BRICKS-WM: BUILDING REUSABLE DYNAMICS NETWORKS FOR COMPOSITIONAL WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning (MBRL) has achieved remarkable success in continuous control by leveraging latent world models. However, monolithic world models couple distinct dynamics in a single transition dynamics model, making it difficult to reuse learned dynamics separately from the original predictor. To address this, we introduce **BRICKS-WM** (***B**uilding **R**eusable dynam**IC**s networ**KS** for compositional **W**orld **M**odels*), a framework for composing world models from reusable dynamics modules. Motivated by the insight that the physical world is composed of entities, we model global dynamics as a composition of distinct dynamics modules interacting via latent interfaces. As a minimal instantiation, we factorize the environment dynamics into an actuated Agent dynamics module and an external Background dynamics module, bridged by a learned latent interface. For reuse, we recompose the world model by combining a previously learned Background dynamics module with a newly learned Agent dynamics module through an adapted latent interface. Empirically, BRICKS-WM achieves control performance comparable to strong monolithic baselines when trained from scratch, and enables the reuse of background dynamics across scenarios.

1 INTRODUCTION

Model-based Reinforcement Learning (MBRL) has achieved remarkable success by learning latent world models that facilitate planning in low-dimensional spaces (Hafner et al., 2019; 2023; Schrittwieser et al., 2020; Hansen et al., 2024). By compressing high-dimensional sensory data into compact latent states, these agents can effectively imagine future outcomes to master complex control tasks. However, despite their performance, monolithic world models couple distinct dynamics in a single transition dynamics model, making it difficult to reuse learned dynamics separately from the original dynamics predictor. When a scenario changes, some learned dynamics may remain applicable, but their coupling to other parts of the monolithic dynamics model hinders their reuse in a new composition.

In this paper, we adopt a different perspective: *composing world models from reusable dynamics modules*. Driven by the insight that the physical world is composed of entities, we posit that global dynamics can be modeled as a composition of distinct dynamics modules interacting via latent interfaces. This view allows learned dynamics modules to be reused and recomposed across scenarios, analogous to the assembly of physical bricks. In this view, the key technical challenge is not merely to learn factorized representations or dynamics (Gmelin et al., 2023; Pore et al., 2024; Fu et al., 2021; Wang et al., 2022; Pan et al., 2022), but to learn dynamics modules whose interfaces support recomposition without breaking predictive consistency.

This modular objective aligns with broader trends in compositional representation learning and object-centric modeling (Locatello et al., 2020; Greff et al., 2019; Burgess et al., 2019; Engelcke et al., 2020; Lin et al., 2020; Kipf et al., 2022; Jiang et al., 2024) and with modularity for transfer in control (Devin et al., 2017; Huang et al., 2020; Gupta et al., 2022; He et al., 2025). However, existing object-centric world models for control tasks (Wang et al., 2025; Mosbach et al., 2025; Feng et al., 2025) typically remain functionally coupled: even when the latent state is factorized, the learned dynamics remain coupled within the original predictor, making it difficult to reuse and recompose individual dynamics modules in a new scenario. The closest modular mechanism approach is COMET (Lei et al., 2024), which promotes decomposition but does not directly reuse learned dynamics modules through recomposition across scenarios.

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.

As a foundational step towards this vision, we introduce **BRICKS-WM** (***B**uilding **R**eusable dynam**IC**s networ**KS** for compositional **W**orld **M**odels*). While the principle of modularity is general, this paper operationalizes it through a minimal yet meaningful instantiation: recomposing environment dynamics with an *Actuated Agent* dynamics module and an *External Background* dynamics module. We propose a learned latent interface for the interaction between the agent and background dynamics modules. The background transition conditions on its own state and the interface, through which the agent's influence is transmitted. This structure avoids direct dependence of the background transition on agent states or actions, providing a dynamics module that can be reused across scenarios. For reuse, we recompose a learned background dynamics module with newly learned agent dynamics and an adapted interface to construct a world model for a new scenario. The target agent learns interface inputs compatible with the reused background dynamics, turning the transfer problem into a protocol matching problem. While sharing conceptual similarities with modular transfer in policy learning (Devin et al., 2017; Huang et al., 2020), our work is distinguished by its focus on *modular reuse at the dynamics level*. Specifically, we propose reusing dynamics modules of the world model rather than transferring policies.

Our approach bridges the gap between structured representations and high-performance control. We evaluate both control performance from scratch and reuse across scenarios on the DeepMind Control Suite (Tassa et al., 2018) and CarRacing (Towers et al., 2025). Our specific contributions are as follows:

- **Framework for Modular Dynamics Composition**: We propose BRICKS-WM, a framework designed to compose structured world models from reusable dynamics modules. As a foundational instantiation, we factorize global dynamics into Agent and Background dynamics mediated by a latent interface.
- **SOTA-Level Control with Object-Centric Models from Scratch**: We demonstrate that object-centric structure need not compromise control efficacy. When trained from scratch, BRICKS-WM achieves performance on par with the monolithic state-of-the-art baselines, DreamerV3 (Hafner et al., 2023) and TD-MPC2 (Hansen et al., 2024). To the best of our knowledge, this is the ***first*** object-centric world model approach to match them on dynamic ***locomotion*** tasks solely from raw pixels, without ground-truth state supervision or any pretrained model.
- **First Demonstration of Modular Dynamics Reuse**: We introduce a mechanism that recomposes a learned background dynamics module with a newly learned agent module through an adapted latent interface. This enables the reuse of environmental dynamics without retraining the whole world model, yielding competitive control performance especially in related target settings.

2 PRELIMINARIES

Problem formulation. We model the environment as a Partially Observable Markov Decision Process (POMDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, \gamma)$, where Ω denotes the observation, or emission, kernel. At time t, the environment is in an unobserved ground-truth state $s_t^* \in \mathcal{S}$ and emits an observation $o_t \sim \Omega(\cdot \mid s_t^*)$. Given the history $\mathcal{H}_t \triangleq (o_{1:t}, a_{1:t-1})$, the agent executes an action $a_t \sim \pi(\cdot \mid \mathcal{H}_t)$, receives a reward $r_t \sim \mathcal{R}(\cdot \mid s_t^*, a_t)$, and causes the state to evolve as $s_{t+1}^* \sim \mathcal{T}(\cdot \mid s_t^*, a_t)$. The next observation is then generated as $o_{t+1} \sim \Omega(\cdot \mid s_{t+1}^*)$. We define a trajectory as $\tau = \{(o_t, a_t, r_t)\}_{t=1}^T$. The standard control objective is to learn a policy $\pi(a_t \mid \mathcal{H}_t)$ that maximizes the expected return $\mathbb{E}_{\pi,\mathcal{M}}\big[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\big]$.

Compositional Reuse Setting. Unlike standard monolithic modeling, we approach the problem through *modular composition*. We posit that global dynamics arise from interactions among distinct, reusable dynamics. While this principle is general, this paper focuses on a minimal instantiation: an *actuated agent* dynamics module and an *external background* dynamics module. We define a transfer scenario involving a source task $\mathcal{M}_{src}$ and a target task $\mathcal{M}_{tgt}$. Crucially, these tasks share a similar background dynamics module (governing external objects and terrain) but employ distinct agent modules or tasks. Our goal is to enable reuse of the background dynamics module learned from $\mathcal{M}_{src}$, recomposing it with an agent dynamics module trained from scratch in $\mathcal{M}_{tgt}$ under a standardized interaction protocol, thereby avoiding the need to re-learn the whole environment from scratch.

Recurrent State-Space Model. To handle high-dimensional observations in POMDPs, we leverage the Recurrent State-Space Model (RSSM) (Hafner et al., 2019). The RSSM maintains a deterministic

recurrent state h_t and a stochastic latent state s_t. The dynamics model contains a recurrent model $h_t = f_\theta(h_{t-1}, s_{t-1}, a_{t-1})$ and a stochastic transition model $p_\theta(s_t \mid h_t)$. For notational brevity, we omit explicit references to h_t in the subsequent sections. While effective, standard RSSMs learn a monolithic state representation s_t. In this work, we adopt the RSSM backbone but modify the internal factorization to enable the modular reuse described above.

3 Related Work

Latent World Models for Control. Model-based reinforcement learning (MBRL) has achieved significant success by learning latent dynamics models that facilitate planning in low-dimensional spaces. The Dreamer series (Hafner et al., 2020; 2021; 2023) establishes a strong baseline by learning deterministic and stochastic components of the environment dynamics. MuZero (Schrittwieser et al., 2020) achieves superhuman performance via planning with a learned model on Atari and board games such as Go, SimPLe (Kaiser et al., 2020) demonstrates sample-efficient model-based RL on Atari 100k, and TD-MPC (Hansen et al., 2022; 2024) learns task-oriented latent dynamics for short-horizon MPC in continuous control. Recent advances have also improved model expressivity via Transformer-based world models (Micheli et al., 2023; Zhang et al., 2023) and large-scale generative interactive environments (Bruce et al., 2024), as well as diffusion-based future modeling beyond step-by-step rollout (Ding et al., 2024) and diffusion-based world modeling that emphasizes high-fidelity visual prediction (Alonso et al., 2024). However, despite their diversity, these approaches typically learn a monolithic latent state that entangles the agent's dynamics with the environment. This coupling necessitates retraining the entire world model when the agent changes, whereas BRICKS-WM proposes a modular factorization to enable the direct reuse of environmental dynamics.

Object-Centric Representation and Structured Dynamics. To improve generalization and reasoning, a growing body of work focuses on decomposing scenes into distinct entities using object-centric inductive biases. Foundational methods such as Slot Attention (Locatello et al., 2020), IODINE (Greff et al., 2019), MONet (Burgess et al., 2019), SPACE (Lin et al., 2020), and GENESIS (Engelcke et al., 2020) demonstrate how to unsupervisedly segment static scenes into constituent slots. In the temporal domain, models like SAVi (Kipf et al., 2022) and PSB (Singh et al., 2024) extend these representations to video. Related lines of work model physical interactions with relational or interaction-network inductive biases (Battaglia et al., 2016; Kipf et al., 2018; Veerapaneni et al., 2019; Kipf et al., 2020; Goyal et al., 2021). Separately, COMET (Lei et al., 2024) studies modular world models via independent mechanisms. Recent methods have successfully integrated these structured representations into control, as seen in Dyn-O (Wang et al., 2025), SOLD (Mosbach et al., 2025), FIOC-WM (Feng et al., 2025) and OC-STORM (Zhang et al., 2026). However, while these approaches leverage factorization to improve physical reasoning, they typically learn dynamics that remain functionally coupled, preventing independent module reuse. Whereas COMET (Lei et al., 2024) reuses interaction primitives across object pairs, BRICKS-WM reuses the learned dynamics of an explicitly separated subsystem, such as the background, with its own latent state and transition model.

Modular Transfer and Cross-Embodiment Generalization. Cross-embodiment transfer has been extensively explored from the perspective of policy learning. Techniques such as modular policy decompositions (Devin et al., 2017) and Shared Modular Policies (Huang et al., 2020) enable transfer by separating task-specific goals from robot-specific actuation. More recently, large-scale pretraining has sought to learn universal controllers (Gupta et al., 2021; 2022) or cross-embodiment world models for specific domains like manipulation (He et al., 2025). Uniquely, BRICKS-WM targets modular world-model reuse. Rather than transferring a policy or learning a universal model from massive datasets, we propose a mechanism to freeze and transplant a pretrained background dynamics module. By relearning only agent-side components under a standardized protocol, we enable dynamics reuse in target tasks without the need for large-scale multi-robot data.

4 Method

We propose BRICKS-WM (***B**uilding **R**eusable dynam**IC**s networ**KS** for compositional **W**orld **M**odels*), a framework designed to enable *modular reuse* in model-based reinforcement learning. We first introduce the compositional viewpoint of world models that BRICKS-WM operationalizes (Section 4.1). Then we instantiate this view with two modules (Section 4.2), describe how the resulting composition is learned from pixels (Section 4.3), and explain how a learned module is reused (Section 4.4). Figure 1 shows the framework.

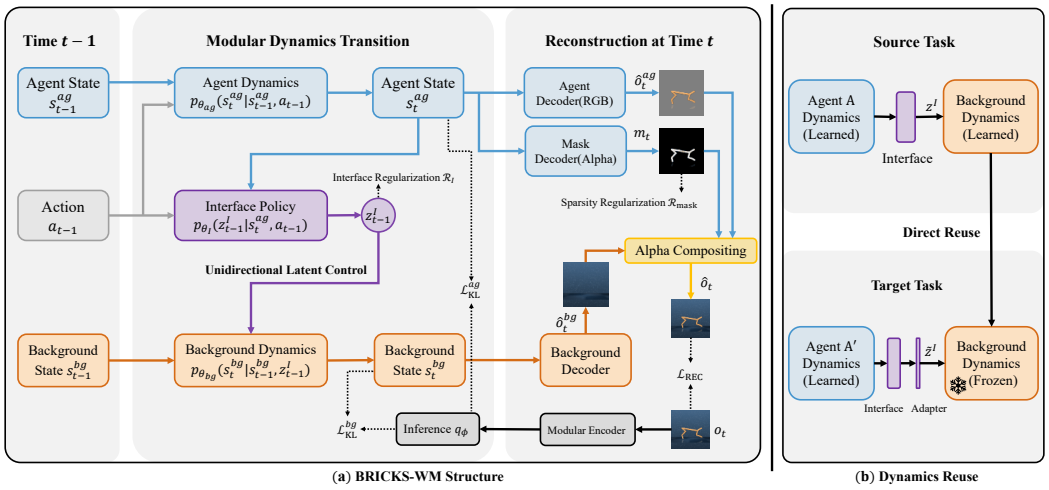


Figure 1: **Framework of BRICKS-WM.** (a) The overall architecture decomposes global dynamics into Agent and Background modules mediated by a latent interface. (b) The dynamics reuse logic, illustrating how a frozen background module is transferred to a new agent in a target task via an interface adapter.

4.1 World Models as Compositions of Dynamics Modules

A monolithic latent world model models the environment using a transition $p(s_t \mid s_{t-1}, a_{t-1})$ over a single latent state. When one part of the scenario changes, the whole transition has to be relearned, because nothing in a monolithic s_t indicates which part of the dynamics changed and which did not. BRICKS-WM instead treats the world model as a composition of dynamics modules. Each module owns a latent state and a transition over that state. The modules influence one another only through low-dimensional interface variables, and the global transition is the product of the module transitions chained by these interfaces. Based on this structure, three operations become available: (1) A learned module can be kept, with its transition frozen. (2) A partner module can be replaced and relearned. (3) The modules can be *recomposed* into a new world model for a new scenario. In the following parts, we study this principle through a minimal two-module instantiation.

4.2 A Minimal Agent–Background Instantiation

As a minimal two-module instantiation, we instantiate the composition with an actuated *Agent* module (what changes across tasks) and an external *Background* module comprising terrain, external objects and camera (what persists). We stress that the Background transition is not a nuisance to discard or ignore, but precisely the module we aim to preserve and reuse. At time t, the latent state is factorized into $s_t \triangleq (s^{ag}_t, s^{bg}_t)$, where $s^{ag}_t \in \mathbb{R}^{d_{ag}}$ encodes agent features and $s^{bg}_t \in \mathbb{R}^{d_{bg}}$ encodes background information. An interface latent $z^{\mathcal{I}}_t \in \mathbb{R}^{d_{\mathcal{I}}}$ mediates the agent-side influence on the background transition. Following the RSSM notation in the preliminaries, deterministic recurrent states are omitted from the notation and the corresponding recurrent context remains implicit in each transition. A state decomposition alone does not isolate reusable dynamics. To enable modular transfer, we adopt the following restriction on the latent transition model.

Assumption 4.1 (Unidirectional Latent Control)**.** The agent's influence on the background is mediated solely by the latent interface variable $z^{\mathcal{I}}_{t-1}$. Conditioned on $(s^{bg}_{t-1}, z^{\mathcal{I}}_{t-1})$, the background transition has no additional dependence on the agent states (s^{ag}_{t-1}, s^{ag}_t) or action a_{t-1}. The agent transition receives no explicit background-to-agent input.

Under Assumption 4.1, the joint transition factorizes into agent dynamics, interface generation, and background dynamics:

$$p_\theta(s^{ag}_t, s^{bg}_t, z^{\mathcal{I}}_{t-1} \mid s^{ag}_{t-1}, s^{bg}_{t-1}, a_{t-1}) = \underbrace{p_{\theta_{ag}}(s^{ag}_t \mid s^{ag}_{t-1}, a_{t-1})}_{\text{agent dynamics}} \cdot \underbrace{p_{\theta_{\mathcal{I}}}(z^{\mathcal{I}}_{t-1} \mid s^{ag}_t, a_{t-1})}_{\text{interface policy}} \cdot \underbrace{p_{\theta_{bg}}(s^{bg}_t \mid s^{bg}_{t-1}, z^{\mathcal{I}}_{t-1})}_{\text{background dynamics}}. \tag{1}$$

Here each component plays a distinct role in our modular architecture. The **Agent Dynamics** $p_{\theta_{ag}}$ models the agent's own transition, without directly conditioning on the background state. The

Interface Policy $p_{\theta_\mathcal{I}}$ is the key bridging module. It generates the abstract interaction signal $z^{\mathcal{I}}_{t-1}$ (e.g., forces, ego-motion) given the agent's action a_{t-1} and its resultant state s^{ag}_t. Conditioning on the generated *next* state s^{ag}_t allows the policy to generate signals consistent with the realized movement. The **Background Dynamics** $p_{\theta_{bg}}$ simulates the transition of the global scene driven by $z^{\mathcal{I}}_{t-1}$. Since this module does not directly observe a_{t-1}, it remains agnostic to the agent's dynamics, supporting reuse across different agents.

Physical interactions are generally bidirectional, such as ground reaction forces. We adopt Unidirectional Latent Control as a trade-off between physical completeness and modularity. Explicitly modeling the *background → agent* feedback would introduce additional coupling between the two dynamics modules and make the inter-module control structure more complex. Instead, we rely on the *implicit representation capacity* of the agent state. Environmental feedback observed by the agent can be captured in the posterior ($q(s^{ag}_t \mid o_{\leq t})$ in Section 4.3) and subsequently modeled by the agent dynamics $p_{\theta_{ag}}$. Meanwhile, the agent influences the background explicitly through $z^{\mathcal{I}}_t$. This design keeps the explicit interaction between modules simple and decoupled, while still allowing environmental feedback to be modeled implicitly.

4.3 Learning the Composition from Pixels

In this section, we describe how we learn the module states, dynamics, and interface from pixels.

Modular representation. Given an observation o_t, we infer two entity slots via a modular encoder. We leverage Slot Attention (Locatello et al., 2020) (see Appendix C.1 for implementation details) to enforce object binding, yielding a factorized posterior:

$$q_\phi(s^{ag}_t, s^{bg}_t \mid o_t) = q_{\phi_{ag}}(s^{ag}_t \mid o_t) \cdot q_{\phi_{bg}}(s^{bg}_t \mid o_t). \tag{2}$$

This factorization acts as an inductive bias, where the background representation remains conditionally independent of the agent's identity during posterior inference.

Asymmetric composite decoder. To reconstruct the observation o_t, we employ an asymmetric decoding strategy that reflects the physical composition of the scene. The agent state s^{ag}_t is decoded into an RGBA component (color image $\hat{o}^{ag}_t$ and an alpha mask m_t), while the background state s^{bg}_t is decoded strictly into an RGB background $\hat{o}^{bg}_t$:$(\hat{o}^{ag}_t, m_t) = D_{\theta_{ag}}(s^{ag}_t)$, where $m_t \in [0,1]^{H\times W}$, $\hat{o}^{bg}_t = D_{\theta_{bg}}(s^{bg}_t)$. The final predicted image $\hat{o}_t$ is formed via alpha compositing:

$$\hat{o}_t = m_t \odot \hat{o}^{ag}_t + (1 - m_t) \odot \hat{o}^{bg}_t. \tag{3}$$

Because Slot Attention is permutation-equivariant, to encourage the semantic binding of slots, we introduce a regularization term. We add an L1 penalty term for m_t as a sparsity regularizer to minimize the locations that are identified as regions of agents:

$$\mathcal{R}_{\texttt{mask}} = \sum_{t=1}^{T} \|m_t\|_1. \tag{4}$$

Dynamics learning. We optimize a variational evidence lower bound (ELBO) on the action-conditioned likelihood of observation sequences. The objective is derived from the interplay between our structured generative model and a factorized inference network.

We define the joint distribution over observations $o_{1:T}$ and structured latent variables, with Agent states $s^{ag}_{1:T}$, Background states $s^{bg}_{1:T}$, and Interface latents $z^{\mathcal{I}}_{0:T-1}$. Leveraging the compositional factorization from Eq. 1, the generative process is given by: $p_\theta\big(o_{1:T}, s^{ag}_{1:T}, s^{bg}_{1:T}, z^{\mathcal{I}}_{0:T-1} \mid a_{0:T-1}\big) = \prod_{t=1}^{T}[p_{\theta_o}(o_t \mid s^{ag}_t, s^{bg}_t) \cdot p_{\theta_{ag}}(s^{ag}_t \mid s^{ag}_{t-1}, a_{t-1}) \cdot p_{\theta_\mathcal{I}}(z^{\mathcal{I}}_{t-1} \mid s^{ag}_t, a_{t-1}) \cdot p_{\theta_{bg}}(s^{bg}_t \mid s^{bg}_{t-1}, z^{\mathcal{I}}_{t-1})]$. To approximate the posterior, we employ a variational encoder q_ϕ that factorizes consistently with the generative structure: $q_\phi\big(s^{ag}_{1:T}, s^{bg}_{1:T}, z^{\mathcal{I}}_{0:T-1} \mid o_{1:T}, a_{0:T-1}\big) = \prod_{t=1}^{T}[q_{\phi_{ag}}(s^{ag}_t \mid s^{ag}_{t-1}, a_{t-1}, o_t) \cdot q_{\phi_\mathcal{I}}(z^{\mathcal{I}}_{t-1} \mid s^{ag}_t, a_{t-1}) \cdot q_{\phi_{bg}}(s^{bg}_t \mid s^{bg}_{t-1}, z^{\mathcal{I}}_{t-1}, o_t)]$. Crucially, the interface posterior $q_{\phi_\mathcal{I}}$ is inferred solely from the agent's transitions. This reflects our design that interaction events are implicit in the agent's kinematics, preventing the interface from peeking at the background state directly. Then we can obtain the evidence lower bound (ELBO):

$$\log p_\theta(o_{1:T} \mid a_{0:T-1}) \geq \mathcal{L}_{\texttt{ELBO}} \triangleq \sum_{t=1}^{T} \mathbb{E}_{q_\phi}\Big[\log p_{\theta_o}(o_t \mid s^{ag}_t, s^{bg}_t)\Big] - \mathcal{R}_{\texttt{KL}}. \tag{5}$$

where $\mathcal{R}_{\mathrm{KL}}$ sums the KL divergences for the Agent and Background modules (see Appendix A for full derivation):

$$\mathcal{R}_{\mathrm{KL}} \triangleq \sum_{t=1}^{T} \mathbb{E}_q \Big[D_{\mathrm{KL}}\big(q_{\phi_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1}, o_t) \,\|\, p_{\theta_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1})\big) + D_{\mathrm{KL}}\big(q_{\phi_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}}, o_t) \,\|\, p_{\theta_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}})\big) \Big]. \tag{6}$$

Interface regularization. In addition to the ELBO, we regularize the interface distribution toward a common prior:

$$\mathcal{R}_{\mathcal{I}} = \sum_{t=1}^{T} \mathbb{E}_q \Big[D_{\mathrm{KL}}\big(q_{\phi_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1}) \,\|\, \mathcal{N}(0, I)\big) \Big]. \tag{7}$$

By limiting the interface channel capacity, this term functions as an *Information Bottleneck*. Since $z^{\mathcal{I}}$ is the *sole* channel modulating the background, the capacity constraint encourages the interface to encode only the most salient, domain-invariant variations (e.g., global displacement, force magnitude) necessary to minimize background reconstruction error. Conversely, agent-specific details (e.g., limb articulation, local texture), which provide no additional predictive benefit for background prediction, are actively filtered out to conserve the information budget. Thus, $\mathcal{R}_{\mathcal{I}}$ acts not merely as a prior constraint, but promotes functional separation between agent-specific internal details and the interaction information required by the background module, supporting a compact interface for dynamics reuse (see Section 4.4 for further discussion).

Overall training loss. The total objective combines the ELBO (Eq. 5) with the structural constraints derived from our sparsity principle (Eq. 4) and protocol standardization (Eq. 7).

$$\mathcal{L} = -\mathcal{L}_{\texttt{ELBO}} + \beta_{\texttt{mask}} \mathcal{R}_{\texttt{mask}} + \beta_{\mathcal{I}} \mathcal{R}_{\mathcal{I}}. \tag{8}$$

4.4 Reuse by Recomposition

As shown in Figure 1(b), we construct a target BRICKS-WM by composing an inherited background dynamics module with a new agent dynamics module and interface policy. We inherit source parameters $\theta^{src}_{\texttt{share}}$ for the shared perceptual components, background dynamics, and background decoder, while re-initializing agent-specific components $\theta^{tgt}_{\texttt{new}}$, including the agent dynamics, interface policy, and reward/continuation heads. The background recurrent core and its input projections are frozen, while the RSSM output layers are allowed to adapt. Appendix C.2 provides the complete parameter partition and freezing strategy. The rest of the content in this section explains the rationale behind this approach.

Definition 4.2 (Reusable dynamics module via protocol matching)**.** A dynamics module $p_{\theta_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}})$ is *reusable* across agents if, for a new agent, there exists an interface policy producing $z_{t-1}^{\mathcal{I}}$ such that the induced interface process matches the protocol expected by the module (i.e., aligning with the distribution learned during source domain training).

Proposition 4.3 (Sufficient condition for background-module reuse)**.** *Consider two agents A (source) and A' (target) operating with the same frozen background module $p_{\theta_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}})$. If for all t, the conditional interface distributions match under comparable histories:*

$$p^{A}(z_{t-1}^{\mathcal{I}} \mid \mathcal{H}_{t-1}) = p^{A'}(z_{t-1}^{\mathcal{I}} \mid \mathcal{H}_{t-1}), \tag{9}$$

then the induced background transition distributions also match:

$$p^{A}(s_t^{bg} \mid s_{t-1}^{bg}, \mathcal{H}_{t-1}) = p^{A'}(s_t^{bg} \mid s_{t-1}^{bg}, \mathcal{H}_{t-1}). \tag{10}$$

Proposition 4.3 formalizes our key insight: *the recomposition reuse is fundamentally a protocol-matching problem*. The sufficient condition for reuse is not merely matching a canonical prior, but matching the interface process expected by the frozen background module. In particular, if the target agent induces the same conditional interface distribution as the source agent under comparable histories, then the frozen background transition induces the same conditional background dynamics. Our regularization term $\mathcal{R}_{\mathcal{I}}$ in Sec. 4.3 provides a capacity constraint and canonicalization bias: it discourages the interface from carrying arbitrary agent-specific information and empirically facilitates

protocol alignment. To adapt target interface inputs to the reused background transition, we insert a lightweight two-layer residual adapter A_ψ before the background module

$$\tilde{z}_t^{\mathcal{I}} = A_\psi(z_t^{\mathcal{I}}) = z_t^{\mathcal{I}} + \mathrm{MLP}_\psi(z_t^{\mathcal{I}}), \tag{11}$$

where the residual branch is initialized to output zero. The adapter therefore acts as the identity map at initialization, without adding an initial transformation to the interface generator's outputs. Compatibility with the reused background transition must still be learned during target adaptation. Overall, BRICKS-WM transforms the transfer learning challenge into a *protocol-matching* problem: the target agent learns to control the environment by communicating with the frozen background dynamics module, enabling sample-efficient reuse of environmental dynamics in related settings. For implementation details, see Appendix C.

5 Experiments

We evaluate whether BRICKS-WM can learn competitive controllers from scratch and subsequently reuse learned dynamics as composable modules. Our experiments address four questions: (1) Can the modular architecture approach the control performance of monolithic world models when trained from scratch, and can its learned background dynamics be recomposed with a new agent module for another task? (Section 5.1); (2) Does the learned interface exhibit interpretable semantics consistent with the intended dynamics decomposition? (Section 5.2); (3) Does reusing background dynamics itself improve transfer performance? (Section 5.3); (4) How does interface regularization affect control performance? (Section 5.4). All shaded regions in fugures represent 95% confidence intervals.

Setup. We evaluate BRICKS-WM on four locomotion tasks from the DeepMind Control Suite (DMC) (Tassa et al., 2018): *Walker Walk*, *Walker Run*, *Cheetah Run*, and *Hopper Hop*, and on two CarRacing variants: *Sport Car* and *Heavy Car*. Benchmark details are provided in Appendix C.4. We compare with two strong monolithic world models, **DreamerV3** (Hafner et al., 2023) and **TD-MPC2** (Hansen et al., 2024), trained from scratch. Prior object-centric approaches such as SOLD (Mosbach et al., 2025) use different training settings involving pretraining or state supervision, so we focus on whether modules learned from scratch support both competitive control and subsequent reuse. BRICKS-WM trained from scratch also serves as the within-architecture reference for reuse.

5.1 Control Performance from Scratch and Modular Reuse

Figure 2 shows that BRICKS-WM trained from scratch reaches final returns broadly comparable to those of DreamerV3 and TD-MPC2 on the four DMC tasks and two CarRacing tasks, without pretrained representations or ground-truth state supervision used in related works (Mosbach et al., 2025; Zhang et al., 2026). Reusing the background dynamics improves early learning between related DMC tasks (*Walker Walk* $\leftrightarrow$ *Walker Run*) and remains effective for some changes in morphology (*Walker Walk* $\rightarrow$ *Hopper Hop*). On CarRacing, reuse improves learning compared with BRICKS-WM trained from scratch in both directions. Conversely, scenarios where transfer performance lags behind the baselines under significant domain gaps serve to empirically delineate the boundaries of effective reuse. Detailed returns and AUCs are given in Appendix D.1.

5.2 Analyzing Interface Semantics and Dynamics Decoupling

To examine the functional role of the learned interface, we conduct a latent modulation experiment on *Cheetah Run*. We sample an episode trajectory and employ the pretrained model for observation reconstruction. During the generation process, we manually substitute the interface latent $z^{\mathcal{I}}$, originally produced by the agent interface policy, with a synthesized vector. Specifically, utilizing a 4-dimensional interface space, we modulate one dimension with a sinusoidal signal while fixing the remaining dimensions to zero. Figure 3 visualizes the results. We observe that the floor in the reconstructed observation exhibits vertical oscillation with the injected sine wave, while showing no horizontal displacement. This suggests that the modulated dimension selectively encodes the vertical offset of the background. These observations suggest that this interface dimension influences vertical background motion, consistent with the intended functional separation. We further quantify this effect by comparing perturbed rollouts with their unperturbed baselines. For each interface dimension, we inject a sinusoidal perturbation during the first 50 imagined steps on 5 held-out episodes. A pixel is counted as changed if the mean absolute RGB difference exceeds 0.05. The foreground region is the agent. For the background region, we use the lower half of the frame, which captures the floor motion induced by locomotion. As shown in Table 1, the proportion of changed pixels in the

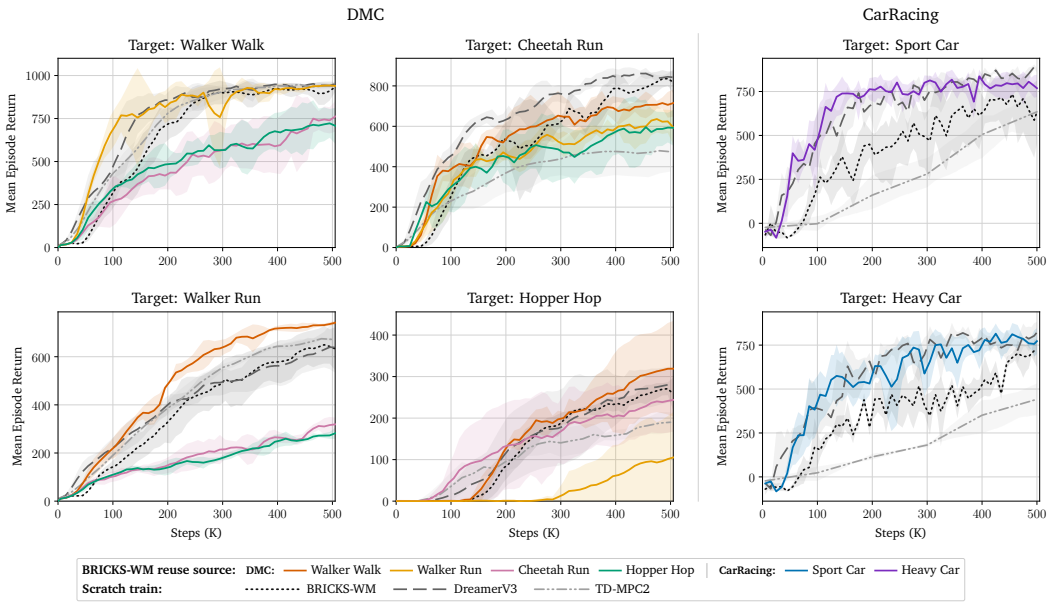


Figure 2: **Control from scratch and modular reuse on DMC and CarRacing.** Each panel corresponds to a target task: four DMC tasks in the left two columns and two CarRacing variants in the right column. Colored solid curves show reuse from the indicated source and monochrome curves show training from scratch.

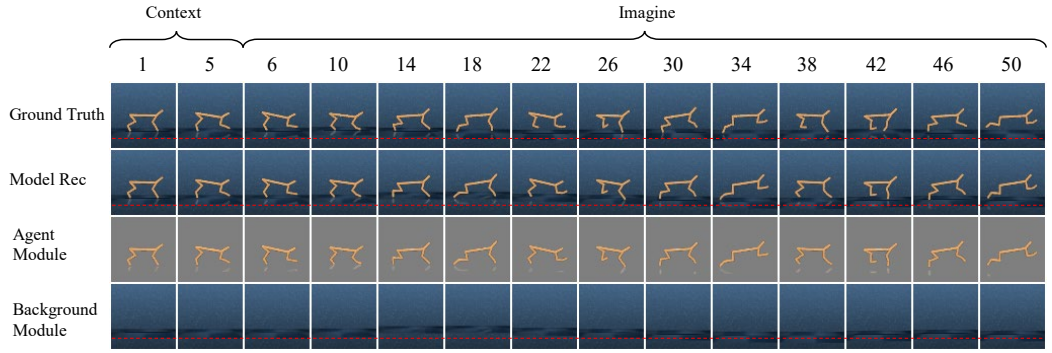


Figure 3: **Latent modulation analysis on *Cheetah Run*.** The model receives the first 5 frames as context and performs open-loop imagination from timestep $t = 6$. During imagination, we modulate one dimension of the interface latent $z^{\mathcal{I}}$ with a sinusoidal signal. **Rows 1–4** show the ground truth, full reconstruction, agent component, and background component, respectively. The **red dashed lines** highlight the vertical floor oscillation induced by the modulation. Notably, unlike the ground truth (**Row 1**), both the full reconstruction (**Row 2**) and background (**Row 4**) display clear vertical floor oscillations.

background region is 3.7–5.2 times that in the foreground region. This quantitative result corroborates the qualitative visualization, indicating that the learned interface primarily modulates background dynamics rather than leaking detailed agent appearance.

5.3 Contribution of Reused Background Dynamics

To test whether transfer benefits extend beyond inherited perception and decoding, we compare BRICKS-WM with a perception-only warm start variant, **BRICKS-WM-RBG** (Reinitialized Background dynamics). This variant uses the same inherited encoder and decoder but reinitializes the background RSSM core before target-task training. Table 2 shows higher early mean returns with

Table 1: **Interface perturbations on *Cheetah Run*.** Change rates for background (BG) and foreground (FG).

Dim.	Region	Change	BG/FG
Dim 0	BG	0.156 ± 0.005	3.727
	FG	0.042 ± 0.016	
Dim 1	BG	0.171 ± 0.010	5.206
	FG	0.033 ± 0.011	
Dim 2	BG	0.166 ± 0.010	3.951
	FG	0.042 ± 0.011	
Dim 3	BG	0.165 ± 0.010	4.413
	FG	0.037 ± 0.015	

Table 2: **Perception-only warm start on DMC.** Mean episode returns after 100K, 200K, and 500K target-environment steps. WW: Walker Walk; WR: Walker Run; CR: Cheetah Run; HH: Hopper Hop.

Transfer	Method	100K	200K	500K
WW→WR	BRICKS-WM	247 ± 51	509 ± 50	746 ± 4
	BRICKS-WM-RBG	221 ± 45	472 ± 70	742 ± 26
WW→CR	BRICKS-WM	465 ± 90	575 ± 116	722 ± 78
	BRICKS-WM-RBG	316 ± 145	543 ± 102	689 ± 98
WW→HH	BRICKS-WM	0 ± 0	150 ± 24	321 ± 100
	BRICKS-WM-RBG	21 ± 43	98 ± 113	167 ± 215
WR→WW	BRICKS-WM	750 ± 131	885 ± 58	940 ± 9
	BRICKS-WM-RBG	661 ± 175	892 ± 34	951 ± 11

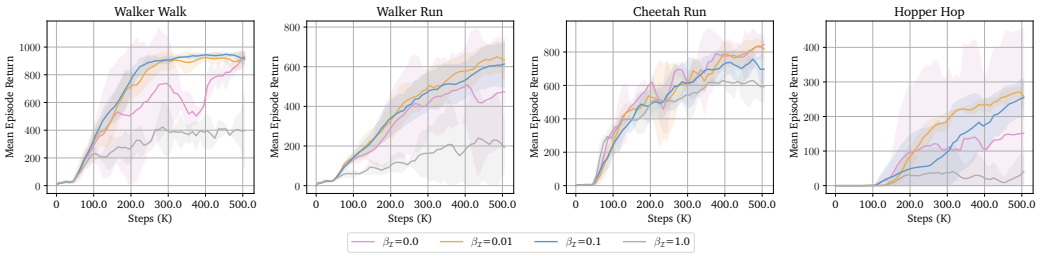


Figure 4: **Effect of the interface regularization coefficient on control performance.**

dynamics reuse in most transfer pairs. For *Walker Walk* → *Hopper Hop*, reuse also yields a higher mean final return, whereas final returns are similar in the two Walker transfer directions. These results support a contribution from the learned background dynamics beyond inherited perception and decoding, with benefits that depend on the transfer pair and training budget.

5.4 Ablation Study

We analyze the necessity of the interface regularization mechanism, $\mathcal{R}_{\mathcal{I}}$, by varying its coefficient $\beta_{\mathcal{I}}$. This term acts as an information bottleneck intended to enforce a standardized interaction protocol. Figure 4 compares the performance. When $\beta_{\mathcal{I}} = 0$, the training exhibits substantial variance and instability. This suggests that without the information bottleneck constraint, the interface fails to filter out agent-specific details. This leakage undermines the intended modular decoupling. Conversely, an overly strong regularization (e.g., $\beta_{\mathcal{I}} = 1.0$) excessively restricts the interface capacity. This prevents the model from encoding high-frequency physical quantities required for dynamic locomotion, leading to performance degradation in highly dynamic tasks like *Cheetah Run* and *Walker Run*. Consequently, we adopt $\beta_{\mathcal{I}} = 0.01$ as a common default setting, balancing between enforcing an agent-agnostic protocol and maintaining sufficient information for control. See Appendix D.2 and D.3 for additional ablation results.

6 Conclusion and Future Work

In this paper, we introduced BRICKS-WM, a framework for composing world models from reusable dynamics modules connected through learned latent interfaces. Our Agent–Background instantiation supports recomposition by reusing a learned background dynamics module while learning new agent dynamics and adapting their interface. Across DMC and CarRacing, BRICKS-WM trained from scratch on pixel observations achieves final control performance broadly comparable to strong monolithic baselines. Reusing the background dynamics module improves early target-task learning in related settings, with gains that depend on the source–target pair. These results provide evidence that learned dynamics components can remain useful when recomposed in a new scenario, and highlight interface compatibility as a central consideration for reuse. Our interface analyses characterize the signals exchanged between dynamics modules during learning and reuse. Future work could build on these analyses to explore adaptive regularization schedules for easier configuration across tasks.

AI USE STATEMENT

In this work, we used generative AI tools to assist in checking the correctness and internal consistency of mathematical formulas and derivations. We have not used generative AI tools for any other tasks with required disclosure, and the remaining required-disclosure tasks that are not part of this work are not applicable. Additionally, we used generative AI tools to polish writing to improve its readability, including grammar, phrasing, and clarity. We have reviewed all AI-assisted work. In particular, all AI-assisted checks of mathematical formulas and derivations were independently verified by the authors, and all AI-assisted language edits were reviewed to ensure that they preserved the intended technical meaning and did not alter the scientific claims or results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Appendix C provides details of the model architecture, training and module reuse procedures, hyperparameters, and computational resources. Benchmark configurations and observation preprocessing are described in Appendix C.4, and Algorithms 1 and 2 summarize source-task training and target-task reuse. We submit the implementation code as supplementary material to support reproduction of the reported experiments.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J. Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *Advances in Neural Information Processing Systems*, 2024.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, 2016.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *International Conference on Machine Learning*, 2024.

Christopher P. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew M. Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *CoRR*, 2019.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *International Conference on Robotics and Automation*, 2017.

Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning. *CoRR*, 2024.

Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations*, 2020.

Fan Feng, Phillip Lippe, and Sara Magliacane. Learning interactive world model for object-centric reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi S. Jaakkola. Learning task informed abstractions. In *International Conference on Machine Learning*, 2021.

Kevin Gmelin, Shikhar Bahl, Russell Mendonca, and Deepak Pathak. Efficient RL via disentangled environment and agent representations. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning*, 2023.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations*, 2021.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew M. Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, 2019.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature Communications*, 2021.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. In *International Conference on Learning Representations*, 2022.

Danijar Hafner, Timothy P. 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*, 2019.

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

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *CoRR*, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Zihao He, Bo Ai, Tongzhou Mu, Yulin Liu, Weikang Wan, Jiawei Fu, Yilun Du, Henrik I. Christensen, and Hao Su. Scaling cross-embodiment world models for dexterous manipulation. *CoRR*, 2025.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, 2020.

Jindong Jiang, Fei Deng, Gautam Singh, Minseung Lee, and Sungjin Ahn. Slot state space models. In *Advances in Neural Information Processing Systems*, 2024.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

Thomas Kipf, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *International Conference on Learning Representations*, 2022.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, 2018.

Thomas N. Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations*, 2020.

Anson Lei, Frederik Nolte, Bernhard Schölkopf, and Ingmar Posner. Compete and compose: Learning independent mechanisms for modular world models. *CoRR*, 2024.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, 2020.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. SOLD: slot object-centric latent dynamics models for relational manipulation learning from pixels. In *International Conference on Machine Learning*, 2025.

Minting Pan, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Iso-dream: Isolating and leveraging noncontrollable visual dynamics in world models. In *Advances in Neural Information Processing Systems*, 2022.

Ameya Pore, Riccardo Muradore, and Diego Dall'Alba. DEAR: disentangled environment and agent representations for reinforcement learning without reconstruction. *CoRR*, 2024.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020.

Gautam Singh, Yue Wang, Jiawei Yang, Boris Ivanovic, Sungjin Ahn, Marco Pavone, and Tong Che. Parallelized spatiotemporal slot binding for videos. In *International Conference on Machine Learning*, 2024.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, 2018.

Mark Towers, Ariel Kwiatkowski, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, J. K. Terry, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. In *Advances in Neural Information Processing Systems*, 2025.

Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Annual Conference on Robot Learning*, 2019.

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*, 2022.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-o: Building structured world models with object-centric representations. In *Advances in Neural Information Processing Systems*, 2025.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. STORM: efficient stochastic transformer based world models for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Weipu Zhang, Adam Jelley, Trevor McInroe, Amos Storkey, and Gang Wang. Object-centric world models from few-shot annotations for sample-efficient reinforcement learning. In *International Conference on Learning Representations*, 2026.

A Derivation of the Evidence Lower Bound

In this section, we provide the detailed derivation of the Evidence Lower Bound (ELBO) objective used in Sec. 4.3. An episode is denoted by $\{(o_t, a_t)\}_{t=1}^{T}$ and we condition on the action sequence. We follow the indexing convention where latent states s_t generate observations o_t, and the transition from $t-1$ to t is driven by action a_{t-1}. We consider observations $o_{1:T}$ and condition on actions $a_{0:T-1}$, where a_0 is a fixed dummy action used to initialize the first transition. The initial states s_0^{ag} and s_0^{bg} are also fixed.

A.1 Generative Model and Factorization

We consider an episode of length T. The model introduces three sets of latent variables: Agent states $s_{1:T}^{ag}$, Background states $s_{1:T}^{bg}$, and Interface latents $z_{0:T-1}^{\mathcal{I}}$ (where $z_{t-1}^{\mathcal{I}}$ mediates the transition from $t-1$ to t).

We define fixed *dummy* initial conditions s_0^{ag}, s_0^{bg}. The probabilistic dependence at $t=1$ is expressed as a transition from these dummy states (e.g., $p(s_1^{ag}) \triangleq p(s_1^{ag}|s_0^{ag}, a_0)$).

Based on the modular factorization described in Eq. 1, the joint distribution is

$$p_\theta(o_{1:T}, s_{1:T}^{ag}, s_{1:T}^{bg}, z_{0:T-1}^{\mathcal{I}} \mid a_{0:T-1}) = \prod_{t=1}^{T} \Bigg[\underbrace{p_{\theta_o}(o_t \mid s_t^{ag}, s_t^{bg})}_{\text{Observation Decoder}} \cdot \underbrace{p_{\theta_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1})}_{\text{Agent Dynamics}} \cdot \underbrace{p_{\theta_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1})}_{\text{Interface Policy}} \cdot \underbrace{p_{\theta_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}})}_{\text{Background Dynamics}} \Bigg]. \tag{12}$$

Note that for $t=1$, the initial interface latent is sampled from $p_{\theta_{\mathcal{I}}}(z_0^{\mathcal{I}} \mid s_1^{ag}, a_0)$.

A.2 Variational Inference and ELBO Decomposition

The inference model q_ϕ approximates the posterior using the current observation o_t. Consistent with the generative structure, the interface latent is inferred via the agent's transition:

$$q_\phi(s_{1:T}^{ag}, s_{1:T}^{bg}, z_{0:T-1}^{\mathcal{I}} \mid o_{1:T}, a_{0:T-1}) = \prod_{t=1}^{T} \Big[q_{\phi_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1}, o_t) \cdot q_{\phi_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1}) \cdot q_{\phi_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}}, o_t) \Big]. \tag{13}$$

Starting from the marginal likelihood,

$$\begin{aligned} \log p_\theta(o_{1:T} \mid a_{0:T-1}) &= \log \int p_\theta(o_{1:T}, s_{1:T}^{ag}, s_{1:T}^{bg}, z_{0:T-1}^{\mathcal{I}} \mid a_{0:T-1})\, ds_{1:T}^{ag}\, ds_{1:T}^{bg}\, dz_{0:T-1}^{\mathcal{I}} \\ &= \log \mathbb{E}_{q_\phi} \left[\frac{p_\theta(o_{1:T}, s_{1:T}^{ag}, s_{1:T}^{bg}, z_{0:T-1}^{\mathcal{I}} \mid a_{0:T-1})}{q_\phi(s_{1:T}^{ag}, s_{1:T}^{bg}, z_{0:T-1}^{\mathcal{I}} \mid o_{1:T}, a_{0:T-1})} \right]. \end{aligned} \tag{14}$$

Applying Jensen's inequality yields the ELBO:

$$\log p_\theta(o_{1:T} \mid a_{0:T-1}) \geq \mathbb{E}_{q_\phi} \left[\log p_\theta(o_{1:T}, s_{1:T}^{ag}, s_{1:T}^{bg}, z_{0:T-1}^{\mathcal{I}} \mid a_{0:T-1}) - \log q_\phi(s_{1:T}^{ag}, s_{1:T}^{bg}, z_{0:T-1}^{\mathcal{I}} \mid o_{1:T}, a_{0:T-1}) \right]. \tag{15}$$

Substituting the factorizations in Eq. 12 and Eq. 13 into Eq. 15, we decompose the ELBO into reconstruction terms and KL divergences

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} = &\sum_{t=1}^{T} \mathbb{E}_{q_\phi}\Big[\log p_{\theta_o}(o_t \mid s_t^{ag}, s_t^{bg})\Big] \\ &- \sum_{t=1}^{T} \mathbb{E}_{q_\phi}\left[\log \frac{q_{\phi_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1}, o_t)}{p_{\theta_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1})}\right] \\ &- \sum_{t=1}^{T} \mathbb{E}_{q_\phi}\left[\log \frac{q_{\phi_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}}, o_t)}{p_{\theta_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}})}\right] \\ &- \sum_{t=1}^{T} \mathbb{E}_{q_\phi}\left[\log \frac{q_{\phi_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1})}{p_{\theta_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1})}\right].\end{aligned} \tag{16}$$

Cancellation of the Interface KL Term. Consider the specific KL divergence term associated with the interface latent $z^{\mathcal{I}}$. In our framework, the interface is an intermediate variable bridging the Agent and Background modules. During training, we define the variational posterior $q_{\phi_{\mathcal{I}}}$ to share the same parameters as the generative policy $p_{\theta_{\mathcal{I}}}$ (both conditioned on the agent's state):

$$q_{\phi_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1}) \equiv p_{\theta_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1}). \tag{17}$$

Consequently, the log-ratio of these distributions is zero, and the KL term vanishes from the standard ELBO:

$$\mathbb{E}_{q_\phi}\left[\log \frac{q_{\phi_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1})}{p_{\theta_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}} \mid s_t^{ag}, a_{t-1})}\right] = 0. \tag{18}$$

This derivation confirms why the interface variable $z^{\mathcal{I}}$ KL term does not appear in the dynamics regularization term $\mathcal{R}_{\text{KL}}$ (Eq. 6) in the main text.

Therefore, the ELBO contains **no** KL term for $z^{\mathcal{I}}$, consistent with the main-text objective where $z^{\mathcal{I}}$ is treated as an intermediate interaction/protocol variable.

The reconstruction term can be written as:

$$\mathcal{L}_{\text{REC}} = \sum_{t=1}^{T} \mathbb{E}_{q_\phi}\Big[\log p_{\theta_o}(o_t \mid s_t^{ag}, s_t^{bg})\Big]. \tag{19}$$

The remaining ratio terms can be written as KL divergences:

$$\mathcal{L}_{\text{KL}}^{ag} = \sum_{t=1}^{T} \mathbb{E}_{q_\phi}\Big[D_{\text{KL}}\Big(q_{\phi_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1}, o_t) \,\Big\|\, p_{\theta_{ag}}(s_t^{ag} \mid s_{t-1}^{ag}, a_{t-1})\Big)\Big], \tag{20}$$

$$\mathcal{L}_{\text{KL}}^{bg} = \sum_{t=1}^{T} \mathbb{E}_{q_\phi}\Big[D_{\text{KL}}\Big(q_{\phi_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}}, o_t) \,\Big\|\, p_{\theta_{bg}}(s_t^{bg} \mid s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}})\Big)\Big]. \tag{21}$$

Combining Eqs. 19–21, we obtain the final ELBO:

$$\mathcal{L}_{\text{ELBO}} = \mathcal{L}_{\text{REC}} - \Big(\mathcal{L}_{\text{KL}}^{ag} + \mathcal{L}_{\text{KL}}^{bg}\Big), \tag{22}$$

which corresponds to Eq. 5 and Eq. 6 in Sec. 4.3.

B Pseudo Code

The complete training procedure for the source task is presented in Algorithm 1, while the dynamics reuse mechanism for the target task is detailed in Algorithm 2. For brevity, we omit predictors for other task information, retaining only the reward predictor.

Algorithm 1 BRICKS-WM Training (Source Task)

Initialize: Dataset $\mathcal{D}$ collected by random policy, World Model parameters θ (comprising Agent θ_{ag}, Background θ_{bg}, Interface $\theta_{\mathcal{I}}$), Encoder parameters ϕ (Slot Attention), Actor parameters ψ^a, and Critic parameters ξ

for training step $t_1 = 1...T_1$ **do**

  for update step $t_2 = 1...T_2$ **do**

    // `Dynamics & Interface Learning`

    Sample minibatch sequence $\{(o_t, a_t, r_t)\}_{t=k}^{k+L} \sim \mathcal{D}$

    Perception: Infer factorized posteriors $(s_t^{ag}, s_t^{bg}) \sim q_\phi(s_t|o_t)$ via Modular Encoder

    Generation:

      Compute Agent prior $s_t^{ag} \sim p_{\theta_{ag}}(s_t^{ag}|s_{t-1}^{ag}, a_{t-1})$

      Generate Interface Latent $z_{t-1}^{\mathcal{I}} \sim p_{\theta_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}}|s_t^{ag}, a_{t-1})$

      Compute Background prior $s_t^{bg} \sim p_{\theta_{bg}}(s_t^{bg}|s_{t-1}^{bg}, z_{t-1}^{\mathcal{I}})$

    Reconstruction:

      Decode Agent components $(\hat{o}_t^{ag}, m_t) = D_{\theta_{ag}}(s_t^{ag})$ (RGB and Alpha Mask)

      Decode Background $\hat{o}_t^{bg} = D_{\theta_{bg}}(s_t^{bg})$

      Composite image $\hat{o}_t = m_t \odot \hat{o}_t^{ag} + (1 - m_t) \odot \hat{o}_t^{bg}$

    Optimization:

      Calculate $\mathcal{L}_{ELBO}$ (Reconstruction and Dynamics KL)

      Calculate Regularizations: Interface $\mathcal{R}_{\mathcal{I}}$ and Mask Sparsity $\mathcal{R}_{mask}$

      Update parameters θ, ϕ on Loss $\mathcal{L} = -\mathcal{L}_{ELBO} + \beta_{mask}\mathcal{R}_{mask} + \beta_{\mathcal{I}}\mathcal{R}_{\mathcal{I}}$

    // `Behavior Learning`

    Imagination: Rollout trajectories $\{(s_\tau, a_\tau, r_\tau)\}_{\tau=t}^{t+H}$ from current posteriors:

      Sample action $a_\tau \sim \pi_{\psi^a}(a_\tau|s_\tau^{ag}, s_\tau^{bg})$

      Step Modular Dynamics: $s_\tau^{ag} \xrightarrow{p_{\theta_{ag}}} s_{\tau+1}^{ag} \xrightarrow{p_{\theta_{\mathcal{I}}}} z_\tau^{\mathcal{I}} \xrightarrow{p_{\theta_{bg}}} s_{\tau+1}^{bg}$

      Predict reward $\hat{r}_\tau$

    Update Actor ψ^a and Critic ξ using imagined returns

  end for

  for rollout step $t_3 = 1...T_3$ **do**

    Observe o_t

    Infer factorized state $(s_t^{ag}, s_t^{bg}) \sim q_\phi(s_t|o_t)$

    Sample action from exploration policy $a_t \sim \pi_{\psi^a}(a_t|s_t^{ag}, s_t^{bg})$

    Execute action: $r_t, o_{t+1} \leftarrow$ `env.step(`a_t`)`

    Add experience to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)\}$

  end for

end for

Algorithm 2 BRICKS-WM Dynamics Reuse (Target Task)

Input: Source parameters θ^{src}, Target Dataset $\mathcal{D}_{tgt}$
Transfer Setup:
1. **Inherit:** $\theta_{bg}, \theta_{enc}, \theta_{dec} \leftarrow \theta^{src}$ // Load BG Dynamics, Encoder, Decoder
2. **Partial Freeze:**
   Freeze θ_{enc} (Spatial & Slot Attn weights), θ_{dec}.
   Freeze θ_{bg}^{core} (RNN Cell & Input Layers).
   Unfreeze θ_{bg}^{heads} (Output Layers) for domain adaptation.
3. **Re-initialize:** θ_{ag} (Agent Dynamics), $\theta_{\mathcal{I}}$ (Interface), ϕ_{ag_init} (Agent Slot Query).
4. **Add Adapter:** Initialize Interface Adapter A_ψ (as identity).

for training step $t = 1...T$ **do**
  // `Inference & Reconstruction`
  Sample batch o_t from $\mathcal{D}_{tgt}$
  $(s_t^{ag}, s_t^{bg}) \sim q_\phi(s_t|o_t)$ // Encoder uses re-inited Agent Query
  // `Modified Dynamics Step with Adapter`
  Generation (Recomposed):
    $s_t^{ag} \sim p_{\theta_{ag}}(s_t^{ag}|s_{t-1}^{ag}, a_{t-1})$ // Learned Agent Dynamics
    $z_{t-1}^{\mathcal{I}} \sim p_{\theta_{\mathcal{I}}}(z_{t-1}^{\mathcal{I}}|s_t^{ag}, a_{t-1})$ // New Interface Policy
    $\tilde{z}_{t-1}^{\mathcal{I}} = A_\psi(z_{t-1}^{\mathcal{I}}) = z_{t-1}^{\mathcal{I}} + \text{MLP}(z_{t-1}^{\mathcal{I}})$ // Interface Adapter
    $s_t^{bg} \sim p_{\theta_{bg}}(s_t^{bg}|s_{t-1}^{bg}, \tilde{z}_{t-1}^{\mathcal{I}})$ // Frozen Core, Adapted Heads
  Optimization:
    Update $\theta_{ag}, \theta_{\mathcal{I}}, \phi_{ag_init}, A_\psi, \theta_{bg}^{heads}$ to minimize $\mathcal{L}$
end for

C Implementation Details

C.1 Modular Encoder Details

To functionally decouple the visual scene into entity-centric representations, we employ a Modular Encoder composed of a Convolutional Neural Network (CNN) backbone followed by a Slot Attention module.

Convolutional Backbone. Given an input observation $\mathbf{o}_t \in \mathbb{R}^{H\times W\times 3}$, we first extract a feature map using a CNN encoder. Unlike standard encoders that flatten the spatial dimensions immediately, we preserve the spatial structure to allow for object-centric binding. The encoder produces a feature map $\mathbf{E}_{\text{spatial}} \in \mathbb{R}^{H'\times W'\times D_{\text{enc}}}$. We flatten the spatial dimensions to obtain a set of $N = H' \times W'$ input feature vectors $\mathbf{V}_{\text{in}} \in \mathbb{R}^{N\times D_{\text{enc}}}$, which serve as the input to the attention mechanism.

Slot Attention. We adapt the Slot Attention mechanism (Locatello et al., 2020) to map the N input vectors to $N_s = 2$ object slots: the Agent slot s^{ag} and the Background slot s^{bg}. Standard Slot Attention initializes slots by sampling from a shared Gaussian distribution, which leads to permutation equivariant slots (order does not matter). However, for BRICKS-WM, we require semantic binding (i.e., the first slot must always represent the Agent and the second the Background) to facilitate modular reuse. Therefore, instead of random sampling, we define the initial slots $\mathbf{S}_0 \in \mathbb{R}^{N_s\times D_{\text{slot}}}$ as distinct learnable parameters:

$$\mathbf{S}_0 = \begin{bmatrix} \boldsymbol{\mu}_{\text{ag}} \\ \boldsymbol{\mu}_{\text{bg}} \end{bmatrix}, \tag{23}$$

where $\boldsymbol{\mu}_{\text{ag}}, \boldsymbol{\mu}_{\text{bg}} \in \mathbb{R}^{D_{\text{slot}}}$ are independent learnable vectors initialized with Gaussian noise. The slots are refined iteratively over T_{iter} iterations (set to 3 in our experiments) via a competitive attention mechanism. At each iteration $l = 1, \ldots, T_{\text{iter}}$, the update proceeds as follows:

1. **Projection:** We project the inputs $\mathbf{V}_{\text{in}}$ into keys $\mathbf{K}$ and values $\mathbf{V}$, and the layer-normalized slots $\mathbf{S}_{l-1}$ into queries $\mathbf{Q}$:
$$\mathbf{K} = \text{LayerNorm}(\mathbf{V}_{in})\mathbf{W}_k, \quad \mathbf{V} = \text{LayerNorm}(\mathbf{V}_{in})\mathbf{W}_v, \quad \mathbf{Q} = \text{LayerNorm}(\mathbf{S}_{l-1})\mathbf{W}_q. \tag{24}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ are learnable linear projections.

2. **Attention:** We compute the dot-product attention scores. Crucially, we apply the softmax function over the *slots* dimension N_s. This forces the slots to compete for explaining each part of the input image:

$$\mathbf{M} = \frac{1}{\sqrt{D_{\text{slot}}}} \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N_s \times N}, \tag{25}$$

$$\alpha_{i,j} = \frac{\exp(M_{i,j})}{\sum_{k=1}^{N_s} \exp(M_{k,j})}, \tag{26}$$

where $\alpha_{i,j}$ represents the attention weight of slot i for input feature vector j.

3. **Aggregation:** To ensure the update magnitude is independent of the object size (number of pixels), we normalize the attention weights over the spatial dimension N before aggregating the values:

$$w_{i,j} = \frac{\alpha_{i,j}}{\sum_{p=1}^{N} \alpha_{i,p} + \epsilon}, \tag{27}$$

$$\mathbf{U}_i = \sum_{j=1}^{N} w_{i,j} \mathbf{V}_j. \tag{28}$$

Here, $\mathbf{U}_i$ is the aggregated update for the i-th slot.

4. **Update:** The aggregated updates $\mathbf{U}$ are used to update the slots using a Gated Recurrent Unit (GRU), followed by an MLP with a residual connection:

$$\hat{\mathbf{S}}_l = \text{GRU}(\mathbf{U}, \mathbf{S}_{l-1}), \tag{29}$$

$$\mathbf{S}_l = \hat{\mathbf{S}}_l + \text{MLP}(\text{LayerNorm}(\hat{\mathbf{S}}_l)). \tag{30}$$

Note that the GRU parameters are shared across all slots.

After T_{iter} iterations, the final slots are projected via a linear layer to match the dimension required by the dynamics modules. Finally these dynamics modules yield the factorized posterior state inputs s_t^{ag} and s_t^{bg}.

C.2 Warmup and Reuse Details

Warmup Phase. To facilitate the stable emergence of object-centric representations before imposing dynamics constraints, we employ a warmup strategy during the initial phase of training. Specifically, for the first $N_{\texttt{warmup}}$ updates (set to 10,000 for source tasks and 5,000 for target tasks), we suppress the dynamics consistency loss and the interface regularization. We set the coefficients $\beta_{dyn} = \beta_{rep} = \beta_{\mathcal{I}} = 0$, reducing the objective to pure reconstruction maximization: $\mathcal{L} \approx -\mathcal{L}_{\texttt{REC}} + \beta_{\texttt{mask}} \mathcal{R}_{\texttt{mask}}$. This warmup emphasizes reconstruction and mask sparsity before introducing KL-based dynamics regularization, encouraging an initial separation of the agent and background representations.

Dynamics Reuse and Freezing Strategy. When transferring the background module to a new agent, we strictly partition the model parameters into inherited, frozen, and re-initialized sets to enforce the reuse protocol:

- **Parameter Inheritance:** We initialize the target model by loading the pretrained parameters θ_{src} for the domain-invariant components: the Spatial Encoder (CNN), the core of Slot Attention (Keys, Values, GRU, and Background Query), the Background Dynamics module (θ_{bg}), and the Decoder.
- **Re-initialization:** Agent-specific components are initialized from scratch. These include the Agent Query in Slot Attention (ϕ_{ag_init}), the Agent Dynamics module (θ_{ag}), the Interface Policy ($\theta_{\mathcal{I}}$), and the Task Heads (Reward and Continuation).
- **Perceptual Freezing and Slot Adaptation:** During training on the target task, we freeze the weights of the Spatial Encoder to preserve the visual feature space. For the Slot Attention module, we freeze its shared extraction mechanisms—specifically the linear projections

for queries, keys, and values, the recurrent update function (GRU), and the feedforward networks (MLP). This ensures the object-centric binding logic remains consistent. Crucially, while the learnable background slot query (ϕ_{bg}) is frozen to anchor the scene representation, the agent slot query (ϕ_{ag}) is re-initialized and trainable, enabling the model to learn to bind to the new agent using the pre-established attention mechanism.

- **Core Dynamics Freezing:** For the Background Dynamics, we enforce structural consistency by freezing the recurrent GRU cells and input projection layers (core Background Dynamics). However, we allow the RSSM output layers to be fine-tuned to adapt to domain shifts. Additionally, a zero-initialized residual adapter (MLP) is inserted before the background module to gradually align the interface distribution $\tilde{z}^{\mathcal{I}}$ without disrupting the initial transfer.

C.3 Hyperparameters and Time Cost

Table 3 presents the primary hyperparameters of BRICKS-WM. Our implementation is built upon the DreamerV3 codebase [1]. The Agent and Background modules are instantiated as two separate Recurrent State-Space Models (RSSMs) but share the same structural hyperparameters (e.g., recurrent units and latent dimensions) as listed in the table. The behavior policy operates on the concatenated features extracted from both the Agent and Background modules ($s_t = [s_t^{ag}, s_t^{bg}]$) to ensure the agent can react to environmental changes. The experiments are conducted on NVIDIA RTX 4090 GPUs. With a batch size of 16 and sequence length of 64, training on DMC tasks requires approximately 24 hours for 500K environment steps.

Table 3: Hyperparameters for BRICKS-WM

Hyperparameter	**Value**
General	
Action Repeat	2
Batch Size	16
Batch Length	64
Optimizer	Adam
Learning Rate (Model)	1×10^{-4}
Learning Rate (Actor/Critic)	3×10^{-5}
Loss Scales	
Dynamics Scale β_{dyn}	0.5
Representation Scale β_{rep}	0.1
Interface KL Scale $\beta_{\mathcal{I}}$	0.01
Mask Loss Scale β_{mask}	1.0
Free Bits (KL)	1.0
Modular RSSM (Agent s_t^{ag} & Background s_t^{bg})	
Discrete latent dimensions	32
Discrete latent classes	32
GRU recurrent units (Deterministic)	512
Dense hidden units	512
Activation	SiLU
Normalization	LayerNorm
Interface & Attention	
Interface latent dimension ($z^{\mathcal{I}}$)	4
Slot Number	2 (Agent + Background)
Slot Dimension	64
Slot Attention Iterations $T_{\texttt{iter}}$	3
Warmup Update Times $N_{\texttt{warmup}}$	10,000 for source, 5,000 for reuse

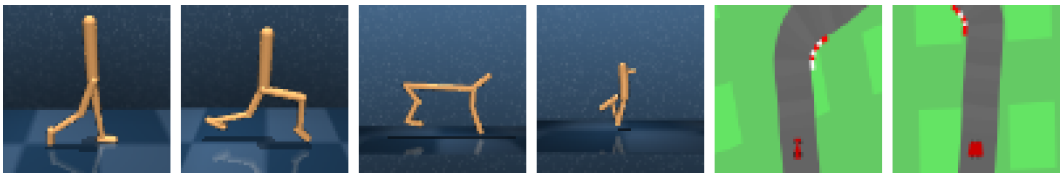


Figure 5: Example observations from the six benchmark tasks. From left to right: *Walker Walk*, *Walker Run*, *Cheetah Run*, *Hopper Hop*, *CarRacing Sport Car*, and *CarRacing Heavy Car*.

C.4 Details about the Benchmark

Benchmark environments. Our benchmark comprises four standard locomotion tasks from the DeepMind Control Suite (DMC) (*Walker Walk*, *Walker Run*, *Cheetah Run*, and *Hopper Hop*) and two CarRacing variants (*Sport Car* and *Heavy Car*), as illustrated in Figure 5. The DMC tasks retain their standard dynamics and reward functions, and use 64×64 RGB pixels. For CarRacing, Sport Car uses the original vehicle configuration, whereas Heavy Car has a wider, shorter trapezoidal silhouette, $1.5\times$ the hull density, and $0.7\times$ the steering response of Sport Car. Both variants share the same track-generation procedure, reward function, and continuous steering, throttle, and brake controls; vehicle color, collision geometry, wheel geometry, engine power, and tire-road friction are unchanged. We remove the dashboard from CarRacing observations and resize the remaining scene to 64×64 RGB pixels.

D Additional Experimental Results

This section supplements the DMC and CarRacing learning curves in Section 5.1 with quantitative transfer results and analyses of modular reuse. We report final returns and normalized return AUCs, then examine the roles of interface regularization, mask sparsity, and interface adaptation in control performance and module composition. We also analyze how the learned interface distribution varies with regularization strength and develops over the course of training.

D.1 Transfer AUC and Return

Tables 4 and 5 report final returns and normalized return AUCs for modular reuse at a 500K target-data budget on DMC and CarRacing. Table 6 reports the corresponding results for baselines trained from scratch. Transfers between the two Walker tasks achieve final returns of 746 ± 4 for *Walker Walk* $\rightarrow$ *Walker Run* and 940 ± 9 for *Walker Run* $\rightarrow$ *Walker Walk*. Mean AUCs exceed those of both scratch baselines in these two directions, although *Walker Run* $\rightarrow$ *Walker Walk* is nearly tied with DreamerV3 at 500K. Results for transfers between different morphologies are more variable, indicating that the benefits of reuse depend on the source–target pair.

Training budgets are measured in environment steps. Transfer budgets include only target-task data and exclude source pretraining. CarRacing entries report the mean $\pm$ sample standard deviation across four training seeds. For each CarRacing run, we use linear interpolation between evaluations to obtain the return at the specified budget. We compute normalized return AUC by integrating the interpolated return curve and dividing by the length of the evaluated training interval. The CarRacing results are computed without curve smoothing or extrapolation.

[1] https://github.com/NM512/dreamerv3-torch

Table 4: **Final return at a 500K target-data budget.** Rows denote source tasks and columns denote target tasks for transfer within each domain.

DMC				
Source \ Target	**Walker Walk**	**Walker Run**	**Cheetah Run**	**Hopper Hop**
Walker Walk	–	746 ± 4	722 ± 78	321 ± 100
Walker Run	940 ± 9	–	566 ± 112	97 ± 88
Cheetah Run	780 ± 37	318 ± 23	–	249 ± 69
Hopper Hop	653 ± 124	309 ± 22	598 ± 99	–
CarRacing				
Source \ Target	**Sport Car**		**Heavy Car**	
Sport Car	–		772 ± 90	
Heavy Car	769 ± 77		–	

Table 5: **Normalized return AUC at a 500K target-data budget.** Rows denote source tasks and columns denote target tasks for transfer within each domain.

DMC				
Source \ Target	**Walker Walk**	**Walker Run**	**Cheetah Run**	**Hopper Hop**
Walker Walk	–	508 ± 18	538 ± 48	158 ± 31
Walker Run	776 ± 44	–	454 ± 34	28 ± 29
Cheetah Run	477 ± 67	186 ± 17	–	143 ± 62
Hopper Hop	506 ± 59	173 ± 9	435 ± 78	–
CarRacing				
Source \ Target	**Sport Car**		**Heavy Car**	
Sport Car	–		562 ± 68	
Heavy Car	652 ± 38		–	

Table 6: **Performance of baselines trained from scratch at a 500K training budget.** Each entry lists final return / normalized return AUC.

DMC				
Method	**Walker Walk**	**Walker Run**	**Cheetah Run**	**Hopper Hop**
DreamerV3	$958 \pm 12/774 \pm 50$	$636 \pm 113/421 \pm 30$	$821 \pm 56/662 \pm 16$	$281 \pm 42/147 \pm 56$
TD-MPC2	$952 \pm 10/741 \pm 21$	$662 \pm 20/449 \pm 9$	$455 \pm 58/372 \pm 68$	$194 \pm 20/114 \pm 17$
CarRacing				
Method	**Sport Car**		**Heavy Car**	
DreamerV3	$870 \pm 15/633 \pm 64$		$822 \pm 98/587 \pm 55$	
TD-MPC2	$634 \pm 166/253 \pm 60$		$442 \pm 106/178 \pm 32$	

Early Transfer AUC. We additionally report normalized return AUC at a 200K target-data budget to assess early learning efficiency on DMC and CarRacing. This measure complements final return by capturing differences in learning progress before performance approaches saturation, as in *Walker Run* $\rightarrow$ *Walker Walk*.

Table 7: **Normalized return AUC at a 200K target-data budget.** Rows denote source tasks and columns denote target tasks for transfer within each domain.

DMC				
Source \ Target	**Walker Walk**	**Walker Run**	**Cheetah Run**	**Hopper Hop**
Walker Walk	–	257 ± 14	358 ± 50	29 ± 4
Walker Run	593 ± 70	–	296 ± 34	0 ± 0
Cheetah Run	270 ± 44	104 ± 15	–	63 ± 51
Hopper Hop	322 ± 31	104 ± 7	297 ± 70	–
CarRacing				
Source \ Target	**Sport Car**		**Heavy Car**	
Sport Car	–		337 ± 102	
Heavy Car	461 ± 71		–	

Table 8: **Normalized return AUC for baselines trained from scratch at a 200K training budget.** Results are reported on the same target tasks as the transfer experiments.

DMC				
Method	**Walker Walk**	**Walker Run**	**Cheetah Run**	**Hopper Hop**
DreamerV3	547 ± 108	245 ± 35	461 ± 29	37 ± 39
TD-MPC2	477 ± 28	229 ± 11	245 ± 41	48 ± 17
CarRacing				
Method	**Sport Car**		**Heavy Car**	
DreamerV3	406 ± 119		345 ± 94	
TD-MPC2	34 ± 27		35 ± 20	

As shown in Tables 7 and 8, *Walker Run* $\rightarrow$ *Walker Walk* achieves a normalized return AUC of 593 ± 70 at 200K steps, with a mean higher than those of both scratch baselines on *Walker Walk*. This early advantage may become less apparent in final returns as performance on the target task approaches saturation. On CarRacing, the 200K mean AUC increases from 164 for BRICKS-WM trained from scratch to 461 with reuse on *Sport Car*, and from 141 to 337 on *Heavy Car*. These comparisons show that composing a new agent module with the retained background core can improve mean learning performance within a smaller target-data budget.

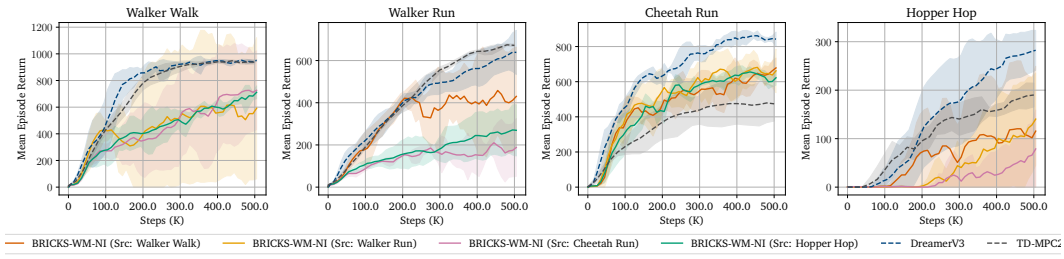


Figure 6: **Transfer performance without interface regularization (BRICKS-WM-NI).** Each panel corresponds to a target task. The transfer curves use the source tasks indicated in the legend, while DreamerV3 and TD-MPC2 are trained from scratch.

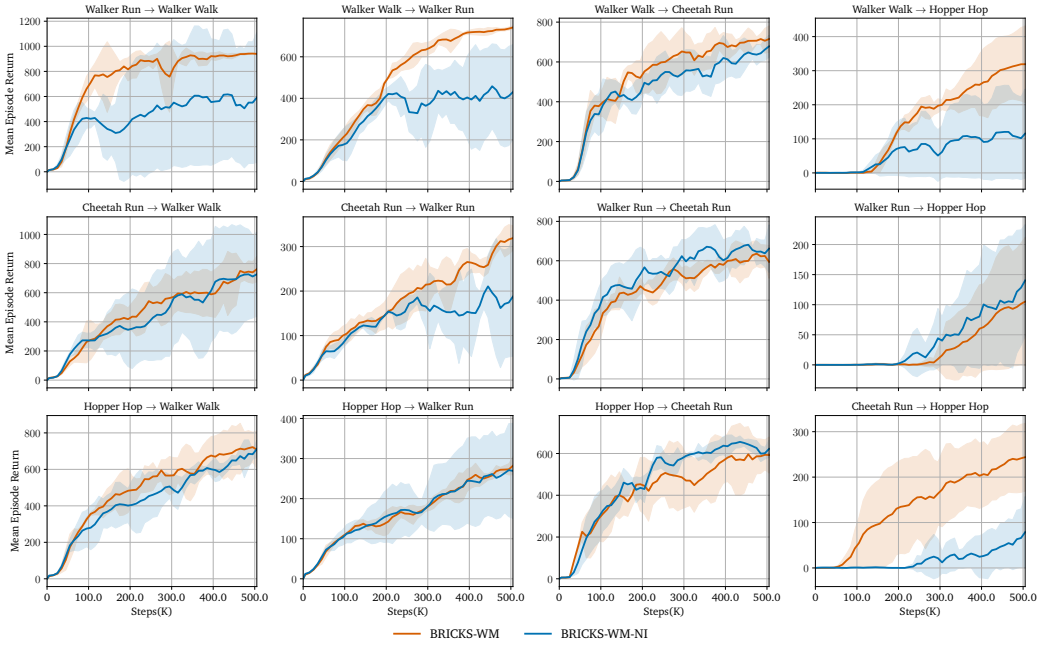


Figure 7: **Effect of interface regularization on modular reuse.** Each panel compares BRICKS-WM and BRICKS-WM-NI for one source–target task pair.

D.2 Role of Interface Regularization in Transfer

To assess the contribution of interface regularization $\mathcal{R}_{\mathcal{I}}$ to modular reuse, we evaluate **BRICKS-WM-NI**, a variant without interface regularization. We set $\beta_{\mathcal{I}} = 0$ during both source-task and target-task training, removing the explicit KL penalty on the interface distribution. Figures 6 and 7 report the resulting transfer performance. The regularized model achieves higher mean returns in many transfer directions, with differences evident during learning and at the end of the evaluated training budget. The benefit varies across source–target pairs, and some transfers favor the unregularized variant. These results support interface regularization as a useful inductive bias for modular reuse. The control results alone do not establish that regularization guarantees matching interface distributions across source and target tasks.

D.3 Mask Sparsity Ablation

We examine the effect of mask sparsity regularization, $\mathcal{R}_{\texttt{mask}}$, on control performance and the spatial assignment of visual information to modules. The shared Slot Attention update is equivariant to permutations of the initial slots and does not assign fixed agent or background semantics to individual

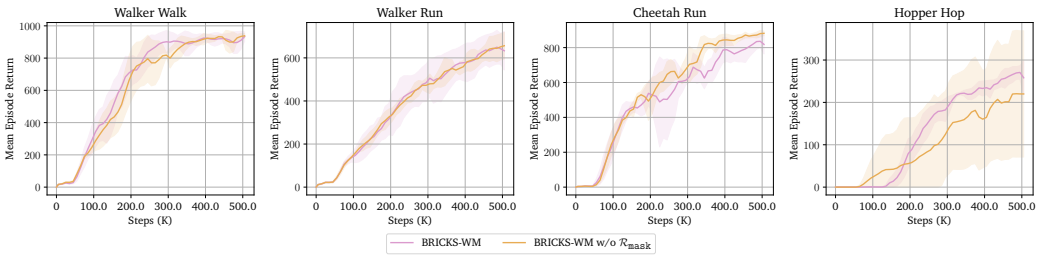


Figure 8: **Control performance when trained from scratch with and without mask sparsity regularization.** Each panel compares BRICKS-WM with a variant that omits $\mathcal{R}_{\texttt{mask}}$ on the indicated DMC task.

slots. BRICKS-WM uses distinct learnable slot initializations and asymmetric decoders to encourage these separate roles. The mask penalty provides an additional inductive bias based on the assumption that the agent typically occupies a smaller image region than the background. By penalizing the total mass of the mask generated by the designated agent slot, it encourages a spatially localized agent component.

Figure 8 compares BRICKS-WM trained from scratch with the variant denoted BRICKS-WM w/o $\mathcal{R}_{\texttt{mask}}$. The two Walker tasks show broadly similar mean learning curves with and without regularization. With regularization, the mean final return is lower on *Cheetah Run* and higher on *Hopper Hop* in the displayed runs. These results indicate that the effect on control performance is task-dependent. We next examine the penalty's intended role in shaping the spatial decomposition.

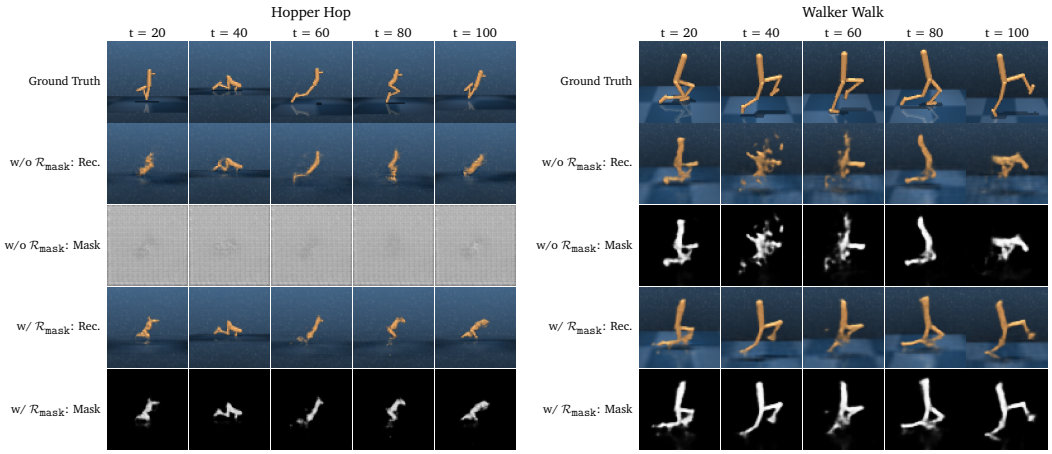


Figure 9: **Effect of $\mathcal{R}_{\texttt{mask}}$ on spatial decomposition at 105K source-training steps.** All four models are trained from scratch and evaluated at the same training budget. Within each task group, rows show *Ground Truth* (the input image), *Rec.* and *Mask* without $\mathcal{R}_{\texttt{mask}}$, followed by *Rec.* and *Mask* with $\mathcal{R}_{\texttt{mask}}$. *Rec.* denotes the composite RGB reconstruction. *Mask* denotes the decoder's soft blending mask $m_t \in [0,1]^{H\times W}$, with brighter pixels assigning greater weight to the agent component. Both variants use a blending mask, and the ablation removes only its sparsity penalty. For each task, the two models receive identical observations and actions from a fixed evaluation episode generated by a policy obtained late in training with the regularizer. The values $t = 20, 40, 60, 80, 100$ index observations in that episode. Each reconstruction is conditioned on the current image rather than generated by open-loop prediction. The examples were selected to show visible differences. *Hopper Hop* uses seed 1 in both conditions. *Walker Walk* uses seed 3 with the regularizer and seed 2 without it.

Spatial Module Assignment. Accurate composite reconstruction alone does not uniquely determine the spatial allocation of information between modules. Figure 9 illustrates how the sparsity penalty

biases this allocation toward a localized agent component. In the selected *Hopper Hop* pair, the unregularized mask assigns substantial weight to the agent component across background regions, whereas the regularized mask concentrates around the robot. The mean soft-mask value, averaged over all pixels in the first 150 observations, is 0.6855 without the penalty and 0.0203 with it. The corresponding mean squared errors of the composite reconstructions are similar at 0.002731 and 0.002685, respectively. These errors are computed over the same observations using RGB values in $[0, 1]$. Thus, similar reconstruction errors can accompany substantially different spatial decompositions, motivating an explicit inductive bias for module assignment. In the selected *Walker Walk* examples, the unregularized model shows fragmented or missing torso structure, particularly at $t = 40$ and $t = 100$, while the regularized model retains a more complete outline. These examples illustrate how $\mathcal{R}_{\texttt{mask}}$ can influence spatial decomposition early in training. Because the examples are selected and the Walker comparison uses different training seeds, they do not quantify an average improvement in segmentation or control performance.

D.4 Source–Target Interface Alignment through the Adapter

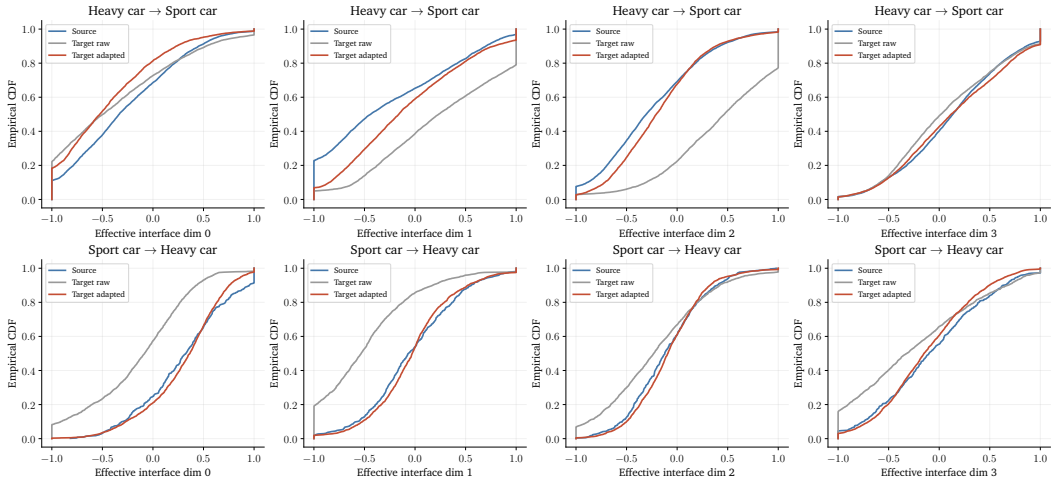


Figure 10: **Marginal interface alignment before and after the adapter.** Columns correspond to the four interface dimensions. Empirical cumulative distribution functions (CDFs) compare source interface means, target interface means at the adapter input (*Target raw*), and the outputs obtained by applying the adapter to those same target means (*Target adapted*). All plotted values are clipped to $[-1, 1]$. For each target run, the raw and adapted values come from the same trained model using deterministic posterior inference. Source models use seed 3 at the 505K checkpoint, and target models use their final saved checkpoints. The top row shows *Heavy Car* $\rightarrow$ *Sport Car* using target seed 3, selected from four completed runs. For each model, we use ten final-evaluation episodes and retain all observations after the first ten. The bottom row shows *Sport Car* $\rightarrow$ *Heavy Car*, pooling four target seeds with three episodes per run and observation indices 10–179. The source uses three episodes and the same observation window. Each model is evaluated on its own trajectories, and the two rows use different sampling windows.

The adapter is intended to translate the target agent's interface into coordinates compatible with the reused background dynamics. Figure 10 examines this role by comparing the marginal distributions of source interface means, target interface means, and the adapter outputs computed from those target means. For *Sport Car* $\rightarrow$ *Heavy Car*, all four marginals move closer to the source, and the Wasserstein-1 distance averaged over dimensions decreases from 0.3015 to 0.0525. In the selected *Heavy Car* $\rightarrow$ *Sport Car* run, the corresponding distance decreases from 0.3366 to 0.1166. Three dimensions move closer to the source, while dimension 0 moves farther away. The adapter reduces the average marginal discrepancy in both displayed comparisons, consistent with its intended role in aligning interface inputs with the retained background transition core. These results characterize the learned transformation through the marginal distributions of clipped interface means and adapter outputs. Because the comparison uses the inputs and outputs of the same trained adapter, it does not isolate

the effect of including the adapter during training. The stronger condition of history-conditioned distribution matching in Proposition 4.3 is not assessed by these marginal plots.

D.5 Interface Regularization and the Rate–Performance Trade-off

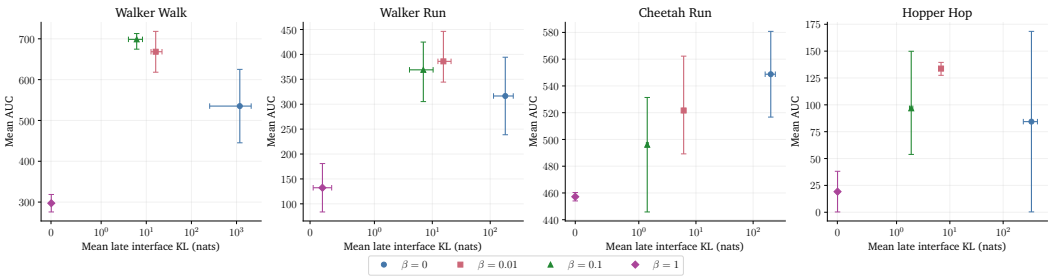


Figure 11: **Interface KL and control performance across regularization strengths.** Each point summarizes the available runs trained from scratch for one source task at a fixed coefficient $\beta_{\mathcal{I}}$ (denoted β in the legend). The horizontal coordinate is the conditional interface KL to $\mathcal{N}(0, I)$, averaged over 400K–500K environment steps and then across seeds. The vertical coordinate is the mean normalized return AUC at a 500K training budget. Horizontal and vertical error bars show marginal 95% bootstrap confidence intervals for the respective means, computed using the same seed resamples for both coordinates. Each task uses two seeds per coefficient at $\beta_{\mathcal{I}} \in \{0, 1\}$ and four per coefficient at $\beta_{\mathcal{I}} \in \{0.01, 0.1\}$, except *Walker Run*, which uses six seeds at 0.1. The horizontal axis uses a symmetric logarithmic scale.

The coefficient $\beta_{\mathcal{I}}$ controls the strength of regularization, whereas the measured conditional KL quantifies the deviation of the learned interface distribution from the canonical prior. Figure 11 relates this empirical interface rate to control performance during source training. Without regularization, the mean KL over the late training interval ranges from approximately 182 to 1,182 nats across tasks. At $\beta_{\mathcal{I}} = 0.01$, the mean KL falls to approximately 6–16 nats, while mean AUC increases relative to the unregularized setting on both Walker tasks and *Hopper Hop*. Stronger compression does not consistently improve control performance. At $\beta_{\mathcal{I}} = 1$, the interface distribution approaches the prior, but mean AUC is the lowest among the tested settings on all four tasks. The coefficient associated with the highest observed mean AUC also depends on the task. *Walker Walk* achieves its highest mean AUC at 0.1, whereas *Cheetah Run* achieves its highest mean AUC without regularization. These results support selecting the regularization strength by considering control performance alongside interface KL. Here, "rate" refers to the measured conditional KL to a fixed prior. This analysis of source training does not directly estimate mutual information or evaluate transfer performance.

D.6 Evolution of the Interface Distribution during Training

Figure 12 tracks the evolution of the interface distribution and complements the aggregate rate–performance comparison. The curves begin to diverge near the end of reconstruction warmup. Warmup completion is first recorded at 55K environment steps in all 50 runs. This temporal association does not establish a causal effect of warmup. With $\beta_{\mathcal{I}} = 0$, KL remains large late in training and generally increases, while the conditional standard deviation becomes small. Together, these statistics indicate sharply concentrated conditional distributions that differ substantially from the canonical prior. At $\beta_{\mathcal{I}} = 1$, low KL and conditional standard deviations near one are consistent with the interface distribution approaching the canonical Gaussian. Intermediate coefficients yield nonzero KL and levels of conditional variability that differ across tasks. Because Gaussian KL depends on both the conditional mean and variance, KL alone cannot attribute changes to either component. The lower row reports the conditional scale directly, helping to distinguish changes in this scale from other contributions to KL. This statistic describes variability within the conditional interface distribution, while the shaded ranges describe variability across training seeds. The trajectories illustrate how the realized interface rate and conditional variability depend on the regularization coefficient, task, and stage of training.

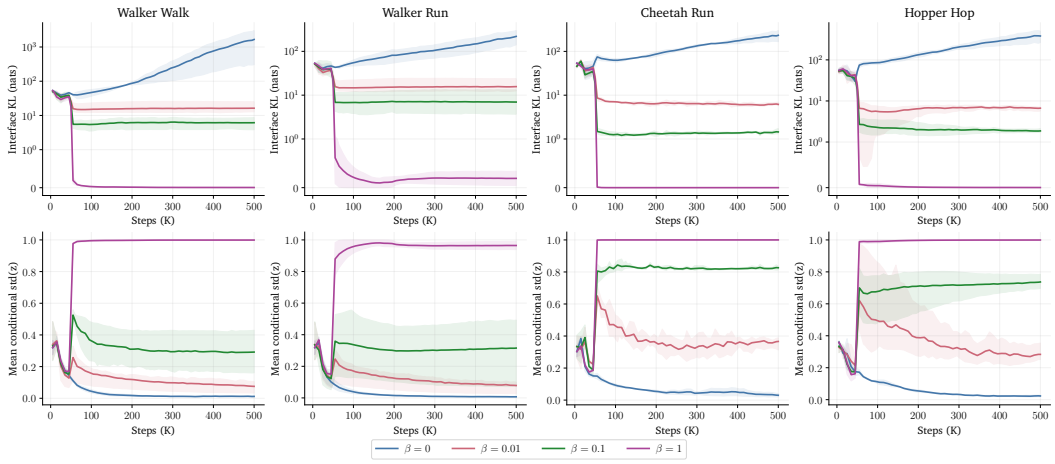


Figure 12: **Interface statistics throughout source training.** The top row shows the conditional interface KL to $\mathcal{N}(0, I)$. The bottom row shows the conditional standard deviation of the Gaussian interface distribution, averaged over interface dimensions and training samples. Columns correspond to the four DMC tasks and use the same training runs as Figure 11. Lines show means across seeds. Shaded regions show the range from the minimum to the maximum across seeds, rather than confidence intervals. *Steps (K)* denotes environment steps in thousands. The KL axes use a symmetric logarithmic scale.