, Dima Damen, and Walterio Mayol-Cuevas. Who's better? who's best? pairwise deep ranking for skill determination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6057–6066, 2018.

Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pp. 1447–1456. PMLR, 2018.

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

Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, et al. Benchmarks for deep off-policy evaluation. In *International Conference on Learning Representations*, 2020b.

Ge Gao, Qitong Gao, Xi Yang, Miroslav Pajic, and Min Chi. A reinforcement learning-informed pattern mining framework for multivariate time series classification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022a.

Ge Gao, Song Ju, Markel Sanz Ausin, and Min Chi. Hope: Human-centric off-policy evaluation for e-learning and healthcare. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

Qitong Gao, Davood Hajinezhad, Yan Zhang, Yiannis Kantaros, and Michael M Zavlanos. Reduced variance deep reinforcement learning with temporal logic specifications. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 237–248, 2019.

Qitong Gao, Michael Naumann, Ilija Jovanov, Vuk Lesi, Karthik Kamaravelu, Warren M Grill, and Miroslav Pajic. Model-based design of closed loop deep brain stimulation controller using reinforcement learning. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 108–118. IEEE, 2020a.

Qitong Gao, Miroslav Pajic, and Michael M Zavlanos. Deep imitative reinforcement learning for temporal logic robot motion planning with noisy semantic observations. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8490–8496. IEEE, 2020b.

Qitong Gao, Stephen L Schmidt, Karthik Kamaravelu, Dennis A Turner, Warren M Grill, and Miroslav Pajic. Offline policy evaluation for learning-based deep brain stimulation controllers. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 80–91. IEEE, 2022b.

Qitong Gao, Dong Wang, Joshua D Amason, Siyang Yuan, Chenyang Tao, Ricardo Henao, Majda Hadziahmetovic, Lawrence Carin, and Miroslav Pajic. Gradient importance learning for incomplete observations. *International Conference on Learning Representations*, 2022c.

Stephen Gould, Tianshi Gao, and Daphne Koller. Region-based segmentation and object detection. *Advances in Neural Information Processing Systems*, 22, 2009.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

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

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2020b.

Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *Advances in Neural Information Processing Systems*, 31, 2018.

Botao Hao, Xiang Ji, Yaqi Duan, Hao Lu, Csaba Szepesvari, and Mengdi Wang. Bootstrapping fitted q-evaluation for off-policy inference. In *International Conference on Machine Learning*, pp. 4074–4084. PMLR, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 652–661. PMLR, 2016.

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

Ilya Kostrikov and Ofir Nachum. Statistical bootstrapping for uncertainty estimation in off-policy evaluation. *arXiv preprint arXiv:2007.13609*, 2020.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.

Alex Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33, 2020.

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM International Conference on Web Search and Data Mining*, pp. 297–306, 2011.

Cong Lu, Philip J Ball, Tim GJ Rudner, Jack Parker-Holder, Michael A Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *arXiv preprint arXiv:2206.04779*, 2022.

Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, volume 1077, 2014.

Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2243–2251, 2018.

Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *Advances in Neural Information Processing Systems*, 32:2318–2328, 2019.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pp. 1154–1168. PMLR, 2021.

Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6917–6925, 2018.

Simone Rossi, Pietro Michiardi, and Maurizio Filippone. Good initializations of variational bayes for deep models. In *International Conference on Machine Learning*, pp. 5487–5497. PMLR, 2019.

Oleh Rybkin, Chuning Zhu, Anusha Nagabandi, Kostas Daniilidis, Igor Mordatch, and Sergey Levine. Model-based reinforcement learning via latent-space collocation. In *International Conference on Machine Learning*, pp. 9190–9201. PMLR, 2021.

Yuta Saito, Takuma Udagawa, Haruka Kiyohara, Kazuki Mogi, Yusuke Narita, and Kei Tateno. Evaluating the robustness of off-policy evaluation. In *Fifteenth ACM Conference on Recommender Systems*, pp. 114–123, 2021.

Yassir Saquil, Da Chen, Yuan He, Chuan Li, and Yong-Liang Yang. Multiple pairwise ranking networks for personalized video summarization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1718–1727, 2021.

Avi Segal, Kobi Gal, Ece Kamar, Eric Horvitz, and Grant Miller. Optimizing interventions via offline policy evaluation: Studies in citizen science. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Shengpu Tang and Jenna Wiens. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Machine Learning for Healthcare Conference*, pp. 2–35. PMLR, 2021.

Ziyang Tang, Yihao Feng, Lihong Li, Dengyong Zhou, and Qiang Liu. Doubly robust bias reduction in infinite horizon off-policy estimation. In *International Conference on Learning Representations*, 2019.

Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148. PMLR, 2016.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3024–3033, 2021.

Junfeng Wen, Bo Dai, Lihong Li, and Dale Schuurmans. Batch stationary distribution estimation. In *International Conference on Machine Learning*, pp. 10203–10213. PMLR, 2020.

Mengjiao Yang, Ofir Nachum, Bo Dai, Lihong Li, and Dale Schuurmans. Off-policy evaluation via the regularized lagrangian. *Advances in Neural Information Processing Systems*, 33:6551–6561, 2020.

Mengjiao Yang, Bo Dai, Ofir Nachum, George Tucker, and Dale Schuurmans. Offline policy selection under uncertainty. In *Deep RL Workshop NeurIPS 2021*, 2021.

Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Yes, but did it work?: Evaluating variational inference. In *International Conference on Machine Learning*, pp. 5581–5590. PMLR, 2018.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

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*, pp. 7444–7453. PMLR, 2019.

Michael R Zhang, Thomas Paine, Ofir Nachum, Cosmin Paduraru, George Tucker, Mohammad Norouzi, et al. Autoregressive dynamics models for offline policy evaluation and optimization. In *International Conference on Learning Representations*, 2020a.

Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. In *International Conference on Learning Representations*, 2020b.

Shangtong Zhang, Bo Liu, and Shimon Whiteson. Gradientdice: Rethinking generalized offline estimation of stationary values. In *International Conference on Machine Learning*, pp. 11194–11203. PMLR, 2020c.

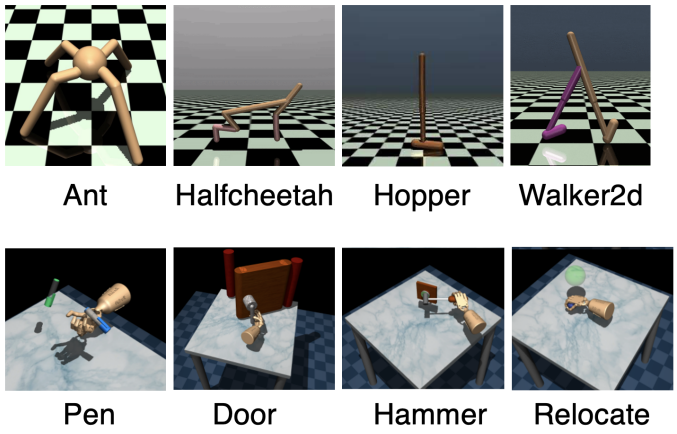

Figure 9: The Gym-Mujoco and Adroit environments considered by the D4RL branch of DOPE.

A Additional Experimental Details and Results

Additional Results and Discussions. Rank correlations, regret@1 and MAEs for all 32 tasks are documented in Tables 1- 6 below.[6] The mean and standard deviation (in subscripts) over 3 random seeds are reported. Note that in each column, performance of multiple methods may be highlighted in bold, meaning they all achieve the best performance and do not significantly outperform each other. The fact that VLBM outperforms the ablation baselines in most cases suggests that the RSA loss term and branching architecture can effectively increase model expressiveness, and allow to learn the dynamics underlying the MDP more accurately and robustly from *offline* data that provide limited exploration coverage. Yet, smaller margins are attained between the VLBM and VLM+RSA in Hopper-M-E and Hopper-M. It is likely because Hopper has relatively lower dimensional state space compared to the other three environments, from which the underlying dynamics can be sufficiently captured by the VLM+RSA. Fig. 10 and 11 shows the correlation between estimated (y-axis) and true returns (x-axis) for all the OPE tasks we consider. It can be found that for Halfcheetah-R, -M-R, -M, most of the model-based methods cannot significantly distinguish the returns across target policies. The cause could be that the offline trajectories provided for this task are relatively more challenging, compared to the other OPE tasks. Such an effect appears to affect IS, VPM, DICE, DR and FQE at larger scale. It can be observed from the scatter plots reported in the DOPE benchmark (Fu et al., 2020b) that these methods could hardly tell the scale of returns across different target policies; as the dots almost form a horizontal line in each plot. However, the estimated returns from VLBM and IS still preserve the rank, which leads to high rank correlations and low regrets.

Implementation Details and Hyper-parameter. The model-based methods are evaluated by directly interacting with each target policy for 50 episodes, and the mean of discounted total returns ($\gamma = 0.995$) over all episodes is used as estimated performance for the policy. We choose the neural network architectures as follows. For the components involving LSTMs, which include $q_\psi(z_t|z_{t-1}, a_{t-1}, s_t)$ and $p_\phi(z_t|z_{t-1}, a_{t-1})$, their architecture include one LSTM layer with 64 nodes, followed by a dense layer with 64 nodes. All other components do not have LSTM layers involved, so they are constituted by a neural network with 2 dense layers, with 128 and 64 nodes respectively. The output layers that determine the mean and diagonal covariance of diagonal Gaussian distributions use linear and softplus activations, respectively. The ones that determine the mean of Bernoulli distributions (*e.g.*, for capturing early termination of episodes) are configured to use sigmoid activations. VLBM and the two ablation baselines, VLM and VLM+RSA, are trained using offline trajectories provided by DOPE, with max_iter in Alg. 1 set to 1,000 and minibatch size set to 64. Adam optimizer is used to perform gradient descent. To determine the learning rate, we perform grid search among $\{0.003, 0.001, 0.0007, 0.0005, 0.0003, 0.0001, 0.00005\}$. Exponential decay is applied to the learning rate, which decays the learning rate by 0.997 every iteration. To train VLBM, we set the constants from equation 10 following $C_1 = C_2$, and perform grid search among

[6] Some VPM entries are absent since they were not reported in Fu et al. (2020b), nor the code is open-sourced.

$\{5, 1, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0001\}$. To train VLM+RSA, the constant C from equation 8 is determined by grid search among the same set of parameters above. L2-regularization with decay of 0.001 and batch normalization are applied to all hidden layers. Consider that some of the environments (*e.g.*, Ant, Hopper, Walker2d, Pen) may terminate an episode, before timeout, if the state meets specific conditions; details for VLBM to capture such early termination behavior is introduced in Appendix D.

The DOPE Benchmark. The deep OPE (DOPE) benchmark (Fu et al., 2020b) provides standardized training and evaluation procedure for OPE works to follow, which facilitates fair and comprehensive comparisons among various OPE methods. Specifically, it utilizes existing environments and training trajectories provided by D4RL[7] and RLUnplugged[8], which are two benchmark suites for offline RL training, and additionally provide target policies for OPE methods to evaluate. In the D4RL branch, the training trajectories are originally collected from various sources including random exploration, human teleoperation, and RL-trained policies with limited exploration; thus, can provide varied levels of coverage over the state-action space. Moreover, the target policies are trained using online RL algorithms, which can in general lead to different state-action visitations than in the training trajectories. We leverage the D4RL branch as our test base, since the OPE tasks it provides are considered challenging, *i.e.*, the limited coverage introduced by training data, as well as the discrepancy between the behavioral and target policies. Graphical illustrations of the Gym-Mujoco and Adroit environments considered are shown in Fig. 9. Details on the environments and datasets used are shown in Tables 7 and 8, from the perspectives of state and action dimensions, if episodes can be terminated before timeout, if controls are performed over continuous space, and the size of the offline trajectories used for training. In contrast, in the RLUnplugged branch, the training trajectories are always collected using online RL training, which can result in adequate coverage over the state-action space. The target policies are trained by applying offline RL over the training trajectories, so that behavioral and target policies can lead to similar state-action visitation distributions. As discussed in DOPE (Fu et al., 2020b), such tasks are suitable for studies where ideal data are needed, such as complexity comparisons.

Evaluation Metrics. Following from (Fu et al., 2020b), we consider rank correlation, regret@1 and mean absolute error (MAE) as the evaluation metrics. Specifically, rank correlation measures the strength and direction of monotonic association between the rank of OPE-estimated returns and true returns over all target policies. It is is captured by Spearsman's correlation coefficient between the ordinal rankings between estimated and true returns. Regret@1 is captured by the difference between the return of the policy corresponding to the highest return as estimated by OPE and the return of the policy that actually produces the highest true return. In other words, regret@1 evaluates how worse the policy resulting in the highest OPE-estimated return would perform than the actual best policy. The two metrics above evaluate how useful OPE would be to facilitate important applications such as policy selection. Finally, we also consider MAE which is commonly used in estimation/regression tasks. Mathematical definitions of these metrics can be found in (Fu et al., 2020b).

Implementation of AR Ensembles. For fair comparisons with VLBM, in experiments we train an ensemble of the state-of-the-art model-based OPE method, auto-regressive (AR) models (Zhang et al., 2020a), as one of the baselines. Specifically, we train an ensemble of 10 AR models to learn $p(s_{t+1}, r_t|s_t, a_t)$ following the auto-regressive manner, with each individual model following the design introduced in (Zhang et al., 2020a), *i.e.*,

$$s_{t+1}^{(j)} \sim p(s_{t+1}^{(j)}|s_t, a_t, s_{t+1}^{(1)}, \ldots, s_{t+1}^{(j-1)}), \tag{11}$$

with $s_{t+1}^{(j)}$ representing the element located at the j-th dimension of the state variable, and D the dimension of state space. The reward is treated as an additional dimension of the states, *i.e.*, $r_t \sim p(r_t|s_t, a_t, s_{t+1}^{(1)}, \ldots, s_{t+1}^{(D)})$. However, in the original literature (Zhang et al., 2020a) it does not introduce in details regarding which specific ensemble architecture is used (*e.g.*, overall averaging or weighted averaging). As a result, we choose the same weighted averaging procedure as used in VLBM branching, to sort out the influence of different ensemble architectures and facilitate fair comparisons. Specifically, a total of 10 AR models, parameterized by $\{\theta_1, \ldots, \theta_{10}\}$, along with 10

[7] https://github.com/rail-berkeley/d4rl

[8] https://github.com/deepmind/deepmind-research/tree/master/rl_unplugged

weight variables $\{w_1^\theta, \ldots, w_{10}^\theta | \sum_i w_i^\theta = 1\}$, are trained. Similar to weighted averaging architecture used in VLBM, *i.e.*, equation 9, the mean and variance of the prediction $s_{t+1}^{(j)}$, captured by normal distribution $\mathcal{N}(\mu, \sigma^2)$, follow

$$\mu = \sum_{i=1}^{10} w_i^\theta \cdot \mu_{\theta_i}(s_{t+1}^{(j)}), \quad \sigma^2 = \sum_{i=1}^{10} (w_i^\theta)^2 \cdot \sigma_{\theta_i}^2(s_{t+1}^{(j)}), \tag{12}$$

where $\mu_{\theta_i}(s_{t+1}^{(j)})$ and $\sigma_{\theta_i}^2(s_{t+1}^{(j)})$ are the mean and variance produced from each individual AR model in the ensemble.

Training Resources. Training of the proposed method, and baselines, are facilitated by Nvidia Quadro RTX 6000, NVIDIA RTX A5000, and NVIDIA TITAN XP GPUs.

License. The use of DOPE[9] and D4RL (Fu et al., 2020a) follow the Apache License 2.0.

[9] https://github.com/google-research/deep_ope

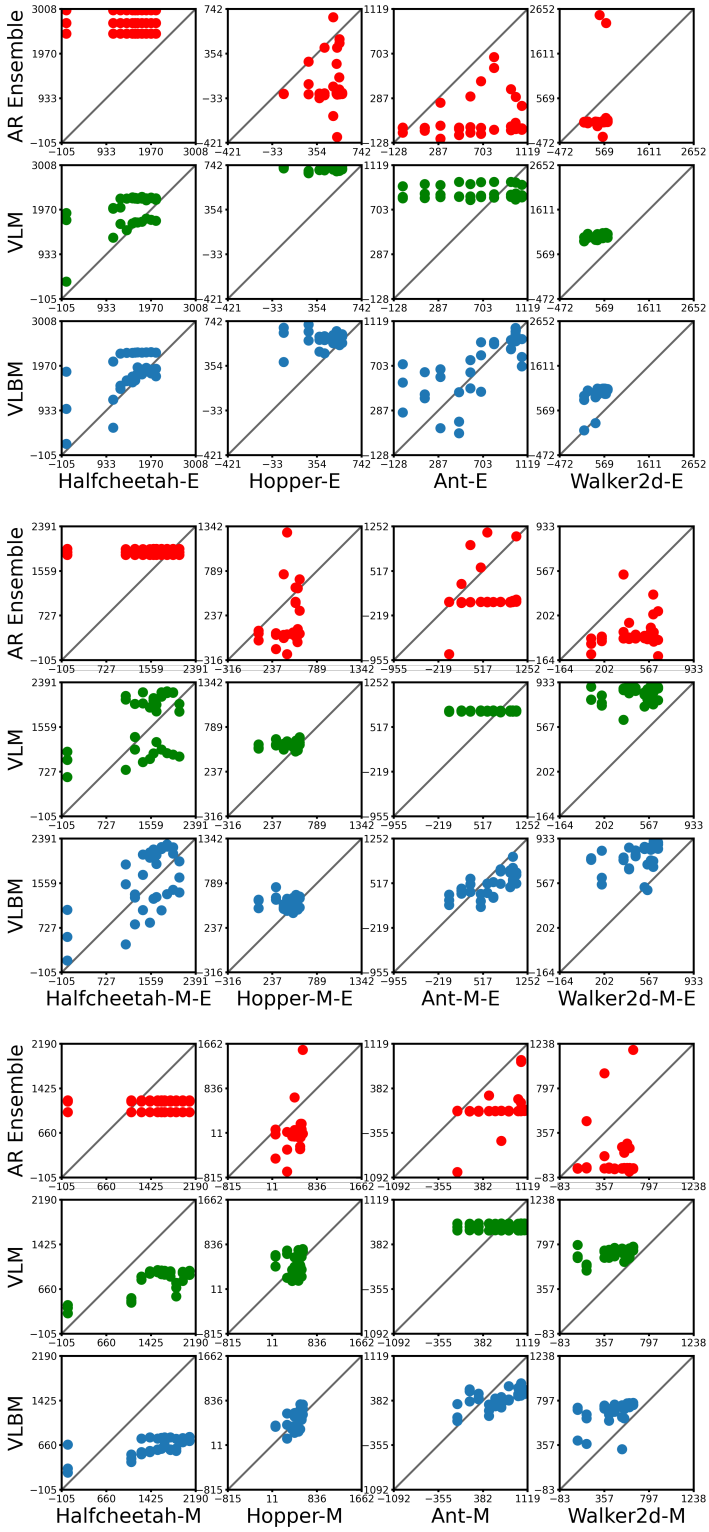

Figure 10: Scatter plots between OPE-estimated (y-axis) and true (x-axis) returns over all 20 Gym-Mujoco tasks that are considered. Part 1.

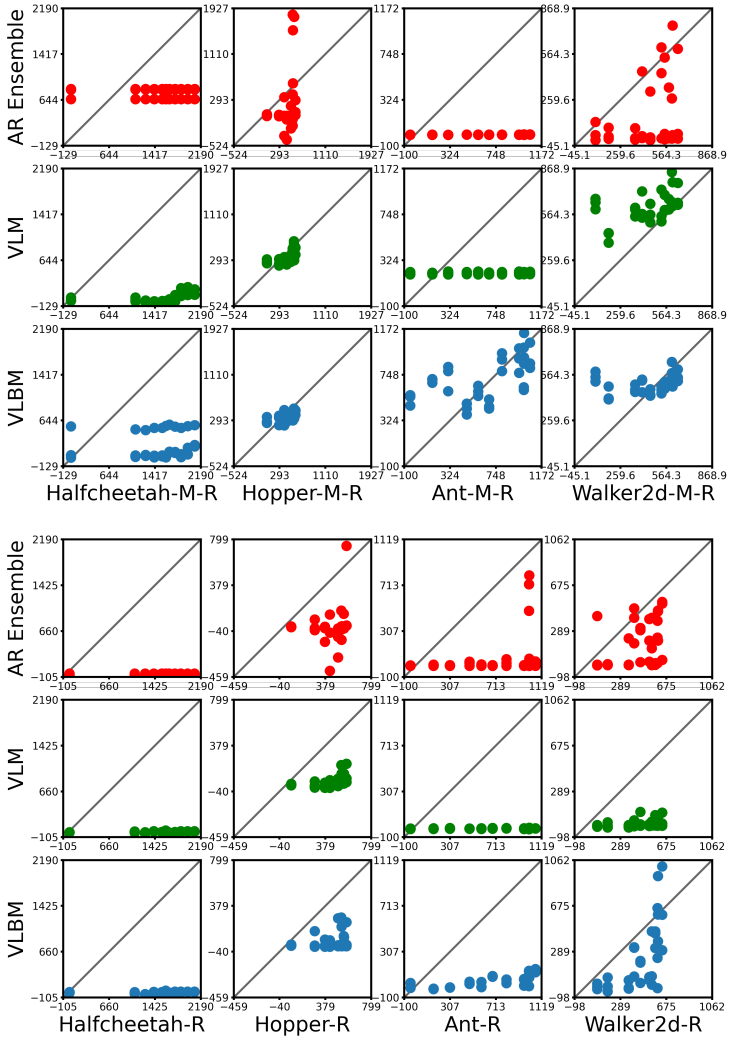

Figure 11: Scatter plots between OPE-estimated (y-axis) and true (x-axis) returns over all 20 Gym-Mujoco tasks that are considered. Part 2.

B MORE t-SNE VISUALIZATIONS

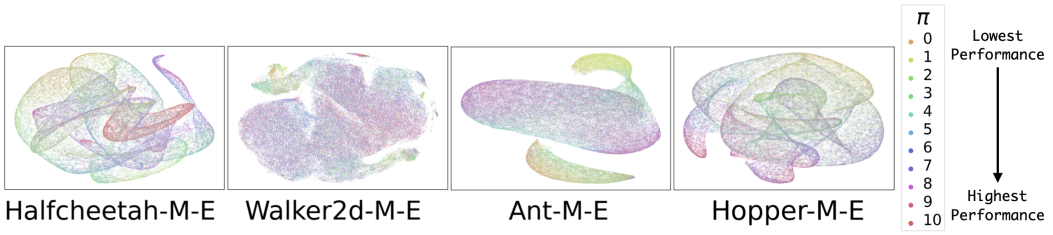

Figure 12: t-SNE visualization over the latent space captured by VLM, illustrating encoded state-action visitations induced from all target policies. Each point is colored by the corresponding policy from which it is generated. Policies in the legend are sorted in the order of increasing performance.

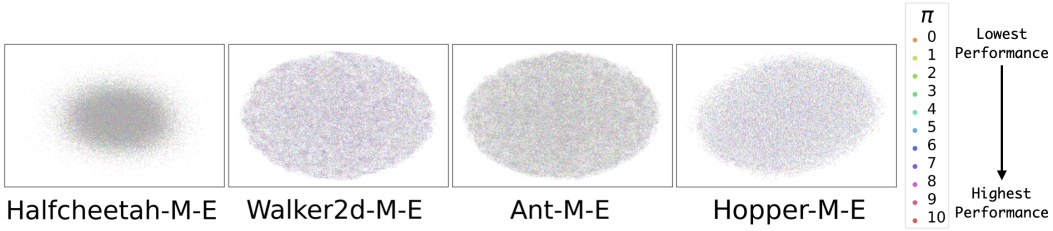

Figure 13: t-SNE visualization over the latent space captured by VLM+RSA(MSE), illustrating encoded state-action visitations induced from all target policies. Each point is colored by the corresponding policy from which it is generated. Policies in the legend are sorted in the order of increasing performance.

Figures 12 and 13 above visualize the latent space captured by two ablation baselines, VLM and VLM+RSA(MSE), respectively. It can be observed that comparing to the latent space captured by VLM are not disentangled well compared to VLBM (shown in Figure 8), as the state-action pairs induced by policies with different levels of performance are generally cluster together without explicit boundaries. Such a finding illustrated the importance of the use of RSA loss (7) empirically, as it can effectively regularize $p_\psi(z_t|z_{t-1}, a_{t-1}, s_t)$ and allows the encoder to map the MDP states to an expressive and compact latent space from which the decoder can reconstruct states and rewards accurately. Moreover, Figure 13 shows that the latent representations of the state-action pairs captured by VLM+RSA(MSE) distributed almost uniformly over the latent space. This justifies the rationale provided in Sec. 2.3 where MSE is too strong to regularize the hidden states of the encoder and decoder, and is also consistent with the results reported in Figure 3 that MSE+RSA(MSE) performs worse than VLM in general.

C ALGORITHMS FOR TRAINING AND EVALUATING VLBM

Algorithm 1 Train VLBM.

Input: Model weights $\psi, \phi_1, \ldots, \phi_B, w_1, \ldots, w_B$, offline trajectories ρ^β, and learning rate α.
Begin:
Initialize $\psi, \phi_1, \ldots, \phi_B, w_1, \ldots, w_B$
for $iter$ in $1 : max\_iter$ **do**
  Sample a trajectory $[(s_0, a_0, r_0, s_1), \ldots, (s_{T-1}, a_{T-1}, r_{T-1}, s_T)] \sim \rho^\beta$
  $z_0^\psi \sim q_\psi(z_0|s_0)$
  $z_0^{\phi_b} \sim p(z_0)$, for all $b \in [1, B]$
  Run forward pass of VLBM following (3), (5) and (9) for $t = 1 : T$, and collect all variables needed to evaluate $\mathcal{L}_{VLBM}$ as specified in (10).
  $\psi \leftarrow \psi + \alpha \nabla_\psi \mathcal{L}_{VLBM}$
  for b in $1 : B$ **do**
    $\phi_b \leftarrow \phi_b + \alpha \nabla_{\phi_b} \mathcal{L}_{VLBM}$
    $w_b \leftarrow w_b + \alpha \nabla_{w_b} \mathcal{L}_{VLBM}$
  end for
end for

Algorithm 2 Evaluate VLBM.

Input: Trained model weights $\psi, \phi_1, \ldots, \phi_B, w_1, \ldots, w_B$
Begin:
Initialize the list that stores the accumulated returns over all episodes $\mathcal{R} = []$
for epi in $1 : max\_epi$ **do**
  Initialize the variable $r = 0$ that tracks the accumulated return for the current episode
  Initialize latent states from the prior, *i.e.*, $z_0^{\phi_b} \sim p(z_0)$ for all $b \in [1, B]$
  Initialize LSTM hidden states $h_0^{\phi_b} = 0$ for all $b \in [1, B]$
  Sample $s_0^{\phi_b} \sim p_\phi(s_0|z_t^{\phi_b})$ for all $b \in [1, B]$ and generate initial MDP state s_0^ϕ following (9)
  for t in $1 : T$ **do**
    Determine the action following the target policy π, *i.e.*, $a_{t-1} \sim \pi(a_{t-1}|s_{t-1}^\phi)$
    for b in $1 : B$ **do**
      Update $h_t^{\phi_b}, \tilde{h}_t^{\phi_b}, z_t^{\phi_b}, s_t^{\phi_b}, r_{t-1}^{\phi_b}$ following (5).
    end for
    Generate the next state s_t^ϕ following (9), as well as the reward $r_{t-1}^\phi \sim p_\phi(r_{t-1}|z_t^{\phi_1}, \ldots, z_t^{\phi_B}) = \mathcal{N}\Big(\boldsymbol{\mu} = \sum_b w_b \cdot \mu(r_{t-1}^{\phi_b}), \boldsymbol{\Sigma}_{diag} = \sum_b w_b^2 \cdot \Sigma_{diag}(r_{t-1}^{\phi_b})\Big)$
    Update $r \leftarrow r + \gamma^{t-1} r_{t-1}^\phi$, with γ being the discounting factor
  end for
  Append r into $\mathcal{R}$
end for
Average over all elements in $\mathcal{R}$, which serves as the estimated return over π

D EARLY TERMINATION OF ENVIRONMENTS

Given that some Gym-Mujoco environments, including Ant, Hopper, Walker2d and Pen, may terminate an episode before reaching the maximum steps, if the state violates specific constraints. Below we introduce how VLM and VLBM can be enriched to capture such early termination behaviors.

VLM For VLM, we introduce an additional component $d_t^\phi \sim p_\phi(d_t|z_t^\phi)$ to the generative process equation 5, where d_t^ϕ is a Bernoulli variable determining if an episode should be terminated at its t-th step. Specifically, $p_\phi(d_t|z_t^\phi)$ follows Bernoulli distribution, with mean determined by an MLP with sigmoid activation applied to the output layer. As a result, the generative process now follows

$$\begin{aligned} h_t^\phi = f_\phi(h_{t-1}^\phi, z_{t-1}^\phi, a_{t-1}), \quad \tilde{h}_t^\phi = g_\phi(h_t^\phi), \quad z_t^\phi \sim p_\phi(\tilde{h}_t^\phi), \\ s_t^\phi \sim p_\phi(s_t|z_t^\phi), \quad r_{t-1}^\phi \sim p_\phi(r_{t-1}|z_t^\phi), \quad d_t^\phi \sim p_\phi(d_t|z_t^\phi), \quad a_t \sim \pi(a_t|s_t^\phi). \end{aligned} \tag{13}$$

Moreover, we add in a new term to VLM's training objective, in order to update the component introduced above during training, *i.e.*,

$$\mathcal{L}_{VLM}^{early\_term}(\psi, \phi) = \mathcal{L}_{VLM}(\psi, \phi) + \sum\nolimits_{t=0}^{T} \log p_\phi(d_t|z_t), \tag{14}$$

with $\mathcal{L}_{VLM}(\psi, \phi)$ being the original objective of VLM, as presented in equation 8.

VLBM For VLBM, the termination of an episode is determined following, *i.e.*,

$$d_t^\phi \sim p_\phi(d_t|z_t^{\phi_1}, \ldots, z_t^{\phi_B}) = Bernoulli(\boldsymbol{\mu} = \sum_b w_b \cdot \mu_d(d_t^{\phi_b})), \tag{15}$$

where $\mu_d(d_t^{\phi_b}) = \phi_{b,\mu_d}^{MLP}(z_t^{\phi_b})$ is the mean of $d_t^{\phi_b}$ produced from the b-th branch of the decoder, and ϕ_{b,μ_d}^{MLP} is the corresponding MLP that maps $z_t^{\phi_b}$ to $\mu_d(d_t^{\phi_b})$. To update the components involved in the procedure above, we introduce a new term to the VLBM's objective, *i.e.*,

$$\mathcal{L}_{VLBM}^{early\_term}(\psi, \phi_1, \ldots, \phi_B, w_1, \cdots, w_B) \tag{16}$$

$$= \mathcal{L}_{VLBM}(\psi, \phi_1, \ldots, \phi_B, w_1, \cdots, w_B) + \sum\nolimits_{t=0}^{T} \log p_\phi(d_t^\phi|z_t^{\phi_1}, \ldots, z_t^{\phi_B}), \tag{17}$$

with $\mathcal{L}_{VLBM}$ being the original objective of VLBM, as presented in equation 10.

E BOUND DERIVATION

We now derive the evidence lower bound (ELBO) for the joint log-likelihood distribution, *i.e.*,

$$\log p_\phi(s_{0:T}, r_{0:T-1}) \tag{18}$$

$$= \log \int_{z_{1:T} \in \mathcal{Z}} p_\phi(s_{0:T}, z_{1:T}, r_{0:T-1}) dz \tag{19}$$

$$= \log \int_{z_{1:T} \in \mathcal{Z}} \frac{p_\phi(s_{0:T}, z_{1:T}, r_{0:T-1})}{q_\psi(z_{0:T}|s_{0:T}, a_{0:T-1})} q_\psi(z_{0:T}|s_{0:T}, a_{0:T-1}) dz \tag{20}$$

$$\geq \mathbb{E}_{q_\psi}[\log p(z_0) + \log p_\phi(s_{0:T}, z_{1:T}, r_{0:T-1}|z_0) - \log q_\psi(z_{0:T}|s_{0:T}, a_{0:T-1})] \tag{21}$$

$$= \mathbb{E}_{q_\psi}\Big[\log p(z_0) + \log p_\phi(s_0|z_0) + \sum\nolimits_{t=1}^{T} \log p_\phi(s_t, z_t, r_{t-1}|z_{t-1}, a_{t-1}) - \log q_\psi(z_0|s_0) - \sum\nolimits_{t=1}^{T} \log q_\psi(z_t|z_{t-1}, a_{t-1}, s_t) \Big] \tag{22}$$

$$= \mathbb{E}_{q_\psi}\Big[\log p(z_0) - \log q_\psi(z_0|s_0) + \log p_\phi(s_0|z_0) + \sum\nolimits_{t=1}^{T} \log \big(p_\phi(s_t|z_t) p_\phi(r_{t-1}|z_t) p_\phi(z_t|z_{t-1}, a_{t-1}) \big) - \sum\nolimits_{t=1}^{T} \log q_\psi(z_t|z_{t-1}, a_{t-1}, s_t) \Big] \tag{23}$$

$$= \mathbb{E}_{q_\psi}\Big[\sum\nolimits_{t=0}^{T} \log p_\phi(s_t|z_t) + \sum\nolimits_{t=1}^{T} \log p_\phi(r_{t-1}|z_t) - KL\big(q_\psi(z_0|s_0)||p(z_0)\big) - \sum\nolimits_{t=1}^{T} KL\big(q_\psi(z_t|z_{t-1}, a_{t-1}, s_t)||p_\phi(z_t|z_{t-1}, a_{t-1})\big) \Big]. \tag{24}$$

Note that the transition from equation 20 to equation 21 follows Jensen's inequality.

F BASICS OF VARIATIONAL INFERENCE

Classic variational auto-encoders (VAEs) are designed to generate synthetic data that share similar characteristics than the ones used for training (Kingma & Welling, 2013). Specifically, VAEs learn an approximated posterior $q_\psi(z|x)$ and a generative model $p_\phi(x|z)$, over the prior $p(z)$, with x being the data and z the latent variable. It's true posterior $p_\phi(z|x)$ is intractable, ***i.e.***,

$$p_\phi(z|x) = \frac{p_\phi(x|z)p(z)}{p_\phi(x)}; \tag{25}$$

since the marginal likelihood in the denominator, $p_\phi(x) = \int_z p_\phi(x|z)p(z)dz$, requires integration over the unknown latent space. For the same reason, VAEs cannot be trained to directly maximize the marginal log-likelihood, $\max \log p_\phi(x)$. To resolve this, one could maximize a lower bound of $p_\phi(x)$, ***i.e.***,

$$\max_{\psi,\phi} -KL(q_\psi(z|x)||p(z)) + \mathbb{E}_{q_\psi}[\log p_\phi(x|z)], \tag{26}$$

which is the evidence lower bound (ELBO).

Reparameterization. During training, it is required to sample from $q_\psi(z|x)$ and $p_\phi(x|z)$ constantly. The reparameterization technique is introduced in (Kingma & Welling, 2013), to ensure that the gradients can flow through such sampling process during back-propagation. For example, if both distributions ($q_\psi(z|x)$ and $p_\phi(x|z)$) follow diagonal Gaussians, with mean and diagonal covariance determined by MLPs, ***i.e.***,

$$z \sim q_\psi(z|x) = \mathcal{N}\Big(\boldsymbol{\mu} = \psi_\mu^{MLP}(x), \quad \boldsymbol{\Sigma} = \psi_\Sigma^{MLP}(x)\Big), \tag{27}$$

$$x \sim p_\phi(x|z) = \mathcal{N}\Big(\boldsymbol{\mu} = \phi_\mu^{MLP}(z), \quad \boldsymbol{\Sigma} = \phi_\Sigma^{MLP}(z)\Big); \tag{28}$$

here, $\psi_\mu^{MLP}, \psi_\Sigma^{MLP}, \phi_\mu^{MLP}, \phi_\Sigma^{MLP}$ are the MLPs that generate the means and covariances. The sampling processes above can be captured by reparameterization, ***i.e.***,

$$z = \psi_\mu^{MLP}(x) + \psi_\Sigma^{MLP}(x) \cdot \boldsymbol{\epsilon}, \tag{29}$$

$$x = \phi_\mu^{MLP}(z) + \phi_\Sigma^{MLP}(z) \cdot \boldsymbol{\epsilon}, \tag{30}$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. Consequently, the gradients over ψ and ϕ can be calculated following the chain rule, and used for back-propagation during training. We direct readers to (Kingma & Welling, 2013) for a comprehensive review of reparameterization.

G ADDITIONAL RELATED WORKS

Overview of latent-model based RL methods. In SLAC, latent representations are used to improve the sample efficiency of model-free RL training algorithms, by jointly modeling and learning dynamics and controls over the latent space. Similarly, SOLAR improves data efficiency for multi-task RL by first learning high-level latent representations of the environment, which can be shared across different tasks. Then, local dynamics models are inferred from the abstraction, with controls solved by linear-quadratic regulators. PlaNet and Dreamer further improve the architecture and training objectives of latent models, allowing them to look ahead multiple steps and plan for longer horizon. There also exist LatCo which directly performs trajectory optimization over the latent space, allowing the agent to temporarily bypass dynamical constraints and quickly navigate to the high-reward regions in early training stage. To summarize, methods above leverage latent representations to gain sufficient exploration coverage and quickly navigate to high-reward regions, improving sample efficiency for policy optimization. Note that they mostly require *online* interactions with the environment to formulate a *growing* experience replay buffer for policy learning, which have different goals than OPE which requires learning from a *fixed* set of *offline* trajectories.

Rank Corr.	Ant -E	Ant -M-E	Ant -M	Ant -M-R	Ant -R
IS	$.14_{.41}$	$-.21_{.35}$	$-.17_{.32}$	$.07_{.39}$	$.26_{.34}$
VPM	$-.42_{.38}$	$-.28_{.28}$	$-.2_{.31}$	$-.26_{.29}$	$.24_{.31}$
DICE	$-.13_{.37}$	$-.33_{.4}$	$-.36_{.28}$	$-.24_{.39}$	$-.21_{.35}$
DR	$-.28_{.32}$	$.35_{.35}$	$\mathbf{.66_{.26}}$	$.45_{.32}$	$.01_{.33}$
FQE	$-.13_{.32}$	$.37_{.35}$	$\mathbf{.65_{.25}}$	$\mathbf{.57_{.28}}$	$.04_{.33}$
AR Ensemble	$.40_{.12}$	$.44_{.25}$	$.56_{.01}$	$\mathbf{.54_{.16}}$	$.48_{.17}$
VLM	$.28_{.14}$	$.39_{.16}$	$.37_{.03}$	$.37_{.19}$	$.36_{.07}$
VLM+RSA (MSE)	$.33_{.11}$	$.29_{.13}$	$.35_{.22}$	$.30_{.42}$	$.17_{.14}$
VLM+RSA	$.40_{.03}$	$.53_{.19}$	$.42_{.12}$	$\mathbf{.53_{.19}}$	$.40_{.11}$
VLM+RSA Ens.	$.62_{.16}$	$.76_{.02}$	$\mathbf{.65_{.07}}$	$\mathbf{.62_{.13}}$	$0._{.60}$
VLBM	$\mathbf{.79_{.01}}$	$\mathbf{.81_{.05}}$	$\mathbf{.65_{.06}}$	$\mathbf{.59_{.14}}$	$\mathbf{.78_{.24}}$
Rank Corr.	Halfcheetah -E	Halfcheetah -M-E	Halfcheetah -M	Halfcheetah -M-R	Halfcheetah -R
IS	$.01_{.35}$	$-.06_{.37}$	$\mathbf{.80_{.11}}$	$\mathbf{.59_{.26}}$	$-.24_{.36}$
VPM	$.18_{.35}$	$-.47_{.29}$	-	$-.07_{.36}$	$.27_{.36}$
DICE	$-.44_{.30}$	$-.08_{.35}$	$-.26_{.07}$	$-.15_{.41}$	$-.70_{.22}$
DR	$.77_{.17}$	$.62_{.27}$	$.32_{.32}$	$.32_{.37}$	$-.02_{.38}$
FQE	$.78_{.15}$	$.62_{.27}$	$.34_{.17}$	$.26_{.37}$	$-.11_{.41}$
AR Ensemble	$.65_{.11}$	$\mathbf{.65_{.07}}$	$.60_{.09}$	$\mathbf{.59_{.14}}$	$\mathbf{.60_{.06}}$
VLM	$.75_{.19}$	$.45_{.06}$	$.33_{.1}$	$\mathbf{.64_{.06}}$	$.43_{.09}$
VLM+RSA (MSE)	$.54_{.31}$	$.49_{.03}$	$.6_{.08}$	$.47_{.11}$	$.13_{.27}$
VLM+RSA	$.80_{.17}$	$.54_{.08}$	$.65_{.21}$	$\mathbf{.61_{.03}}$	$.51_{.08}$
VLM+RSA Ens.	$.71_{.14}$	$.66_{.08}$	$.64_{.02}$	$\mathbf{.60_{.05}}$	$.45_{.17}$
VLBM	$\mathbf{.88_{.01}}$	$\mathbf{.74_{.13}}$	$\mathbf{.81_{.13}}$	$\mathbf{.64_{.04}}$	$\mathbf{.60_{.06}}$
Rank Corr.	Walker2d -E	Walker2d -M-E	Walker2d -M	Walker2d -M-R	Walker2d -R
IS	$.22_{.37}$	$.24_{.33}$	$-.25_{.35}$	$\mathbf{.65_{.24}}$	$-.05_{.38}$
VPM	$.17_{.32}$	$.49_{.37}$	$.44_{.21}$	$-.52_{.25}$	$-.42_{.34}$
DICE	$-.37_{.27}$	$-.34_{.34}$	$.12_{.38}$	$.55_{.23}$	$-.19_{.36}$
DR	$.26_{.34}$	$.19_{.33}$	$.02_{.37}$	$-.37_{.39}$	$.16_{.29}$
FQE	$.35_{.33}$	$.25_{.32}$	$-.09_{.36}$	$-.19_{.36}$	$.21_{.31}$
AR Ensemble	$.54_{.11}$	$.25_{.33}$	$\mathbf{.55_{.14}}$	$.38_{.17}$	$.36_{.29}$
VLM	$.57_{.13}$	$.16_{.13}$	$.18_{.30}$	$.39_{.18}$	$.44_{.18}$
VLM+RSA (MSE)	$.27_{.28}$	$.20_{.25}$	$.09_{.18}$	$.10_{.11}$	$.36_{.19}$
VLM+RSA	$.56_{.11}$	$\mathbf{.57_{.11}}$	$.46_{.08}$	$.43_{.14}$	$.59_{.29}$
VLM+RSA Ens.	$.62_{.17}$	$\mathbf{.57_{.25}}$	$.43_{.20}$	$-.14_{.09}$	$.39_{.14}$
VLBM	$\mathbf{.70_{.13}}$	$\mathbf{.55_{.17}}$	$\mathbf{.66_{.15}}$	$\mathbf{.60_{.07}}$	$\mathbf{.72_{.14}}$
Rank Corr.	Hopper -E	Hopper -M-E	Hopper -M	Hopper -M-R	Hopper -R
IS	$\mathbf{.37_{.27}}$	$\mathbf{.35_{.26}}$	$-.55_{.26}$	$-.16_{.03}$	$.23_{.34}$
VPM	$.21_{.32}$	-	$.13_{.37}$	$-.16_{.03}$	$-.46_{.20}$
DICE	$-.08_{.32}$	$.08_{.14}$	$.19_{.33}$	$.27_{.28}$	$-.13_{.39}$
DR	$-.41_{.27}$	$-.08_{.30}$	$-.31_{.34}$	$.05_{.17}$	$-.19_{.36}$
FQE	$-.33_{.30}$	$.01_{.08}$	$-.29_{.33}$	$.45_{.13}$	$-.11_{.36}$
AR Ensemble	$.23_{.30}$	$.14_{.29}$	$.53_{.03}$	$.28_{.18}$	$.26_{.10}$
VLM	$-.05_{.22}$	$.22_{.11}$	$.34_{.08}$	$.46_{.21}$	$.36_{.03}$
VLM+RSA (MSE)	$-.18_{.24}$	$.05_{.09}$	$.51_{.20}$	$.43_{.18}$	$.58_{.14}$
VLM+RSA	$.15_{.28}$	$.26_{.10}$	$.51_{.11}$	$.53_{.06}$	$.55_{.19}$
VLM+RSA Ens.	$.09_{.21}$	$.13_{.12}$	$-.01_{.3}$	$.66_{.07}$	$.63_{.16}$
VLBM	$.28_{.16}$	$\mathbf{.32_{.10}}$	$\mathbf{.70_{.03}}$	$\mathbf{.75_{.07}}$	$\mathbf{.77_{.04}}$

Table 1: Rank correlation between estimated and ground-truth returns for all Gym-Mujoco tasks. Results are obtained by averaging over 3 random seeds used for training, with standard deviations shown in subscripts.

Rank Corr.	Door human	Door cloned	Door expert	Pen human	Pen cloned	Pen expert
IS	$-.12_{.35}$	$.66_{.22}$	$.76_{.17}$	$.28_{.28}$	$.71_{.08}$	$-.45_{.31}$
VPM	-	$-.29_{.36}$	$.65_{.23}$	-	-	$.08_{.33}$
DICE	$-.02_{.20}$	$.18_{.31}$	$-.06_{.32}$	$.17_{.33}$	$-.07_{.26}$	$-.53_{.30}$
DR	$.01_{.18}$	$.60_{.28}$	$.76_{.13}$	$-.36_{.29}$	$.39_{.25}$	$.52_{.28}$
FQE	$.07_{.09}$	$.55_{.27}$	$\mathbf{.89_{.09}}$	$-.31_{.21}$	$.06_{.42}$	$-.01_{.33}$
AR Ens.	$.58_{.06}$	$.52_{.13}$	$.61_{.07}$	$\mathbf{.33_{.07}}$	$.42_{.08}$	$\mathbf{.60_{.09}}$
VLBM	$\mathbf{.80_{.14}}$	$\mathbf{.78_{.18}}$	$\mathbf{.93_{.03}}$	$\mathbf{.34_{.17}}$	$\mathbf{.82_{.07}}$	$\mathbf{.58_{.15}}$
Rank Corr.	Hammer human	Hammer cloned	Hammer expert	Relocate human	Relocate cloned	Relocate expert
IS	$\mathbf{.39_{.07}}$	$\mathbf{.58_{.27}}$	$\mathbf{.64_{.24}}$	$-.23_{.07}$	$-.22_{.18}$	$\mathbf{.52_{.23}}$
VPM	-	$-.77_{.22}$	$.39_{.31}$	-	-	$.39_{.31}$
DICE	$.11_{.18}$	$.35_{.38}$	$-.42_{.31}$	$-.23_{.16}$	$.22_{.16}$	$-.27_{.34}$
DR	$-.04_{.25}$	$-.70_{.20}$	$.49_{.31}$	$\mathbf{.65_{.19}}$	$.10_{.16}$	$-.40_{.24}$
FQE	$.14_{.10}$	$-.15_{.33}$	$.29_{.34}$	$\mathbf{.62_{.11}}$	$.15_{.17}$	$-.57_{.28}$
AR Ens.	$\mathbf{.44_{.12}}$	$.40_{.20}$	$.53_{.11}$	$.42_{.23}$	$.30_{.10}$	$\mathbf{.54_{.23}}$
VLBM	$.34_{.14}$	$\mathbf{.58_{.18}}$	$\mathbf{.70_{.20}}$	$\mathbf{.68_{.17}}$	$\mathbf{.80_{.04}}$	$\mathbf{.58_{.17}}$

Table 2: Rank correlation between estimated and ground-truth returns for all Adroit tasks. Results are obtained by averaging over 3 random seeds used for training, with standard deviations shown in subscripts.

Regret@1	Ant -E	Ant -M-E	Ant -M	Ant -M-R	Ant -R
IS	$.47_{.32}$	$.46_{.18}$	$.61_{.18}$	$.16_{.23}$	$.56_{.22}$
VPM	$.88_{.22}$	$.32_{.24}$	$.4_{.21}$	$.72_{.43}$	$.15_{.24}$
DICE	$.62_{.15}$	$.60_{.16}$	$.43_{.1}$	$.64_{.13}$	$.50_{.29}$
DR	$.43_{.22}$	$.37_{.13}$	$.12_{.18}$	$.05_{.09}$	$.28_{.15}$
FQE	$.43_{.22}$	$.36_{.14}$	$.12_{.18}$	$.05_{.09}$	$.28_{.15}$
AR Ensemble	$.18_{.09}$	$.17_{.20}$	$\mathbf{.05_{0}}$	$.31_{.20}$	$\mathbf{.03_{.02}}$
VLM	$.38_{.24}$	$\mathbf{.07_{.02}}$	$.20_{.25}$	$\mathbf{.08_{.02}}$	$.14_{.16}$
VLM+RSA (MSE)	$\mathbf{.05_{0.}}$	$.26_{.21}$	$.28_{.4}$	$.48_{.33}$	$.43_{.44}$
VLM+RSA	$.18_{.09}$	$.13_{.12}$	$.14_{.16}$	$\mathbf{.17_{.24}}$	$.07_{.02}$
VLM+RSA Ens.	$.13_{.08}$	$\mathbf{.05_{0.}}$	$\mathbf{.03_{.02}}$	$\mathbf{.03_{.02}}$	$.52_{.37}$
VLBM	$\mathbf{.05_{0.}}$	$\mathbf{.05_{0.}}$	$\mathbf{.05_{0.}}$	$\mathbf{.11_{.09}}$	$\mathbf{0_{.0.}}$
Regret@1	Halfcheetah -E	Halfcheetah -M-E	Halfcheetah -M	Halfcheetah -M-R	Halfcheetah -R
IS	$.15_{.08}$	$.73_{.42}$	$\mathbf{.05_{.05}}$	$.13_{.10}$	$.31_{.11}$
VPM	$.14_{.09}$	$.80_{.34}$	$.33_{.19}$	$.25_{.09}$	$.12_{.07}$
DICE	$.32_{.40}$	$.38_{.37}$	$.82_{.29}$	$.30_{.07}$	$.81_{.30}$
DR	$.11_{.08}$	$.14_{.07}$	$.37_{.15}$	$.33_{.18}$	$.31_{.10}$
FQE	$.12_{.07}$	$.14_{.07}$	$.38_{.13}$	$.36_{.16}$	$.37_{.08}$
AR Ensemble	$\mathbf{.02_{.03}}$	$\mathbf{.11_{.07}}$	$.13_{.10}$	$\mathbf{.07_{.05}}$	$\mathbf{.04_{.05}}$
VLM	$.11_{.04}$	$.12_{.06}$	$.25_{.01}$	$\mathbf{.04_{.03}}$	$.23_{0.}$
VLM+RSA (MSE)	$.09_{.08}$	$.22_{.09}$	$.20_{.06}$	$\mathbf{.09_{.08}}$	$.27_{.05}$
VLM+RSA	$.08_{.02}$	$.17_{.05}$	$.09_{.12}$	$\mathbf{.02_{.03}}$	$.23_{0.}$
VLM+RSA Ens.	$.13_{.05}$	$.19_{.13}$	$.07_{.09}$	$\mathbf{.02_{.03}}$	$.69_{.44}$
VLBM	$.14_{.04}$	$\mathbf{.09_{.02}}$	$\mathbf{0_{.0.}}$	$\mathbf{.07_{.09}}$	$.15_{.07}$
Regret@1	Walker2d -E	Walker2d -M-E	Walker2d -M	Walker2d -M-R	Walker2d -R
IS	$.43_{.26}$	$.13_{.07}$	$.70_{.39}$	$\mathbf{.02_{.05}}$	$.74_{.33}$
VPM	$.09_{.19}$	$.24_{.42}$	$.08_{.06}$	$.46_{.31}$	$.88_{.20}$
DICE	$.35_{.36}$	$.78_{.27}$	$.27_{.43}$	$.18_{.12}$	$.39_{.33}$
DR	$\mathbf{.06_{.07}}$	$.30_{.12}$	$.25_{.09}$	$.68_{.23}$	$.15_{.20}$
FQE	$\mathbf{.06_{.07}}$	$.22_{.14}$	$.31_{.10}$	$.24_{.20}$	$.15_{.21}$
AR Ensemble	$.13_{.11}$	$.17_{.19}$	$.16_{.15}$	$.14_{.16}$	$\mathbf{.16_{.02}}$
VLM	$.10_{.05}$	$.51_{.25}$	$.30_{.39}$	$.33_{.38}$	$\mathbf{.08_{.07}}$
VLM+RSA (MSE)	$.49_{.16}$	$.39_{.30}$	$.43_{.35}$	$.86_{0.}$	$.31_{.29}$
VLM+RSA	$.10_{.07}$	$.11_{.02}$	$.18_{.15}$	$.34_{.37}$	$\mathbf{.08_{.04}}$
VLM+RSA Ens.	$.11_{.04}$	$.14_{.16}$	$\mathbf{.02_{.02}}$	$.86_{0.}$	$.58_{.20}$
VLBM	$\mathbf{.05_{.04}}$	$\mathbf{.05_{.01}}$	$\mathbf{.03_{.04}}$	$.14_{.16}$	$\mathbf{.06_{.06}}$
Regret@1	Hopper -E	Hopper -M-E	Hopper -M	Hopper -M-R	Hopper -R
IS	$\mathbf{.06_{.03}}$	$\mathbf{.10_{.12}}$	$.38_{.28}$	$.88_{0.}$	$\mathbf{.05_{.05}}$
VPM	$.13_{.10}$	-	$\mathbf{.10_{.14}}$	-	$.26_{.10}$
DICE	$.20_{.08}$	$.16_{.08}$	$.18_{.19}$	$.16_{.13}$	$.30_{.15}$
DR	$.34_{.35}$	$.34_{.39}$	$.32_{.32}$	$.34_{.24}$	$.41_{.17}$
FQE	$.41_{.20}$	$.42_{.08}$	$.32_{.32}$	$.18_{.23}$	$.36_{.22}$
AR Ensemble	$\mathbf{.07_{.05}}$	$.23_{.11}$	$.14_{.09}$	$\mathbf{.06_{.02}}$	$.12_{.11}$
VLM	$.76_{.18}$	$.35_{.22}$	$.22_{.22}$	$.14_{.15}$	$\mathbf{.07_{.02}}$
VLM+RSA (MSE)	$.42_{.34}$	$.51_{0.}$	$.33_{.39}$	$.26_{.13}$	$\mathbf{.06_{.04}}$
VLM+RSA	$.62_{.38}$	$.18_{.23}$	$\mathbf{.13_{.12}}$	$.25_{.15}$	$.33_{.39}$
VLM+RSA Ens.	$.31_{.18}$	$.51_{0.}$	$.47_{.36}$	$\mathbf{.03_{.02}}$	$\mathbf{.06_{.04}}$
VLBM	$\mathbf{.10_{.03}}$	$\mathbf{.10_{.03}}$	$\mathbf{.11_{.11}}$	$\mathbf{.04_{0.}}$	$\mathbf{.03_{.04}}$

Table 3: Regret@1 for all Gym-Mujoco tasks. Results are obtained by averaging over 3 random seeds used for training, with standard deviations shown in subscripts.

Regret@1	Door human	Door cloned	Door expert	Pen human	Pen cloned	Pen expert
IS	$.45_{.40}$	$\mathbf{.02_{.07}}$	$\mathbf{.01_{.04}}$	$.17_{.15}$	$.14_{.09}$	$.31_{.10}$
VPM	$.69_{.24}$	$.81_{.33}$	$\mathbf{.03_{.03}}$	$.28_{.12}$	$.36_{.18}$	$.25_{.13}$
DICE	$.10_{.27}$	$.65_{.45}$	$.37_{.27}$	$\mathbf{.04_{.09}}$	$.12_{.08}$	$.33_{.20}$
DR	$\mathbf{.05_{.09}}$	$.11_{.08}$	$.05_{.07}$	$\mathbf{.09_{.08}}$	$.13_{.06}$	$\mathbf{.05_{.07}}$
FQE	$\mathbf{.05_{.08}}$	$.11_{.06}$	$\mathbf{.03_{.03}}$	$\mathbf{.07_{.05}}$	$.12_{.07}$	$\mathbf{.11_{.14}}$
AR Ens.	$\mathbf{.08_{.10}}$	$.44_{.31}$	$.10_{.09}$	$\mathbf{.09_{.08}}$	$.14_{.05}$	$\mathbf{.08_{.07}}$
VLBM	$\mathbf{.03_{.04}}$	$\mathbf{.03_{.04}}$	$\mathbf{.02_{.03}}$	$.29_{.07}$	$\mathbf{.08_{.06}}$	$\mathbf{.09_{.02}}$
Regret@1	Hammer human	Hammer cloned	Hammer expert	Relocate human	Relocate cloned	Relocate expert
IS	$.19_{.30}$	$.03_{.15}$	$\mathbf{.01_{.04}}$	$.63_{.41}$	$.63_{.41}$	$.18_{.14}$
VPM	$.18_{.29}$	$.72_{.39}$	$.04_{.07}$	$.77_{.18}$	$.11_{.29}$	$.76_{.23}$
DICE	$\mathbf{.04_{.08}}$	$.67_{.48}$	$.24_{.34}$	$.97_{.11}$	$.96_{.18}$	$.97_{.07}$
DR	$.46_{.23}$	$.78_{.38}$	$.09_{.09}$	$.17_{.15}$	$.18_{.27}$	$.98_{.08}$
FQE	$.46_{.23}$	$.36_{.39}$	$.05_{.04}$	$.17_{.14}$	$.29_{.42}$	$1.00_{.06}$
AR Ens.	$\mathbf{.08_{.06}}$	$.05_{.05}$	$\mathbf{0_{.0}}$.	$.26_{.33}$	$.63_{.35}$	$.26_{.33}$
VLBM	$\mathbf{.08_{0.}}$	$\mathbf{0_{.0}}$.	$\mathbf{.01_{.01}}$	$\mathbf{.08_{.08}}$	$\mathbf{.02_{.02}}$	$\mathbf{.07_{.07}}$

Table 4: Regret@1 for all Adroit tasks. Results are obtained by averaging over 3 random seeds used for training, with standard deviations shown in subscripts.

MAE	Ant -E	Ant -M-E	Ant -M	Ant -M-R	Ant -R
IS	605_{104}	604_{102}	594_{104}	603_{101}	606_{103}
VPM	607_{108}	604_{106}	570_{109}	612_{105}	570_{99}
DICE	558_{108}	471_{100}	495_{90}	583_{110}	530_{92}
DR	584_{114}	326_{66}	$\mathbf{345_{66}}$	421_{72}	$\mathbf{404_{106}}$
FQE	583_{122}	319_{67}	$\mathbf{345_{64}}$	410_{79}	$\mathbf{398_{111}}$
AR Ensemble	551_{81}	629_{14}	574_{35}	642_{1}	575_{61}
VLM	331_{15}	315_{20}	$\mathbf{310_{31}}$	486_{6}	663_{2}
VLM+RSA (MSE)	343_{13}	324_{4}	$\mathbf{306_{3}}$	463_{21}	661_{8}
VLM+RSA	351_{7}	314_{23}	$\mathbf{305_{25}}$	448_{3}	665_{4}
VLM+RSA Ens.	242_{20}	312_{37}	$\mathbf{345_{80}}$	464_{6}	667_{20}
VLBM	$\mathbf{202_{4}}$	$\mathbf{269_{55}}$	$\mathbf{331_{43}}$	$\mathbf{265_{2}}$	598_{11}
MAE	Halfcheetah -E	Halfcheetah -M-E	Halfcheetah -M	Halfcheetah -M-R	Halfcheetah -R
IS	1404_{152}	1400_{146}	1217_{123}	1409_{154}	1405_{155}
VPM	945_{164}	1427_{111}	1374_{153}	1384_{148}	1411_{154}
DICE	944_{161}	1078_{132}	1382_{130}	1440_{158}	1446_{156}
DR	1025_{95}	1015_{103}	1222_{134}	1001_{129}	$\mathbf{949_{126}}$
FQE	1031_{95}	1014_{101}	1211_{130}	1003_{132}	$\mathbf{938_{125}}$
AR Ensemble	1226_{222}	$\mathbf{480_{24}}$	$\mathbf{553_{64}}$	$\mathbf{846_{64}}$	1537_{16}
VLM	520_{242}	526_{49}	624_{53}	1478_{27}	1490_{1}
VLM+RSA (MSE)	469_{159}	$\mathbf{426_{49}}$	689_{39}	1432_{10}	1489_{0}
VLM+RSA	414_{155}	$\mathbf{446_{50}}$	622_{153}	1473_{20}	1492_{6}
VLM+RSA Ens.	$\mathbf{253_{20}}$	773_{139}	1306_{113}	1468_{41}	1525_{22}
VLBM	$\mathbf{201_{22}}$	$\mathbf{456_{30}}$	$\mathbf{517_{50}}$	1281_{170}	1495_{2}
MAE	Walker2d -E	Walker2d -M-E	Walker2d -M	Walker2d -M-R	Walker2d -R
IS	405_{62}	436_{62}	428_{60}	427_{60}	430_{61}
VPM	$\mathbf{367_{68}}$	425_{61}	426_{60}	424_{64}	440_{58}
DICE	437_{60}	322_{60}	273_{31}	374_{51}	419_{57}
DR	519_{179}	$\mathbf{217_{46}}$	368_{74}	296_{54}	347_{74}
FQE	453_{142}	$\mathbf{233_{42}}$	350_{79}	313_{73}	354_{73}
AR Ensemble	530_{102}	408_{4}	444_{6}	327_{106}	383_{42}
VLM	538_{30}	380_{12}	$\mathbf{250_{17}}$	$\mathbf{160_{46}}$	452_{8}
VLM+RSA (MSE)	521_{30}	340_{20}	361_{19}	236_{14}	443_{15}
VLM+RSA	522_{41}	358_{86}	$\mathbf{253_{9}}$	$\mathbf{125_{7}}$	326_{161}
VLM+RSA Ens.	538_{9}	386_{23}	$\mathbf{201_{38}}$	$\mathbf{168_{11}}$	441_{24}
VLBM	517_{24}	$\mathbf{288_{72}}$	$\mathbf{244_{33}}$	$\mathbf{156_{28}}$	$\mathbf{262_{22}}$
MAE	Hopper -E	Hopper -M-E	Hopper -M	Hopper -M-R	Hopper -R
IS	$\mathbf{106_{29}}$	360_{47}	405_{48}	438_{11}	412_{45}
VPM	442_{43}	-	433_{44}	-	438_{44}
DICE	259_{54}	266_{40}	215_{41}	398_{2}	$\mathbf{122_{16}}$
DR	426_{99}	234_{77}	307_{85}	298_{14}	289_{50}
FQE	282_{76}	252_{28}	283_{73}	295_{7}	261_{42}
AR Ensemble	369_{16}	292_{11}	393_{42}	477_{34}	454_{34}
VLM	148_{31}	$\mathbf{136_{19}}$	210_{22}	$\mathbf{138_{9}}$	382_{66}
VLM+RSA (MSE)	246_{40}	186_{10}	232_{29}	124_{12}	415_{15}
VLM+RSA	270_{2}	$\mathbf{140_{15}}$	$\mathbf{117_{28}}$	$\mathbf{117_{16}}$	412_{20}
VLM+RSA Ens.	253_{23}	$\mathbf{149_{42}}$	233_{62}	$\mathbf{115_{17}}$	306_{47}
VLBM	266_{8}	$\mathbf{140_{4}}$	$\mathbf{126_{47}}$	$\mathbf{124_{21}}$	385_{27}

Table 5: MAE between estimated and ground-truth returns for all Gym-Mujoco tasks. Results are obtained by averaging over 3 random seeds used for training, with standard deviations shown in subscripts.

MAE	Door human	Door cloned	Door expert	Pen human	Pen cloned	Pen expert
IS	870_{173}	891_{188}	$\mathbf{648_{122}}$	3926_{128}	1707_{128}	4547_{222}
VPM	862_{163}	1040_{188}	879_{182}	$\mathbf{1569_{215}}$	2324_{129}	2325_{136}
DICE	1108_{199}	697_{79}	856_{134}	4193_{244}	1454_{219}	2963_{279}
DR	$\mathbf{379_{65}}$	$\mathbf{424_{73}}$	1353_{218}	2846_{200}	1323_{98}	2013_{564}
FQE	$\mathbf{389_{60}}$	$\mathbf{438_{81}}$	1343_{84}	2872_{170}	1232_{105}	$\mathbf{1057_{281}}$
AR Ens.	734_{3}	826_{7}	2236_{16}	2161_{12}	1981_{106}	1803_{226}
VLBM	710_{152}	933_{1}	$\mathbf{600_{84}}$	$\mathbf{1637_{286}}$	$\mathbf{669_{270}}$	$\mathbf{1002_{262}}$
MAE	Hammer human	Hammer cloned	Hammer expert	Relocate human	Relocate cloned	Relocate expert
IS	7352_{1118}	7403_{1126}	3052_{608}	$\mathbf{638_{217}}$	632_{215}	2731_{147}
VPM	7105_{1107}	7459_{1114}	7312_{1117}	806_{166}	586_{135}	$\mathbf{620_{214}}$
DICE	$\mathbf{5677_{936}}$	$\mathbf{4169_{839}}$	3963_{758}	4526_{474}	1347_{485}	1095_{221}
DR	$\mathbf{5768_{751}}$	6101_{679}	3485_{590}	$\mathbf{606_{116}}$	$\mathbf{412_{124}}$	1193_{350}
FQE	$\mathbf{6000_{612}}$	5415_{558}	$\mathbf{2950_{728}}$	$\mathbf{593_{113}}$	$\mathbf{439_{125}}$	1351_{393}
AR Ens.	6897_{27}	7240_{12}	3057_{8}	823_{7}	662_{6}	2138_{4}
VLBM	$\mathbf{6184_{479}}$	7267_{402}	$\mathbf{2682_{146}}$	$\mathbf{624_{25}}$	$\mathbf{388_{183}}$	2021_{270}

Table 6: MAE between estimated and ground-truth returns for all Adroit tasks. Results are obtained by averaging over 3 random seeds used for training.

	State Dim.	Action Dim.	Early Term.	Continuous Ctrl.	Dataset	Dataset Size
Ant	27	8	Yes	Yes	random	999,427
					medium-replay	301,698
					medium	999,175
					medium-expert	1,998,158
					expert	999,036
Halfcheetah	17	6	No	Yes	random	999,000
					medium-replay	201,798
					medium	999,000
					medium-expert	1,998,000
					expert	999,000
Hopper	11	3	Yes	Yes	random	999,999
					medium-replay	401,598
					medium	999,998
					medium-expert	1,998,966
					expert	999,061
Walker2d	17	6	Yes	Yes	random	999,999
					medium-replay	301,698
					medium	999,322
					medium-expert	1,998,318
					expert	999,000

Table 7: Summary of the Gym-Mujoco environments and datasets used to train VLBM and baselines.

	State Dim.	Action Dim.	Early Term.	Continuous Ctrl.	Dataset	Dataset Size
Pen	45	24	Yes	Yes	human	4,975
					cloned	496,264
					expert	494,248
Door	39	28	No	Yes	human	6,704
					cloned	995,642
					expert	995,000
Hammer	46	26	No	Yes	human	11,285
					cloned	996,394
					expert	995,000
Relocate	39	30	No	Yes	human	9,917
					cloned	996,242
					expert	995,000

Table 8: Summary of the Adroit environments and datasets used to train VLBM and baselines.