# DISTRACTOR-ROBUST REINFORCEMENT LEARNING VIA VARIATIONAL BISIMULATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Model-based reinforcement learning (MBRL) promises data efficiency and generalization, but typical reconstruction-based objectives encourage models to waste representational capacity on task-irrelevant distractors. We introduce VIBES (Variational Inference for Bisimulation-based Encoded States), a new objective that replaces pixel reconstruction with an adversarial term, which enforces that latent states suffice to predict both rewards and the *next* latent state. We show theoretically that, under mild assumptions, global optima of this objective correspond to encoders that induce bisimulation relations, ensuring that latent states capture task-relevant information while discarding irrelevant variation. Our method serves as a drop-in replacement for Dreamer's model-learning component, achieves state-of-the-art performance on the Distracting Control Suite, and extends to visually complex manipulation in Realistic ManiSkill. Unlike prior approaches, it does not rely on image-specific augmentations and applies equally well to high-dimensional vector-state tasks, demonstrated on a 100-link swimmer robot. Finally, latent-space analyses (UMAP embeddings and nearest-neighbor probes) confirm that the learned representations are sensitive to task-relevant structure while invariant to distractors.

## 1 INTRODUCTION

Model-based reinforcement learning (MBRL) promises a powerful set of advantages: it can achieve high data efficiency by reusing learned dynamics for many policy updates, support transfer of knowledge across tasks through shared models, enable long-horizon reasoning and counterfactual planning, and, when paired with uncertainty estimates, drive intelligent exploration. These benefits make MBRL an attractive alternative to model-free methods.

However, in practice, learned models are inevitably imperfect—limited by capacity and the cost of collecting data and performing optimization—and this discrepancy between the model and the true environment often leads to suboptimal policies. One might hope that at least model-based RL methods would consistently find the "best" of the available set of imperfect candidate models; however, this is not even the case, because typical model training objectives are not aligned with the ultimate goal of improving control performance. This issue is referred to as objective mismatch (Lambert et al., 2020). Worse still, typical objectives actively incentivize models to expend representational capacity on irrelevant *distractors*—features of the observation that have no bearing on the task.

Consider the case of autonomous driving: while the agent must model the road, traffic signals, and surrounding vehicles, it need not track details such as the positions of leaves on nearby trees. Yet typical model-learning objectives reward the model for reconstructing those leaves just as faithfully as traffic lights, thereby wasting capacity on irrelevant information. This problem has been shown to significantly degrade the performance of otherwise strong algorithms such as Dreamer (Hafner et al., 2020), which performs well on clean image-based benchmarks but struggles in the presence of distractors (Nguyen et al., 2021; Zhu et al., 2023). This phenomenon exemplifies objective mismatch because the agent actively chooses a model that is poor for policy improvement, even if a model that ignores distractors and is well-suited to policy improvement exists within its model class.

A natural way to mitigate objective mismatch is through state abstraction. State abstraction filters observations so that only task-relevant information is retained. In MBRL, this allows a dynamics model to focus its capacity on predicting what matters for decision-making, rather than reconstructing irrelevant features. Numerous approaches to state abstraction have been proposed, but many rely on heuristic, domain-specific strategies, such as image augmentations (Laskin et al., 2020a; Yarats



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

et al., 2021; 2022; Laskin et al., 2020b; Lee et al., 2020). This partially defeats the purpose of RL, which is to solve tasks generally, without relying on domain knowledge.

In this work, we instead pursue a principled approach based on *bisimulation relations* (Hennessy & Milner, 1985; Givan et al., 2003). Bisimulation provides a theoretical foundation for abstraction by requiring that states be equivalent if and only if they yield identical immediate rewards and identical distributions over future abstract states. This ensures that task-relevant features are preserved while irrelevant variation is discarded. Our work builds on recent approaches that use *bisimulation metrics*—continuous distance functions that approximate behavioral equivalence (Zhang et al., 2021; Castro, 2020). In contrast, we directly learn bisimulation *relations*, showing that our variational objective is optimized only when the encoder induces partitions of histories that satisfy reward and dynamics sufficiency. This yields a representation aligned with the theoretical definition of bisimulation while avoiding the need for brittle and hyperparameter-sensitive distance metrics. We call our resulting model-based RL algorithm **VIBES**: *Variational Inference for Bisimulation-based Encoded States*.

We evaluate our method on the challenging Distracting Control Suite (DCS) (Stone et al., 2021), where standard DeepMind Control Suite tasks are augmented with strong visual distractors, and on Realistic ManiSkill (Zhu et al., 2023), where robotic manipulation tasks from ManiSkill2 (Gu et al., 2023) are set against real indoor scenes from Matterport3D (Chang et al., 2017). On DCS, our approach consistently outperforms prior state-of-the-art RL methods, both model-free and model-based, for distractor robustness, despite requiring no specialized data augmentation. On Realistic ManiSkill, it performs statistically indistinguishably from DreamerV3-12M (Hafner et al., 2025a) with fewer than one-third as many learned parameters. Moreover, we demonstrate that the learned representations generalize to unseen distractors and also yield strong performance in a high-dimensional vector-state control task (a swimmer robot with a 401-dimensional observation space). Finally, latent-space analyses confirm that our method learns representations that preserve task-relevant structure, discard distractors, and capture symmetries in the dynamics (e.g., rotational invariances), yielding a more compact and general abstraction.

In summary, the contributions of this paper are as follows:

1. We propose a principled state abstraction method based on bisimulation relations, integrated into a variational inference framework for model learning.
2. We demonstrate strong empirical performance on image-based tasks with heavy distractors (DCS and Realistic ManiSkill) and on high-dimensional vector-state control, without requiring data augmentations.
3. We provide quantitative analyses showing that our learned representations preserve task-relevant features while discarding irrelevant variation.

## 2 Background

### 2.1 Variational Inference for Learning State Representations




(a) PGM assumed by *Dreamer*.

(b) PGM assumed by *VIBES*.

Figure 1: Side-by-side comparison of graphical model assumptions. These Probabilistic Graphical Models (PGMs) model environment dynamics, where the latent state $\tilde{s}_t$ represents the true, underlying hidden state from which observations and rewards arise. Actions may be either fixed exogenously or depend on observations through a policy. **Left (Dreamer):** the observation at time $t$ is conditionally independent of prior observations and actions given the current latent state, i.e., $o_t \perp\!\!\!\perp o_{<t}, a_{<t} \mid \tilde{s}_t$. **Right (Ours):** we allow $o_t$ to depend on the full history of observations (and, in general, on the history of states/actions). Under Dreamer's assumption, any information in $o_{<t}$ needed to predict $o_t$ must be encoded into $s_t$; in our formulation this pressure is removed from $s_t$ because past observations can inform $o_t$ directly.

Variational inference (VI) provides a general framework for learning latent-variable models. In such models, observations are assumed to be generated from unobserved latent variables through a probabilistic process. Latent-variable models are powerful because they enable compact, structured representations that capture essential features of the data. In the context of RL, VI enables learning latent state representations together with dynamics and reward models. Dreamer[1] is a prominent VI formulation that maximizes a lower bound on $\log p(o_{0:T}, r_{0:T} \mid a_{0:T-1})$ using a decoder $\hat{p}(o_t \mid s_t)$, a transition $\hat{p}(s_t \mid s_{t-1}, a_{t-1})$, and a reward model $\hat{p}(r_t \mid s_t, a_t)$. A key modeling assumption is $o_t \perp\!\!\!\perp (o_{<t}, a_{<t}) \mid s_t$, which pressures $s_t$ to carry any history information useful for pixel prediction—even if irrelevant for control (Figure 1a). This "remember everything" incentive misaligns model learning with decision making, especially when in tasks with distractors.

In Sec. 3, we propose a modified objective that relaxes this conditional independence assumption. By allowing observations to depend on past observations directly, we remove the pressure for $\tilde{s}_t$ to transmit irrelevant predictive information, freeing the latent representation to focus on features that matter for predicting rewards and dynamics (Fig. 1b).

### 2.2 Bisimulation Relations

Bisimulation originates in the theory of Markov decision processes (MDPs) and provides a principled notion of when two states can be considered behaviorally equivalent. Intuitively, two states are bisimilar if they yield the same rewards and, under any action, transition to future states that are themselves bisimilar. This recursive definition ensures that bisimilar states are indistinguishable for the purposes of planning and control.

In our setting, the agent does not observe the environment state directly, only observations from which the state must be inferred. We therefore work in a partially observable MDP (POMDP) with action space $\mathcal{A}$. In a POMDP, the history $H_t = (o_{\leq t}, a_{<t})$ plays the role of the state (Ni et al., 2024; Simão et al., 2023), so a bisimulation relation partitions the space of all histories into groups of equivalent histories.

Formally, let $\mathcal{H}$ denote the space of all histories. A relation $\sim$ on $\mathcal{H}$ is a *bisimulation relation* if, for all $H^i, H^j \in \mathcal{H}$,

$$H^i \sim H^j \iff \begin{cases} R(H^i, a) = R(H^j, a) & \forall a \in \mathcal{A}, \\ P_{\tilde{H}'|H,a}(\tilde{H}' \mid H^i, a) = P_{\tilde{H}'|H,a}(\tilde{H}' \mid H^j, a) & \forall a \in \mathcal{A},\ \forall \tilde{H}' \in \tilde{\mathcal{H}}, \end{cases} \tag{1}$$

where $\tilde{\mathcal{H}}$ is the set of all groups of equivalent histories and $\tilde{H}'$ is a particular group of equivalent histories; $P_{\tilde{H}'|H,a}(\tilde{H}' \mid H, a) = \sum_{H' \in \tilde{H}'} P(H' \mid H, a)$ is the probability that the next history falls in group $\tilde{H}'$, where $P(H' \mid H, a)$ is the probability of observing $o'$ after taking action $a$ given history $H$, so that the next history is $H' = (H, a, o')$; and $R(H, a)$ is the distribution over the reward received after taking action $a$ given history $H$. Put differently, equivalent histories must yield identical immediate reward distributions under every action and identical distributions over groups of next histories.

The coarsest such relation partitions $\mathcal{H}$ into *bisimulation equivalence classes*, which can be treated as abstract states without loss of optimal control performance (Givan et al., 2003; Li et al., 2006), making bisimulation relations an ideal basis for a state representation.

**Bisimulation for RL** Bisimulation is appealing for reinforcement learning because it formalizes the idea of *state abstraction*: discarding irrelevant details of the environment while preserving task-relevant information. A representation that encodes bisimulation equivalence classes retains precisely the information needed for reward prediction and decision making. By contrast, states that differ only in irrelevant features (e.g., background distractors) are collapsed into the same abstract state.

Recent work has sought to approximate bisimulation in continuous or high-dimensional settings by learning embeddings that preserve bisimulation distances (Ferns et al., 2004; Zhang et al., 2021). We refer the reader to Appendix D for an extended review of the landscape of comparable representation-learning methods. In practice, exact bisimulation is infeasible in complex environments, but the principle provides a powerful guide: representations should be invariant to irrelevant

[1]Unless a version is specified, "Dreamer" refers to the variational formulation that the Dreamer family builds on (Hafner et al., 2019; 2020; 2021; 2025a). Empirically, we compare against the strongest applicable Dreamer variant (Sec. 4.1).

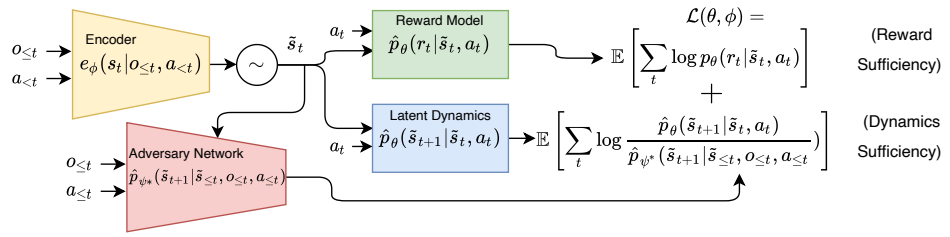



Figure 2: **Model diagram and objective.** Our model comprises an *encoder* (CNN+GRU) that maps history $(o_{\leq t}, a_{<t})$ to a stochastic latent $\tilde{s}_t$; a *dynamics model* that predicts $\tilde{s}_{t+1}$ from $(\tilde{s}_t, a_t)$; a *reward model* $p(r_t \mid \tilde{s}_t, a_t)$; and an *adversary network* that predicts $\tilde{s}_{t+1}$ from the history and current action, $(o_{\leq t}, a_{\leq t})$. Our objective has two parts: a *reward sufficiency* term, maximized when the latent state contains all information needed to predict the current reward, and a *dynamics sufficiency* term, maximized when the latent state is as informative as the full history for predicting the *next* latent state.

variation while distinguishing states that differ in reward or in their distribution over future behaviors. In this work, we show how to incorporate this principle directly into a variational inference framework for model-based RL, yielding latent representations that approximate bisimulation relations and are robust to distractors.

## 3 Learning Bisimulation Relations with Variational Inference

We adopt a variational approach to learn a latent state, a dynamics model, and a reward model, but replace pixel reconstruction with an *adversarial* term that measures whether the latent state at time $t$ is as informative as the full history for predicting the *next* latent state. The resulting objective has two parts: a *reward sufficiency* term (the latent must predict rewards) and a *dynamics sufficiency* term (the latent must be as predictive of the next latent as history). A full derivation is deferred to Appendix A.

### 3.1 Model Objective

Let $\tilde{s}_t \sim e_\phi(o_{\leq t}, a_{<t})$ denote the stochastic latent at time $t$, where $e_\phi$ is an *encoder network*. In practice, our encoder outputs a Gaussian with fixed standard deviation $0.01$, and a mean with an $\ell_2$ norm of 1, thus fixing an upper bound on the amount of information that can be encoded into $\tilde{s}_t$.

Our model objective is

$$\mathcal{L}(\theta, \phi) \;=\; \mathbb{E}\left[\sum_t \log \frac{\hat{p}_\theta(\tilde{s}_t \mid \tilde{s}_{t-1}, a_{t-1})}{\hat{p}_{\psi^*}(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t})} \;+\; \log \hat{p}_\theta(r_t \mid \tilde{s}_t, a_t)\right]. \tag{2}$$

The full derivation is given in Appendix A. Here $\hat{p}_\theta(\tilde{s}_t \mid \tilde{s}_{t-1}, a_{t-1})$ is the latent dynamics; $\hat{p}_\theta(r_t \mid \tilde{s}_t, a_t)$ is the reward model; and $\hat{p}_{\psi^*}(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t})$ is an *adversary* (the optimal history-conditioned predictor of $\tilde{s}_t$), obtained by an inner maximization. In practice, we approximate $\psi^*$ by training the adversary with a higher learning rate than $\phi$, so that $\psi$ tracks close to its optimum at each step. In our implementation, the dynamics model additionally maintains a deterministic recurrent state $h_{t-1}$, summarizing past latents and actions, alongside $\tilde{s}_{t-1}$ (see Appendix C.1). We write $\tilde{s}_{t-1}$ throughout for readability.

**Intuitive explanation of $\mathcal{L}$.** The reward sufficiency term $\log p(r_t|\tilde{s}_t, a_t)$ encourages $\tilde{s}_t$ to be sufficient for predicting $r_t$ given $a_t$. The dynamics sufficiency term $\frac{\hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_{t-1})}{\hat{p}_{\psi*}(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t})}$ compares two predictors of $\tilde{s}_t$: our latent dynamics model $\hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_{t-1})$ that only sees $(\tilde{s}_{t-1}, a_{t-1})$ vs. an optimal predictor that sees the full history, $\hat{p}_{\psi*}(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t})$. Even though $\tilde{s}_{t-1}$ is itself built from the full history by the encoder, it is a (potentially) lossy compression of that history, whereas the adversary observes the raw history directly. This makes the adversary a useful training-time signal: whenever the encoder has discarded task-relevant information, the adversary can still recover it and out-predict the dynamics model, and this gap is exactly what pressures the encoder to retain that information. This log ratio is upper-bounded by 0 and equals 0 iff $\tilde{s}_{t-1}$ contains all history information needed to predict $\tilde{s}_t$. Maximizing equation 2 therefore pushes the representation toward *reward* and *dynamics* sufficiency.

### 3.2 WHY $\mathcal{L}$ YIELDS BISIMULATION

We formalize the guarantee with a lemma (sufficiency ⇒ bisimulation) and a theorem (optimality of $\mathcal{L}$ ⇒ sufficiency). Full proofs are in App. B.

**Lemma 3.1** (Sufficiency induces bisimulation)**.** *Let $\tilde{s}_t = e_\phi(o_{\leq t}, a_{<t})$. Suppose for all $t$ and actions $a_t$:*

$$\textit{(Dynamics sufficiency)} \quad \tilde{s}_t \perp\!\!\!\perp (o_{<t}, a_{<t-1}) \mid \tilde{s}_{t-1}, a_{t-1}, \tag{3}$$

$$\textit{(Reward sufficiency)} \quad r_t \perp\!\!\!\perp (o_{\leq t}, a_{<t}) \mid \tilde{s}_t, a_t. \tag{4}$$

*Then the history partition induced by $e_\phi$ is a bisimulation relation.*

**Theorem 3.2** (Global optima of $\mathcal{L}$ yield bisimulation)**.** *Assume sufficiently expressive model classes for rewards and dynamics. Any global maximizer of equation 2 satisfies, for all $t$,*

$$\hat{p}_\theta(\tilde{s}_t \mid \tilde{s}_{t-1}, a_{t-1}) = p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}), \qquad \hat{p}_\theta(r_t \mid \tilde{s}_t, a_t) = p(r_t \mid o_{\leq t}, a_{\leq t}).$$

*These equalities imply the sufficiency conditions in Lemma 3.1, hence the encoder-induced partition is a bisimulation.*

*Proof Sketch.* By Jensen's inequality, $\mathcal{L}$ is maximized if and only if

$$\hat{p}_\theta(\tilde{s}_t \mid \tilde{s}_{t-1}, a_{t-1}) = p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}), \qquad \hat{p}_\theta(r_t \mid \tilde{s}_t, a_t) = p(r_t \mid o_{\leq t}, a_t)$$

which implies that $e_\phi$ satisfies equation 3 and equation 4, which by Lemma 3.1 implies that $e_\phi$ is a bisimulation relation. A complete proof is provided in Appendix B.2.

**Optimizing the Model Objective** Optimizing our model objective in equation 2 is slightly more involved than in Dreamer because it contains an inner maximization problem (equation 13) used to compute $\psi^*$. Fortunately, as we prove in Appendix B.3, the derivative with respect to the explicit occurrence of $\phi$ in the adversarial objective vanishes, *i.e.,* $\frac{\partial}{\partial \phi}\ \mathbb{E}[\log \hat{p}_{\psi^*}(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t})] = 0$, allowing us to treat $\psi^*$ as fixed when computing $\nabla_\theta \mathcal{L}$ and $\nabla_\phi \mathcal{L}$. This allows the model training to be reduced from a bilevel optimization problem, to a simpler alternating optimization, where $\psi$ is optimized before each gradient update of $\theta$ and $\phi$. In practice, we found it sufficient to perform one step of gradient ascent on $\psi$ for each step of gradient ascent on $\theta$ and $\phi$. Our complete model training procedure is summarized in Algorithm 1 in Appendix C.2.

**Integration into Online Reinforcement Learning** The objective in equation 2 serves as a direct drop-in replacement for the model-learning component of Dreamer. Once trained, the resulting world model is used in a manner similar to Dreamer: to simulate imagined rollouts in the latent state space for policy and value learning. Thus, apart from replacing the reconstruction loss with our bisimulation-driven objective, the rest of the Dreamer framework (policy optimization, value estimation, and training loop) remains largely unchanged. Specific details of the RL training framework are presented in Appendix C.1.

## 4 EXPERIMENTS

We evaluate VIBES in four settings. We first benchmark it on the Distracting Control Suite (DCS) and ablate the adversarial objective (Section 4.1). We then test robotic manipulation under realistic background distractors on Realistic ManiSkill (Section 4.2) and high-dimensional control from vector states (Section 4.3). Finally, we analyze the learned latent spaces to test whether they preserve task-relevant features while discarding distractors (Section 4.4). Architectures, hyperparameters, and benchmark configurations are in Appendix C.

### 4.1 DISTRACTING CONTROL SUITE

The Distracting Control Suite (DCS) (Stone et al., 2021) extends the DeepMind Control Suite with three types of distractors: (i) *background distractors*, where high-resolution natural videos play behind the agent, (ii) *color distractors*, where the agent's body color changes dynamically, and (iii) *camera distractors*, where the camera viewpoint shifts during the episode. Each task requires fine-grained control of the agent's state while the distractors add large amounts of task-irrelevant variation, so DCS directly tests whether an algorithm can overcome the objective mismatch problem from Section 1. Our DCS configuration is detailed in Appendix C.4.

We compare against SAC+RAD (Laskin et al., 2020b), QT-Opt (Kalashnikov et al., 2018) variants with RAD and DrQ (Yarats et al., 2021) regularization, RSAC (Gupta et al., 2022), CURL (Laskin

| Method | Mean | BiC-Catch | C-swingup | C-run | F-spin | R-easy | W-walk |
|---|---|---|---|---|---|---|---|
| SAC+RAD† | $270 \pm 55$ | $366 \pm 116$ | $297 \pm 41$ | $198 \pm 76$ | $338 \pm 116$ | $173 \pm 22$ | $249 \pm 271$ |
| QT-Opt+RAD† | $343 \pm 43$ | $490 \pm 125$ | $467 \pm 24$ | $170 \pm 16$ | $393 \pm 178$ | $428 \pm 133$ | $109 \pm 24$ |
| QT-Opt+DrQ† | $265 \pm 25$ | $395 \pm 76$ | $431 \pm 35$ | $126 \pm 20$ | $203 \pm 65$ | $343 \pm 104$ | $91 \pm 6$ |
| RSAC† | $275 \pm 25$ | $181 \pm 63$ | $465 \pm 41$ | $292 \pm 20$ | $86 \pm 108$ | $145 \pm 61$ | $482 \pm 39$ |
| CURL† | $391 \pm 22$ | $102 \pm 39$ | $432 \pm 29$ | $233 \pm 26$ | $648 \pm 63$ | $253 \pm 78$ | $678 \pm 69$ |
| CoRe† | $586 \pm 38$ | $798 \pm 59$ | $499 \pm 43$ | $423 \pm 43$ | $\mathbf{713 \pm 159}$ | $340 \pm 118$ | $\mathbf{744 \pm 78}$ |
| SAR | $173 \pm 17$ | $286 \pm 84$ | $351 \pm 29$ | $240 \pm 70$ | $14 \pm 16$ | $108 \pm 31$ | $36 \pm 3$ |
| DBC | $68 \pm 12$ | $99 \pm 45$ | $119 \pm 21$ | $62 \pm 49$ | $10 \pm 7$ | $89 \pm 21$ | $29 \pm 2$ |
| HRSSM | $599 \pm 58$ | $591 \pm 247$ | $651 \pm 13$ | $454 \pm 34$ | $688 \pm 218$ | $598 \pm 63$ | $611 \pm 82$ |
| DreamerV3 | $277 \pm 10$ | $103 \pm 36$ | $339 \pm 10$ | $309 \pm 42$ | $33 \pm 15$ | $332 \pm 17$ | $547 \pm 7$ |
| VIBES (Ours) | $\mathbf{701 \pm 29}$ | $\mathbf{844 \pm 95}$ | $\mathbf{696 \pm 50}$ | $\mathbf{489 \pm 22}$ | $670 \pm 167$ | $\mathbf{930 \pm 12}$ | $574 \pm 56$ |
| No Adv. Obj. | $606 \pm 27$ | $593 \pm 158$ | $658 \pm 28$ | $436 \pm 27$ | $469 \pm 57$ | $925 \pm 21$ | $556 \pm 23$ |
| Recon. Obj. | $417 \pm 49$ | $144 \pm 96$ | $543 \pm 72$ | $469 \pm 11$ | $347 \pm 164$ | $360 \pm 280$ | $639 \pm 101$ |

Table 1: Performance on Distracting Control Suite tasks at 500K environment steps. Mean episode return $\pm$ 95% confidence interval across 5 seeds. Bold marks the highest mean in each column; underlined results have confidence intervals that overlap the bolded entry. † denotes results taken from prior work (Srivastava et al., 2021), converted from mean $\pm$ SE to 95% confidence intervals.

et al., 2020a), CoRe (Srivastava et al., 2021), SAR (Liang et al., 2024), the bisimulation-based methods DBC (Zhang et al., 2021) and HRSSM (Sun et al., 2024), and DreamerV3 (Hafner et al., 2025a), the strongest Dreamer variant applicable to our setting. We exclude DreamerV4 (Hafner et al., 2025b), which targets a different problem: learning multi-task agents from offline human demonstrations. All methods are evaluated at 500K environment steps on a held-out set of background video distractors, directly testing the generalization of the learned representation.

As shown in Table 1, VIBES achieves the highest mean return across tasks and the highest per-task mean on four of six tasks. It is the only method to solve *Reacher-easy*, even though this task is structurally simpler than the others. We hypothesize that this stems from Reacher-easy's randomized goal location. Many existing methods learn representations that emphasize agent-centric dynamics—the aspects of the system directly influenced by actions. Since the goal position is unaffected by actions, it is often underrepresented in the latent even though it determines the reward. VIBES instead requires the latent to preserve all reward-predictive information, so both the agent state and the goal location are encoded.

**Ablation Studies** Removing the adversarial objective (No Adv. Obj.) lowers the mean return on every task, confirming that the term provides a measurable benefit, although the ablation still has a higher mean return than every baseline. We hypothesize that the adversarial term matters least when the task-relevant state is (1) governed by deterministic dynamics and (2) fully observable from the observations. Under these conditions, both $e_\phi$ and $p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t})$ become deterministic. The adversarial objective therefore reduces to a constant and can be omitted without changing the optimal solution. Condition (1) holds in our setting because the underlying MuJoCo simulator is deterministic. Condition (2) can fail when limited rendering resolution or distractors obscure the agent's state (e.g., when background or camera motion overlaps with agent movement). The adversarial term is therefore most valuable where observations are imperfectly informative about the underlying state, which explains its empirical benefits.

We also evaluate Recon. Obj., which keeps our architecture but replaces the adversarial objective with a Dreamer-style pixel reconstruction objective:
$\mathcal{L}_{\text{recon}}(\theta, \phi) = \mathbb{E}_{\vec{o},\vec{r},\vec{a},\vec{s}} \left[ \sum_t \left( \log \hat{p}_\theta(o_t|\tilde{s}_t) + \log \hat{p}_\theta(r_t|\tilde{s}_t, a_t) + \log \frac{\hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_{t-1})}{e_\phi(\tilde{s}_t|o_{\leq t}, a_{<t})} \right) \right]$. [2]

Since this is the closest analogue of Dreamer's objective that our architecture admits, comparing it with VIBES isolates the contribution of the objective. Replacing the adversarial objective with reconstruction substantially lowers mean performance, though *Cheetah-run* is nearly unaffected and *Walker-walk* improves; when reconstruction remains competitive is an open question. Recon. Obj. is not a Dreamer model: it inherits our architecture[3] and, because the encoder's standard deviation is fixed, drops the encoder entropy term from the KL (footnote [2]). These differences may explain why it outperforms DreamerV3 (Hafner et al., 2025a).

[2] Because our encoder uses a fixed variance, the term in gray is constant and therefore omitted during optimization.

[3] Our encoder outputs a unit-norm mean with fixed standard deviation and uses a dedicated GRU. In DreamerV1, the posterior mean and standard deviation are learned without a norm constraint, while DreamerV2 and V3 use categorical latents. In all three, the posterior shares the dynamics model's recurrent state. Full architectural details are in Appendix C.1.

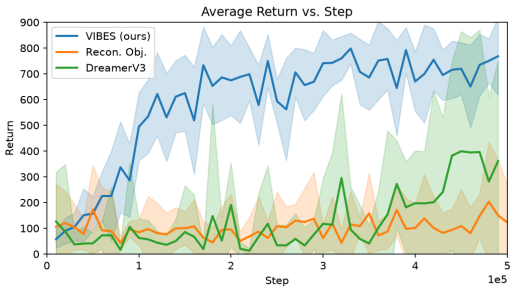



Figure 3: **100-link swimmer (vector states): learning curves.** Mean episode return over 4 runs versus environment steps, with 95% confidence intervals shaded, for VIBES, Recon. Obj. (Section 4.1), and DreamerV3 on the 100-link swimmer task with 401-dimensional vector observations. VIBES learns faster and reaches higher returns than both reconstruction-based baselines.

### 4.2 Realistic ManiSkill

Following HRSSM (Sun et al., 2024), we evaluate VIBES on two tasks from Realistic ManiSkill (Zhu et al., 2023), a variant of ManiSkill2 (Gu et al., 2023): Lift Cube and Turn Faucet, which require controlling a robot arm to lift a cube above a target height and rotate a faucet handle to a target angle, respectively. The default plain scene behind the robot and object is replaced with real indoor scenes from the Matterport3D dataset (Chang et al., 2017), sampled from a pool of 88 scenes at each episode reset and held fixed within the episode. This tests whether VIBES uses model capacity for task-relevant information rather than background reconstruction. Full details of this task are provided in Appendix C.5.

| Method | Learned params. | Lift Cube | Turn Faucet |
|---|---|---|---|
| TIA† | 13.750M | $274 \pm 196$ | $47 \pm 26$ |
| Denoised MDP† | 7.483M | $155 \pm 80$ | $71 \pm 31$ |
| RePo† | 1.671M | $83 \pm 18$ | $92 \pm 70$ |
| HRSSM† | 10.991M | $284 \pm 68$ | $278 \pm 78$ |
| DreamerV3-1M | 0.687M | $434.50 \pm 86.54$ | $104.91 \pm 27.45$ |
| DreamerV3-12M | 10.495M | $\mathbf{607.22 \pm 9.95}$ | $\mathbf{851.17 \pm 69.61}$ |
| VIBES (Ours) | 3.257M | $\underline{561.61 \pm 68.96}$ | $\underline{757.19 \pm 76.31}$ |

Table 2: Model capacity and performance on Realistic ManiSkill. Values are mean $\pm$ 95% confidence intervals across 6 seeds. † denotes results taken from HRSSM (Sun et al., 2024), converted from mean $\pm$ SD to 95% confidence intervals. Bolded numbers indicate the highest mean return in each column. Underlined numbers indicate results whose confidence intervals overlap with the bolded entry, suggesting statistically indistinguishable performance at the reported level.

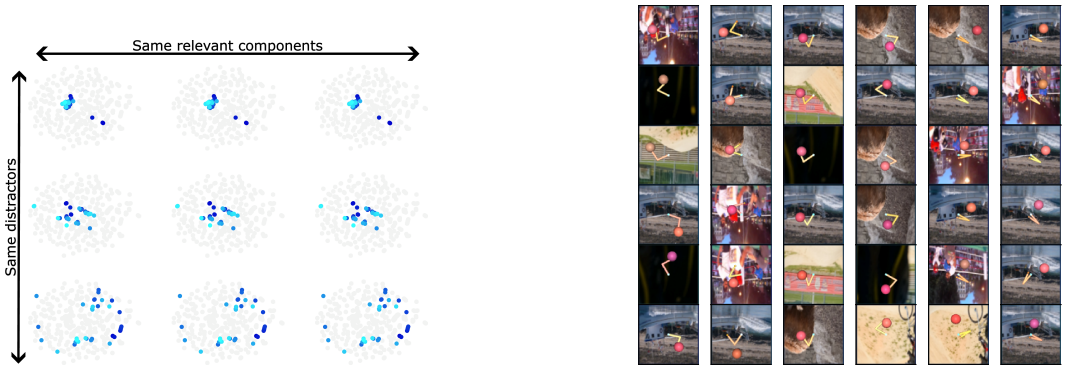
VIBES is statistically indistinguishable from the strongest baseline, DreamerV3-12M, on both tasks, reaching 92.5% and 89.0% of its returns with fewer than one-third as many learned parameters. It achieves a higher mean return than the remaining five methods on both tasks, with non-overlapping confidence intervals in every case except DreamerV3-1M on Lift Cube.

### 4.3 High-Dimensional Continuous Control from Vector States

To test generality beyond image-based tasks, we extend the *Swimmer* environment from the DeepMind Control Suite (Tassa et al., 2018) to a 100-link planar snake robot with a 204-dimensional state and a 401-dimensional observation—a high-dimensional, vector-based control task. The agent must propel its nose to a fixed target, with a reward that decays smoothly with distance according to a Lorentzian function. Rather than controlling all 99 joints directly, we parameterize the action space with a spatiotemporal sinusoid that maps four parameters to every joint torque. Implementation details are in Appendix C.6.

This setting tests whether VIBES scales to high-dimensional vector states, where reconstruction is difficult, and whether it works without the image-specific augmentations, such as random crops and color jittering, that prior approaches rely on.

We compare VIBES against DreamerV3 and against Recon. Obj., the reconstruction ablation from Section 4.1. Across 4 runs, VIBES learns faster and more stable policies and achieves higher returns than both baselines. Recon. Obj. struggles to make progress in the high-dimensional state space, and DreamerV3 improves only late in training, with high variance across runs. These results rein-




(a) UMAP embeddings (3×3). (b) Nearest neighbors in latent space.

Figure 4: **Latent-space analyses on *Reacher-easy*.** (a) UMAP projections of 9 encoded trajectories in a $3 \times 3$ grid; rows share task states and columns share background distractors. Latent states change with the task state but not with the background. (b) Five nearest latent neighbors for six query observations (top of each column). Neighbors differ in background but match the query's arm pose relative to the goal, sometimes rotated about the origin, showing that the representation is task-relevant and invariant to joint rotations of the arm and goal.

force our claim that bisimulation-based objectives offer a principled and general approach to state abstraction, applicable to both visual and non-visual domains.

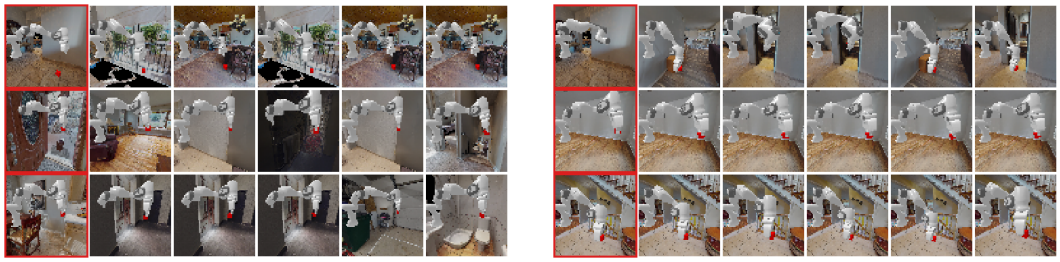
### 4.4 Latent Space Analysis

We generate 10 *Reacher-easy* trajectories from distinct initial states and render each over 10 background videos, producing 100 unique trajectory-background pairs. Each trajectory is a sequence of 100 states (the agent's joint angles and velocities and the goal position). We encode the frames of every pair and map the resulting latent states to two dimensions with UMAP (McInnes et al., 2018). Figure 4a shows a $3 \times 3$ subset. Each panel displays the latent states from all 100 pairs in gray and highlights those of one pair in blue. Moving across a row varies the background while keeping the trajectory fixed; moving down a column varies the trajectory while keeping the background fixed. The blue points barely move within a row but shift across rows, indicating that the representation varies with task state and is largely invariant to background distractors.

The UMAP analysis does not reveal which aspects of the observation are preserved versus collapsed, so we also conduct a nearest-neighbor analysis. We encode the dataset collected during online training on *Reacher-easy*, select 20 random latent states, and retrieve each one's five nearest neighbors from other trajectories under standardized L2 distance, which normalizes each dimension by its variance. Figure 4b shows six representative queries.

The analysis supports two main findings. First, as expected, the encoder is largely invariant to distractors: states with different backgrounds and agent colors are grouped together in latent space. Second, the encoder appears to have learned a rotational symmetry of the Reacher problem: because the optimal action sequence is unchanged by jointly rotating the arm and goal location about the origin, the encoder clusters together configurations that are equivalent under such rotations. This suggests that the learned representation is even more compact than the true underlying vector state, which preserves absolute angles. These results indicate that the encoder captures task-relevant structure while discarding both distractors and redundant symmetries, consistent with the bisimulation perspective.

We repeat the nearest-neighbor analysis on Realistic ManiSkill. For each of 6 seeds, we encode 50 episodes (5,000 frames) and retrieve each frame's five nearest neighbors under the same standardized L2 distance, excluding frames from the query's own episode so that adjacent, near-identical frames do not dominate the result. Figure 5 shows the difference: DreamerV3-12M organizes its latent space by background, while VIBES organizes it by task state. To quantify this, we measure the proportion of neighbors that share the query's background scene. Across six seeds (mean $\pm$ 95% confidence interval), this is $8.0 \pm 3.3\%$ for VIBES versus $40.9 \pm 7.1\%$ for DreamerV3-12M on Lift Cube, and $5.6 \pm 2.4\%$ versus $41.0 \pm 7.1\%$ on Turn Faucet. An encoder with no dependence on the background would score about $1/88 \approx 1.1\%$, since scenes are drawn from a pool of 88. This suggests that VIBES performs comparably to DreamerV3-12M with a third of the parameters



(a) VIBES. (b) DreamerV3-12M.

Figure 5: **Nearest latent neighbors on Lift Cube (query outlined in red).** (a) VIBES: neighbors come from different scenes but share the query's gripper and cube configuration. (b) DreamerV3-12M: neighbors share the query's scene, even when gripper and cube configuration differ (top row).

(Table 2) partly because it does not spend capacity on background distractors. Additional latent-space visualizations are provided in Appendix E.

## 5 Conclusions

We introduced **VIBES** (Variational Inference for Bisimulation-based Encoded States), a model-learning objective that replaces pixel reconstruction with two control-aligned terms: a *reward sufficiency* term and a *dynamics sufficiency* term. We proved that, under mild expressivity assumptions, global optima of VIBES correspond to encoders that induce bisimulation relations on histories, ensuring that the learned representation preserves task-relevant structure while collapsing distractors. Empirically, VIBES serves as a drop-in replacement for Dreamer's model-learning component and achieves state-of-the-art performance on the Distracting Control Suite without image-specific augmentations. On Realistic ManiSkill, VIBES extends this robustness to robotic manipulation with realistic background distractors, performing statistically indistinguishably from DreamerV3-12M with a third of its learned parameters. We further demonstrated applicability to high-dimensional vector-state control, and latent-space analyses showed that the learned latents are invariant to distractors, capture symmetries of the dynamics, and, unlike the reconstruction-based DreamerV3-12M, are organized by task-relevant state rather than background scene. Overall, VIBES advances representation learning for MBRL by bridging the gap between the theoretically appealing framework of bisimulation relations and the empirically successful and robust framework of VI.

## 6 Limitations and Future Work

1. **Approximation to bisimulation.** Our guarantees hold at global optima with sufficiently expressive model classes. In practice, limited capacity, finite data, and imperfect optimization mean we cannot quantify how close the learned representation is to a true bisimulation relation.
2. **Choice of abstraction.** Bisimulation is not the only task-aligned notion of state abstraction. Alternatives—e.g., value-equivalence, successor features, or predictive state representations—may be preferable for certain objectives (e.g., policy transfer or exploration). Extending VIBES to interpolate among these notions is an open direction.
3. **Computation.** The adversary adds compute and memory overhead compared to pure reconstruction-free objectives.
4. **Generalizability.** VIBES learns an abstraction relative to a single reward function, discarding information that doesn't influence the reward. This yields distractor robustness at small model scale, but limits transfer: information discarded for one task may be essential for another, so the representation must be relearned when the reward changes. Large-scale world models such as DreamerV4 (Hafner et al., 2025b) make the opposite trade-off, pretraining task-agnostic representations on unlabeled video at the cost of substantially larger models and datasets.
5. **Scope of evaluation.** We evaluate image-based control with distractors, robotic manipulation, and vector-state control in simulation. Evaluation on physical robots is outside this scope.
6. **Design choices in the bottleneck.** Our fixed-variance, unit-norm latent parameterization imposes a specific information bottleneck that may interact with task difficulty. Learning the bottleneck (e.g., via learnable variance or rate–distortion trade-offs) could improve adaptability across domains.

### Reproducibility Statement

We have made efforts to ensure that our results are reproducible. Appendix C contains the full implementation details: the model architecture (C.1), the training algorithm (C.2), all hyperparameters (C.3), and the environment configurations for the Distracting Control Suite (C.4), Realistic ManiSkill (C.5), and the 100-link swimmer (C.6). All reported results include the number of seeds and 95% confidence intervals. Anonymized code will be made available to reviewers and area chairs through the OpenReview discussion forum once it opens.

## References


Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10069–10076, 2020.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *International Conference on 3D Vision*, 2017.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *UAI*, volume 4, pp. 162–169, 2004.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning task informed abstractions. In *International Conference on Machine Learning*, pp. 3480–3491. PMLR, 2021.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial intelligence*, 147(1-2):163–223, 2003.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Surbhi Gupta, Gaurav Singal, Deepak Garg, and Swagatam Das. RSAC: a robust deep reinforcement learning strategy for dimensionality perturbation. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(5):1157–1166, 2022.

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*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025a.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b.

Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, January 1985. ISSN 0004-5411. doi: 10.1145/2455.2460. URL https://doi.org/10.1145/2455.2460.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control*, pp. 761–770. PMLR, 2020.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pp. 5639–5650. PMLR, 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020b.

Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for MDPs. *AI&M*, 1(2):3, 2006.

Dayang Liang, Qihang Chen, and Yunlong Liu. Sequential action-induced invariant representation for reinforcement learning. *Neural Networks*, 179:106579, 2024.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *International conference on machine learning*, pp. 2391–2400. PMLR, 2017.

Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8130–8139. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/nguyen21h.html`.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. In *The Twelfth International Conference on Learning Representations*, 2024.

Thiago D. Simão, Marnix Suilen, and Nils Jansen. Safe policy improvement for POMDPs via finite-state controllers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 15109–15117, 2023.

Nitish Srivastava, Walter Talbott, Martin Bertran Lopez, Shuangfei Zhai, and Joshua M Susskind. Robust robotic control from pixels using contrastive recurrent state-space models. In *Deep RL Workshop NeurIPS 2021*, 2021.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The Distracting Control Suite–A challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Ruixiang Sun, Hongyu Zang, Xin Li, and Riashat Islam. Learning latent dynamic robust representations for world models. In *International Conference on Machine Learning*, pp. 47234–47260. PMLR, 2024.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tongzhou Wang, Simon S. Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In *International Conference on Machine Learning*, pp. 22591–22612. PMLR, 2022.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations (ICLR)*, 2021.

Chuning Zhu, Max Simchowitz, Siri Gadipudi, and Abhishek Gupta. RePo: Resilient model-based reinforcement learning by regularizing posterior predictability. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

## A Derivation of the Model Objective

**Notation** Throughout the derivation, we use $\vec{o}$, $\vec{a}$, $\vec{r}$, and $\vec{\tilde{s}}$ to denote temporal sequences of observations, actions, rewards, and state representations, respectively. We use $p(\cdot)$ to denote *true* probability distributions (*i.e., those specified by the environment*) and $\hat{p}(\cdot)$ to denote approximations to said true distributions (*i.e.*, those specified by our model). We use $\tilde{s}_t \sim e_\phi(\cdot|o_{\leq t}, a_{<t})$ to denote the distribution over the state representation at time $t$, which depends on the observations before and including time $t$, and the actions up to time $t$, and is parameterized by $\phi$. Note that we do not place a circumflex ( $\hat{}$ ) over $e_\phi$, because this is our *chosen* state representation and does not represent an approximation to some underlying true distribution.

As in PlaNet (Hafner et al., 2019) and Dreamer (Hafner et al., 2020), we form a variational lower bound by marginalizing over the latent sequence and applying Jensen's inequality to the logarithm. This gives a lower bound on the expected log ratio of the approximate and true distributions over our observations and rewards given actions,

$$\mathbb{E}_{\vec{o},\vec{r},\vec{a}\sim D}\left[\log \frac{\hat{p}(\vec{o},\vec{r}|\vec{a})}{p(\vec{o},\vec{r}|\vec{a})}\right] \geq \mathbb{E}_{\vec{o},\vec{r},\vec{a}\sim D}\mathbb{E}_{\vec{\tilde{s}}\sim e_\phi}\left[\log \frac{\hat{p}(\vec{o},\vec{r},\vec{\tilde{s}}|\vec{a})}{e_\phi(\vec{\tilde{s}}|\vec{o},\vec{a})p(\vec{o},\vec{r}|\vec{a})}\right]. \tag{5}$$

From here, we diverge from the Dreamer derivation, factoring both the numerator and denominator differently.

**Numerator** In the Dreamer derivation, $\hat{p}(\vec{o},\vec{r},\vec{\tilde{s}}|\vec{a})$ factors into an *observation decoder* $\hat{p}_\theta(o_t|\tilde{s}_t)$, a *latent dynamics model* $\hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_{t-1})$, and a *reward decoder* $\hat{p}_\theta(r_t|\tilde{s}_t, a_t)$. We instead factor the numerator according to

$$\hat{p}(\vec{o},\vec{r},\vec{\tilde{s}}|\vec{a}) = \prod_{t=1}^{T} p(o_t|\tilde{s}_{\leq t}, o_{<t}, a_{<t})\hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_t)\hat{p}_\theta(r_t|\tilde{s}_t, a_t). \tag{6}$$

Note that the term in red has two key differences from the corresponding term in Dreamer. First, it conditions on $o_{<t}$, thereby avoiding the assumption that the system is Markovian in $s$. This allows us to avoid learning a state representation that is incentivized to encode all (potentially irrelevant) information from past observations that is predictive for the current observation (see Figure 1 right for a visual representation of these assumptions). Second, we use the *true* (but unknown) distribution over $p(o_t|\tilde{s}_{\leq t}, o_{<t}, a_{<t})$ rather than a model-approximated distribution, allowing us to avoid having to learn an observation decoder and the associated challenges of representing high-dimensional probability distributions. As we shall see, the fact that this distribution is unknown will not impact our ability to optimize the resulting model objective because this term will cancel.

**Denominator** In the Dreamer derivation, $p(\vec{o},\vec{r}|\vec{a})$ is treated as a constant and eliminated. We instead chose to group it together with the encoder distribution term and re-factor it.

We recognize that the denominator $e_\phi(\vec{\tilde{s}}|\vec{o},\vec{a})\,p(\vec{o},\vec{r}|\vec{a})$ is the *true* joint distribution over $\vec{o}$, $\vec{r}$, $\vec{\tilde{s}}$ given $\vec{a}$. This is because $e_\phi$ describes our chosen state representation, not an approximation to some underlying distribution over latent states, so $p(\vec{\tilde{s}}|\vec{o},\vec{a}) = e_\phi(\vec{\tilde{s}}|\vec{o},\vec{a})$. Since the encoder never observes rewards, conditioning on $\vec{r}$ does not change this. Therefore

$$p(\vec{o}, \vec{r}, \vec{\tilde{s}}|\vec{a}) = p(\vec{\tilde{s}}|\vec{o}, \vec{r}, \vec{a})\, p(\vec{o}, \vec{r}|\vec{a}) = e_\phi(\vec{\tilde{s}}|\vec{o}, \vec{a})\, p(\vec{o}, \vec{r}|\vec{a}). \tag{7}$$

We factor this joint in three steps. First, group variables by timestep:

$$p(\vec{o}, \vec{r}, \vec{\tilde{s}}|\vec{a}) = \prod_{t=1}^{T} p(o_t, r_t, \tilde{s}_t \mid o_{<t}, r_{<t}, \tilde{s}_{<t}, a_{\leq t}). \tag{8}$$

Second, apply the chain rule within each timestep:

$$= \prod_{t=1}^{T} p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, r_{<t}, a_{<t})\, p(o_t \mid \tilde{s}_{\leq t}, o_{<t}, r_{<t}, a_{<t})\, p(r_t \mid \tilde{s}_{\leq t}, o_{\leq t}, r_{<t}, a_{\leq t}). \tag{9}$$

Third, we drop redundant conditioning. Observations and state representations are conditionally independent of past rewards given past observations and actions. Because $\tilde{s}_{<t}$ is computed from $o_{<t}$ and $a_{<t}$ (plus independent encoder noise), $r_t$ is conditionally independent of $\tilde{s}_{\leq t}$ given $o_{\leq t}$ and $a_{\leq t}$. This gives us

$$p(\vec{o}, \vec{r}, \vec{\tilde{s}}|\vec{a}) = \prod_{t=1}^{T} p(o_t|\tilde{s}_{\leq t}, o_{<t}, a_{<t}) p(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t}) p(r_t|o_{\leq t}, a_{\leq t}). \tag{10}$$

Substituting equation 6 and equation 10 back into equation 5 gives us

$$\mathbb{E}\left[\log \frac{\hat{p}(\vec{o}, \vec{r}|\vec{a})}{p(\vec{o}, \vec{r}|\vec{a})}\right] \geq \mathbb{E}\left[\log \frac{\prod_{t=1}^{T} \cancel{p(o_t|\tilde{s}_{\leq t}, o_{<t}, a_{\leq t})} \hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_{t-1}) \hat{p}_\theta(r_t|\tilde{s}_t, a_t)}{\prod_{t=1}^{T} \cancel{p(o_t|\tilde{s}_{\leq t}, o_{<t}, a_{<t})} p(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t}) p(r_t|o_{\leq t}, a_{\leq t})}\right] \tag{11}$$

$$= \mathbb{E}\left[\sum \log \frac{\hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_{t-1})}{p(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t})} + \log \hat{p}_\theta(r_t|\tilde{s}_t, a_t)\right] + \mathcal{H}(p(\vec{r}|\vec{o}, \vec{a})). \tag{12}$$

Because $\mathcal{H}(p(\vec{r}|\vec{o}, \vec{a}))$ does not depend on our model parameters, it can be treated as a constant and ignored. $p(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t})$ is unknown; however, for a sufficiently flexible class of function approximators $\Psi$, by Gibbs' inequality, it is given by

$$p(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t}) = \hat{p}_{\psi*}(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t}) = \operatorname*{argmax}_{\psi \in \Psi} \mathbb{E}\left[\log \hat{p}_\psi(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t})\right]. \tag{13}$$

Substituting this term into equation 12 yields our final model objective,

$$\mathcal{L}(\theta, \phi) = \mathbb{E}\left[\sum \log \frac{\hat{p}_\theta(\tilde{s}_t|\tilde{s}_{t-1}, a_{t-1})}{\hat{p}_{\psi*}(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t})} + \log \hat{p}_\theta(r_t|\tilde{s}_t, a_t)\right]. \tag{14}$$

We refer to $\hat{p}_{\psi*}$ as the *adversary network* because optimizing it *reduces* $\mathcal{L}$, placing $\psi$ in an adversarial game with $\phi$ and $\theta$. Similarly, we refer to $\mathbb{E}\left[\sum_t \log \frac{1}{\hat{p}_{\psi*}(\tilde{s}_t|\tilde{s}_{<t}, o_{<t}, a_{<t})}\right]$ as the *adversarial objective component*. In practice, to ensure that the adversary network never underperforms the dynamics model in predicting $\tilde{s}_t$, we clip each per-dimension term $\mathbb{E}\left[\log \frac{\hat{p}_\theta(\tilde{s}_t^{(i)}|\tilde{s}_{t-1}, a_{t-1})}{\hat{p}_{\psi*}(\tilde{s}_t^{(i)}|\tilde{s}_{<t}, o_{<t}, a_{<t})}\right]$ to be non-positive. This modification does not alter the theoretical objective, since the term is upper-bounded by zero, but it improves stability in practice when the adversary is slow to converge.

# B PROOFS

## B.1 PROOF OF LEMMA 3.1

Here we restate lemma 3.1 more rigorously:

**Lemma 3.1** (Encoder-induced equivalence and bisimulation).

*Let $H_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$ denote the history of observations and actions up to time $t$. Let $\tilde{s}_t = e_\phi(H_t)$ be the abstract state produced by an encoder $e_\phi$. The encoder induces an equivalence relation $\sim_{e_\phi}$ on histories, where*

$$H_t^i \sim_{e_\phi} H_t^j \quad \Longleftrightarrow \quad e_\phi(H_t^i) = e_\phi(H_t^j).$$

*Suppose $e_\phi$ satisfies the following sufficiency conditions for all $t$ and actions $a_t$:*

$$\textbf{(Dynamics sufficiency)} \quad \tilde{s}_{t+1} \perp\!\!\!\perp H_t \mid \tilde{s}_t, a_t, \tag{15}$$

$$\textbf{(Reward sufficiency)} \quad r_t \perp\!\!\!\perp H_t \mid \tilde{s}_t, a_t. \tag{16}$$

*Then the equivalence relation $\sim_{e_\phi}$ is a bisimulation relation on histories, i.e., for all $H_t^i \sim_{e_\phi} H_t^j$,*

$$P_{\tilde{s}'|H,a}(\tilde{s}_{t+1}|H_t^i, a_t) = P_{\tilde{s}'|H,a}(\tilde{s}_{t+1}|H_t^j, a_t) \quad \forall a_t \in \mathcal{A}, \quad \tilde{s}_{t+1} \in \tilde{S}, \tag{17}$$

$$P_{r|H,a}(r_t|H_t^i, a_t) = P_{r|H,a}(r_t|H_t^j, a_t) \quad \forall a_t \in \mathcal{A}, \tag{18}$$

where $P_{r|H,a}$ is the reward distribution. Note that, while classical bisimulation is often defined in terms of deterministic reward functions $R(H, a)$, this is equivalent to requiring equality of the degenerate reward distribution $P_{r|H,a}(r|H, a) = \delta(r - R(H, a))$. Our formulation in terms of $P_{r|H,a}$ generalizes this definition to stochastic rewards and is thus more naturally aligned with the probabilistic framework of variational inference.

*Proof*: For all $H_t^i \sim_{e_\phi} H_t^j$, for all $\tilde{s}_{t+1} \in \tilde{\mathcal{S}}$, and for all $a_t \in \mathcal{A}$, we have

$$\begin{aligned} P(\tilde{s}_{t+1}|H_t^i, a_t) &= P(\tilde{s}_{t+1}|H_t^i, e_\phi(H_t^i), a_t) \quad (\text{Because } \tilde{s}_{t+1} \perp\!\!\!\perp e_\phi(H_t^i)|H_t^i) && (19)\\ &= P(\tilde{s}_{t+1}|e_\phi(H_t^i), a_t) \quad (\text{By equation 15}) && (20)\\ & && (21)\end{aligned}$$

We also have

$$\begin{aligned} P(\tilde{s}_{t+1}|H_t^j, a_t) &= P(\tilde{s}_{t+1}|H_t^j, e_\phi(H_t^j), a_t) \quad (\text{Because } \tilde{s}_{t+1} \perp\!\!\!\perp e_\phi(H_t^i)|H_t^i) && (22)\\ &= P(\tilde{s}_{t+1}|e_\phi(H_t^j), a_t) \quad (\text{By equation 15}) && (23)\\ &= P(\tilde{s}_{t+1}|e_\phi(H_t^i), a_t) \quad (\text{Because } e_\phi(H_t^j) = e_\phi(H_t^i)) && (24)\\ &= P(\tilde{s}_{t+1}|H_t^i, a_t), && (25)\end{aligned}$$

which is exactly the condition for bisimulation specified by equation 17.

Similarly for reward, we have

$$\begin{aligned} P(r_t|H_t^i, a_t) &= P(r_t|H_t^i, e_\phi(H_t^i), a_t) \quad (\text{Because } r_t \perp\!\!\!\perp e_\phi(H_t^i)|H_t^i) && (26)\\ &= P(r_t|e_\phi(H_t^i), a_t) \quad (\text{By equation 16}) && (27)\\ & && (28)\end{aligned}$$

and

$$\begin{aligned} P(r_t|H_t^j, a_t) &= P(r_t|H_t^j, e_\phi(H_t^j), a_t) \quad (\text{Because } r_t \perp\!\!\!\perp e_\phi(H_t^i)|H_t^i) && (29)\\ &= P(r_t|e_\phi(H_t^j), a_t) \quad (\text{By equation 16}) && (30)\\ &= P(r_t|e_\phi(H_t^i), a_t) \quad (\text{Because } e_\phi(H_t^j) = e_\phi(H_t^i)) && (31)\\ &= P(r_t|H_t^i, a_t), && (32)\end{aligned}$$

which is exactly the condition for bisimulation specified by equation 18. Therefore, if $e_\phi$ satisfies equation 15 and equation 16, then the equivalence relation induced by $e_\phi$, $\sim_{e_\phi}$, is a bisimulation relation. □

### B.2 Proof of Theorem 3.2

**Theorem 3.2** (Optimality of $\mathcal{L}$ yields bisimulation). *Suppose the model classes for the latent dynamics and reward distributions are sufficiently expressive. Then any global maximizer of the objective in equation 2 satisfies*

$$\hat{p}_\theta(\tilde{s}_t \mid \tilde{s}_{t-1}, a_{t-1}) = p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}), \qquad \hat{p}_\theta(r_t \mid \tilde{s}_t, a_t) = p(r_t \mid o_{\leq t}, a_{\leq t}),$$

*for all $t$. These equalities imply the latent dynamics and reward sufficiency conditions, and therefore, by Lemma 3.1, the encoder-induced partition of histories is a bisimulation.*

*Proof.* By adding/subtracting the constant $\mathbb{E}[\log p(r_t \mid o_{\leq t}, a_{\leq t})]$, $\mathcal{L}$ can be re-written as

$$\mathcal{L} = \text{const} - \mathbb{E}\left[\sum_t D_{\mathrm{KL}}(p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}) \parallel \hat{p}_\theta(\tilde{s}_t \mid \tilde{s}_{t-1}, a_{t-1})) + D_{\mathrm{KL}}(p(r_t \mid o_{\leq t}, a_t) \parallel \hat{p}_\theta(r_t \mid \tilde{s}_t, a_t))\right].$$

By nonnegativity of KL divergence, $\mathcal{L}$ is maximized if and only if both KL terms vanish almost surely, i.e.

$$\hat{p}_\theta(\tilde{s}_t \mid \tilde{s}_{t-1}, a_{t-1}) = p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}), \qquad \hat{p}_\theta(r_t \mid \tilde{s}_t, a_t) = p(r_t \mid o_{\leq t}, a_t).$$

These identities assert that the history influences the true conditionals only through $(\tilde{s}_t, a_t)$, which is exactly the latent dynamics and reward sufficiency conditions. Lemma 3.1 then guarantees that the encoder-induced partition is a bisimulation. □

### B.3 Proof that gradients with respect to the explicit occurrence of $\phi$ in the adversarial objective vanish

We now justify the claim made in Sec. 3 that $\psi^*$ can be treated as fixed when computing $\nabla_\theta \mathcal{L}$ and $\nabla_\phi \mathcal{L}$. Specifically, we show that the expected gradient with respect to the *explicit* occurrence of $\phi$ in the adversarial objective vanishes. This property has been used by prior work employing adversarial predictors of latent variables; we follow a similar argument to that of Mescheder et al. (2017):

$$\frac{\partial}{\partial \phi}\mathbb{E}\left[\sum_t \log \hat{p}_{\psi^*}(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t})\right] \tag{33}$$

$$= \frac{\partial}{\partial \phi}\mathbb{E}\left[\sum_t \log p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi)\right] \quad \text{(By equation 13).} \tag{34}$$

The expectation in equation 34 is over all of $o_{<t}, a_{<t}, \tilde{s}_{<t}, o_t, \tilde{s}_t$. By the law of iterated expectations, we split it into an expectation over the current-timestep variables, conditioned on the past, and an outer expectation over the past:

$$\frac{\partial}{\partial \phi}\sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}}\left[\mathbb{E}_{o_t, \tilde{s}_t | o_{<t}, a_{<t}, \tilde{s}_{<t}}[\log p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi)]\right].$$

Writing the inner expectation as an integral over the joint density $p(o_t, \tilde{s}_t \mid o_{<t}, a_{<t}, \tilde{s}_{<t})$, and factoring that density as $p(o_t \mid o_{<t}, a_{<t}) \cdot e_\phi(\tilde{s}_t \mid o_{\leq t}, a_{<t})$, yields

$$= \frac{\partial}{\partial \phi}\sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}}\left[\int p(o_t | o_{<t}, a< t) e_\phi(\tilde{s}_t | o_{\leq t}, a_{<t}) \log p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) do_t d\tilde{s}_t\right]. \tag{35}$$

This same joint density can also be factored the other way, which will let us conveniently integrate out the $o_t$ term next. Factoring it as $p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) \cdot p(o_t \mid \tilde{s}_{\leq t}, o_{<t}, a_{<t})$ yields

$$= \frac{\partial}{\partial \phi} \sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}} \left[ \int p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) p(o_t | \tilde{s}_{\leq t}, o_{<t}, a < t) \log p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) do_t d\tilde{s}_t \right]. \tag{36}$$

Since $\int p(o_t \mid \tilde{s}_{\leq t}, o_{<t}, a_{<t})\, do_t = 1$, the $o_t$ integral drops out entirely, yielding

$$= \frac{\partial}{\partial \phi} \sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}} \left[ \int p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) \log p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) d\tilde{s}_t \right]. \tag{37}$$

By the Leibniz integral rule, we move the derivative inside the integral, yielding

$$= \sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}} \left[ \int p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) \frac{\partial}{\partial \phi} \log p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) d\tilde{s}_t \right]. \tag{38}$$

Applying the log-derivative identity $\frac{\partial}{\partial \phi} \log p = \frac{\nabla_\phi p}{p}$ yields

$$= \sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}} \left[ \int p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) \frac{\nabla_\phi p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi)}{p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi)} d\tilde{s}_t \right]. \tag{39}$$

The density in the numerator cancels against the denominator, and moving $\nabla_\phi$ outside the integral yields

$$= \sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}} \left[ \nabla_\phi \int p(\tilde{s}_t | \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi) d\tilde{s}_t \right]. \tag{40}$$

Since $\int p(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t}, \phi)\, d\tilde{s}_t = 1$, and the gradient of a constant vanishes, this becomes

$$= \sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}} \left[ \nabla_\phi 1 \right] = \sum_t \mathbb{E}_{o_{<t}, a_{<t}, \tilde{s}_{<t}} \left[ 0 \right] = 0. \tag{41}$$

# C Implementation Details

## C.1 Model Architecture and Policy Learning

Our model consists of four main components: an observation encoder, a latent dynamics model, a reward model, and an adversary network.

**Observation Encoder.** The encoder maps the agent's history up to time $t$, $(o_{\leq t}, a_{<t})$, into a distribution over the latent state $\tilde{s}_t$. The architecture is composed of a convolutional neural network (CNN) followed by a GRU. Unlike Dreamer, which shares a GRU between the encoder and dynamics model, we use a dedicated GRU for the encoder. The encoder outputs a mean vector $\mu_t$, normalized to have $\ell_2$ norm equal to one. We then sample $\tilde{s}_t$ from a Gaussian distribution with mean $\mu_t$ and fixed standard deviation $0.01$. This normalization and fixing of the standard deviation introduces an information bottleneck: only a fixed amount of information can flow through the latent state, encouraging abstraction.

**Latent Dynamics Model.** The dynamics model resembles that of Dreamer, with both stochastic and deterministic components. It maintains a deterministic hidden state $h_t$ via a GRU and outputs the parameters of a Gaussian distribution over $\tilde{s}_t$. The mean is again normalized to unit norm, while the variance is learned but constrained to be isotropic (identical across latent dimensions).

**Reward Model.** Our reward model differs slightly from Dreamer's: it conditions only on the stochastic latent state $\tilde{s}_t$ (and the current action $a_t$), not on the deterministic hidden state $h_t$. It outputs the parameters of a Gaussian distribution over rewards $r_t$.

**Adversary Network.** Finally, the adversary network shares a similar architecture with the encoder (CNN + GRU). The adversary additionally conditions on $s_t$ through an auxiliary input before the

GRU, and predicts the *next* latent state $\tilde{s}_{t+1}$ instead of the current one. Similar to the dynamics model, the adversary network outputs a unit-norm mean and an isotropic variance vector.

**Actor.** The actor is an MLP that conditions on the concatenation of the stochastic latent $\tilde{s}_t$ and the dynamics model's deterministic state $h_t$, and outputs a tanh-transformed Gaussian over actions. As in DreamerV1, it is trained by backpropagating $\lambda$-returns through imagined rollouts of the learned dynamics ($H = 15$), started from every latent state in the sampled batch. Unlike DreamerV1, the actor loss includes an entropy bonus with fixed coefficient 1, as in DreamerV2.

**Critic.** The critic is an MLP that takes the same input as the actor and outputs a scalar value estimate. It is trained by squared-error regression onto $\lambda$-returns ($\lambda = 0.95$, $\gamma = 0.99$) computed from the reward model's mean prediction. Unlike DreamerV1, bootstrap values come from a target critic, as in DreamerV2, which is updated by a hard copy every 50 updates. Actor and critic gradients are never applied to world-model parameters.

### C.2 Model Training Algorithm

**Algorithm 1:** VIBES Model Training Algorithm

**Inputs:** Replay buffer $\mathcal{D}$ with sequences $(o_{0:T}, a_{0:T-1}, r_{0:T-1})$

**Models:** Encoder $e_\phi$: $(o_{\le t}, a_{<t}) \mapsto \tilde{s}_t \sim \mathcal{N}(\mu_t, \sigma_e^2 I)$ with $\|\mu_t\|_2 = 1$ and fixed $\sigma_e^2$;
Deterministic state update $h_t = f_\theta(h_{t-1}, \tilde{s}_{t-1}, a_{t-1})$ (GRU);
Dynamics $p_\theta(\tilde{s}_t \mid h_{t-1}, \tilde{s}_{t-1}, a_{t-1})$: Gaussian with unit-norm mean, isotropic variance;
Reward $p_\theta(r_t \mid \tilde{s}_t, a_t)$: Gaussian;
Adversary $\hat{p}_\psi(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t})$: Gaussian with unit-norm mean, isotropic variance

**Hyperparams:** Learning rates $\eta_\theta, \eta_\phi, \eta_\psi$ with $\eta_\psi > \eta_\phi$; batch size $B$; sequence length $T$

**Initialize:** Randomly initialize $\phi, \theta, \psi$; set $h_0 = \mathbf{0}$

**for** *training steps* $= 1, 2, \ldots$ **do**
- Sample $B$ trajectory chunks of length $T$ from $\mathcal{D}$
- `// Encode latents with reparameterization`
- **for** $t = 0$ **to** $T$ **do**
  - $(\mu_t, \sigma_e^2) \leftarrow e_\phi(o_{\le t}, a_{<t})$, enforce $\|\mu_t\|_2 \leftarrow 1$
  - Sample $\epsilon_t \sim \mathcal{N}(0, I)$, $\tilde{s}_t \leftarrow \mu_t + \sigma_e \odot \epsilon_t$
- `// Update deterministic hidden state, then compute latent dynamics and reward`
- **for** $t = 1$ **to** $T$ **do**
  - $h_t \leftarrow f_\theta(h_{t-1}, \tilde{s}_{t-1}, a_{t-1})$ `// GRU update`
  - Compute $p_\theta(\tilde{s}_t \mid h_{t-1}, \tilde{s}_{t-1}, a_{t-1})$ `// latent transition`
  - Compute $p_\theta(r_t \mid \tilde{s}_t, a_t)$ `// reward model`
- `// Adversary predictions (condition on full history)`
- Compute $\hat{p}_\psi(\tilde{s}_t \mid \tilde{s}_{<t}, o_{<t}, a_{<t})$ for $t = 1{:}T$
- `// Per-dimension non-positivity clipping of the ratio term`
- **for** $t = 1$ **to** $T$ **do**
  - **for** *each latent dim* $i$ **do**
    - $\ell_t^{(i)} \leftarrow \sum_{b=1}^{B} \log \dfrac{p_\theta(\tilde{s}_{t,b}^{(i)} \mid h_{t-1,b}, \tilde{s}_{t-1,b}, a_{t-1,b})}{\hat{p}_\psi(\tilde{s}_{t,b}^{(i)} \mid \tilde{s}_{<t,b}, o_{<t,b}, a_{<t,b})}$
    - $\tilde{\ell}_t^{(i)} \leftarrow \min(\ell_t^{(i)}, 0)$ `// clip so term is non-positive`
- `// Objective (minibatch average)`
- $\mathcal{L} \leftarrow \dfrac{1}{TB} \displaystyle\sum_{t=1}^{T} \left[ \sum_i \tilde{\ell}_t^{(i)} + \sum_{b=1}^{B} \log p_\theta(r_{t,b} \mid \tilde{s}_{t,b}, a_{t,b}) \right]$
- `// Outer ascent updates (treat` $\psi^*$ `as fixed; Appendix B.3)`
- $\theta \leftarrow \theta + \eta_\theta \nabla_\theta \mathcal{L}$; $\phi \leftarrow \phi + \eta_\phi \nabla_\phi \mathcal{L}$
- `// Inner ascent update on adversary (maximize equation 13)`
- $\mathcal{J}_\psi \leftarrow \dfrac{1}{TB} \displaystyle\sum_{b=1}^{B} \sum_{t=1}^{T} \log \hat{p}_\psi(\tilde{s}_{t,b} \mid \tilde{s}_{<t,b}, o_{<t,b}, a_{<t,b})$
- $\psi \leftarrow \psi + \eta_\psi \nabla_\psi \mathcal{J}_\psi$

### C.3 Hyperparameters

Table 3 lists the hyperparameters used by VIBES.

| Name | Value |
|---|---|
| *Encoder* | |
| Latent dimension $s_{\text{dim}}$ | 50 |
| Encoder standard deviation $\sigma_e$ | 0.01 (fixed) |
| Mean normalization | unit $\ell_2$ norm |
| Encoder GRU layers | 1 |
| Encoder layer norm | yes |
| Encoder learning rate | $10^{-4}$ |
| *Dynamics model* | |
| Hidden dimension | 200 |
| GRU layers | 1 |
| Variance | isotropic (shared across dimensions) |
| Model learning rate | $5 \times 10^{-4}$ |
| *Adversary* | |
| GRU layers | 1 |
| Variance | isotropic, per-dimension (not shared) |
| Adversary updates per model update | 1 |
| Adversary learning rate | $3 \times 10^{-4}$ |
| *Reward model* | |
| Reward normalization | yes |
| Minimum reward std | 0.1 |
| *Loss coefficients* | |
| Adversarial loss coefficient | 0.01 |
| Dynamics loss coefficient | 0.01 |
| Reward loss coefficient | 1 |
| *Actor and critic* | |
| Actor hidden dimension | 200 |
| Actor layers | 4 |
| Actor learning rate | $10^{-4}$ |
| Entropy coefficient $\alpha$ | 1 (fixed) |
| Imagination horizon $H$ | 15 |
| Critic hidden dimension | 200 |
| Critic layers | 3 |
| Critic learning rate | $10^{-4}$ |
| Discount $\gamma$ | 0.99 |
| Return lambda $\lambda$ | 0.95 |
| Target critic update interval | 50 (hard copy) |
| *Training* | |
| Sequence length $L$ | 50 |
| Batch size | 10 |
| Gradient steps per iteration | 100 |
| Prefill episodes | 5 |
| Image size | $84 \times 84$ |
| Environment steps | 500K |

Table 3: VIBES hyperparameters

### C.4 DCS CONFIGURATION

We evaluate on the Distracting Control Suite (Stone et al., 2021) at the easy difficulty tier with dynamic distractors, and enable all three distractor types (background, camera, and color) simultaneously. Background distractors are drawn from the DAVIS dataset: following the package's easy-tier default, four videos are drawn from the DAVIS training split for training and four from the validation split for evaluation, so the evaluation distractors are held out and disjoint from those seen during training. Observations are $84 \times 84$ RGB images. Action repeat is 2 for Walker-walk and Finger-spin, 4 for Cheetah-run, Reacher-easy, and Ball-in-cup-catch, and 8 for Cartpole-swingup, giving episode lengths of $1000/\text{action repeat}$ steps. All methods are trained for 500K environment steps.

### C.5 Realistic ManiSkill Configuration

We use the released HRSSM Realistic ManiSkill environment and its pool of 88 Matterport3D (Chang et al., 2017) indoor scenes. Agents receive $64 \times 64$ RGB observations and use the seven-dimensional `pd_ee_delta_pose` controller. We use the released dense rewards and an action repeat of one. Episodes are truncated after 100 actions, and VIBES does not learn a continuation model. Lift Cube uses the released reward and fixed initial state. Turn Faucet uses faucet model 5021. Training and evaluation sample the same scene pool.

The comparison between VIBES and DreamerV3 uses six training seeds and ten stochastic evaluation episodes per seed. Both methods receive about 100K learner updates, corresponding to 100K environment interactions for VIBES and 203K for DreamerV3. Parameter counts exclude target and EMA copies.

The rows marked with † in Table 2 are taken from HRSSM Table 2: TIA, Denoised MDP, RePo, and HRSSM. We ran the remaining experiments. The DreamerV3-1M and DreamerV3-12M rows use the official DreamerV3 implementation, and the VIBES row uses our implementation.

### C.6 Swimmer Configuration

We build on the *Swimmer* environment from the DeepMind Control Suite (Tassa et al., 2018), a procedurally generated $k$-link planar snake robot moving in a viscous fluid, and set $k = 100$. The physical state is 204-dimensional: the 3 position coordinates of the head and the 99 joint angles, plus their velocities. The agent receives a 401-dimensional vector observation, consisting of the 99 joint angles, the 2-dimensional nose-to-target vector, and the 300-dimensional body velocities; no images are rendered. The target location is fixed across episodes, and the reward decays smoothly with the nose-to-target distance according to a Lorentzian function.

Directly controlling a 99-dimensional action space is impractical and not the focus of this study. Instead, the agent outputs the four parameters $[A,\, c_1,\, c_2,\, c_3]$ of a spatiotemporal sinusoid, and the torque applied to joint $i$ at timestep $t$ is $\tau_i(t) = A \cdot \sin(c_1 i + c_2 t) + c_3$. Each parameter is scaled for stable control. We use an action repeat of 10 to prevent the agent from changing the sinusoid parameters at unrealistically high frequencies.

VIBES, Recon. Obj., and DreamerV3 are each trained for 500K environment steps with 4 seeds.

## D Related Work

Several classes of representation-learning methods have been proposed to learn useful representations of high-dimensional observations. We review the classes most relevant to our setting and discuss the key algorithms that belong to each.

**1. Reconstruction-Based Methods.** These methods encode latents that can be decoded back into the original observation. Because the objective rewards faithful reconstruction of every pixel, these representations have no incentive to discard distractors. As a result, they perform poorly in environments with heavy distractor content. Representative algorithms are PlaNet (Hafner et al., 2019), and the Dreamer family of algorithms (Hafner et al., 2020; 2021; 2025a;b).

**2. Contrastive and Self-Supervised Methods.** These methods define an anchor observation, a positive example, and one or more negative examples, and train a lower-dimensional encoding via a contrastive loss that pulls the anchor and positive together while pushing the anchor and negatives apart. The choice of what counts as a positive determines what the representation ends up invariant to, and this choice can lead to problematic outcomes. We look at three representative examples:

CURL (Laskin et al., 2020a) (Contrastive Unsupervised Representations for Reinforcement Learning) treats two randomly-cropped versions of the same observation as a matching pair, and trains the encoder to recognize them as the same while telling them apart from crops of other observations.

CoRe (Srivastava et al., 2021) (Contrastive Recurrent State-Space Model) defines its matching pairs temporally instead: the anchor is the real encoding of the current observation, and the positive is the predicted latent for that step — computed from the past latent and past action, before the real observation is seen — with negatives drawn from other real and predicted encodings in the same training batch. Since the positive is generated by predicting forward from the past latent and action, what survives in the latent is whatever is temporally predictable.

RAD (Laskin et al., 2020b) (Reinforcement Learning with Augmented Data) drops the contrastive loss, and the whole notion of a matching pair, entirely. It simply replaces the clean observation with a randomly-distorted one and trains the ordinary RL objective on that. Because the same underlying transition is repeatedly redrawn with a different random distortion applied each time, the network is pushed toward invariance to that distortion simply by having to fit one consistent output despite the varying input.

In all three cases, what stays in the latent is decided by something other than the reward: an augmentation choice, or a notion of temporal predictability, not by whether the discarded information actually mattered for the task. These methods may discard information the agent needed, or just as easily retain information that was irrelevant.

**3. Explicit Factorization.** These methods split the latent into separately labeled pieces and only hand the policy the piece that's supposed to matter. We look at two representative examples:

TIA (Fu et al., 2021) (Task Informed Abstractions) splits the latent into a task feature and a distractor feature. Both features cooperate to reconstruct the full image. The policy only sees the task feature. This split assumes the task and distractor evolve completely independently; the authors call this assumption "too strong" themselves, and it breaks whenever a distractor actually interacts with the task instead of sitting apart from it.

Denoised MDP (Wang et al., 2022) generalizes this to two axes instead of one (controllable vs. uncontrollable, crossed with reward-relevant vs. not), keeping only the controllable-and-reward-relevant quadrant. It only knows how to handle a small set of pre-defined dependency patterns between signal and noise. If a distractor and the task signal share a common cause, something driving both at once, that case falls outside what the method handles, and it has no clean way to separate them.

Both methods build a fixed factorization structure ahead of time and rely on the environment actually matching it. When it does, the split works; when task and distractor interact instead of evolving independently, both degrade, with no mechanism to detect or correct for that at run time.

**4. Bisimulation Metrics.** These methods target the same signal we do — reward and dynamics. But they get there through approximations, not a proof that the result is a bisimulation relation. We look at two representative examples:

DBC (Zhang et al., 2021) (Deep Bisimulation for Control) trains an encoder so the distance between two latent codes matches a bisimulation-metric target: reward difference plus how differently the two states are predicted to evolve. The metric is defined recursively: calculating the distance between two states' predicted next-states requires already knowing the bisimulation distance between the current states, the very thing being solved for. DBC's implementation substitutes this ground distance with Euclidean distance instead, an incorrect approximation that shows up in practice: DBC shows poor performance when tested on out-of-distribution distractors (Nguyen et al., 2021).

HRSSM (Sun et al., 2024) (Hybrid Recurrent State-Space Model) combines a masking strategy with a bisimulation loss on top of DreamerV3. HRSSM's own analysis admits each half is individually insufficient. Masking alone can accidentally hide task-critical detail along with the distractor. The bisimulation loss alone isn't guaranteed to filter anything if the dynamics model it depends on is imperfect. The two are combined on the empirical hope that they cover each other's gaps, not a proof that they do. RePo (Zhu et al., 2023) introduced Realistic ManiSkill, and HRSSM (Sun et al., 2024) later evaluated its model on the same benchmark. Section 4.2 adds comparisons with the official DreamerV3 1M and 12M configurations.

Note: TIA, HRSSM, and CURL each rely on an image-specific operation — pixel-space mask composition, spatio-temporal masking, and random crop, respectively — so none of them can be adapted to vector-based inputs the way our method can.

In contrast, VIBES does not approximate a bisimulation distance. Its adversarial objective is proven (Theorem 3.2) to induce an exact bisimulation relation over histories at every global optimum of the objective. It requires no domain-specific augmentation heuristics and no pixel reconstruction. The objective keeps everything needed to predict reward and dynamics, and the encoder's bottleneck pushes it to drop the rest.

## E LATENT SPACE ANALYSIS

This appendix shows the full nearest-neighbor results summarized in Section 4.4. Figure 6 shows all 20 Reacher-easy queries. Figures 7 and 8 show 20 Lift Cube queries each for VIBES and DreamerV3-12M. In every figure, the leftmost image in a row is the query, and the remaining five are its nearest neighbors in latent space under standardized L2 distance, excluding frames from the query's own episode.

## F LLM USAGE

In this work, we used generative AI coding assistants to implement competing baseline algorithms and to write the code for some of our experiments, including scripts that run them and format the resulting data. The authors read and verified the baseline codebases and checked all data presented in the paper against the experiment logs. Coding assistants also helped compute the learned-parameter counts in Table 2, and the authors verified the counting methodology. We did not use generative AI tools to develop the theoretical framework, formulate mathematical claims, write proofs, propose or refine hypotheses, design or provide feedback on our research methodology or experiments, or interpret results, and the implementation of VIBES itself was written without AI assistance. Generating datasets, translation, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to draft and edit parts of the paper for readability and concision, write figure captions and appendix text, format LaTeX tables and references, and locate and verify details of prior work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.



Figure 6: **Nearest-neighbor analysis in latent space (all 20 queries).** For each of 20 query observations (left-most image in each row), we display the five nearest neighbors in the learned latent space beneath it (standardized $L_2$ distance). The visualization shows invariance to visual distractors and clustering of rotationally symmetric arm–goal configurations.

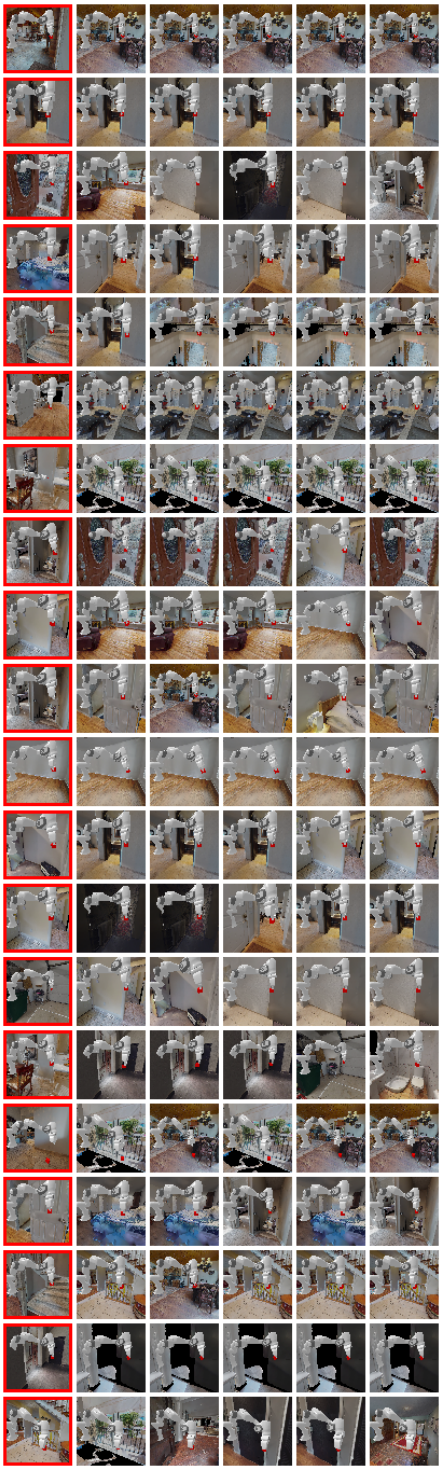


Figure 7: **Nearest latent neighbors on Lift Cube for VIBES (all 20 queries).** Each row shows a query frame (outlined in red) followed by its five nearest neighbors in latent space. Neighbors match the query's gripper and cube configuration but mostly come from different scenes.



Figure 8: **Nearest latent neighbors on Lift Cube for DreamerV3-12M (all 20 queries).** Layout as in Figure 7. Neighbors frequently share the query's scene, even when the gripper and cube configuration differ.