DREAMSPI: WORLD MODELS FOR SAFE POLICY IMPROVEMENT THROUGH LATENT IMAGINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In model-based reinforcement learning (MBRL), world models promise to support policy improvement with fewer environment interactions. Safe policy improvement (SPI) can reason about such updates, but its guarantees are local to the behavior policy and difficult to combine with changing encoders, deep representations, and replay-heavy model learning. We introduce `DreamSPI`, a principled on-policy MBRL algorithm that translates SPI locality into practical updates,while updating the encoder, model, and latent actor-critic. It tokenizes spatial encoder outputs, predicts categorical next-token embeddings, imagines fixed-horizon latent rollouts from on-policy vectorized data, and stabilizes coupled updates with Kronecker-factored optimization. On ALE-57 and Octax, `DreamSPI` outperforms strong baselines while ablations identify on-policy latent imagination and the token-embedding/codebook interface as the key design.

1 INTRODUCTION

Model-based reinforcement learning (MBRL) learns a *model of the environment* to improve the agent's policy. *World models* (Ha & Schmidhuber, 2018; Dawid & LeCun, 2023; Delgrange, 2026) make this idea practical by learning to predict the dynamics directly in a latent space, as in `PlaNet` and `Dreamer` (Hafner et al., 2019; 2021; 2025). Most successful approaches obtain environment data coverage through *experience replay buffers*: the agents train on experienced trajectories and optimize from "imagined" rollouts, drawn in the latent space (Janner et al., 2019; Schrittwieser et al., 2020). In this context, *safe policy improvement* methods (SPI) ask whether a candidate policy can be certified not to underperform the behavior policy that generated the data, up to explicit uncertainty or model-error terms (Thomas et al., 2015; Ghavamzadeh et al., 2016; Laroche et al., 2019; Simão et al., 2020; Castellini et al., 2023).[1] Replay complicates this criterion: the model-training and actor-training distributions may differ from the behavior distribution in the SPI bound. Theoretically, local world-model errors transfer only to policies in a behavior-policy neighborhood (Delgrange et al., 2025); empirically, even MuZero's learned model generalizes poorly to policies outside its training distribution (He et al., 2024). With replay, low average model error can thus hide larger errors on trajectories generated by the current behavior policy.

We introduce `DreamSPI`, which learns a world model from on-policy trajectories collected in B independent environments — a *vectorized environment*. `DreamSPI` learns its policy together with an encoder ϕ, which maps each state s to a latent representation $z = \phi(s)$. The actor, a latent policy $\bar{\pi}$, selects actions from z, giving $\pi(a \mid s) = \bar{\pi}(a \mid z)$. The world model uses the same representation. `DreamSPI` treats *simulated trajectories* ("imagined" rollouts) as *local* data: useful only near the behavior distribution used to fit the model. Aggressive policy updates can move imagined rollouts beyond the region where the model was fitted. Changes to the encoder also change the representations used by both the actor and the world model. If these components do not adapt together, the model's predictions can become inaccurate and mislead the actor. `DreamSPI` keeps the updated composed policy $\bar{\pi}^{new} \circ \phi$ close enough to the actor that its model-error estimates remain meaningful for exploiting the world model and improving the policy.

`DeepSPI`, a deep representation learning algorithm for RL, targets this specific problem (Delgrange et al., 2025). It replaces global tabular SPI requirements by local reward and transition losses under

[1] Here, "safe" has the SPI meaning: the update is judged relative to the behavior-policy return and explicit local model-error terms. It is not a deployment-safety or constrained-MDP claim.

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.

the behavior policy, paired with a next-policy neighborhood that implicitly constrains changes to the encoder. `DeepSPI`'s main algorithm uses the model only to regularize the encoder; the policy is still updated from real-world trajectories. `DreamSPI` addresses this gap by combining policy updates from real and imagined trajectories. At each iteration, `DreamSPI` learns from on-policy trajectories, i.e., collected by the current policy, without replay. Building on SPI principles, it constrains actor and encoder updates to keep the resulting policy close to the policy that collected these data. Our results suggest that training on imagined rollouts can improve on strong on-policy baselines, even when the world model learns only from on-policy trajectories.

Delgrange et al. (2025) found that combining `DeepSPI` with imagination was less effective than learning from real trajectories alone. Our encoder organizes visual features in the form of *token embeddings*: vectors associated with spatial locations. The world model predicts the next representation from a learned finite set of candidate embeddings, the *codebook*. This limits its possible predictions while letting the actor use distances between latent states. The actor and critic map these embeddings to action scores and value estimates through a single linear layer, the *readout*. During imagination, only this layer is updated; the encoder and model remain fixed. Transition predictions are matched to the current encoder's outputs, without reconstruction or detached targets (Grill et al., 2020; Assran et al., 2023; Bardes et al., 2024). We optimize joint real-data updates with `Kron` (Pooladzandi & Li, 2024), whose benefits for non-stationary learning were demonstrated by Creus Castanyer et al. (2025). This challenge is particularly acute here: the encoder changes the representation, while the world model changes the rewards and transitions used to train the actor. `Kron` adapts updates to the differing curvature and gradient scales of these evolving components.

Our contributions are:

1. **Algorithm.** We introduce `DreamSPI`, an on-policy algorithm that learns and exploits a world model without replay, using SPI-guided updates of the encoder, model, and policy.
2. **Principled design.** We ground `DreamSPI`'s design in SPI theory, explaining how the codebook and readout-only updates support model regularity and controlled policy changes.
3. **Empirical evidence.** We show gains over strong on-policy baselines on stochastic ALE (Bellemare et al., 2013) and Octax (Radji et al., 2026). Atari-5 (Aitchison et al., 2023) ablations show that these gains depend on the representation, readout-only updates, imagination, and optimizer.

2 The Recipe Starts Here: Background and Problem Setting

To model the environment, we consider the standard *Markov Decision Process* (MDP) framework $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$ Puterman (1994) with *initial-state distribution* ρ_0, *discount* $\gamma \in (0, 1)$, *value, action-value, and advantage* notation (V^π, Q^π, A^π), where π is a *policy* mapping states from $\mathcal{S}$ to distributions over actions in $\mathcal{A}$ (cf. Appendix A for details). We write ν^π for the *state-action visitation distribution*. The goal of the agent is to maximize the *return* $J_{\mathcal{M}}(\pi) = \mathbb{E}_{s_0 \sim \rho_0}[V^\pi(s_0)]$.

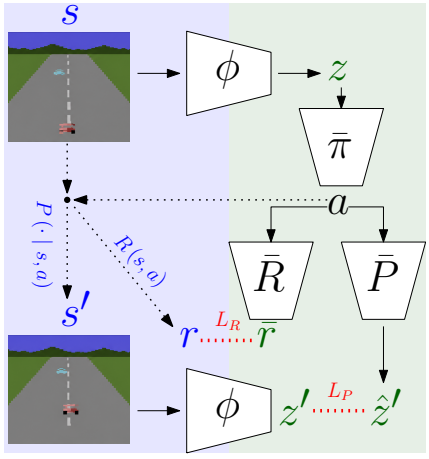


Figure 1: The representation ϕ, reward $\bar{R}$, and transition $\bar{P}$ models are trained from real data $(s, a, r, s') \sim \nu^{\pi_b}$.

In high-dimensional environments, an encoder $\phi : \mathcal{S} \to \mathcal{Z}$ maps a raw state s to $z = \phi(s)$, and a latent policy $\bar{\pi}$ is deployed through the composed policy $\pi(a \mid s) = \bar{\pi}(a \mid \phi(s))$. Changing ϕ changes the deployed policy even if $\bar{\pi}$ is fixed. To plan in this representation, we need a model of how it evolves. An arbitrary learned representation need not make the encoded real process Markov (Hafner et al., 2021). We make the Markov assumption explicit: our *planning model* is a latent MDP $\bar{\mathcal{M}} = (\mathcal{Z}, \mathcal{A}, \bar{P}, \bar{R}, \bar{\rho}_0, \gamma)$, with learned reward $\bar{R}(z, a)$ and transition $\bar{P}(\cdot \mid z, a)$ evaluated at $z = \phi(s)$, so we can measure and control model errors in this latent space. We call the encoder ϕ the *representation*, and the latent MDP $\bar{\mathcal{M}}$ the *abstraction* learned over that representation. For control, the latent return $J_{\bar{\mathcal{M}}}(\bar{\pi})$ must predict the deployed policy's environment return $J_{\mathcal{M}}(\bar{\pi} \circ \phi)$. The reward and transition losses below bound the difference between these returns.

We study the link between $\mathcal{M}$ and $\bar{\mathcal{M}}$ through *local reward and transition losses* under a *behavior policy* π_b, namely the composed policy that generated the current data: $\pi_b = \bar{\pi} \circ \phi$, where $\bar{\pi}_b$ is the latent actor used to collect this data. For a real transition (s_t, a_t, r_t, s_{t+1}), with observed reward $r_t = R(s_t, a_t)$, draw $\hat{z}_{t+1} \sim \bar{P}(\cdot \mid \phi(s_t), a_t)$ and define the *sample errors*

$$\ell_{R,t} = \left| r_t - \bar{R}(\phi(s_t), a_t) \right|, \qquad \ell_{P,t} = \left\| \phi(s_{t+1}) - \hat{z}_{t+1} \right\|. \tag{1}$$

The corresponding *population losses* average over the behavior distribution:

$$L_R^{\pi_b} = \mathbb{E}_{(s_t,a_t)\sim\nu^{\pi_b}} \left[\ell_{R,t} \right], \qquad L_P^{\pi_b} = \mathbb{E}_{\substack{(s_t,a_t)\sim\nu^{\pi_b},\, s_{t+1}\sim P(\cdot|s_t,a_t) \\ \hat{z}_{t+1}\sim\bar{P}(\cdot|\phi(s_t),a_t)}} \left[\ell_{P,t} \right].$$

We predict the next latent state rather than reconstructing observations (in contrast to Ha & Schmidhuber, 2018; Hafner et al., 2019), and use the current encoder rather than a detached target (in contrast to Grill et al., 2020; Assran et al., 2023; Bardes et al., 2024). The question is then how $L_R^{\pi_b}$ and $L_P^{\pi_b}$ can support improvement while ϕ, $\bar{R}$, and $\bar{P}$ move. *Deep* SPI provides a part of the answer.

From safe policy improvement to deep SPI. SPI asks whether a candidate can be guaranteed not to underperform the behavior policy (Thomas et al., 2015; Laroche et al., 2019). ***Deep*** SPI trusts a learned model only on that behavior distribution and transfers its errors to a nearby candidate (Delgrange et al., 2025). Concretely, the first ingredient we rely on is a **local policy *neighborhood***. For $\rho_{\pi'|\pi}(s,a) = \pi'(a \mid s)/\pi(a \mid s)$ and $\epsilon \in (0,1)$, the formal neighborhood is

$$\mathcal{N}^\epsilon(\pi) = \left\{ \pi' \;\middle|\; \begin{array}{ll} \operatorname{supp} \pi'(\cdot \mid s) = \operatorname{supp} \pi(\cdot \mid s) & \forall s \in \mathcal{S}, \\ 1 - \epsilon \le \rho_{\pi'|\pi}(s,a) \le 1 + \epsilon & \forall s,\ a \in \operatorname{supp} \pi(\cdot \mid s) \end{array} \right\}. \tag{2}$$

Notice that for $\pi' = \bar{\pi}' \circ \phi'$, the neighborhood implicitly constrains the combined effect of actor and encoder updates on action probabilities. The corresponding policy-improvement step is $\pi_{n+1} \in \arg\max_{\pi' \in \mathcal{N}^\epsilon(\pi_n)} \mathbb{E}_{s\sim\nu^{\pi_n},\, a\sim\pi'(\cdot|s)} \left[A^{\pi_n}(s,a) \right]$. The sequence converges to an optimal policy in finite state-action spaces (Appendix A). This step constrains *where* the next policy may move, enabling SPI. In practice, Delgrange et al. (2025) approximate this pointwise neighborhood with `PPO`-style clipping Schulman et al. (2017): the actor loss replaces the `PPO` advantage with a model-error-aware utility. Using the sample errors in Eq. (1), define $U_t = \hat{A}_t - \alpha_R \cdot \ell_{R,t} - \alpha_P \cdot \ell_{P,t}$ with $\hat{A}_t$ the estimated advantage and $\alpha_R, \alpha_P \ge 0$. For $\rho_t = {\bar{\pi}(a_t|\phi(s_t))}/{\pi_b(a_t|s_t)}$, the clipped surrogate is

$$\mathcal{L}_{\texttt{DeepSPI}} = -\mathbb{E}_t[\min\left(\rho_t U_t,\ \mathrm{clip}\left(\rho_t,\ 1 \pm \epsilon\right) U_t\right)]. \tag{3}$$

The clipping radius approximates the neighborhood of Eq. (2) in the sampled surrogate. The reward loss trains ϕ through the model input $\phi(s_t)$; the transition loss also trains it through the target $\phi(s_{t+1})$. Including both losses in U_t ties these encoder gradients to the ratio-weighted objective used for policy learning. Clipping approximates this constraint; it does not ensure that every update satisfies it.

The second ingredient is **abstraction quality**: how closely returns in the latent MDP match those in the environment. The local losses $L_R^{\pi_b}$ and $L_P^{\pi_b}$ bound this mismatch for any composed policy $\pi^{new} := (\bar{\pi} \circ \phi) \in \mathcal{N}^\epsilon(\pi_b)$ *that remains in the ϵ-neighborhood* of the behavior policy π_b, with $\epsilon \le \epsilon_{\max} := {1}/{\gamma} - 1$ (formal statement in Appendix C). For this bound, assume $\pi_b = \bar{\pi}_b \circ \phi$ for a latent policy $\bar{\pi}_b$, so both policies use the candidate encoder. More formally, we have:

$$J_{\mathcal{M}}(\bar{\pi} \circ \phi) - J_{\mathcal{M}}(\pi_b) \ge J_{\bar{\mathcal{M}}}(\bar{\pi}) - J_{\bar{\mathcal{M}}}(\bar{\pi}_b) - \zeta, \tag{4}$$

where $\zeta \propto {L_R^{\pi_b}}/{\gamma} + K_V L_P^{\pi_b}$ is a *modeling-error* and K_V a constant depending on *regularity* of the world model (for more details, see Eq. 6). Both model losses depend on ϕ, so the modeling-error ζ must be evaluated using the candidate policy's encoder. The bound guarantees improvement in the environment when the gain predicted by the world model exceeds ζ.

The final ingredient is **representation quality**: for any $\eta > 0$, minimizing $L_R^{\pi_b}$, $L_P^{\pi_b}$ makes the value function of the new policy smooth in the latent space with probability at least $1 - \delta(\eta)$:

$$\forall s_1, s_2 \in \mathcal{S}, \quad \left| V^{\pi^{new}}(s_1) - V^{\pi^{new}}(s_2) \right| \le K_V \left\| \phi(s_1) - \phi(s_2) \right\| + \eta, \tag{5}$$

where $\delta(\eta) \propto {1}/{\eta} \left({L_R^{\pi_b}}/{\gamma} + K_V L_P^{\pi_b} \right)$. The statement holds *for any policy* $\pi^{new} = (\bar{\pi} \circ \phi) \in \mathcal{N}^\epsilon(\pi_b)$ *in an* $(\epsilon \le \epsilon_{\max})$*-neighborhood of* π_b. The encoder must retain distinctions that matter for nearby policies: where the bound holds, states with value differences larger than η cannot share an embedding. Reducing the local losses $L_R^{\pi_b}, L_P^{\pi_b}$ lowers the bound on the probability of violating this property.

Deep, safe policy improvement thus requires **controlling the neighborhood radius ϵ**, **the regularity constant K_V**, and **minimizing $L_R^{\pi_b}, L_P^{\pi_b}$** under the data collected *on-policy*, i.e., under π_b.

Regularity and Finite Latent Spaces. Here K_V bounds how fast the *world model's* value function changes between latent states. We assume Lipschitz latent reward and transition kernels: $\exists K_{\bar{R}}^{\bar{\pi}}, K_{\bar{P}}^{\bar{\pi}} > 0$ such that $\forall z_1, z_2 \in \mathcal{Z}$,

$$|\bar{R}^{\bar{\pi}}(z_1) - \bar{R}^{\bar{\pi}}(z_2)| \leq K_{\bar{R}}^{\bar{\pi}} \|z_1 - z_2\|, \qquad \mathcal{W}_{\|\cdot\|}\big(\bar{P}^{\bar{\pi}}(\cdot \mid z_1), \bar{P}^{\bar{\pi}}(\cdot \mid z_2)\big) \leq K_{\bar{P}}^{\bar{\pi}} \|z_1 - z_2\|. \quad (6)$$

Here $\mathcal{W}_{\|\cdot\|}$ is the 1-*Wasserstein distance* defined over the Euclidean norm $\|\cdot\|$ (Villani, 2009). Appendix I gives the exact definitions. Although neural networks are always Lipschitz continuous, controlling these constants usually requires a careful architecture design or loss penalties. When $\gamma K_{\bar{P}}^{\bar{\pi}} < 1$, the latent value function is Lipschitz with $K_V = {K_{\bar{R}}^{\bar{\pi}}}/{(1-\gamma K_{\bar{P}}^{\bar{\pi}})}$. For *latent codes* made of one-hot categorical variables (as in `Dreamer`, cf. Hafner et al., 2021), with code norm $\|z_1 - z_2\|_{\text{code}} = \mathbf{1}\{z_1 \neq z_2\}$, Wasserstein boils down to *total variation* with $K_{\bar{P}}^{\bar{\pi}} \leq 1$, $K_{\bar{R}}^{\bar{\pi}} \leq 2\bar{R}_{\max}$ when $\left\|\bar{R}\right\|_\infty \leq \bar{R}_{\max}$, and $K_V \leq {2\bar{R}_{\max}}/{(1-\gamma)}$, which thus gives known Lipschitz constants. The price is that plain codes use an "all or nothing" geometry: Eq. (5) can separate unequal codes, but it no longer expresses that nearby observations should induce nearby latent values.

Why `DeepSPI` is not yet `DreamSPI`. The practical `DeepSPI` algorithm of Delgrange et al. (2025) uses transition and reward predictors to regularize the representation through Eq. (5). Their imagination variant trains the value function and policy with `PPO` on **imagined rollouts** from a **discrete world model**, but is not competitive with the baselines. The discrete approach is attractive for the reason above: categorical codes make the regularity constant K_V explicit and are successful in model-based RL (e.g., Hafner et al., 2021). However, the code metric treats all distinct codes as equally far apart, so imagined rollouts lose the notion that some latent states are closer than others. Using the learned world model for planning through Eq. (4) adds a second difficulty: model losses are measured under the current behavior policy, while the encoder, world model, and actor defining the next behavior policy move together. This compounds the ordinary non-stationarity of deep RL (Creus Castanyer et al., 2025) with planning in a learned, changing latent environment.

3 DREAMSPI: THE GOLDEN TICKET TO PURE IMAGINATION

The aim is to make policy improvements in the world model carry over to the environment. Our principled design builds upon SPI's model and representation guarantees to address two difficulties: *combining the regularity of finite predictions with distances the actor can use*, and *improving the policy in imagination without changing the encoder or model*.

`DreamSPI` learns the encoder and world model from on-policy trajectories using the local objective of Eq. (3). Our encoder organizes state features into *token embeddings*, one vector per spatial location in its feature map. Together, these vectors form the latent state used by the actor and world model. The transition model predicts the next latent state from a learned finite set of candidate embeddings, the *codebook*. The codebook limits the model's possible predictions; the embedding vectors give the actor a notion of distance between latent states. Imagined rollouts update only the final actor–critic linear layer while keeping the encoder and model fixed. Finally, we use `Kron` (Pooladzandi & Li, 2024) to stabilize updates of the encoder, world model, and actor–critic from the same real data.

In the following, we use stacked images, with a convolutional network (CNN) (Fukushima, 1988) as the first part of our encoder. The design applies to any input that can be mapped to token embeddings.

3.1 THE RECIPE: CURRENT DATA, FROZEN MODELS, AND LOCAL IMAGINATION

`DreamSPI` repeats two phases (Fig. 2; pseudocode in Appendix E). First, it collects T consecutive transitions from each of B parallel environments, producing a new $B \times T$ batch under a behavior-policy snapshot. The real batch trains the encoder and world model with $\mathcal{L}_{\texttt{DeepSPI}}$ (Eq. (3)). With `dyna` enabled (Sutton, 1990), it also trains the actor and critic; otherwise, they learn only in imagination. The encoder and model learn to predict rewards and next latent states. Clipping discourages large changes in action probabilities relative to the behavior policy.

Then comes the *imagination phase*, in which the encoder and world-model parameters are frozen. From each of the BT encoded states in the batch, we simulate H steps in the world model. These trajectories train only the final actor–critic linear layer. Under the assumptions of Eq. (4), a gain in model return greater than ζ also implies a gain in the environment. As we minimize $\mathcal{L}_{\texttt{DeepSPI}}$ in the real-data phase, the model errors $\ell_{R,t}$ and $\ell_{P,t}$ decrease, which reduces ζ and makes it easier to certify

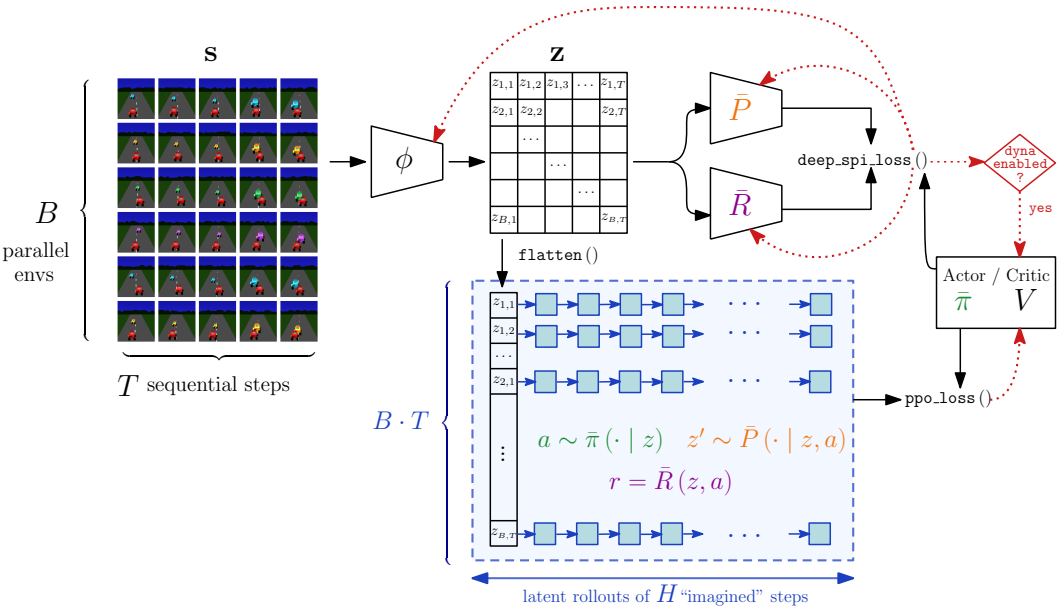


Figure 2: The real $B \times T$ batch updates the encoder and world model with Eq. (3); when `dyna` is enabled, it also updates the actor–critic readout. The batch supplies BT new starts for imagined rollouts of H steps, which update only the readout with $\mathcal{L}_{\texttt{DeepSPI}}$ (Eq. (3)). Red dotted arrows denote gradient-update paths; each arrowhead points to the module receiving that update.

that the next policy improves in the environment. Clipping and a divergence penalty discourage large changes in action probabilities, but do not enforce the neighborhood at every state and action. Appendix N.1 measures how closely the updates satisfy this constraint.

We do not use replay, though SPI itself does not rule it out: the guarantee needs model errors measured under an identified behavior distribution and a candidate policy that stays close to it. A replay mixture would require explicit importance weighting or off-policy correction (Espeholt et al., 2018; Meng et al., 2023). Parallel environments collect data faster and provide BT states from each on-policy batch to start imagined rollouts. Parallelism does not guarantee broader exploration.

3.2 Inside the Codebook Factory: Tokenized Categorical Latents

We now describe how our encoder forms token embeddings and how the world model predicts the next latent state from a finite codebook. Appendix J gives the full construction and proofs. At the top of Fig. 3, our encoder ϕ reshapes its $H \times W \times C$ output into $M = HW$ tokens, each holding the C features of one spatial location. This preserves the encoder's spatial feature groups, which Sokar et al. (2025) identify as a main source of `SoftMoE`'s gains, even with one expert. An optional soft mixture-of-experts layer (`SoftMoE`) processes learned combinations of tokens to form $z \in \mathbb{R}^{M \times C}$. Flattening z gives the input to the actor $\bar{\pi}$ and critic $V^{\bar{\pi}}$.

At the bottom, the world model takes z and action a: $\bar{R}$ predicts the reward $\bar{r}$, while $\bar{P}$ produces logits $o \in \mathbb{R}^{K \times L}$ for K categorical variables with L classes each. Straight-through Gumbel-softmax (Bengio et al., 2013; Jang et al., 2017; Maddison et al., 2017) samples one class per row, giving a one-hot matrix $u^{\omega} \in \{0,1\}^{K \times L}$ and code $\omega \in \Omega = \{1, \ldots, L\}^{K}$. The decoder $\Psi_{F,G}$ maps this code back to token embeddings using two learned tables: $F \in \mathbb{R}^{M \times K}$ combines the choices at each position, then $G \in \mathbb{R}^{M \times L \times C}$ maps them to features using a separate embedding table at each position. Thus the same choice can describe different features at different positions. The result $\hat{z} = \Psi_{F,G}(\omega)$ (z' in the figure) has the same shape as z, so the actor can act on model predictions.

The model predicts a distribution over the finite codebook $\mathcal{C}_{F,G}$. Only predictions pass through a categorical bottleneck; the actor receives continuous features from real observations. One-hot outcomes within each variable are equally far apart. Learning F, G allows different codes to map to nearby embeddings, so the transition loss penalizes near misses less than distant predictions.

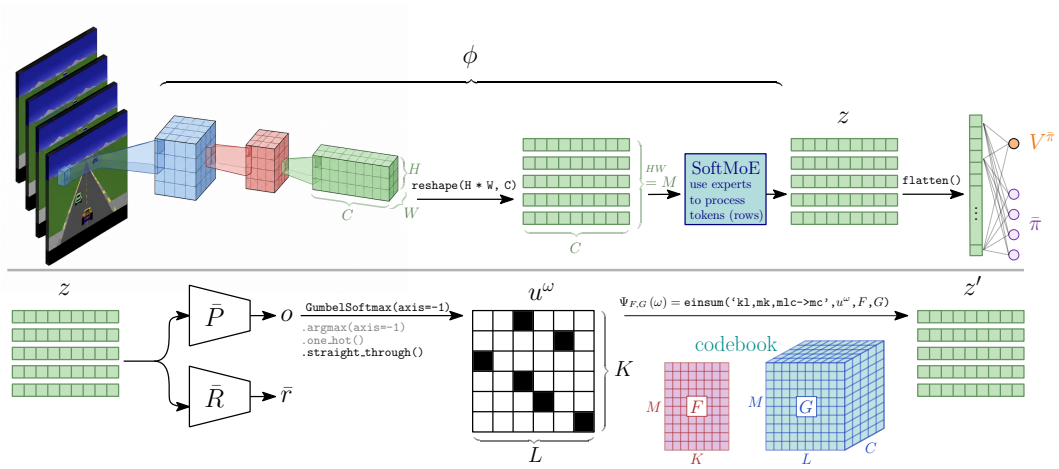


Figure 3: `DreamSPI` architecture. The encoder maps frames to token embeddings $z \in \mathbb{R}^{M \times C}$ for the linear actor and critic. The transition model samples categorical codes and decodes them into predicted next embeddings using F and G. The transition loss matches these predictions to the next encoded observation and trains the encoder together with F and G (Fig. 1).

Claim 1 (Codebook regularity)**.** *Fix a codebook $\mathcal{C}_{F,G}$ with minimum separation $d > 0$ and diameter D. On these embeddings, the world model's policy-induced reward and transition are Lipschitz, with $K^{\bar{\pi}}_{\bar{R},\mathcal{C}} \leq 2\bar{R}_{\max}/d$ and $K^{\bar{\pi}}_{\bar{P},\mathcal{C}} \leq D/d$ (Theorem 1). Here $\bar{R}_{\max}$ is the largest absolute reward prediction over $\mathcal{C}_{F,G} \times \mathcal{A}$; this maximum exists because both sets are finite.*

The codebook thus yields finite Lipschitz bounds while retaining learned distances between states. Their size depends on the diameter and reward scale relative to the separation between embeddings. Appendix J explains how to use these bounds in SPI and states the required conditions.

3.3 No Touching the Machinery: Readout-Only Updates

Small reward and transition losses make the next value function ϵ-Lipschitz in the encoder output (Eq. (5)). The actor–critic therefore has **no hidden layer**: a single affine readout maps flattened z to action scores and values (Fig. 3). During imagination, **only this linear readout is updated**; the encoder and model remain fixed. The importance-sampling ratio (ISR) measures how much each action's probability changes relative to the behavior policy. Large changes can favor trajectories on which the model is poorly trained. Keeping the ISR within $1 \pm \epsilon$ at every state and action places the policy in the SPI neighborhood (Eq. (2)). Under the assumptions of Eq. (4), model errors measured under the behavior policy then bound how much imagined gains can exceed real gains.

Claim 2 (Readout-only ratio control)**.** *For a softmax actor on fixed latent states, limiting changes to its readout weights and biases bounds the ISR above and below.* See Appendix D.

3.4 Keeping the Ingredients from Melting: Non-stationarity and Stabilization

The remaining difficulty is optimization. RL is already non-stationary: the data distribution, world model, and bootstrapped targets move with the policy. Model-based RL also trains the actor and critic in a changing world model. Recent evidence shows that `Kron` helps to stabilize large RL agents (Creus Castanyer et al., 2025). Kron is a second-order optimizer using Kronecker-factored preconditioning. Token maps, model logits, value targets, and policy readouts can have very different curvature and gradient scales, making a shared step size hard to choose. Kron corrects for this curvature through matrix-valued preconditioning, whereas Adam rescales each coordinate separately Kingma & Ba (2015). Appendix H gives the update, normalization, and activation details.

4 Evaluating the DreamSPI Recipe

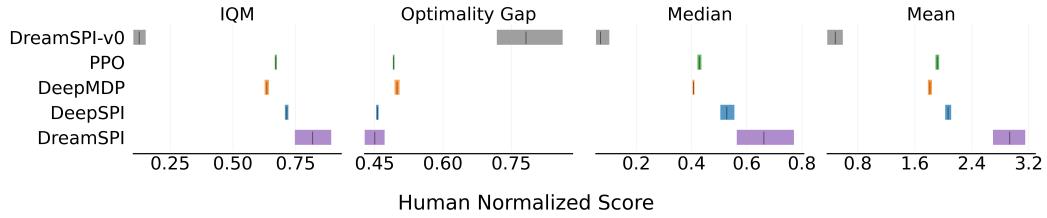


Figure 5: Aggregate ALE-57 results with 95% stratified bootstrap confidence intervals (Agarwal et al., 2021b). Higher mean, median, and IQM are better; lower optimality gap is better.

We ask three questions: 1) Does `DreamSPI` outperform baselines on the *Arcade Learning Environment* (ALE) (Bellemare et al., 2013) when all agents use mirror-learning or regularized policy objectives and learn only from on-policy trajectories, without replay buffers? 2) Do the results transfer to *Octax* Radji et al. (2026), with a different simulator and game mechanics? 3) Which ingredients explain the gain?

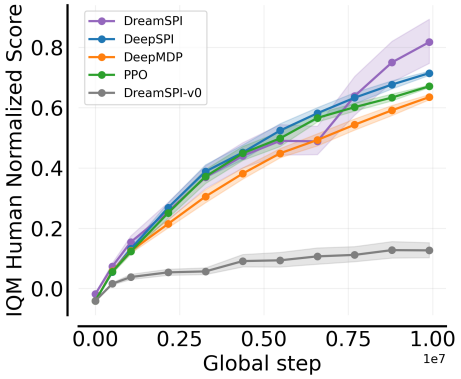


Figure 4: ALE-57 sample efficiency.

Experimental setup. We compare `DreamSPI` with vectorized on-policy baselines on 57 ALE games (Machado et al., 2018) and 15 visual CHIP-8 games from Octax. To test planning when actions have uncertain outcomes, we use sticky actions in both suites: with probability $p = 0.3$, the environment repeats the previously executed action instead of the chosen one. ALE also uses random no-op starts of up to $n = 60$ frames. Both suites use stacks of 4 frames as input.

The baselines are `PPO` (Schulman et al., 2017); `PPO` with auxiliary reward/transition representation losses from `DeepMDP`, unconstrained by the SPI neighborhood (Gelada et al., 2019); `DeepSPI`, which clips these local losses within the SPI-style policy update; and `DreamSPI-v0`, the `DeepSPI` variant that updates `PPO` from imagined latent trajectories (Delgrange et al., 2025) without our combination of token embeddings, readout-only imagination, and `Kron`.

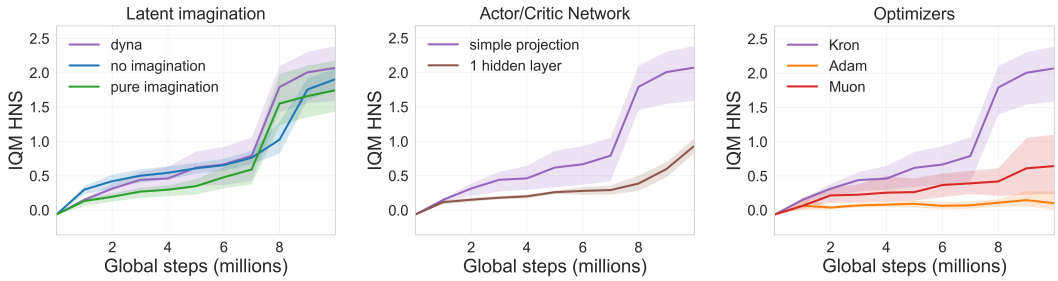
We report *human-normalized score* (HNS) on ALE and *agent-normalized score* (ANS) on Octax. We use five seeds per algorithm and game (three for Atari-5 ablations). ALE gives $5 \times 57 = 285$ runs per algorithm; excluding *Adventure*, where all agents score zero, leaves 280 scores. Mean and median use per-game seed means. Our primary metric, the *interquartile mean* (IQM), averages the middle 50% of scores across runs and games, limiting outliers' influence while using more scores than the median. Following Agarwal et al. (2021b), we report 95% stratified bootstrap confidence intervals from resampling runs within each game, rather than binary significance tests. Appendices L–O report per-game curves, alternative aggregates, unsuccessful ablations, and policy and model diagnostics.

On an Nvidia H200 with 64 parallel environments, real rollouts of 8 steps, and imagined rollouts of 16 steps, the median imagination throughput is 36,177 latent steps/s versus 1,177 real-environment training steps/s, so new latent rollouts are cheap enough to regenerate from each on-policy batch.

Full-suite results on ALE. Figure 5 gives aggregate results after 10M training steps per run on ALE-57. `DreamSPI` has the highest IQM; its 95% confidence interval lies entirely above those of all compared baselines. The gap to `DreamSPI-v0` shows the value of combining imagination with our latent interface, readout restriction, and optimizer.

On the ALE-57 IQM HNS curve (Fig. 4), `DreamSPI` has the highest final IQM, with `DeepSPI` closest. The aggregate gain varies across games. Our post-hoc analysis in Appendix O finds the largest gains for `DreamSPI` in games with frequent rewards. Such feedback may help the actor compare alternative actions in short imagined rollouts.

Appendix Tab. 5 reports final HNS; Appendix L.4 gives full `PPO`/`DeepMDP` comparisons, with IQM and median views to check whether the ranking changes with the aggregation method.



(a) Latent imagination. (b) Projection-only readout. (c) Optimizer.

Figure 7: Atari-5 IQM-HNS control and optimizer ablations. Performance drops when we disable `Dyna`, add an actor–critic hidden layer, or replace `Kron` with `Muon` or `Adam`.

Beyond ALE: Octax validation. On Octax (Radji et al., 2026), `DreamSPI` has the highest IQM ANS at most checkpoints, including the last (Fig. 6). Appendix L.3 defines ANS; Appendix L.5 reports aggregate scores, per-game learning curves, and comparisons with each baseline.

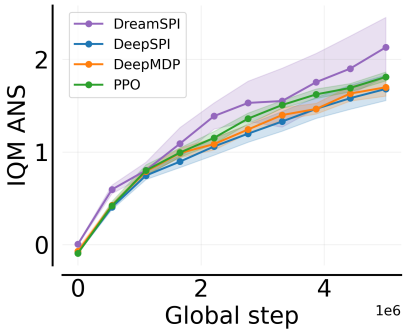


Figure 6: Octax-15 sample efficiency.

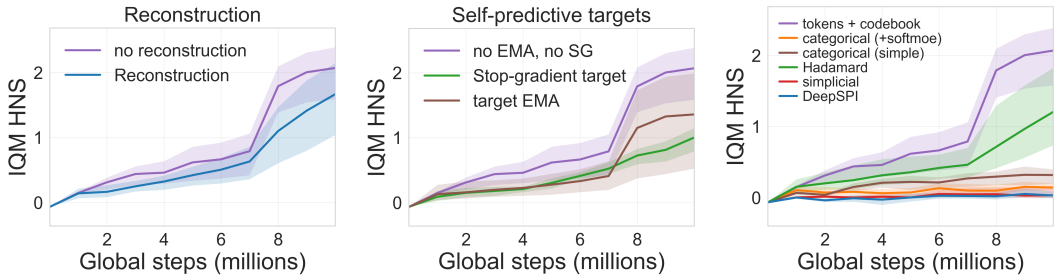
Which components matter? We use Atari-5 (Aitchison et al., 2023), a five-game proxy for the full ALE suite, to reduce ablation cost (Figs. 7–8). **Removing latent imagination and disabling `Dyna`** (Fig. 7a) moderately reduces aggregate performance. `DreamSPI` dominates in all 5 games except *BattleZone* (Appendix Fig. 24). Our ingredients thus transfer both to *pure representation learning* (`DeepSPI`) and *pure imagination* (separating representation learning on real data from actor/critic updates on imagined rollouts). Training the actor and critic on real and imagined trajectories gives the best results, combining deep SPI's representation learning with policy improvement in the model.

Adding a nonlinear hidden layer between the embeddings and final actor–critic outputs reduces performance (Fig. 7b). This extra nonlinear transformation changes how the actor uses the frozen representation, making it harder to control action-probability changes during imagination.

Optimizer choice is decisive. **Replacing `Kron` with `Muon`/`Adam`** reduces IQM. `Muon` also exploits matrix structure by orthogonalizing updates and has recently been used in large-scale neural-network training (Liu et al., 2025b). Its drop suggests that orthogonalized updates alone do not replace curvature-aware preconditioning when encoder updates change the actor's inputs and world-model updates change the rewards and transitions used for learning (Appendix H).

Adding a reconstruction loss decreases performance (Fig. 8a), supporting latent prediction over observation reconstruction (Appendix M). **The encoder-target ablations** test whether stabilizing prediction targets helps (Fig. 8b): `JEPA`-style latent objectives (Assran et al., 2023; Bardes et al., 2024) often stop gradients through the encoded next observation, or use a target encoder with *exponential moving average weights* (EMA). **Both underperform**, consistent with the theory: **the composed-policy neighborhood already restricts encoder changes**, while reward and transition errors use the policy update's representation. We discuss multi-step prediction and contrastive-style alternatives in Appendix M.1. Appendix M.2 tests direct gradient penalties (Gelada et al., 2019), `WGAN-GP` penalties (Gulrajani et al., 2017), and norm-constrained networks (Anil et al., 2019) to limit how sharply the world model's predictions can change.

Finally, we test our token embeddings and codebook (Fig. 8c; Appendix L.6). *Categorical (simple)* **removes tokenization and learned embeddings**: encoder outputs serve as categorical logits, converted to one-hot codes with straight-through gradients. *Categorical (+softmoe)* uses a mixture of experts before the same straight-through categorical bottleneck (Sokar et al., 2025). Hadamard/hypertanh embeddings (Kooi et al., 2024) and simplicial embeddings (Obando-Ceron et al., 2026) **test bounded and simplex-constrained geometries**. These variants naturally constrain the geometry in Eq. (6): Hadamard bounds each latent coordinate, while simplicial embeddings place each token



(a) Reconstruction. (b) Encoder targets. (c) Representation.

Figure 8: Atari-5 IQM-HNS ablations. Adding reconstruction, stop-gradient, or EMA targets reduces performance. Token embeddings with a learned codebook outperform the other tested representations.

on a probability simplex. Our token embeddings perform best. Under the unit code metric, Eq. (5) treats all distinct codes alike; learned distances instead express which states are closer. Hadamard and simplicial constraints reduce performance, though Hadamard remains competitive. Bounding codebook diameter can help regularity, provided separation is also controlled.

5 Related Work

Safe policy improvement and local control. SPI bounds improvement over a baseline from uncertain data (Thomas et al., 2015; Laroche et al., 2019), while trust-region and regularized methods limit policy changes (Schulman et al., 2015; 2017; Geist et al., 2019; Kuba et al., 2022). `DreamSPI` controls the composed policy $\bar{\pi} \circ \phi$. `DeepSPI`'s main algorithm regularizes the encoder with local model losses (Delgrange et al., 2025); `DreamSPI` also trains the policy in this model.

World models and the distribution problem. `PlaNet`, `Dreamer`, `MBPO`, and `MuZero` use replay or accumulated data for imagination, off-policy learning, or search (Hafner et al., 2019; 2021; 2025; Janner et al., 2019; Schrittwieser et al., 2020). `DreamSPI` instead learns from current-policy batches, measuring model errors and limiting policy changes relative to the same behavior policy.

Learned-model agents. `PILCO` uses uncertainty-aware policy search in low-dimensional models (Deisenroth & Rasmussen, 2011); Imagination-Augmented Agents supply rollouts as policy inputs (Racanière et al., 2017); `SimPLe` trains `PPO` in a video model from stored real states (Kaiser et al., 2020). `V-JEPA 2`, `NE-Dreamer`, and `JEDI` replace reconstruction with latent prediction, using EMA or stop-gradient targets (Assran et al., 2025; Bredis et al., 2026; Lim et al., 2026). `DreamSPI` uses neighborhood-regularized updates without EMA or stop-gradient. Adding either, or reconstruction, reduces Atari-5 performance (Fig. 8). Appendix K gives further related work.

6 Conclusion

World models can reduce real-world interactions by training policies on imagined trajectories. Changes to the encoder, model, and policy can make these updates unreliable. We introduce `DreamSPI`, which uses safe policy improvement (SPI) bounds to guide their joint learning from on-policy data, without replay. These bounds relate gains in the model to gains in the environment through local model errors and limits on policy changes. The model predicts from a learned finite set of embeddings; imagined updates keep the encoder and model fixed and change only the final policy and value-function layers. Curvature-aware optimization stabilizes joint updates on real data. On the Arcade Learning Environment and Octax, `DreamSPI` improves aggregate performance over the compared on-policy baselines. World models learned from on-policy data can improve control. Our ablations support these design choices, while per-game results, confidence intervals, and training diagnostics show how performance varies across games and runs. Future work includes using model errors to guide exploration (Pathak et al., 2017), longer rollouts, and memory for partial observability. Extending SPI to multiple behavior policies or goals could allow one world model to generalize to more tasks and lead to "foundation" world models (Letcher et al., 2026; Delgrange, 2026).

AI USE STATEMENT

Generative AI tools, including large language models and coding agents, assisted with software implementation, data analysis (figure and table production), summarization, and editing the manuscript (polishing). The authors reviewed the AI-assisted work through source and provenance checks, automated tests, independent experimental reruns, statistical validation, citation verification, and direct inspection of the manuscript and generated artifacts. The authors take responsibility for the final content, claims, code, and results.

REPRODUCIBILITY STATEMENT

Appendix E specifies the algorithm, and Appendix F reports the evaluation configurations. The remaining appendices provide proofs, environment-level results, ablations, normalization anchors, uncertainty procedures, and implementation diagnostics.

REFERENCES

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29304–29320, 2021b. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 421–438. PMLR, 2023. URL https://proceedings.mlr.press/v202/aitchison23a.html.

Cem Anil, James Lucas, and Roger B. Grosse. Sorting out lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 291–301. PMLR, 2019. URL http://proceedings.mlr.press/v97/anil19a.html.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael G. Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 15619–15629. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01499. URL https://doi.org/10.1109/CVPR52729.2023.01499.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *CoRR*, abs/2506.09985, 2025. doi: 10.48550/ARXIV.2506.09985. URL https://doi.org/10.48550/arXiv.2506.09985.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *The Fourteenth*

International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=SzXDuBN8M1.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=QaCCuDfBk2.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013. doi: 10.1613/JAIR.3912. URL https://doi.org/10.1613/jair.3912.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

George Bredis, Nikita Balagansky, Daniil Gavrilov, and Ruslan Rakhimov. Next embedding prediction makes world models stronger, 2026. URL https://arxiv.org/abs/2603.02765.

Alberto Castellini, Federico Bianchi, Edoardo Zorzi, Thiago D. Simão, Alessandro Farinelli, and Matthijs T. J. Spaan. Scalable safe policy improvement via Monte Carlo tree search. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3732–3756. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/castellini23a.html.

Pablo Samuel Castro. The formalism-implementation gap in reinforcement learning research. *CoRR*, abs/2510.16175, 2025. doi: 10.48550/ARXIV.2510.16175. URL https://doi.org/10.48550/arXiv.2510.16175.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. MICo: Improved representations via sampling-based state similarity for Markov decision processes. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 30113–30126. Curran Associates, Inc., 2021.

Roger Creus Castanyer, Johan Obando Ceron, Lu Li, Pierre-Luc Bacon, Glen Berseth, Aaron Courville, and Pablo Samuel Castro. Stable gradients for stable learning at scale in deep reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 35051–35087. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/32375260090404f907ceae19f3564a7e-Paper-Conference.pdf.

Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: A path towards autonomous machine intelligence. *CoRR*, abs/2306.02572, 2023. doi: 10.48550/ARXIV.2306.02572. URL https://doi.org/10.48550/arXiv.2306.02572.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In Lise Getoor and Tobias Scheffer (eds.), *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 465–472. Omnipress, 2011. URL https://icml.cc/2011/papers/323_icmlpaper.pdf.

Florent Delgrange. Foundation world models for agents that learn, verify, and adapt reliably beyond static environments. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026)*, Paphos, Cyprus, may 2026. doi: 10.65109/WCEI7331. URL https://dl.acm.org/doi/10.65109/WCEI7331. Blue Sky Ideas Track; Best Blue Sky Paper Award.

Florent Delgrange, Raphaël Avalos, and Willem Röpke. Deep SPI: safe policy improvement via world models. *CoRR*, abs/2510.12312, 2025. doi: 10.48550/ARXIV.2510.12312. URL https://doi.org/10.48550/arXiv.2510.12312.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, Proceedings of Machine Learning Research, pp. 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988. doi: 10.1016/0893-6080(88)90014-7. URL https://doi.org/10.1016/0893-6080(88)90014-7.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=7IzeL0kflu.

Samuel Garcin, Trevor McInroe, Pablo Samuel Castro, Christopher G. Lucas, David Abel, Prakash Panangaden, and Stefano V. Albrecht. Studying the interplay between the actor and critic representations in reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=tErHYBGlWc.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2160–2169. PMLR, 2019. URL http://proceedings.mlr.press/v97/geist19a.html.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 2019.

Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2298–2306, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/9a3d458322d70046f63dfd8b0153ece4-Abstract.html.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5767–5777, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/892c3b1c6dccd52936e27cbd0ff683d6-Abstract.html.

Zhaohan Daniel Guo, Bernardo Ávila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask

reinforcement learning. *CoRR*, abs/2004.14646, 2020. doi: 10.48550/ARXIV.2004.14646. URL https://doi.org/10.48550/arXiv.2004.14646.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pp. 2555–2565. PMLR, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse control tasks through world models. *Nat.*, 640(8059):647–653, 2025. doi: 10.1038/S41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Jinke He, Thomas M. Moerland, Joery A. de Vries, and Frans A. Oliehoek. What model does muzero learn? In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz (eds.), *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 1599–1606. IOS Press, 2024. doi: 10.3233/FAIA240666. URL https://doi.org/10.3233/FAIA240666.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G. M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *J. Mach. Learn. Res.*, 23:274:1–274:18, 2022. URL https://jmlr.org/papers/v23/21-1342.html.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12498–12509, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL https://doi.org/10.1016/S0004-3702(98)00023-X.

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.

Tristan Kenneweg, Philip Kenneweg, and Barbara Hammer. JEPA for RL: investigating joint-embedding predictive architectures for reinforcement learning. *CoRR*, abs/2504.16591, 2025.

doi: 10.48550/ARXIV.2504.16591. URL https://doi.org/10.48550/arXiv.2504.16591.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Jacob E. Kooi, Mark Hoogendoorn, and Vincent François-Lavet. Latent assistance networks: Rediscovering hyperbolic tangents in RL. *CoRR*, abs/2406.09079, 2024. doi: 10.48550/ARXIV.2406.09079. URL https://doi.org/10.48550/arXiv.2406.09079.

Jakub Grudzien Kuba, Christian A. Schröder de Witt, and Jakob N. Foerster. Mirror learning: A unifying framework of policy optimisation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7825–7844. PMLR, 2022.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3652–3661. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/laroche19a.html.

Alistair Letcher, Mattie Fellows, Alexander David Goldie, Jonathan Richens, Jakob N. Foerster, and Oliver Richardson. Inverting the bellman equation: From q-values to world models. *CoRR*, abs/2606.21173, 2026. doi: 10.48550/ARXIV.2606.21173. URL https://doi.org/10.48550/arXiv.2606.21173.

Jing Yu Lim, Rushi Shah, Zarif Ikram, Samson Yu, Haozhe Ma, Tze-Yun Leong, and Dianbo Liu. JEDI: joint embedding diffusion world model for online model-based reinforcement learning. *CoRR*, abs/2605.13013, 2026. doi: 10.48550/ARXIV.2605.13013. URL https://doi.org/10.48550/arXiv.2605.13013.

Jiashun Liu, Zihao Wu, Johan S. Obando-Ceron, Pablo Samuel Castro, Aaron C. Courville, and Ling Pan. Measure gradients, not activations! enhancing neuronal activity in deep reinforcement learning. *CoRR*, abs/2505.24061, 2025a. doi: 10.48550/ARXIV.2505.24061. URL https://doi.org/10.48550/arXiv.2505.24061.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training. *CoRR*, abs/2502.16982, 2025b. doi: 10.48550/ARXIV.2502.16982. URL https://doi.org/10.48550/arXiv.2502.16982.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *J. Artif. Intell. Res.*, 61:523–562, 2018. doi: 10.1613/JAIR.5699. URL https://doi.org/10.1613/jair.5699.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

Walter Mayor, Johan S. Obando-Ceron, Aaron C. Courville, and Pablo Samuel Castro. The impact of on-policy parallelized data collection on deep reinforcement learning networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/mayor25a.html.

Wenjia Meng, Qian Zheng, Gang Pan, and Yilong Yin. Off-policy proximal policy optimization. In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 9162–9170. AAAI Press, 2023. doi: 10.1609/AAAI.V37I8.26099. URL https://doi.org/10.1609/aaai.v37i8.26099.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/NATURE14236. URL https://doi.org/10.1038/nature14236.

Johan Obando-Ceron, Walter Mayor, Samuel Lavoie, Scott Fujimoto, Aaron Courville, and Pablo Samuel Castro. Simplicial embeddings improve sample efficiency in actor-critic agents. In *The Fourteenth International Conference on Learning Representations, ICLR 2026*, 2026. URL https://openreview.net/forum?id=mCpq1GCKxA. ICLR 2026 poster.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pp. 2778–2787. PMLR, 2017. URL http://proceedings.mlr.press/v70/pathak17a.html.

Omead Pooladzandi and Xilin Li. Curvature-informed SGD via general purpose lie-group preconditioners. *CoRR*, abs/2402.04553, 2024. doi: 10.48550/ARXIV.2402.04553. URL https://doi.org/10.48550/arXiv.2402.04553.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887. URL https://doi.org/10.1002/9780470316887.

Sébastien Racanière, Theophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter W. Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5690–5701, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9e82757e9a1c12cb710ad680db11f6f1-Abstract.html.

Waris Radji, Thomas Michel, and Hector Piteau. Octax: Accelerated CHIP-8 arcade environments for reinforcement learning in JAX. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BdeUsYlyIf.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4.

John Schulman. Approximating KL divergence. http://joschu.net/blog/kl-approx.html, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu.

Max Schwarzer, Johan S. Obando-Ceron, Aaron C. Courville, Marc G. Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pp. 30365–30380. PMLR, 2023. URL https://proceedings.mlr.press/v202/schwarzer23a.html.

Thiago D. Simão, Romain Laroche, and Rémi Tachet des Combes. Safe policy improvement with an estimated baseline policy. In Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith (eds.), *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pp. 1269–1277. International Foundation for Autonomous Agents and Multiagent Systems, 2020. doi: 10.5555/3398761.3398908. URL https://dl.acm.org/doi/10.5555/3398761.3398908.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pp. 32145–32168. PMLR, 2023. URL https://proceedings.mlr.press/v202/sokar23a.html.

Ghada Sokar, Johan S. Obando-Ceron, Aaron C. Courville, Hugo Larochelle, and Pablo Samuel Castro. Don't flatten, tokenize! unlocking the key to softmoe's efficacy in deep RL. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=8oCrlOaYcc.

Richard S. Sutton. Integrated modeling and control based on reinforcement learning. In Richard P. Lippmann, John E. Moody, and David S. Touretzky (eds.), *Advances in Neural Information Processing Systems 3, [NIPS] Conference, Denver, Colorado, USA, November 26-29, 1990*, pp. 471–478. Morgan Kaufmann, 1990. URL https://proceedings.neurips.cc/paper/1990/hash/d9fc5b73a8d78fad3d6dffe419384e70-Abstract.html.

Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2380–2388. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/thomas15.html.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6306–6315, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

Cédric Villani. *The Wasserstein distances*, pp. 93–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9_6. URL https://doi.org/10.1007/978-3-540-71050-9_6.

Aleksandar Vujinovic and Aleksandar Kovacevic. ACT-JEPA: joint-embedding predictive architecture improves policy representation learning. *CoRR*, abs/2501.14622, 2025. doi: 10.48550/ARXIV.2501.14622. URL https://doi.org/10.48550/arXiv.2501.14622.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 25476–25488, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d5eca8dc3820cad9fe56a3bafda65ca1-Abstract.html.

APPENDIX CONTENTS

A. Background definitions 19
MDP notation, policy neighborhoods, and the clipped `DeepSPI` loss.

B. Scope notes 20
What "safe" means here, why we compare on-policy agents, and limits of the guarantees.

C. `DeepSPI` environment-to-latent transfer bound 20
How local model errors bound return mismatch, policy improvement, and representation quality.

D. Projection-only ratio control 21
Proof that bounded changes to a linear actor readout control action-probability ratios.

E. Algorithmic details 22
Pseudocode for real-data updates and imagination, temporary buffers, and the `Dyna` option.

F. Hyperparameters 24
Selected ALE and Octax settings and environment-specific configurations.

G. Wall-clock diagnostics 25
Training times on ALE, Octax, and Atari-5; optimizer cost and performance tradeoffs.

H. Stabilizer details 26
Changing training distributions, `Kron` preconditioning, Mish activations, and LayerNorm.

I. Probability metrics used in the bounds 27
Wasserstein distance, total variation, and their relation under the discrete code metric.

J. Token embeddings and codebook regularity 28
Token and codebook construction, Lipschitz bounds, and off-codebook error terms.

K. Further related work 32
SPI, world models, representation learning, self-supervised targets, and optimization.

L. Evaluation details and additional plots 33
Metrics and uncertainty, HNS/ANS definitions and reference scores, ALE and Octax results, and Atari-5 ablations.

M. Auxiliary targets and regularity ablations 49
Reconstruction, target encoders, multi-step prediction, and Lipschitz-network alternatives.

N. Empirical SPI diagnostics 57
Measured policy changes and local reward and transition losses throughout training.

O. Grouped environment diagnostics 59
Post-hoc ALE game groups: where `DreamSPI` gains most and where it struggles.

A Background Definitions

Notation details. In the main text, we consider Markov decision processes, defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$, where ρ_0 is the initial-state distribution, $P(\cdot \mid s, a)$ is the transition kernel, $R(s, a)$ is the deterministic one-step reward, and $\gamma \in (0, 1)$ is the discount factor. A policy $\pi(a \mid s)$ induces trajectories $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot \mid s_t)$, $r_t = R(s_t, a_t)$, and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. Equivalently, it induces a Markov chain on states with transition kernel and expected reward

$$P^\pi(\cdot \mid s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) P(\cdot \mid s, a), \qquad R^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) R(s, a),$$

with sums replaced by integrals when the action space is not finite. We use the same convention for latent MDPs, e.g. $\bar{P}^{\bar{\pi}}(\cdot \mid z)$ and $\bar{R}^{\bar{\pi}}(z)$ denote the transition and reward induced by the latent policy. The value, action-value, advantage, and return are

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s \right], \qquad Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a \right],$$
$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s), \qquad J_{\mathcal{M}}(\pi) = \mathbb{E}_{s_0 \sim \rho_0}[V^\pi(s_0)].$$

A stationary distribution of π is a distribution ξ^π over states satisfying the usual fixed-point equation

$$\xi^\pi(s') = \sum_{s \in \mathcal{S}} \xi^\pi(s) \sum_{a \in \mathcal{A}} \pi(a \mid s) P(s' \mid s, a), \qquad s' \in \mathcal{S},$$

with the analogous integral form in general state spaces. See Delgrange et al. (2025, Appendix A and Remark 3) for the episodic construction and for the extension to settings where a stationary distribution is available. For the behavior policy π_b, we denote the corresponding stationary state-action distribution by ν^{π_b}, defined by

$$\nu^{\pi_b}(s, a) = \xi^{\pi_b}(s) \pi_b(a \mid s).$$

Thus $(s, a) \sim \nu^{\pi_b}$ means exactly $s \sim \xi^{\pi_b}$ and then $a \sim \pi_b(\cdot \mid s)$.

Finite-space policy improvement. For finite state and action spaces, Delgrange et al. (2025, Theorem 1) establish the following guarantee. Fix $\epsilon \in (0, 1)$, let π_0 have full support and let μ_{π_n} be a sampling distribution with $\mathrm{supp}(\mu_{\pi_n}) = \mathcal{S}$ for each n. If

$$\pi_{n+1} \in \operatorname*{arg\,max}_{\pi' \in \mathcal{N}^\epsilon(\pi_n)} \mathbb{E}_{s \sim \mu_{\pi_n},\, a \sim \pi'(\cdot \mid s)} \left[A^{\pi_n}(s, a) \right],$$

then V^{π_n} improves monotonically and converges to the optimal value. Note that in their and our implementation, this neighborhood update is approximated with `PPO`-style clipping. The current policy is included in its own neighborhood: $\pi_n \in \mathcal{N}^\epsilon(\pi_n)$ because the support is unchanged and all self-ratios equal one. The neighborhood constrains each state's action distribution separately. Since the state-sampling distribution μ_{π_n} has full support, the maximizer has nonnegative expected advantage at each state (otherwise, keeping the current policy at that state would give zero advantage and increase the objective). Policy improvement then gives $V^{\pi_{n+1}} \geq V^{\pi_n}$ pointwise.

Practical clipped `DeepSPI` loss. Delgrange et al. (2025) approximate this policy update with the clipped loss used in the main text. Given a transition (s_t, a_t, r_t, s_{t+1}) collected by the reference policy π_{old}, the learned world model provides a reward prediction $\bar{R}(\phi(s_t), a_t)$ and a transition prediction $\bar{P}(\cdot \mid \phi(s_t), a_t)$. We measure prediction errors on this transition by

$$\ell_{R,t} = \left| \bar{R}(\phi(s_t), a_t) - r_t \right|, \qquad \ell_{P,t} = \|\hat{z}_{t+1} - \phi(s_{t+1})\|.$$

Here $\hat{z}_{t+1}$ is sampled from $\bar{P}(\cdot \mid \phi(s_t), a_t)$, as in Eq. (1). Averaging $\ell_{R,t}$ estimates $L_R^{\pi_b}$; averaging $\ell_{P,t}$ estimates an upper bound on $L_P^{\pi_b}$, as explained in Appendix C.

Let $\hat{A}_t$ be the critic advantage estimate, e.g. from generalized advantage estimation (Schulman et al., 2016). `DeepSPI` replaces the raw advantage by the model-error-aware utility

$$U_t = \hat{A}_t - \alpha_R \, \ell_{R,t} - \alpha_P \, \ell_{P,t},$$

where $\alpha_R, \alpha_P \geq 0$ tune the penalty paid for local reward and transition errors. For a candidate policy π_ψ and policy ratio

$$\rho_t(\psi) = \frac{\pi_\psi(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)},$$

the clipped DeepSPI actor loss is

$$\mathcal{L}_{\texttt{DeepSPI}}(\psi) = -\mathbb{E}_t \left[\min \left(\rho_t(\psi) U_t,\ \text{clip} \left(\rho_t(\psi), 1-\epsilon, 1+\epsilon \right) U_t \right) \right].$$

Here the clipping radius is the same ϵ as in Eq. (2), so the clipping interval matches the neighborhood ratio interval $[1-\epsilon, 1+\epsilon]$. Setting $\alpha_R = \alpha_P = 0$ recovers the usual PPO actor loss. The model-error penalties are nonnegative; subtracting them lowers U_t or leaves it unchanged. The sign of U_t determines whether the actor is encouraged to increase or decrease the sampled action's probability. In DreamSPI's imagination phase, the model is fixed and these penalties are disabled: the readout learns from imagined advantages with the clipped PPO loss. In practice, we also add a KL penalty to the loss, using a snapshot of the policy taken before imagination as the reference. Finally, note that clipping and the KL penalty encourage the policy to stay within the neighborhood, but do not enforce its ratio bounds at every state.

B Scope Notes

Here, *safe* follows the safe-policy-improvement literature: return is preserved relative to a behavior policy, subject to an explicit local model-error term. DreamSPI is neither a risk-sensitive nor a constrained-MDP method and provides no deployment-safety guarantee.

The theory and implementation use on-policy updates. Agents with replay may learn better world models on Atari, but replay changes the data used to train the model and actor, as well as training time. We ask whether a learned model can improve control when its errors are measured under the policy that collected the current batch. We therefore compare primarily with PPO, a PPO variant of DeepMDP, and DeepSPI.

The algorithm approximates the formal DeepSPI update, whose neighborhood constrains policy ratios pointwise in a discounted MDP. It uses PPO-style clipping as a proxy, categorical predictions to obtain a finite latent model, and token embeddings as actor and critic inputs. Appendix J states the regularity guarantees for the finite model and the additional projection residual needed for the token-embedding implementation.

ALE allows to evaluate the method under image observations, stochastic transitions, and various game mechanics and dynamics. These features make representation learning and local model errors important and reveal where fixed-horizon imagination helps or fails. Appendix O examines these differences across games.

C Deep Safe Policy Improvement: Environment-to-Latent Transfer Bound

For completeness, in the following, we state the DeepSPI bounds (Delgrange et al., 2025) that relate model errors to policy improvement. Let π_b be the behavior policy and ν^{π_b} its stationary state-action distribution. Let $\phi : \mathcal{S} \to \mathcal{Z}$ be a fixed representation during the policy update, let $\bar{\pi}$ be a latent policy, and let $\bar{\pi} \circ \phi$ be the composed policy deployed in the original environment. Write

$$\Delta(\pi_b, \bar{\pi}) = \sup_{s,\ a \in \text{supp}\, \pi_b(\cdot|s)} \frac{(\bar{\pi} \circ \phi)(a \mid s)}{\pi_b(a \mid s)}$$

for the supremum importance ratio from the behavior policy to the composed candidate policy. Assume the two policies have the same action support at every state and rewards are bounded. The latent model starts at $z_0 = \phi(s_0)$, with $s_0 \sim \rho_0$. If $\bar{\pi} \circ \phi \in \mathcal{N}^\epsilon(\pi_b)$, then $\Delta(\pi_b, \bar{\pi}) \leq 1 + \epsilon$. Hence the maximum admissible deep SPI radius is $\epsilon_{\max} = 1/\gamma - 1$, with strict inequality required in the finite bounds below whenever the denominator $1/\Delta(\pi_b, \bar{\pi}) - \gamma$ appears. The local model losses under behavior data are

$$L_R^{\pi_b} = \mathbb{E}_{(s,a)\sim\nu^{\pi_b}} \left| R(s,a) - \bar{R}(\phi(s), a) \right|, \quad L_P^{\pi_b} = \mathbb{E}_{(s,a)\sim\nu^{\pi_b}} \mathcal{W}\big(\phi_\# P(\cdot \mid s,a), \bar{P}(\cdot \mid \phi(s), a)\big). \tag{7}$$

Here we use the more general Wasserstein transition loss, whereas the main text uses Euclidean prediction errors. All bounds below also hold for the Euclidean loss under the same assumptions, since its expectation upper-bounds the Wasserstein loss. With Euclidean distance as the cost, Wasserstein distance is the smallest expected distance over all ways of pairing these distributions (Appendix I). Independent real and model samples give one such pairing, hence

$$\mathbb{E}_{(s,a)\sim\nu^{\pi_b}} \mathcal{W}\big(\phi_{\#}P(\cdot \mid s,a), \bar{P}(\cdot \mid \phi(s),a)\big) \le \mathbb{E}_{\substack{(s,a)\sim\nu^{\pi_b},\ s'\sim P(\cdot|s,a)\\ \hat{z}'\sim\bar{P}(\cdot|\phi(s),a)}} \|\phi(s') - \hat{z}'\|\,.$$

The expected ℓ_2 prediction error therefore upper-bounds the Wasserstein loss used in the guarantees (Delgrange et al., 2025, Appendix C). We use Euclidean prediction errors in the main text for clarity and because they can be estimated directly from sampled transitions. We also use Euclidean distance in our implementation. This avoids the extra optimization needed to compute or approximate Wasserstein distance during training. Assume $\gamma > 1/2$, $\Delta(\pi_b, \bar{\pi}) < 1/\gamma$, and that the policy-induced latent reward and transition kernels are regular, with transition Lipschitz constant $K_{\bar{P}}^{\bar{\pi}} < 1/\gamma$. Let

$$K_V = \frac{K_{\bar{R}}^{\bar{\pi}}}{1 - \gamma K_{\bar{P}}^{\bar{\pi}}},$$

where $K_{\bar{R}}^{\bar{\pi}}$ is the corresponding latent reward Lipschitz constant.

The deep SPI world-model quality theorem first bounds the mismatch between environment return and latent-model return for any candidate policy that remains in the local neighborhood:

$$|J_{\mathcal{M}}(\bar{\pi}\circ\phi) - J_{\bar{\mathcal{M}}}(\bar{\pi})| \le \mathrm{AEL}(\pi_b)\,\frac{L_R^{\pi_b}/\gamma + K_V L_P^{\pi_b}}{1/\Delta(\pi_b,\bar{\pi}) - \gamma}. \tag{8}$$

where $\mathrm{AEL}(\pi_b)$ is the average expected length of a episodes under the behavior policy. Thus, local reward and transition errors under the behavior distribution control the environment-to-latent return mismatch for nearby policies, while the denominator worsens as the candidate approaches the ratio boundary $1/\gamma$.

The safe policy-improvement theorem then adds the assumption that the behavior policy itself is represented in the latent policy class, $\pi_b = \bar{\pi}_b \circ \phi$. The baseline latent value function must also be Lipschitz. For the improvement bound, let K_V bound the Lipschitz constants of both latent value functions. When both transition constants satisfy $\gamma K_{\bar{P}}^{\bar{\pi}} < 1$, one can take $K_V = \max_{\bar{\pi}\in\{\bar{\pi},\bar{\pi}_b\}} K_{\bar{R}}^{\bar{\pi}}/(1-\gamma K_{\bar{P}}^{\bar{\pi}})$. Under these assumptions,

$$J_{\mathcal{M}}(\bar{\pi}\circ\phi) - J_{\mathcal{M}}(\pi_b) \ge J_{\bar{\mathcal{M}}}(\bar{\pi}) - J_{\bar{\mathcal{M}}}(\bar{\pi}_b) - \zeta,$$

where a valid modeling-error penalty is

$$\zeta = \mathrm{AEL}(\pi_b)\left(\frac{L_R^{\pi_b}}{\gamma} + K_V L_P^{\pi_b}\right)\left(\frac{1}{1/\Delta(\pi_b,\bar{\pi}) - \gamma} + \frac{1}{1-\gamma}\right). \tag{9}$$

The first denominator is the model-error cost for the candidate policy; the second is the same value-gap bound evaluated at the behavior policy, where $\Delta(\pi_b, \bar{\pi}_b) = 1$.

Finally, the representation-quality theorem uses the same quantities to state the high-probability almost-Lipschitz property of the next-policy value function. For any $\eta > 0$, define

$$\delta(\eta) = 4\,\frac{L_R^{\pi_b} + \gamma K_V L_P^{\pi_b}}{\eta\,(1/\Delta(\pi_b,\bar{\pi}) - \gamma)}.$$

Let λ be any coupling of ξ^{π_b} with itself. For a pair $(s_1, s_2) \sim \lambda$, with probability at least $1 - \delta(\eta)$,

$$\left|V^{\bar{\pi}\circ\phi}(s_1) - V^{\bar{\pi}\circ\phi}(s_2)\right| \le K_V\,\|\phi(s_1) - \phi(s_2)\| + \eta.$$

Appendix J applies these bounds to policies evaluated on the finite codebook.

D Projection-Only Ratio Control

Theorem 3 (Projection-only ratio control)**.** *Fix encoder/world-model parameters after the real-data phase and write $z = \phi(s;\vartheta)$. For logits $g(z;\psi) = W_\pi \mathrm{vec}(z) + b_\pi$, define $\bar{\pi}(\cdot \mid z;\psi) = \mathrm{softmax}(g(z;\psi))$ on the finite action set. Let $\psi' = \psi + \Delta\psi$ change only (W_π, b_π) and define $\|A\|_{\mathrm{op}} = \sup_{\|x\|_2\le 1}\|Ax\|_\infty$. If $\chi(z) = \|\Delta W_\pi\|_{\mathrm{op}}\,\|\mathrm{vec}(z)\|_2 + \|\Delta b_\pi\|_\infty$,*

$$\exp(-2\chi(z)) \le \frac{\bar{\pi}(a \mid z;\psi')}{\bar{\pi}(a \mid z;\psi)} \le \exp(2\chi(z)).$$

Proof. Recall that $g(z;\psi) = W_\pi \mathrm{vec}(z) + b_\pi$ denotes the actor logits and $\chi(z) = \|\Delta W_\pi\|_{\mathrm{op}} \|\mathrm{vec}(z)\|_2 + \|\Delta b_\pi\|_\infty$. The projection-only head gives

$$g(z;\psi') - g(z;\psi) = \Delta W_\pi \,\mathrm{vec}(z) + \Delta b_\pi,$$

Taking sup norms and using the triangle inequality,

$$\begin{aligned}\|g(z;\psi') - g(z;\psi)\|_\infty &\le \|\Delta W_\pi \,\mathrm{vec}(z)\|_\infty + \|\Delta b_\pi\|_\infty \\ &\le \|\Delta W_\pi\|_{\mathrm{op}} \|\mathrm{vec}(z)\|_2 + \|\Delta b_\pi\|_\infty = \chi(z),\end{aligned}$$

To see the second inequality explicitly, write $\Delta w_{\pi,a}$ for the ath row of ΔW_π. Holder's inequality with $p = q = 2$ gives

$$|(\Delta W_\pi \,\mathrm{vec}(z))_a| = |\langle \Delta w_{\pi,a}, \mathrm{vec}(z)\rangle| \le \|\Delta w_{\pi,a}\|_2 \|\mathrm{vec}(z)\|_2 .$$

Taking the maximum over a gives $\|\Delta W_\pi \,\mathrm{vec}(z)\|_\infty \le \|\Delta W_\pi\|_{\mathrm{op}} \|\mathrm{vec}(z)\|_2$, which is the $\ell_2 \to \ell_\infty$ operator-norm bound. For a logit vector $\mathbf{u}$, $\log \mathrm{softmax}(\mathbf{u})_a = u_a - \log \sum_b \exp(u_b)$. Moreover, for $h(\mathbf{u}) = \log \sum_b \exp(u_b)$, $\nabla h(\mathbf{u}) = \mathrm{softmax}(\mathbf{u})$ and $\|\nabla h(\mathbf{u})\|_1 = 1$. Hence, by the mean-value theorem and Holder's inequality, $|h(\mathbf{u}') - h(\mathbf{u})| \le \|\mathbf{u}' - \mathbf{u}\|_\infty$. Applying this with $\mathbf{u}' = g(z;\psi')$ and $\mathbf{u} = g(z;\psi)$ bounds both the selected-logit change and the log-normalizer change by $\chi(z)$. Therefore

$$|\log \bar\pi(a \mid z;\psi') - \log \bar\pi(a \mid z;\psi)| \le 2\chi(z),$$

and exponentiating gives the ratio bound. □

With the encoder fixed, Theorem 3 bounds policy ratios through the *readout update*, i.e., changes to the weights and biases of the actor's final linear layer. For the composed policies $\pi_\psi(a \mid s) = \bar\pi(a \mid \phi(s);\psi)$ and $\pi_{\psi'}(a \mid s) = \bar\pi(a \mid \phi(s);\psi')$, the ϵ-SPI neighborhood requires $1-\epsilon \le {}^{\pi_{\psi'}(a|s)}/_{\pi_\psi(a|s)} \le 1+\epsilon$ on the relevant state-action support. The bound depends on $\chi(\phi(s))$, hence only on the readout displacement $(\Delta W_\pi, \Delta b_\pi)$ and fixed latent features. The condition $2\chi(\phi(s)) \le \log(1+\epsilon)$ suffices, since $\epsilon \in (0,1)$ implies $\log(1+\epsilon) \le -\log(1-\epsilon)$.

During imagination, `DreamSPI` freezes the encoder and world model and updates only the final actor–critic linear layer. Updating the encoder would add $W_\pi \mathrm{vec}\big(\frac{d}{d\alpha}\phi(s;\vartheta + \alpha\Delta\vartheta)\big|_{\alpha=0}\big)$ to the first-order change in the actor's action scores. Updating hidden layers in the actor would introduce products of their Jacobians through the chain rule. With fixed latent states, the ratio bound depends only on changes to the readout weights and biases.

The softmax actor assigns positive probability to every action: $\bar\pi(a \mid z;\psi) > 0$ and $\bar\pi(a \mid z;\psi') > 0$. Both policies therefore have the same action support, and the ratio in Theorem 3 is well defined. The converse need not hold: bounded ratios need not bound the parameter displacement $(\Delta W_\pi, \Delta b_\pi)$. A common shift of all logits leaves the softmax unchanged. Note that on a finite set of latent states, some readout directions may also leave all evaluated logits unchanged. The theorem thus gives a sufficient condition for ratio control, not a necessary one.

E Algorithmic Details

Algorithm 1 gives the two-phase loop. Each *outer iteration* collects a real rollout, updates the networks on that data, and runs imagination from the encoded states. Both buffers are temporary: we empty the real buffer before collecting the next T steps from B environments and create an imagined buffer for each model rollout of H steps. No transition from an earlier outer iteration is reused. When `dyna` is true, the actor–critic learns from both buffers; otherwise, it learns only from imagination.

In Algorithm 1, bold symbols denote batches of states $\mathbf{s}$, actions $\mathbf{a}$, rewards $\mathbf{r}$, done flags $\mathbf{d}$, and latent states $\mathbf{z}$. We collect B entries per step from the vectorized environments. During imagination, $\mathbf{z}_h$, $\bar{\mathbf{a}}_h$, $\bar{\mathbf{r}}_h$, and $\bar{\mathbf{d}}_{h+1}$ each contain TB entries, one per parallel imagined trajectory.

Imagined advantages $\hat A_h^{\mathrm{imag}}$ and returns $\hat G_h^{\mathrm{imag}}$ use generalized advantage estimation (Schulman et al., 2016), bootstrapping from $V_\psi(z_{H+1})$ unless the rollout has terminated. At the start of imagination cycle m, we save a copy of the actor, $\bar\pi_{\mathrm{roll},m}$. The buffer stores $\log \bar\pi_{\mathrm{roll},m}(a_h \mid z_h)$ for the PPO ratio and $\log \bar\pi_{\mathrm{ref}}(a_h \mid z_h)$ for the reference-divergence penalty. These stored log-probabilities remain fixed as the actor changes.

Algorithm 1: `DreamSPI`.

Input: Encoder/model parameters ϑ, actor-critic readout parameters ψ, real rollout length T, imagination horizon H, number of environments B, imagination rollouts R_{roll}, Boolean `dyna` (actor–critic also learns from real data when true), and reference-penalty coefficient β_{KL}.

Output: Composed policy $\pi(a \mid s) = \bar{\pi}_\psi(a \mid \phi_\vartheta(s))$.

Initialize vectorized environment states $\mathbf{s}_1 \in \mathcal{S}^B$.

repeat

- Set the real-batch reference policy $\pi_{\mathrm{old}} \leftarrow \bar{\pi}_\psi \circ \phi_\vartheta$.
- Initialize an empty real rollout buffer $\mathcal{B}^{\mathrm{real}}$ with leading shape (T, B) for this outer iteration.
- **for** $t \leftarrow 1$ **to** T **do**
  - Sample $\mathbf{a}_t = (a_t^{(1)}, \ldots, a_t^{(B)})$, with $a_t^{(b)} \sim \pi_{\mathrm{old}}(\cdot \mid s_t^{(b)})$ independently for each environment, and store the B log-probabilities.
  - Step all environments in parallel: $(\mathbf{r}_t, \mathbf{s}_{t+1}, \mathbf{d}_{t+1}) \leftarrow \texttt{env.step}(\mathbf{s}_t, \mathbf{a}_t)$, where $\mathbf{d}_{t+1}$ is the done mask (termination or time-limit truncation).
  - Store $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1}, \mathbf{d}_{t+1})$ at time index t in $\mathcal{B}^{\mathrm{real}}$.
- Compute real-batch returns and advantages on $\mathcal{B}^{\mathrm{real}}$; recompute model errors and `DeepSPI` utilities during each minibatch update.
- Shuffle the (T, B) entries of $\mathcal{B}^{\mathrm{real}}$ into minibatches for the real update
- **if** *dyna* **then**
  - Update both (ϑ, ψ) from the real buffer with Eq. (3) plus $\beta_{\mathrm{KL}} \widehat{\mathrm{KL}}(\pi_{\mathrm{old}} \| \bar{\pi}_\psi \circ \phi_\vartheta)$.
- **else**
  - Update only ϑ with the real-batch model/composed-policy terms in Eq. (3), keeping ψ frozen until imagination.
- Encode the states just encountered in this outer iteration and flatten time/environment axes: $\mathbf{z}^{\mathrm{enc}} \leftarrow \texttt{reshape}(\phi_\vartheta(\mathbf{s}_{1:T}), (TB, \cdot))$; these are the TB imagination starts.
- Freeze ϑ during imagination updates and set the pre-imagination reference policy $\bar{\pi}_{\mathrm{ref}} \leftarrow \bar{\pi}_\psi$.
- **for** $m \leftarrow 1$ **to** R_{roll} **do**
  - Set the cycle's rollout-policy snapshot $\bar{\pi}_{\mathrm{roll},m} \leftarrow \bar{\pi}_\psi$.
  - Reset the model-rollout start batch to the encoded real-batch latents: $\mathbf{z}_1 \leftarrow \mathbf{z}^{\mathrm{enc}}$.
  - Initialize an empty latent rollout buffer $\bar{\mathcal{B}}_m$ with leading shape (H, TB) for this imagination cycle.
  - **for** $h \leftarrow 1$ **to** H **do**
    - Sample $\bar{\mathbf{a}}_h \sim \bar{\pi}_{\mathrm{roll},m}(\cdot \mid \mathbf{z}_h)$ and evaluate $V_\psi(\mathbf{z}_h)$.
    - Step the learned model in parallel: $(\bar{\mathbf{r}}_h, \mathbf{z}_{h+1}, \bar{\mathbf{d}}_{h+1}) \leftarrow \texttt{world_model.step}_\vartheta(\mathbf{z}_h, \bar{\mathbf{a}}_h)$.
    - Store the transition, $V_\psi(\mathbf{z}_h)$, $\log \bar{\pi}_{\mathrm{roll},m}(\bar{\mathbf{a}}_h \mid \mathbf{z}_h)$, and $\log \bar{\pi}_{\mathrm{ref}}(\bar{\mathbf{a}}_h \mid \mathbf{z}_h)$ at index h in $\bar{\mathcal{B}}_m$.
  - Compute $\hat{A}_h^{\mathrm{imag}}$ and $\hat{G}_h^{\mathrm{imag}}$ with GAE, including the bootstrap $V_\psi(\mathbf{z}_{H+1})$.
  - Shuffle the (H, TB) entries into minibatches and optimize the imagined PPO objective for the configured number of imagination epochs.
  - Update only ψ; estimate the PPO ratio and configured reference divergence using the two stored log-probabilities defined above.
- Continue real collection from the final vectorized states $\mathbf{s}_{T+1}$.

until *the training budget is exhausted*

F HYPERPARAMETERS

F.1 ATARI

Table 1: `DreamSPI` configuration used for the full ALE evaluation.

Component	Value
Data collection	64 parallel environments, 8 real steps per rollout, batch size 512, effective minibatch size 128
Encoder/interface	tokenized CNN output, post-CNN layer normalization, `SoftMoE` enabled, token embeddings exposed to the actor-critic
Latent transition	categorical next-token model with 32 categories and 32 classes, `Gumbel-softmax` (Jang et al., 2017; Maddison et al., 2017), L2 categorical cost
World-model head	fully connected auxiliary model, $\texttt{SoftMoE}_{\text{hidden width}} = 768$, hidden width 512, zero hidden MLP layers after the expert mixer, eight experts
Activation and normalization	Mish activations with layer normalization in the encoder and reward/transition networks
Discount and advantage	$\gamma = 0.995$, GAE $\lambda = 0.95$, normalized advantages
Imagination	`dyna` enabled, horizon 16, one latent rollout/`PPO` cycle from encoded starts, 4 `PPO` epochs over each latent rollout buffer, encoder/model updated once per 64×8 real rollout
Policy losses	actor–critic `PPO` ratio clip 0.01, reference-penalty coefficient $\beta_{\text{KL}} = 1.02 \cdot 10^{-4}$, entropy coefficient 0.01, value coefficient 0.5
Model losses	reward scale$= 0.01$, transition scale$= (100 \cdot \texttt{num_tokens})^{-1}$, `done` scale 0.5
Learning rates	encoder/world model $2.5 \cdot 10^{-4}$, actor $1.2 \cdot 10^{-5}$, critic $1.1 \cdot 10^{-5}$, all linearly annealed to zero
Optimization	`Kron` for world model and actor-critic, max gradient norm 5, real-data `DeepSPI` ratio clip $\epsilon = 0.1$, no weight decay[2]

Our model also predicts whether an episode ends. We train this predictor with binary cross-entropy, weighted by the `done` scale (0.5), and include it in the reward loss before applying the reward scale.

Note that real-data `DeepSPI` updates use $\epsilon = 0.1$ in Eq. (3), giving a ratio-clipping interval of $[0.9, 1.1]$. Actor–critic `PPO` updates use $\epsilon = 0.01$, giving $[0.99, 1.01]$.

F.2 OCTAX

The Octax-15 evaluation uses two configurations with shared settings (Table 2) and two differences: the transition-loss coefficient and Hadamard embeddings (Table 3). Each configuration uses five seeds and five million environment steps per game. Observations contain four frames, sticky-action probability is 0.3, and training uses raw rewards without random no-op starts.

[2]The effect of weight decay on TD stability depends on which layers are regularized (Gallici et al., 2025). We leave it out to avoid confounding the effects of the representation and policy updates.

Table 2: Parameters shared by the selected Octax-15 `DreamSPI` runs. Table 3 gives the two settings that differ between configurations.

Component	Value
Data collection	64 parallel environments, 8 real steps per rollout, batch size 512, effective minibatch size 128
Encoder/interface	tokenized CNN output, post-CNN layer normalization, `SoftMoE` enabled, token embeddings exposed to the actor-critic
Latent transition	categorical next-token model with 32 categories and 32 classes, `Gumbel-softmax` (Jang et al., 2017; Maddison et al., 2017), L2 categorical cost
World-model head	fully connected auxiliary model, $\texttt{SoftMoE}_{\text{hidden width}} = 768$, hidden width 512, zero hidden MLP layers after the expert mixer, eight experts
Activation and normalization	Mish activations with layer normalization in the encoder and reward/transition networks
Discount and advantage	$\gamma = 0.995$, GAE $\lambda = 0.95$, normalized advantages
Imagination	`dyna` enabled, 16 steps, one latent rollout/`PPO` cycle from encoded starts, 4 `PPO` epochs over each latent rollout buffer, encoder/model updated once per 64×8 real rollout
Policy losses	actor–critic `PPO` ratio clip 0.01, reference-penalty coefficient $\beta_{\mathrm{KL}} = 1.02 \cdot 10^{-4}$, entropy coefficient 0.01, value coefficient 0.5
Model losses	reward scale$= 0.01$, transition scale in Table 3, `done` scale 0.5
Learning rates	encoder/world model $2.5 \cdot 10^{-4}$, actor $1.2 \cdot 10^{-5}$, critic $1.1 \cdot 10^{-5}$, all linearly annealed to zero
Optimization	`Kron` for world model and actor-critic, max gradient norm 5, real-data `DeepSPI` ratio clip $\epsilon = 0.1$, 4 real-data update epochs, no weight decay

Table 3: The two Octax profiles and their retained environments. Apart from the transition-loss coefficient and Hadamard representation, both profiles use Table 2 exactly.

Profile	Transition loss	Hadamard	Environments
Original	0.01	off	Blinky, Brix, Missile, Spacejam, Tank, Tetris, UFO, Worm
QZD	1	on	Airplane, Deep, Filter, Pong, Squash, Vertical Brix, Wipe Off

During our experiments, transition losses grew sharply on a subset of games. We found it useful to bound embedding magnitudes with the Hadamard/hyper-tanh transformation (Appendix L.6.1). This limits codebook diameter, the numerator of the bound on $K^{\pi}_{\widehat{P},\mathcal{C}}$ in Theorem 1; the bound also depends on the minimum separation. The alternative also increases the transition-loss coefficient, so this comparison tests both changes together.

G WALL-CLOCK DIAGNOSTICS

Table 4 reports runtimes on Nvidia H200 GPUs. The selected ALE runs take a median 2.43h (2.44h for the common subset in the table); Octax takes 1.27h, but uses $20M$ frames rather than $40M$. For context, Delgrange et al. (2025) report A40 runtimes of 0.28h for `PPO`, 1.00h for `DeepMDP`, 1.04h for `DeepSPI`, and 1.35h for `DreamSPI-v0`. Published replay-based Atari agents take longer: roughly 40h for `SAC` on an A100 at the same frame budget (Huang et al., 2022), and two days for

DreamerV2 on a V100 (Hafner et al., 2021). Hardware and implementation differences prevent a direct speed comparison.

Table 4: Wall-clock diagnostics from Nvidia H200 runs. Times are hours per finished run; mean reports empirical standard deviation, median reports the interquartile range (Q1–Q3), and IQM reports 95% bootstrap percentile confidence intervals.

Run set	Mean ± std	Median [IQR]	IQM [95% CI]
ALE-57	3.034 ± 1.417	$2.438\,[2.344, 2.492]$	$2.434\,[2.419, 2.447]$
Octax	1.333 ± 0.239	$1.273\,[1.264, 1.281]$	$1.273\,[1.271, 1.275]$
Atari-5 Kron	2.362 ± 0.212	$2.373\,[2.140, 2.542]$	$2.368\,[2.235, 2.491]$
Atari-5 Muon	1.795 ± 0.064	$1.774\,[1.746, 1.855]$	$1.789\,[1.750, 1.842]$
Atari-5 Adam	0.836 ± 0.052	$0.810\,[0.791, 0.894]$	$0.831\,[0.795, 0.882]$

The optimizer rows quantify the tradeoff in Fig. 7c. On Atari-5, Adam and Muon are about $2.8\times$ and $1.3\times$ faster than Kron, but their IQM-HNS scores fall to 0.101 and 0.643, compared with 2.068 for Kron. These faster updates save training time at a substantial cost in return.

H Stabilizer Details

Moving distributions. Even in a stationary environment, the policy, encoder, and model change during training. At iteration k, write ϕ_k, $\bar{\pi}_k$, and $\bar{P}_k$ for the encoder, latent actor, and latent transition model. The real batch is sampled from the current composed policy,

$$\mathcal{D}_k \sim \nu_{\mathcal{M}}^{\pi_k}, \qquad \pi_k(a \mid s) = \bar{\pi}_k(a \mid \phi_k(s)).$$

Changing either $\bar{\pi}_k$ or ϕ_k changes the distribution of the next batch. Learning the representation can therefore change the deployed policy even when the actor parameters are fixed.

Imagined data also depend on the learned transition model. Starting from a real state s, a rollout at iteration k follows

$$\bar{z}_0 = \phi_k(s), \qquad \bar{a}_h \sim \bar{\pi}_k(\cdot \mid \bar{z}_h), \qquad \bar{z}_{h+1} \sim \bar{P}_k(\cdot \mid \bar{z}_h, \bar{a}_h).$$

Updating the encoder changes the model's inputs and next-state targets. The critic's targets also change with its value estimates and, in imagination, with the predicted rewards and next states. When the model only trains the encoder, its errors affect the representation. With imagination, a wrong prediction can also make an action look better than it is in the environment. We fit the model to the current policy's data and keep imagined rollouts short to limit accumulated errors. During imagination, we update only the final actor–critic linear layer; the encoder and model stay fixed.

Preconditioning and smooth activations. Kron uses curvature estimates to rescale gradients jointly across parameters, rather than coordinate by coordinate as in Adam. It stores this preconditioner as smaller matrices along each tensor axis. For a matrix of weights, the update has the form

$$\Delta W \propto -Q_L \, \nabla_W \mathcal{L} \, Q_R,$$

where the learned factors Q_L and Q_R rescale gradient directions with different curvature or magnitude. Muon also operates on parameter matrices: it orthogonalizes updates through a Newton–Schulz iteration and scales to large neural networks (Liu et al., 2025b). Kron instead uses approximate curvature factors to precondition updates. In DreamSPI, the encoder, model, value, and policy losses share a latent representation; their distributions and targets change after each on-policy update.

The optimizer ablations use a shared base learning-rate grid for Adam, Muon, and Kron. For Kron, the value reported by the hyperparameter search is the pre-scaling value: before constructing the optimizer, our code passes one third of this base rate to the Kron transform. This matches PPO/PQN/DQN implementations in Creus Castanyer et al. (2025), which apply `learning_rate /= 3.0` when using Kron, following the upstream `kron_torch` recommendation to start around a $3\times$ smaller rate than Adam.[3] In PSGD-Kron (Pooladzandi & Li, 2024), the learning rate multiplies

[3] `stable-deep-rl-at-scale`; `kron_torch`.

a gradient transformed by the learned preconditioner $P = Q^\top Q$. The same scalar rate therefore need not produce updates of the same size as in `Adam` or `Muon`.[4] The sweep and hyperparameter table use base rates before Kron's $1/3$ scaling. The selected base rates were useful across the three optimizer families after applying this scaling to Kron.

Since ReLU is already Lipschitz, pointwise activation Lipschitzness does not distinguish the variants; the relevant quantity is the regularity of the activation Jacobian. With a piecewise-linear hard gate, small changes in the encoder or in the on-policy data distribution can flip activation patterns and produce discontinuous changes in the layer Jacobian. Mish replaces the hard gate $x\mathbf{1}\{x > 0\}$ by the smooth gate $x\tanh(\mathrm{softplus}(x))$. On bounded preactivation ranges, both the activation and its derivative have finite local Lipschitz constants. Thus, when the representation drifts from ϕ_{ϑ_k} to $\phi_{\vartheta_{k+1}}$, the features and their derivatives change continuously rather than through gate-pattern switches.

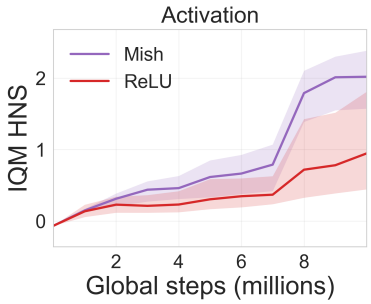


Figure 9: Atari-5 IQM-HNS sample efficiency for Mish and ReLU.

The Atari-5 comparison in Fig. 9 shows sensitivity to this choice: replacing Mish with ReLU, while keeping LayerNorm and the other selected settings fixed, reduces final IQM HNS from 2.068 to 0.958. A perturbation of the encoder changes the actor logits, reward prediction, transition prediction, and critic value. For the composed policy, a first-order logit perturbation has the form

$$\Delta g(s) \approx J_{\mathrm{head}}(h(s;\vartheta))\, D_\vartheta h(s;\vartheta)\, \Delta\vartheta,$$

where $D_\vartheta h$ is the derivative of the features with respect to the encoder parameters. The proof of Theorem 3 gives $|\Delta \log \pi(a \mid s)| \le 2\,\|\Delta g(s)\|_\infty$. The inequality bounds how much action probabilities can change; some changes to the logits leave the policy unchanged. Mish keeps the activation derivative continuous as these quantities change. Negative inputs can also receive gradients, allowing features to keep adapting when the policy and model change the training distribution.

Layer normalization is used mainly in the Atari encoder's first two convolutional blocks and the one-hidden-layer transition and reward networks. The actor–critic readout has no hidden dense layer after the token embeddings. LayerNorm conditions encoder and model updates as the data distribution and bootstrapped targets change.

I Probability Metrics Used in the Bounds

For a metric space (X, d) and Borel probability measures μ, ν with finite first moments, the 1-Wasserstein distance is

$$\mathcal{W}_d(\mu,\nu) = \inf_{\Gamma\in\Pi(\mu,\nu)} \mathbb{E}_{(x,y)\sim\Gamma}\left[d(x,y)\right],$$

where $\Pi(\mu,\nu)$ is the set of couplings with marginals μ and ν. In the main text, $\mathcal{W}_{\|\cdot\|}$ means this distance with ground cost $d(z,z') = \|z - z'\|$ on latent embeddings. Moving probability mass between nearby embeddings costs less than moving it between distant embeddings.

Total variation is the corresponding setwise discrepancy

$$\mathrm{TV}(\mu,\nu) = \sup_{A\in\mathcal{B}(X)} |\mu(A) - \nu(A)|.$$

Here $\mathcal{B}(X)$ is the Borel sigma-algebra. On a finite or countable space, $\mathrm{TV}(\mu,\nu) = \frac{1}{2}\,\|\mu - \nu\|_1$. If the ground cost is the unit mismatch metric $d(x,y) = \mathbf{1}\{x \ne y\}$, then

$$\mathcal{W}_d(\mu,\nu) = \mathrm{TV}(\mu,\nu).$$

Thus the finite-code statements in the paper can be read either as Wasserstein statements under the code metric or as total-variation statements over categorical codes. The continuous token-embedding statements use Wasserstein distance because they depend on the learned latent geometry.

[4] `psgd_torch`.

J Token Embeddings and Codebook Regularity

This appendix expands the construction in Sect. 3.2 and proves Claim 1. We first show how the codebook's diameter, minimum separation, and reward scale bound the world model's Lipschitz constants (Theorem 1). We then apply SPI to policies evaluated on codebook embeddings and state the extra conditions needed when the actor uses encoder outputs directly.

Token embeddings and the finite codebook. We keep the spatial layout of the encoder output in the actor's input (Sokar et al., 2025). For a CNN output $x \in \mathbb{R}^{H\times W\times C}$, each of the $M = HW$ spatial locations gives a token embedding $e_m \in \mathbb{R}^C$. These form the latent state $z = [e_1; \dots; e_M] \in \mathbb{R}^{M\times C}$, with $e_m = z_m$. Optional soft mixture-of-experts layers process the tokens while retaining their spatial positions.

To restrict predictions to a finite set, the transition model outputs K categorical distributions with L classes each, given the current tokens z and action a:

$$q(u \mid z, a) = \prod_{i=1}^{K} \mathrm{Cat}(u_i;\, p_i(\cdot \mid z, a)),$$
$$p_i(\ell \mid z, a) = \mathrm{softmax}_\ell(o_i(z, a)), \quad \ell \in \{1, \dots, L\}.$$

Here, $o_i(z, a) \in \mathbb{R}^L$ contains the logits for variable u_i. We sample one class per variable using straight-through Gumbel-softmax (Bengio et al., 2013; Jang et al., 2017; Maddison et al., 2017). The class indices form a code $\omega \in \Omega = \{1, \dots, L\}^K$, represented by a matrix $u^\omega \in \mathbb{R}^{K\times L}$ with one one-hot row per variable.

We decode the $K \times L$ one-hot code into $M \times C$ token embeddings. Each output position has its own class embeddings, so the same sampled class can represent different features at different positions. We first combine the sampled choices across variables using a learned matrix $F \in \mathbb{R}^{M\times K}$, producing $M \times L$ class weights. At each position m, a learned table $G_m \in \mathbb{R}^{L\times C}$ then maps these weights to C features. Writing $g_{m,\ell} \in \mathbb{R}^C$ for row ℓ of G_m, we obtain

$$\hat{z}_m = \sum_{k=1}^{K} F_{mk}\, g_{m,\omega_k}, \qquad \hat{z} = \Psi_{F,G}(\omega) = [\hat{z}_1; \dots; \hat{z}_M],$$

where ω_k is the class sampled for variable k and $G = (G_1, \dots, G_M)$. Thus, F_{mk} controls how strongly variable k contributes to position m, while g_{m,ω_k} specifies its features there. With one embedding table per variable instead, changing a sampled class would produce the same feature change at every position, scaled by F_{mk}. The set of all possible decoded states forms the finite codebook $\mathcal{C}_{F,G} = \{\Psi_{F,G}(\omega) : \omega \in \Omega\}$. Learning F or G changes the codebook embeddings and their distances.

The decoded prediction $\hat{z} \in \mathbb{R}^{M\times C}$ has the same shape as the encoder output, so it can be fed directly to the actor and compared with $\phi(s')$ in the transition loss. Distances between learned embeddings let the transition loss distinguish near misses from distant predictions.

Lipschitz bounds on codebook embeddings. Hold the model and codebook fixed. The diameter is the largest distance between codebook embeddings; the separation is the smallest distance between distinct embeddings:

$$\mathrm{diam}(\mathcal{C}_{F,G}) = \max_{\omega_1,\omega_2\in\Omega} \|\Psi_{F,G}(\omega_1) - \Psi_{F,G}(\omega_2)\|,$$
$$\mathrm{sep}(\mathcal{C}_{F,G}) = \min_{\substack{z_1,z_2\in\mathcal{C}_{F,G}\\ z_1\neq z_2}} \|z_1 - z_2\|.$$

For a finite codebook with at least two distinct embeddings, this separation is positive. Different codes may still decode to the same embedding.

Let $\bar{p}(\omega' \mid z, a)$ be the learned next-code distribution. For a latent policy $\bar{\pi}$ and $z \in \mathcal{C}_{F,G}$, the model's expected reward and next-embedding distribution are

$$\bar{R}^{\bar{\pi}}_{\mathcal{C}}(z) = \sum_{a\in\mathcal{A}} \bar{\pi}(a \mid z)\bar{R}(z, a),$$
$$\bar{P}^{\bar{\pi}}_{\mathcal{C}}(B \mid z) = \sum_{a\in\mathcal{A}} \bar{\pi}(a \mid z) \sum_{\omega' :\, \Psi_{F,G}(\omega')\in B} \bar{p}(\omega' \mid z, a), \qquad B \subseteq \mathcal{C}_{F,G}.$$

The transition equation averages over actions, samples the next code, and decodes it. Since $\mathcal{C}_{F,G}$ and $\mathcal{A}$ are finite, the maximum absolute reward prediction $\bar{R}_{\max} = \max_{z \in \mathcal{C}_{F,G},\, a \in \mathcal{A}} |\bar{R}(z,a)|$ is finite.

Theorem 1 (Decoded codebooks preserve Lipschitz regularity)**.** *Assume $\mathcal{C}_{F,G}$ has at least two distinct embeddings,* $\mathrm{sep}(\mathcal{C}_{F,G}) > 0$*, and $|\bar{R}(z,a)| \le \bar{R}_{\max}$ for every $z \in \mathcal{C}_{F,G}$ and $a \in \mathcal{A}$. Then the policy-induced decoded reward and transition are Lipschitz under the Euclidean norm, with*

$$K^{\bar{\pi}}_{\bar{R},\mathcal{C}} = \max_{\substack{z_1,z_2 \in \mathcal{C}_{F,G} \\ z_1 \neq z_2}} \frac{\left|\bar{R}^{\bar{\pi}}_{\mathcal{C}}(z_1) - \bar{R}^{\bar{\pi}}_{\mathcal{C}}(z_2)\right|}{\|z_1 - z_2\|} \le \frac{2\bar{R}_{\max}}{\mathrm{sep}(\mathcal{C}_{F,G})},$$

$$K^{\bar{\pi}}_{\bar{P},\mathcal{C}} = \max_{\substack{z_1,z_2 \in \mathcal{C}_{F,G} \\ z_1 \neq z_2}} \frac{\mathcal{W}_{\|\cdot\|}\big(\bar{P}^{\bar{\pi}}_{\mathcal{C}}(\cdot \mid z_1), \bar{P}^{\bar{\pi}}_{\mathcal{C}}(\cdot \mid z_2)\big)}{\|z_1 - z_2\|} \le \frac{\mathrm{diam}(\mathcal{C}_{F,G})}{\mathrm{sep}(\mathcal{C}_{F,G})}.$$

Proof. For any probability measures μ, ν supported on $\mathcal{C}_{F,G}$ and any coupling λ of (μ, ν),

$$\int_{\mathcal{C}_{F,G} \times \mathcal{C}_{F,G}} \|x - y\| \, \mathrm{d}\lambda(x,y) \le \mathrm{diam}(\mathcal{C}_{F,G}).$$

Taking the infimum over couplings gives

$$\mathcal{W}_{\|\cdot\|}(\mu,\nu) \le \mathrm{diam}(\mathcal{C}_{F,G}).$$

If $\mathrm{sep}(\mathcal{C}_{F,G}) > 0$, then every distinct $z_1, z_2 \in \mathcal{C}_{F,G}$ satisfies

$$\|z_1 - z_2\| \ge \mathrm{sep}(\mathcal{C}_{F,G}).$$

The reward difference is at most $2\bar{R}_{\max}$, and the Wasserstein distance between two distributions supported on the decoded codebook is at most its diameter. Dividing those two bounds by the separation gives

$$\frac{\left|\bar{R}^{\bar{\pi}}_{\mathcal{C}}(z_1) - \bar{R}^{\bar{\pi}}_{\mathcal{C}}(z_2)\right|}{\|z_1 - z_2\|} \le \frac{2\bar{R}_{\max}}{\mathrm{sep}(\mathcal{C}_{F,G})}, \qquad \frac{\mathcal{W}_{\|\cdot\|}\big(\bar{P}^{\bar{\pi}}_{\mathcal{C}}(\cdot \mid z_1), \bar{P}^{\bar{\pi}}_{\mathcal{C}}(\cdot \mid z_2)\big)}{\|z_1 - z_2\|} \le \frac{\mathrm{diam}(\mathcal{C}_{F,G})}{\mathrm{sep}(\mathcal{C}_{F,G})}.$$

□

Codes that decode to the same embedding give the same reward and next-code predictions, so merging them leaves the decoded model unchanged. The theorem measures separation between the remaining distinct embeddings. For a singleton codebook, both decoded Lipschitz constants are zero.

Remark 1 (Using the bounds in SPI)**.** *If $\gamma K^{\bar{\pi}}_{\bar{P},\mathcal{C}} < 1$, these constants bound the Lipschitz constant of the candidate's latent value function, as required by Eq.* (8)*. The improvement penalty in Eq.* (9) *requires a bound for both candidate and baseline value functions' Lipschitz constants (Appendix C).*

Connecting encoder outputs to the codebook. In our implementation, encoder outputs $z = \phi(s)$ reach the actor and model directly and need not lie in $\mathcal{C}_{F,G}$. For the analysis below, assign each output to a nearest codebook embedding:

$$\kappa(z) \in \arg\min_{\omega \in \Omega} \|z - \Psi_{F,G}(\omega)\|, \qquad \kappa : \mathcal{Z} \to \Omega.$$

Fix a deterministic tie-breaking order so that κ is measurable. This assignment is used only in the analysis; `DreamSPI` does not project real encoder outputs onto the codebook.

Remark 2 (Encoder outputs outside the codebook)**.** *Theorem 1 bounds the model on inputs in $\mathcal{C}_{F,G}$. For an encoder output outside this set, we must also account for how replacing z by $\Psi_{F,G}(\kappa(z))$ changes reward predictions, transition predictions, and action probabilities. The final calculation defines these additional errors.*

Applying SPI to code indices. For the transfer proof, regard each code $\omega \in \Omega$ as a state. The actor reads $\Psi_{F,G}(\omega)$; the model then predicts a reward and a next code. This gives a finite MDP with L^K states:

$$\bar{R}_{\Omega}(\omega, a) = \bar{R}(\Psi_{F,G}(\omega), a), \qquad \bar{P}_{\Omega}(\omega' \mid \omega, a) = \bar{p}(\omega' \mid \Psi_{F,G}(\omega), a).$$

Start at $\omega_0 = \kappa(\phi(s_0))$, with $s_0 \sim \rho_0$. Under a latent policy $\bar{\pi}$, the expected reward and next-code distribution are

$$\bar{R}^{\bar{\pi}}_{\Omega}(\omega) = \sum_{a \in \mathcal{A}} \bar{\pi}(a \mid \Psi_{F,G}(\omega)) \bar{R}_{\Omega}(\omega, a), \qquad \bar{P}^{\bar{\pi}}_{\Omega}(\omega' \mid \omega) = \sum_{a \in \mathcal{A}} \bar{\pi}(a \mid \Psi_{F,G}(\omega)) \bar{P}_{\Omega}(\omega' \mid \omega, a).$$

Give distinct codes distance one: $d_{\Omega}(\omega_1, \omega_2) = \mathbf{1}\{\omega_1 \neq \omega_2\}$. The next lemma bounds reward and transition changes under this metric for every latent policy. These bounds do not depend on the distances between decoded embeddings.

Lemma 1 (Finite codes are regular under the code metric)**.** *For every latent policy $\bar{\pi}$ and every pair of distinct codes $\omega_1, \omega_2 \in \Omega$,*

$$\mathcal{W}_{d_{\Omega}}\big(\bar{P}^{\bar{\pi}}_{\Omega}(\cdot \mid \omega_1), \bar{P}^{\bar{\pi}}_{\Omega}(\cdot \mid \omega_2)\big) \leq 1 = d_{\Omega}(\omega_1, \omega_2)$$

whenever $\omega_1 \neq \omega_2$. Hence $K^{\bar{\pi}}_{\bar{P}} \leq 1$, independently of the logits network. Similarly, if $\bar{R}_{\Omega}$ is bounded by $\bar{R}_{\max}$, then $K^{\bar{\pi}}_{\bar{R}} \leq 2\bar{R}_{\max}$.

Proof. A coupling λ of probability measures μ and ν on Ω is a probability measure on $\Omega \times \Omega$ whose first and second marginals are μ and ν. Consequently $\lambda(\Omega \times \Omega) = 1$. Because $0 \leq d_{\Omega}(u, v) \leq 1$,

$$\int_{\Omega \times \Omega} d_{\Omega}(u, v) \, \mathrm{d}\lambda(u, v) \leq \lambda(\Omega \times \Omega) = 1.$$

Taking the infimum over couplings gives

$$\mathcal{W}_{d_{\Omega}}(\mu, \nu) \leq 1.$$

Apply this to the two policy-induced transition distributions. For $\omega_1 \neq \omega_2$, the unit metric gives $d_{\Omega}(\omega_1, \omega_2) = 1$, so

$$\frac{\mathcal{W}_{d_{\Omega}}\big(\bar{P}^{\bar{\pi}}_{\Omega}(\cdot \mid \omega_1), \bar{P}^{\bar{\pi}}_{\Omega}(\cdot \mid \omega_2)\big)}{d_{\Omega}(\omega_1, \omega_2)} \leq 1,$$

which proves $K^{\bar{\pi}}_{\bar{P}} \leq 1$ under the code metric. Moreover, if $|\bar{R}_{\Omega}(\omega, a)| \leq \bar{R}_{\max}$, then the policy-induced reward is bounded as

$$\big|\bar{R}^{\bar{\pi}}_{\Omega}(\omega)\big| \leq \bar{R}_{\max}.$$

Consequently, for $\omega_1 \neq \omega_2$,

$$\frac{\big|\bar{R}^{\bar{\pi}}_{\Omega}(\omega_1) - \bar{R}^{\bar{\pi}}_{\Omega}(\omega_2)\big|}{d_{\Omega}(\omega_1, \omega_2)} \leq 2\bar{R}_{\max},$$

so $K^{\bar{\pi}}_{\bar{R}} \leq 2\bar{R}_{\max}$. □

Remark 3 (Bounding predicted rewards)**.** *The reward Lipschitz bound depends on the model's predictions on the codebook. Constraining these predictions to $[-1, 1]$ lets us take $\bar{R}_{\max} = 1$. Clipping observed rewards only bounds the training targets; it does not prevent predictions outside that interval.*

Policy improvement and representation quality. In the following, Theorem 2 applies the two bounds from Appendix C to policies that act on the assigned codebook embedding. The model losses below are measured on data collected by the behavior policy defined in the theorem.

Theorem 2 (Finite-code `DeepSPI` transfer)**.** *Assume the bounded rewards and episodic reset construction of Appendix C, with finite mean cycle length and $\gamma > 1/2$. Hold the encoder, model, and codebook fixed. Assume $|\bar{R}_{\Omega}(\omega, a)| \leq \bar{R}_{\max}$ on the finite codebook. Let $\bar{\pi}_b$ and $\bar{\pi}$ be latent policies, and define their* projected *real-environment policies by*

$$\pi^{\Omega}_b(a \mid s) = \bar{\pi}_b(a \mid \Psi_{F,G}(\kappa(\phi(s)))), \qquad \pi^{\Omega}(a \mid s) = \bar{\pi}(a \mid \Psi_{F,G}(\kappa(\phi(s)))).$$

Let π^{Ω}_b generate the batch. Assume that both projected policies have the same action support at every state, and define the local policy drift

$$\Delta = \sup_{s,\ a \in \operatorname{supp} \pi^{\Omega}_b(\cdot \mid s)} \frac{\pi^{\Omega}(a \mid s)}{\pi^{\Omega}_b(a \mid s)}.$$

If $\pi^\Omega \in \mathcal{N}^\epsilon(\pi_b^\Omega)$, then $\Delta \le 1+\epsilon$; hence the same deep SPI radius condition is met whenever $\epsilon < \epsilon_{\max} = 1/\gamma - 1$. Assume $\Delta < 1/\gamma$.

Define the code-level local losses under the behavior visitation distribution:

$$L_R = \mathbb{E}_{(s,a)\sim\nu^{\pi_b^\Omega}}\left[\left|R(s,a) - \bar{R}_\Omega(\kappa(\phi(s)),a)\right|\right],$$

$$L_P = \mathbb{E}_{(s,a)\sim\nu^{\pi_b^\Omega}}\left[\mathcal{W}_{d_\Omega}\big((\kappa\circ\phi)_\# P(\cdot \mid s,a), \bar{P}_\Omega(\cdot \mid \kappa(\phi(s)),a)\big)\right].$$

By Lemma 1, $K_V = 2\bar{R}_{\max}/(1-\gamma)$ bounds both latent value functions' Lipschitz constants under d_Ω. Applying Eq. (9) gives

$$J_\mathcal{M}(\pi^\Omega) - J_\mathcal{M}(\pi_b^\Omega) \ge J_\Omega(\bar{\pi}) - J_\Omega(\bar{\pi}_b) - \zeta,$$

where J_Ω is the return in the finite code MDP, starting at $\omega_0 = \kappa(\phi(s_0))$ with $s_0 \sim \rho_0$, and

$$\zeta = \mathrm{AEL}(\pi_b^\Omega)\left(\frac{L_R}{\gamma} + K_V L_P\right)\left(\frac{1}{1/\Delta - \gamma} + \frac{1}{1-\gamma}\right).$$

Here AEL *denotes mean reset-to-reset cycle length under the indicated behavior policy. The same local-loss quantities also give the representation-quality statement: for any $\eta > 0$, define*

$$\delta(\eta) = 4\,\frac{L_R + \gamma K_V L_P}{\eta(1/\Delta - \gamma)}.$$

Let $d^{\pi_b^\Omega}$ be the state marginal of $\nu^{\pi_b^\Omega}$ and let λ_Ω be any coupling of $d^{\pi_b^\Omega}$ with itself. For $(s_1,s_2)\sim\lambda_\Omega$, with probability at least $1-\delta(\eta)$,

$$\left|V^{\pi^\Omega}(s_1) - V^{\pi^\Omega}(s_2)\right| \le K_V d_\Omega(\kappa(\phi(s_1)),\kappa(\phi(s_2))) + \eta.$$

Proof. Lemma 1 applies to both latent policies. Their transition constants are at most 1 and their reward constants at most $2\bar{R}_{\max}$, so K_V bounds both latent value functions' Lipschitz constants. The reset assumption relates the stationary and initial state distributions, as required by Eq. (9). Substituting L_R, L_P and the ratio Δ gives the improvement penalty. Applying the sampled-pair bound in Appendix C to $\kappa\circ\phi$ and d_Ω gives the final probability statement. These results concern projected policies and the finite code MDP. □

Relating the bounds to encoder outputs. We now relate the losses in Theorem 2 to prediction errors between token embeddings. Write $z = \phi(s)$, $z' = \phi(s')$, and draw $\hat{\omega} \sim \bar{p}(\cdot \mid z,a)$. For rewards, measure the prediction error and the change caused by replacing the model input with its assigned embedding:

$$L_R^{\mathrm{dec}} = \mathbb{E}_{\nu^{\pi_b^\Omega}}|R(s,a) - \bar{R}(z,a)|, \qquad \mathrm{res}_R^{\mathrm{in}} = \mathbb{E}_{\nu^{\pi_b^\Omega}}|\bar{R}(z,a) - \bar{R}(\Psi_{F,G}(\kappa(z)),a)|,$$

For transitions, measure the expected Euclidean prediction error, the distance from the next encoder output to its assigned embedding, and the change in the next-code distribution:

$$L_P^{\mathrm{dec}} = \mathbb{E}_{\substack{(s,a)\sim\nu^{\pi_b^\Omega},\ s'\sim P(\cdot|s,a)\\ \hat{\omega}\sim\bar{p}(\cdot|z,a)}}\left\|z' - \Psi_{F,G}(\hat{\omega})\right\|,$$

$$\mathrm{res}_P^{\mathrm{tgt}} = \mathbb{E}_{\substack{(s,a)\sim\nu^{\pi_b^\Omega}\\ s'\sim P(\cdot|s,a)}}\left\|z' - \Psi_{F,G}(\kappa(z'))\right\|,$$

$$\mathrm{res}_P^{\mathrm{in}} = \mathbb{E}_{\nu^{\pi_b^\Omega}}\mathcal{W}_{d_\Omega}(\bar{p}(\cdot \mid z,a), \bar{p}(\cdot \mid \Psi_{F,G}(\kappa(z)),a)).$$

For this code-level comparison, additionally assume $\Psi_{F,G}$ is injective and $\mathrm{sep}(\mathcal{C}_{F,G}) > 0$. Couple the assigned real successor code $\kappa(z')$ with $\hat{\omega}$. A code mismatch then costs at least $\mathrm{sep}(\mathcal{C}_{F,G})$ after decoding. The triangle inequality gives

$$L_R \le L_R^{\mathrm{dec}} + \mathrm{res}_R^{\mathrm{in}}, \qquad L_P \le \frac{L_P^{\mathrm{dec}} + \mathrm{res}_P^{\mathrm{tgt}}}{\mathrm{sep}(\mathcal{C}_{F,G})} + \mathrm{res}_P^{\mathrm{in}}.$$

Thus, with an injective decoder, the loss on codes is bounded by the Euclidean prediction error and the errors introduced by assigning inputs and targets to the codebook. The expected distances here

differ from the smooth-L1 training losses (Appendix N.2). The input terms vanish if the model is evaluated at assigned embeddings; the target term vanishes if next encoder outputs already lie in the codebook.

Assigning an embedding can also change the actor's action probabilities. On common action support, measure this change by

$$\mathrm{res}_{\pi}(\bar{\pi}) = \sup_{s,\ a \in \mathrm{supp}\,\bar{\pi}(\cdot|\phi(s))} \left| \log \frac{\bar{\pi}(a \mid \phi(s))}{\bar{\pi}(a \mid \Psi_{F,G}(\kappa(\phi(s))))} \right|, \qquad \bar{\pi} \in \{\bar{\pi}_b, \bar{\pi}\}.$$

If both policy residuals are zero, assigning encoder outputs to the codebook leaves both policies unchanged, and Theorem 2 applies to the deployed policies. Otherwise, the return bound must also account for these changes in action probabilities; the model losses L_R^{dec} and L_P^{dec} alone do not bound them.

K Further Related Work

Safe and regularized policy improvement. SPI methods guarantee improvement over a baseline from uncertain data (Thomas et al., 2015; Laroche et al., 2019), while trust-region and regularized policy optimization stabilize online updates by limiting policy movement (Schulman et al., 2015; 2017; Geist et al., 2019; Kuba et al., 2022). `DreamSPI` limits changes to the composed policy defined by its encoder and latent actor.

Replay-centric world models. World models predict latent states and rewards for planning and policy learning, often using observation reconstruction or generation to train the latent representation. The original `World Models` trains a generative observation model (Ha & Schmidhuber, 2018); `PlaNet` plans in learned latent dynamics (Hafner et al., 2019); and `DreamerV2`/`DreamerV3` learn from replay-seeded imagination (Hafner et al., 2021; 2025); `MBPO` augments off-policy actor-critic learning with fixed-horizon rollouts (Janner et al., 2019); and `MuZero` learns a prediction model for tree search (Schrittwieser et al., 2020). `SimPLe` is also relevant here: it trains `PPO` inside its learned video/latent model for imagined rollouts, while the model itself is updated from an accumulated real-data buffer and rollouts restart from stored real states (Kaiser et al., 2020). These systems handle distribution shift through replay, off-policy value learning, search, or accumulated datasets. `DreamSPI` learns its model from current-policy data and measures model errors under that policy. The model predicts the representation used for control, without reconstructing observations. Our reconstruction ablation shows that adding pixel prediction can reduce return, supporting the use of reward and transition errors to train this representation.

On-policy and near-on-policy learned-model agents. Closer predecessor methods combine learned models with policy gradients or online policy search. `PILCO` learns Gaussian-process dynamics over continuous state-action vectors and optimizes policies through analytic rollout predictions from all collected trials (Deisenroth & Rasmussen, 2011). `PILCO` is designed for low-dimensional observations and policy parameters in continuous control. It collects trials sequentially and fits its model to data accumulated under previous policies. `DreamSPI` fits its model to the current policy's batch. Imagination-Augmented Agents train or reuse environment models whose rollouts are encoded and supplied as context to an `A3C`-style policy (Racanière et al., 2017); these predictions help the policy choose actions. `DreamSPI` also trains the policy on predicted trajectories, with local model-error penalties guiding the update.

Representation learning in RL. `DeepMDP` connects latent transition and reward prediction to representation learning (Gelada et al., 2019). Bisimulation and norm-based approaches study when latent proximity preserves behavioral similarity (Castro et al., 2021; Agarwal et al., 2021a), and recent work stresses actor-critic, data-collection, and gradient-dynamics effects on deep-RL representations (Garcin et al., 2025; Mayor et al., 2025; Sokar et al., 2023; Liu et al., 2025a). `DreamSPI` asks whether such representations remain reliable for locally constrained model-based policy updates.

Discrete representation learning. `VQ-VAE` maps encoder outputs to a finite codebook of learned embeddings (van den Oord et al., 2017). It selects the closest embedding, decodes it to reconstruct the input, and uses auxiliary losses to keep the embedding and encoder output close. `DreamSPI` uses a

finite codebook to predict dynamics from on-policy data. The transition model predicts a codebook embedding and minimizes its distance to the next encoded state. The actor and critic receive the encoder's continuous token embeddings directly and learn from both real and imagined trajectories.

Latent-predictive objectives for RL and world models. Self-predictive representation learning, BYOL, and JEPA-style objectives replace pixel reconstruction by prediction or alignment in representation space (Guo et al., 2020; Grill et al., 2020; Schwarzer et al., 2021; Ye et al., 2021; Schwarzer et al., 2023; Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025). BYOL is a self-supervised image-representation method: it trains an online network to predict a slowly updated target representation of another augmentation of the same image, using stop-gradient and EMA updates to stabilize this self-supervised target. Recent work brings the same latent-prediction principle closer to control. ACT-JEPA combines imitation learning and self-supervision by predicting action chunks and abstract observation sequences (Vujinovic & Kovacevic, 2025). JEPA for RL studies how to adapt image-based JEPA encoders to RL observations, with particular attention to collapse prevention (Kenneweg et al., 2025). NE-Dreamer removes the pixel decoder from a Dreamer-style RSSM and uses a causal transformer with a redundancy-reduction loss to predict the next encoder embedding (Bredis et al., 2026). JEDI learns an online latent diffusion world model on Atari100k by denoising future latents in a JEPA-style framework (Lim et al., 2026). TD-JEPA instead learns offline, reward-free multi-policy latent-predictive representations for zero-shot RL through TD-style successor features (Bagatella et al., 2026).

BYOL and JEPA-style systems use stop-gradient, EMA teachers, masking, or redundancy reduction to stabilize latent targets. DreamSPI jointly trains the encoder and model using reward and transition errors, with regularized updates to the composed policy. Its predictions target the current encoder's token embeddings. This choice matches the SPI bound, which uses model errors in the representation that defines the deployed policy. An EMA target changes the error being measured; stop-gradient alone preserves its value but removes the target's contribution to the encoder gradient. Our SPR, stop-gradient, and EMA ablations do not improve on the current-encoder objective.

Optimization and normalization in deep RL. Large deep-RL networks can degrade when non-stationary training interacts with weak gradients, dormant units, and ill-conditioned updates (Sokar et al., 2023; Liu et al., 2025a; Creus Castanyer et al., 2025). PQN stabilizes online parallelized TD learning with regularization such as LayerNorm (Gallici et al., 2025). DreamSPI uses Kron, Mish, and LayerNorm to stabilize updates as its encoder, model, and policy change.

L Evaluation Details and Additional Plots

L.1 Summary Views Used by the Main Text

Figures 10 and 11 compare DreamSPI with DeepSPI game by game, using HNS for ALE and ANS for Octax. Table 5 reports the final ALE mean, median, and IQM HNS.

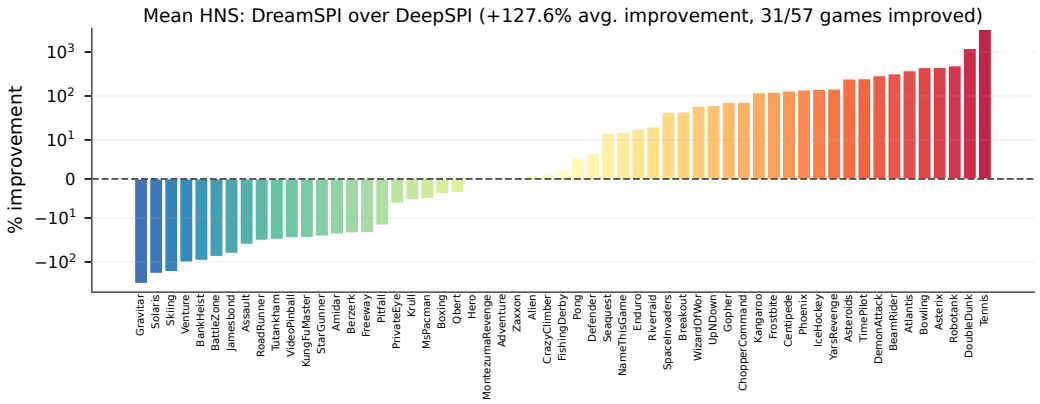


Figure 10: Per-game percentage improvement of `DreamSPI` over `DeepSPI` on the standard 57-game ALE diagnostic set, using final mean HNS per game. Positive values favor `DreamSPI`; the symlog axis keeps large gains and losses visible.

Table 5: Final HNS summary on the common ALE aggregate set.

Algorithm	Mean	Median	IQM
`DeepMDP (PPO)`	1.811	0.409	0.635
`PPO`	1.910	0.426	0.669
`DreamSPI-v0`	0.484	0.070	0.127
`DeepSPI`	2.066	0.528	0.715
`DreamSPI`	**2.934**	**0.663**	**0.810**

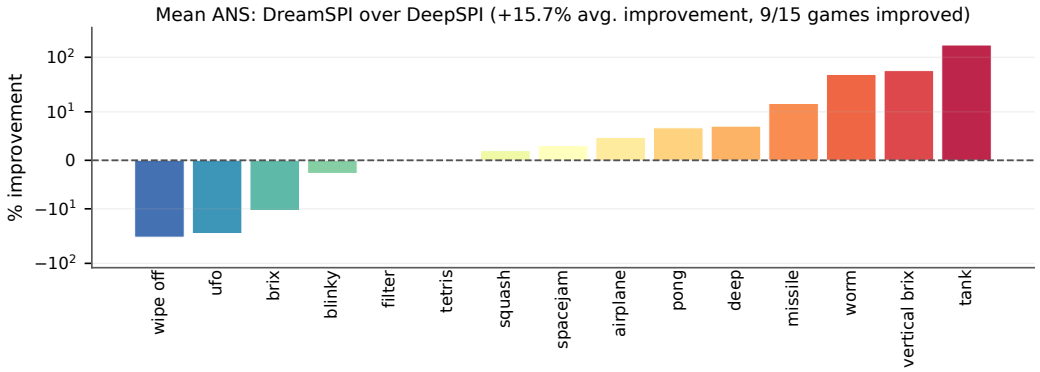


Figure 11: Per-game percentage improvement of `DreamSPI` over `DeepSPI` on Octax-15, using final mean ANS per game. Positive values favor `DreamSPI`.

L.2 Aggregate Metrics

We summarize normalized scores with the mean, median, and interquartile mean (IQM); some figures also report optimality gap. For each algorithm, let $X \in \mathbb{R}^{Q \times E}$ contain the final HNS or ANS, with entry $X_{q,e}$ for seed q and environment e. Learning curves apply the same calculations at each checkpoint. Following Agarwal et al. (2021b), we first average seeds within each environment, $\bar{X}_e = Q^{-1} \sum_{q=1}^{Q} X_{q,e}$, then take the mean or median of these E averages. The mean reflects the size of gains and losses but can be dominated by a few high-score games. The median is less sensitive to extreme scores, but ignores the size of changes away from the middle.

We use IQM as the main summary because it averages many scores while limiting the influence of extreme runs. Flatten X into $N = QE$ scores and sort them as $y_{(1)} \leq \cdots \leq y_{(N)}$. IQM removes the lowest and highest quarters and averages the middle half: it is a 25%-trimmed mean. When N is divisible by four,

$$\mathrm{IQM}(X) = \frac{2}{N} \sum_{i=N/4+1}^{3N/4} y_{(i)}.$$

At the population level, IQM is twice the area under the quantile function between the 25th and 75th percentiles. This makes IQM less sensitive than the mean to exceptional games or seeds, but more informative than the median because it averages the whole central half of the distribution. For the benchmark aggregates, we report 95% stratified bootstrap confidence intervals. For each game, we resample its runs with replacement and recompute the aggregate score. The resulting interval estimates uncertainty due to random seeds on these games.

L.3 Score Normalization

For ALE we report human-normalized score,

$$\mathrm{HNS}(G, e) = \frac{G - r_e}{h_e - r_e},$$

where G is the raw episodic return on environment e, and r_e and h_e are the random-agent and human reference scores. This normalization follows the convention popularized by the original `DQN` Nature paper (Mnih et al., 2015). We use HNS only as a normalized comparison scale, then aggregate with robust statistics rather than relying only on mean score.

For Octax, human scores are not available. We therefore use the analogous agent-normalized score,

$$\mathrm{ANS}(G, e) = \frac{G - r_e^{\mathrm{rand}}}{a_e^{\mathrm{GPT}} - r_e^{\mathrm{rand}}},$$

where G is the raw episodic return, r_e^{rand} is the mean return of five uniform-random episodes, and a_e^{GPT} is a conservative reference from GPT-5.5 (medium) visual playtests. We sum rewards over each episode, as in training, rather than read the final CHIP-8 score register. These references put progress on a comparable scale across games; they do not measure expert human performance.

Use this template to collect new reference scores following the playtest procedure. Replace the bracketed fields before use.

Prompt template: Octax reference scores

```
Collect random-agent and agent reference returns for Octax ANS.
Inputs:  repository [PATH], games [GAME IDS], five seeds [SEEDS],
episode limit [STEPS], and playtest budget per game [BUDGET].
Use the repository's Octax environments and pygame player.  Record the
emulator version, action set, sticky-action probability, frame skip,
and episode limit.  Keep these settings fixed across random and agent
trials.

1. Run five uniform-random episodes per game.  Sum rewards within each
   episode and average these five returns for the random reference.

2. Inspect rendered frames to understand the controls and scoring.
   Test simple action schedules or visual policies within the budget.
   Save each trial's seed, action trace or policy, and summed rewards.
   Choose a conservative agent reference supported by these trials and
   explain the choice.

3. Return one row per game with the random returns, their mean,
   the playtest returns, the chosen agent reference, and its
   justification.  Attach the settings and trial logs.

Use episodic returns, not final score-register values.  Flag
unmeasured references or an agent reference equal to the random mean;
do not invent a score.
```

The per-game plots in Appendices L.4, L.5, and O use these normalized scores. A gain of the same size can therefore correspond to different raw-return gains across games.

L.4 Additional ALE Views

The ALE figures show whether gains also hold against PPO and DeepMDP, and whether the per-game comparisons change when IQM or median replaces the seed mean.

The percentage-improvement diagnostic is computed per game. For each game and baseline, we first aggregate the final normalized scores across seeds for DreamSPI and the baseline, then report

$$100\,\frac{\mathrm{score}_{\texttt{DreamSPI}}(e)-\mathrm{score}_{\mathrm{baseline}}(e)}{\max\{|\mathrm{score}_{\mathrm{baseline}}(e)|,\tau_{\mathrm{num}}\}},$$

with $\tau_{\mathrm{num}}=10^{-6}$ only preventing division by zero. Figure 10 uses this diagnostic for the closest comparator, DeepSPI, with arithmetic means over the five seeds. Figure 12 adds PPO and DeepMDP, showing whether DreamSPI improves over several baselines or only one. Figures 13 and 14 repeat the same computation after replacing the seed mean by IQM and median, respectively. With five seeds, IQM drops the lowest and highest scores for that algorithm and environment and averages the three central scores. These comparisons test whether a few extreme seeds account for the per-game gains.

Figure 15 summarizes how often each algorithm ranks near the top; lower ranks are better. Ranks compare the ordering of algorithms without weighting games by the size of their scores. Figure 16 gives per-game learning curves, showing whether final gaps arise from faster learning, later improvements, variation across seeds, or failure to learn a game.

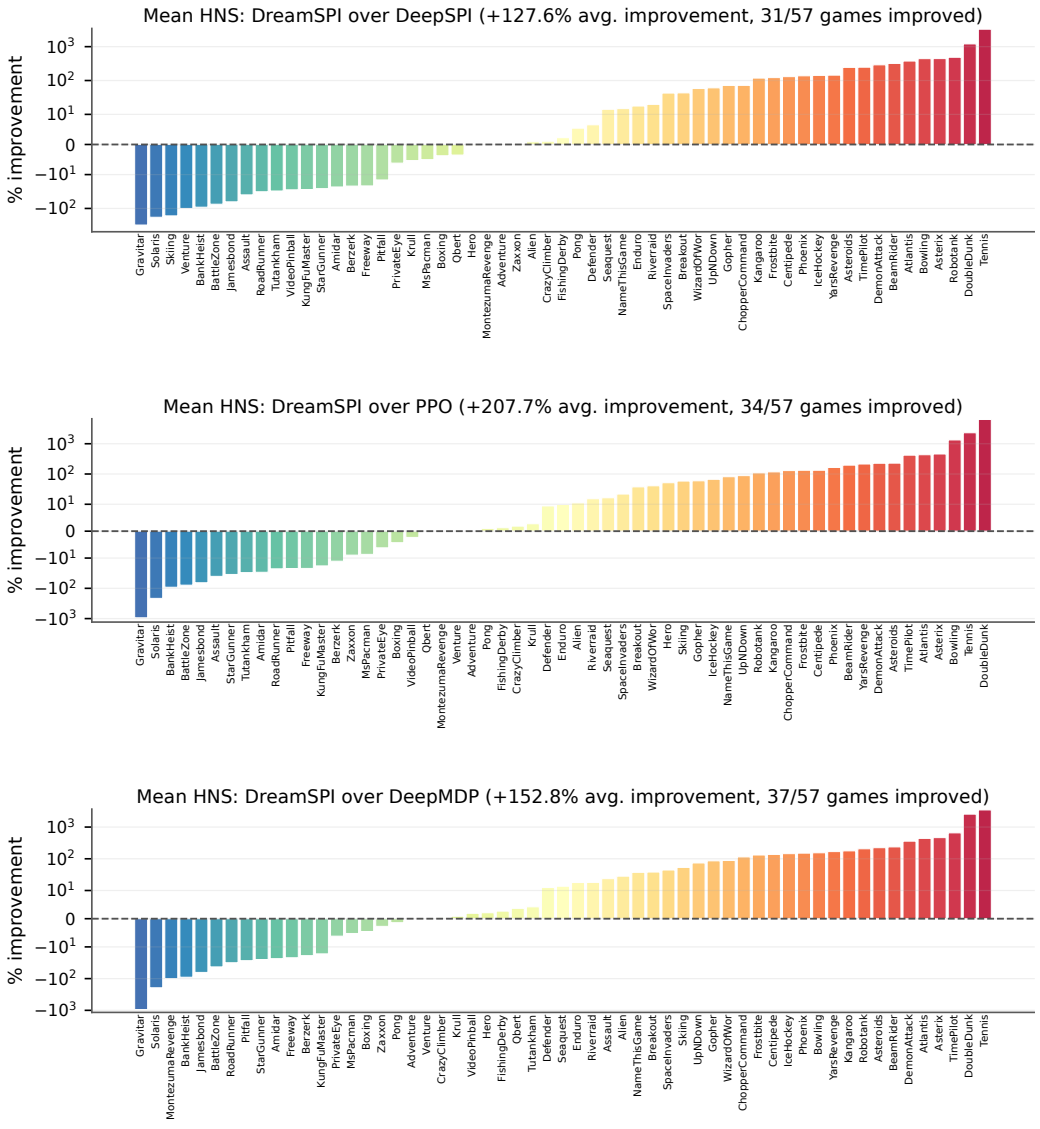


Figure 12: Per-game percentage improvement of `DreamSPI` over `DeepSPI`, `PPO`, and `DeepMDP` across all 57 ALE games. Scores are final HNS means across seeds for each game and algorithm.

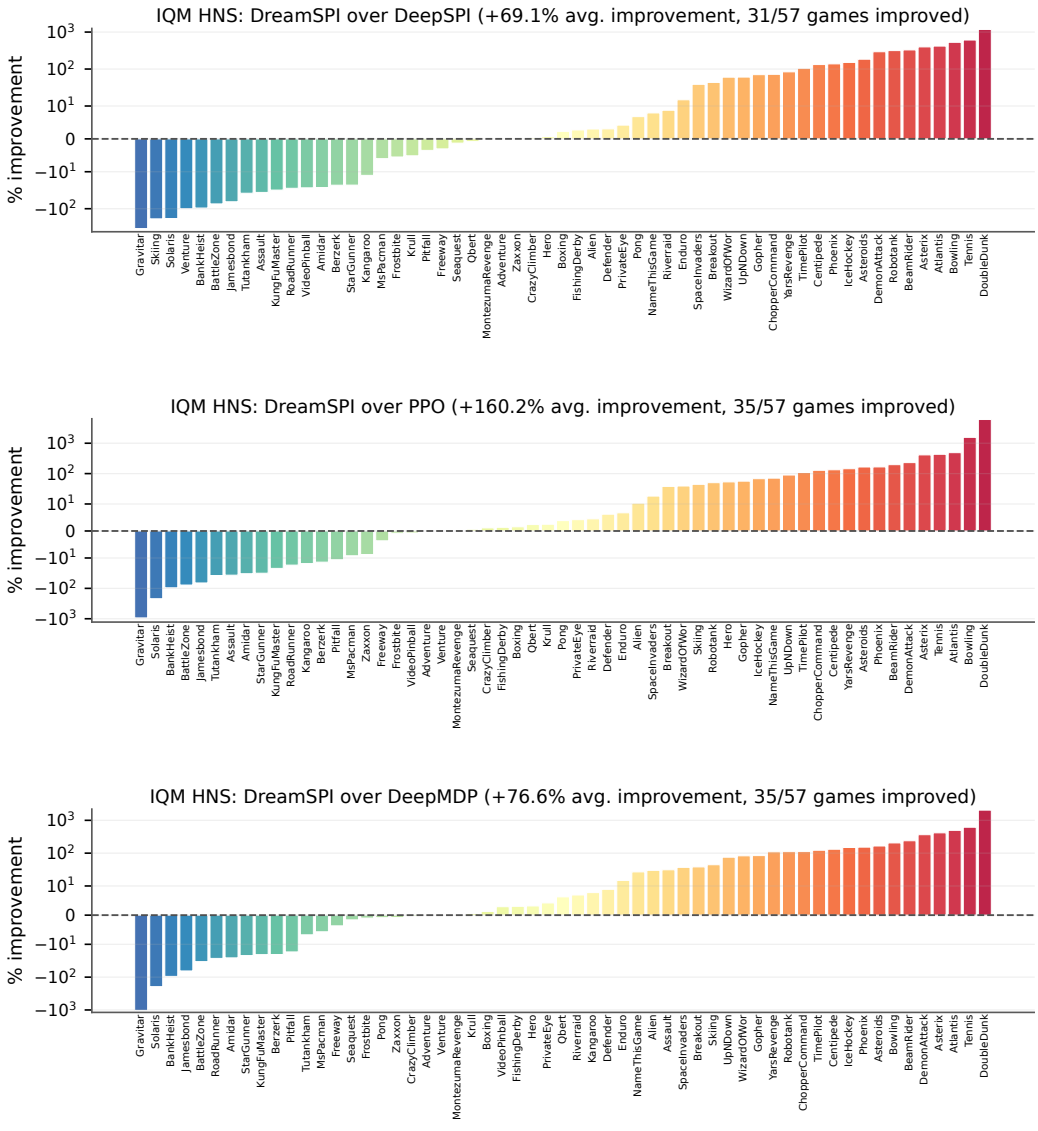


Figure 13: Per-game percentage improvement across all 57 ALE games. We use the IQM HNS across seeds for each game and algorithm to reduce the influence of extreme runs.

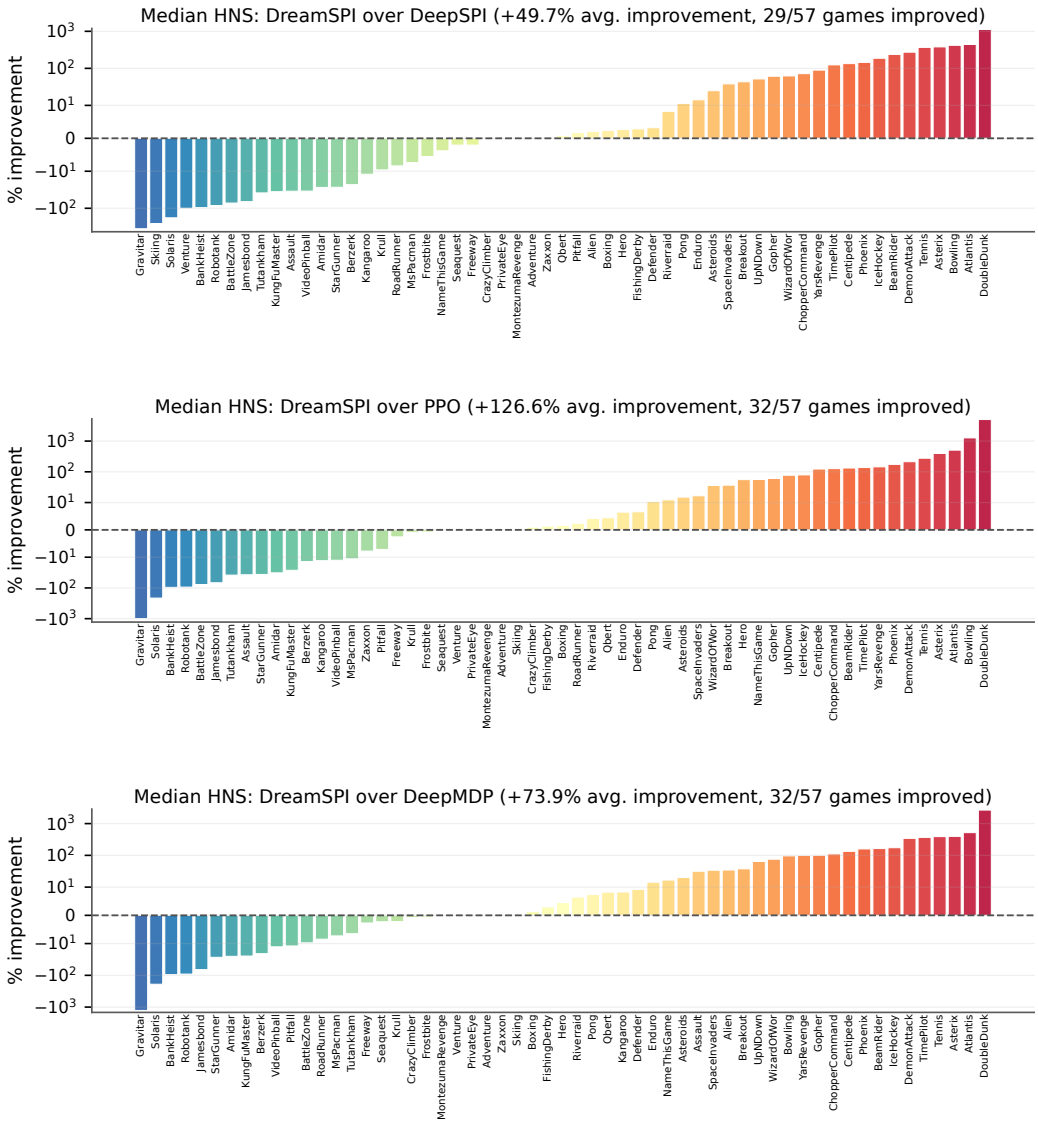


Figure 14: Percentage improvement of `DreamSPI` over each baseline on the 57 ALE games. For each game and algorithm, we first take the median HNS across seeds, rather than the mean, then compute the percentage.

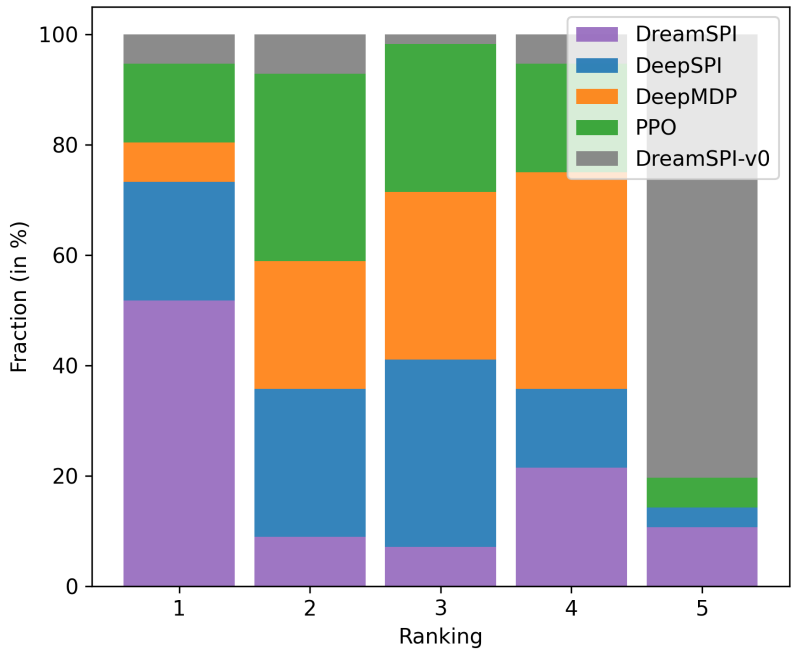


Figure 15: Aggregate ALE ranks. Lower is better. `DreamSPI` has the best rank profile overall, while `DeepSPI` remains the closest comparator.

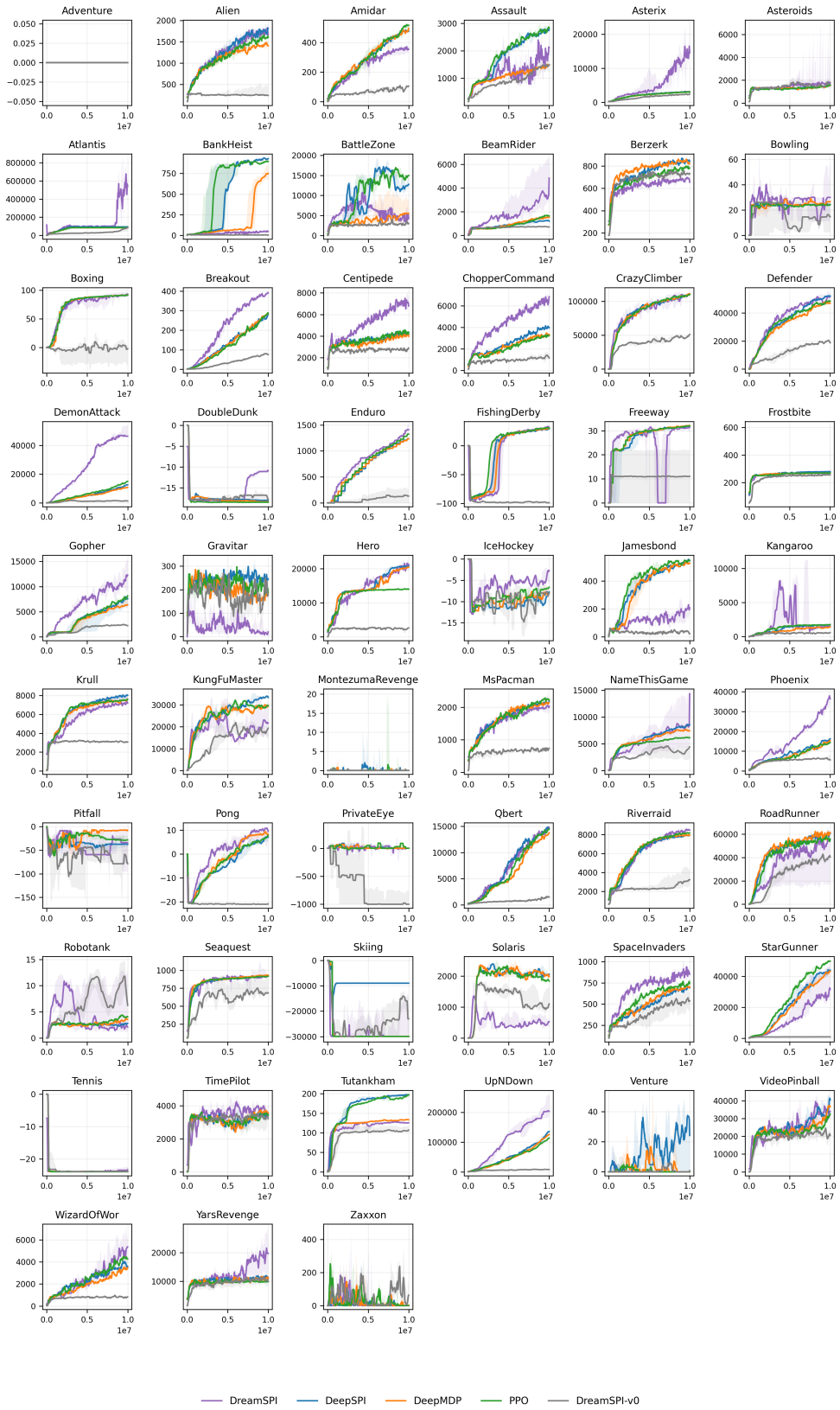


Figure 16: Learning curves (median and IQR) for all 57 ALE games. Appendix O relates the differences across games to their mechanics.

L.5 OCTAX DETAILS

The Octax diagnostics use ANS in place of HNS. Figure 11 compares per-game gains against `DeepSPI` to test whether the ALE result extends to another simulator. Figure 17 also includes `PPO` and `DeepMDP`. Figures 18 and 19 replace seed means with IQM and median to check sensitivity to that choice.

Figure 20 reports final aggregates with confidence intervals, Fig. 21 reports ranks, and Fig. 22 shows per-game learning curves. `DreamSPI` has the highest mean score on seven Octax-15 games, `DeepMDP` on four, and `PPO` and `DeepSPI` on two each. Strong results include *Tank*, *Worm*, *Vertical Brix*, *Squash*, *Pong*, *Spacejam*, and *Airplane*; the weakest cases are *Blinky*, *UFO*, and *Wipe Off*. On several games, `DreamSPI` learns faster and retains higher scores than the baselines. Other games show little difference or favor a baseline.

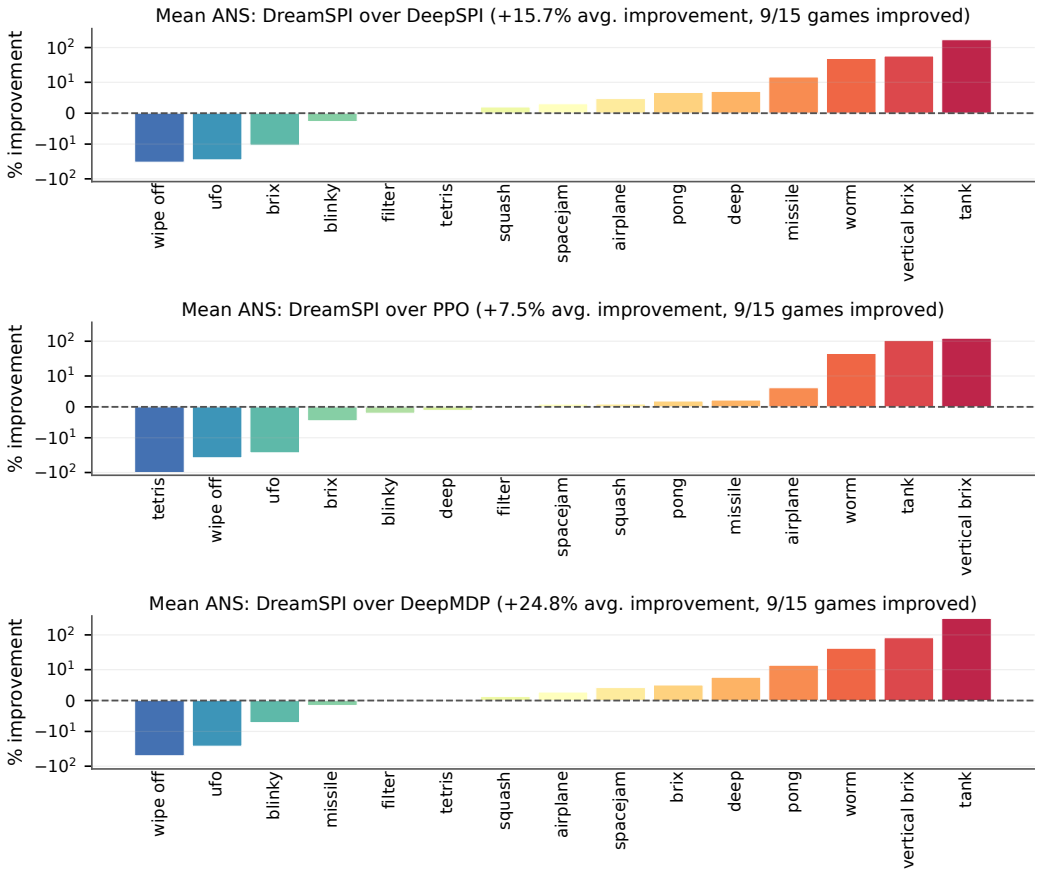


Figure 17: Per-game percentage improvement of `DreamSPI` over `DeepSPI`, `PPO`, and `DeepMDP` on Octax-15, using final ANS means across seeds for each game and algorithm.

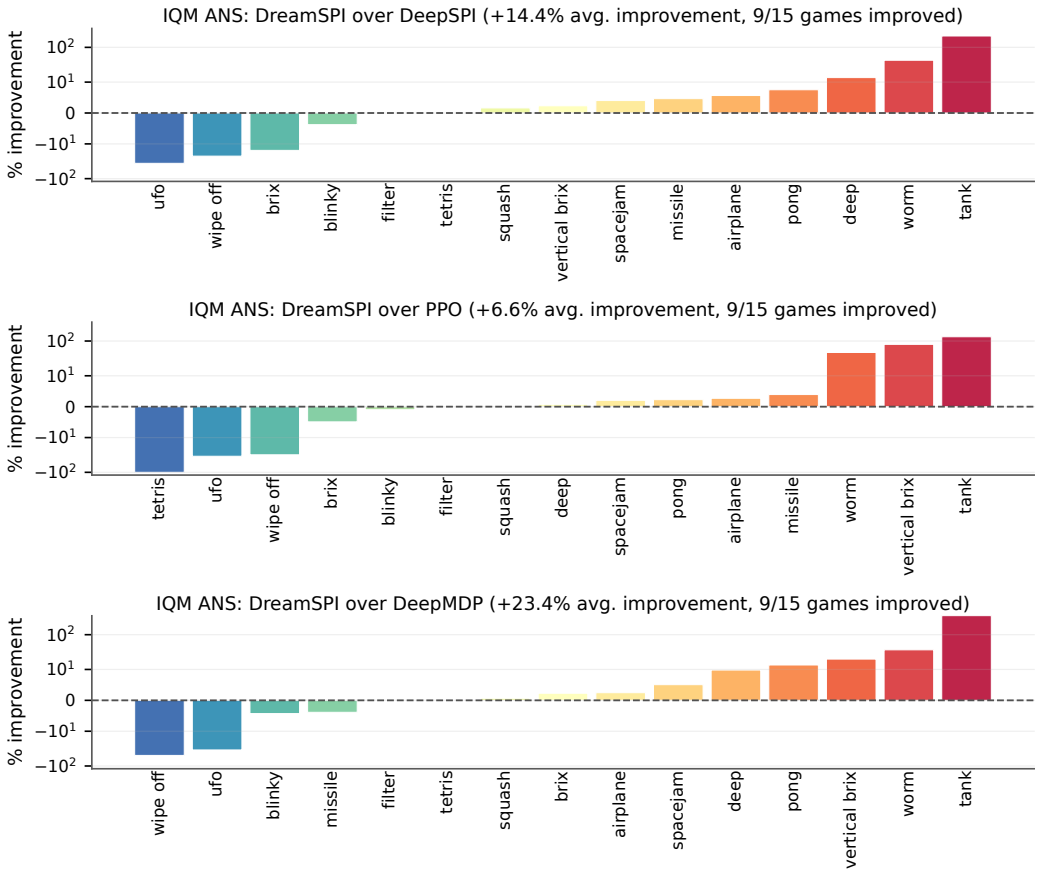


Figure 18: Octax-15 per-game percentage improvement using IQM ANS within each game and algorithm.

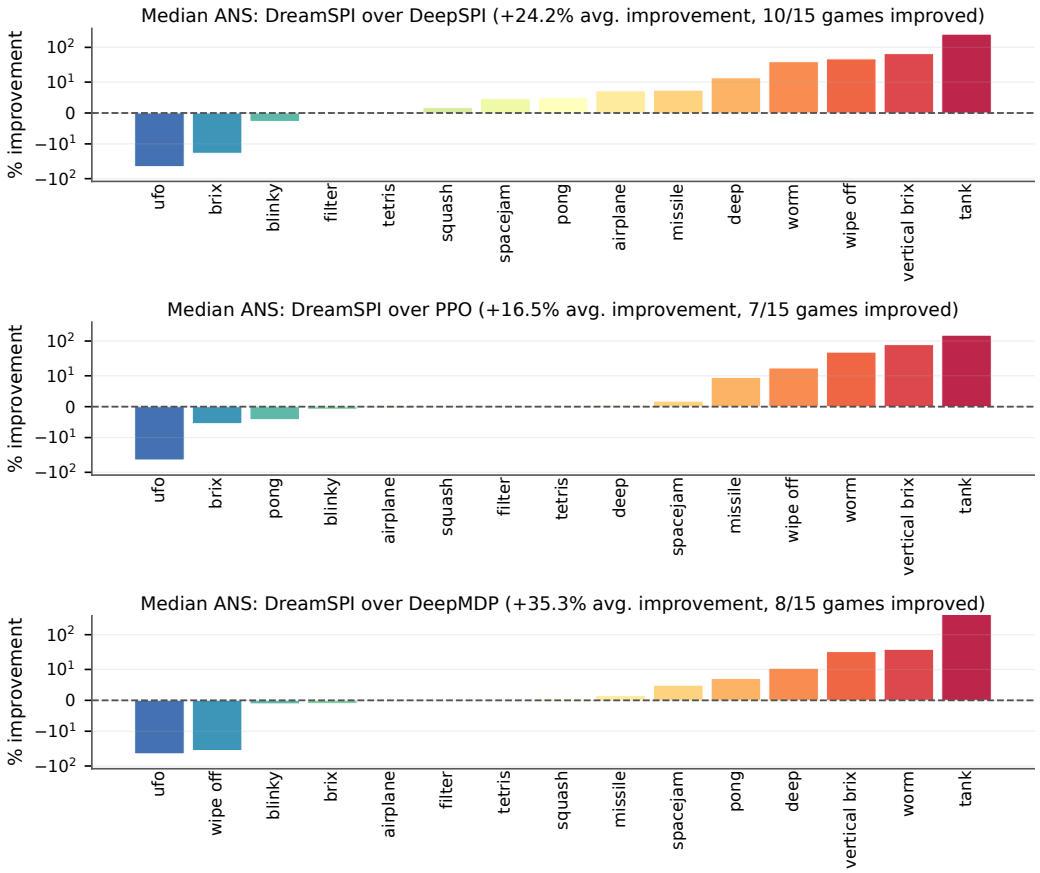


Figure 19: Octax-15 per-game percentage improvement using median ANS within each game and algorithm.

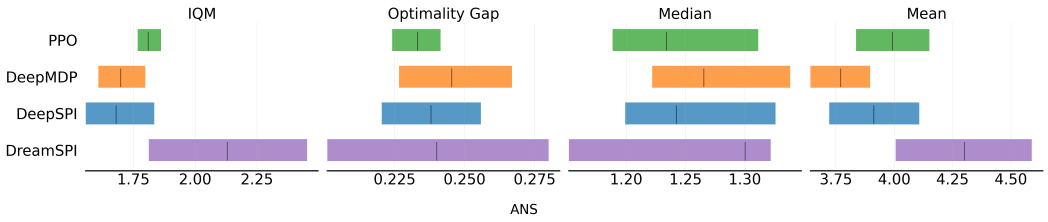


Figure 20: Final aggregate metrics on Octax-15 with 95% stratified bootstrap confidence intervals. Higher mean, median, and IQM are better; lower optimality gap is better.

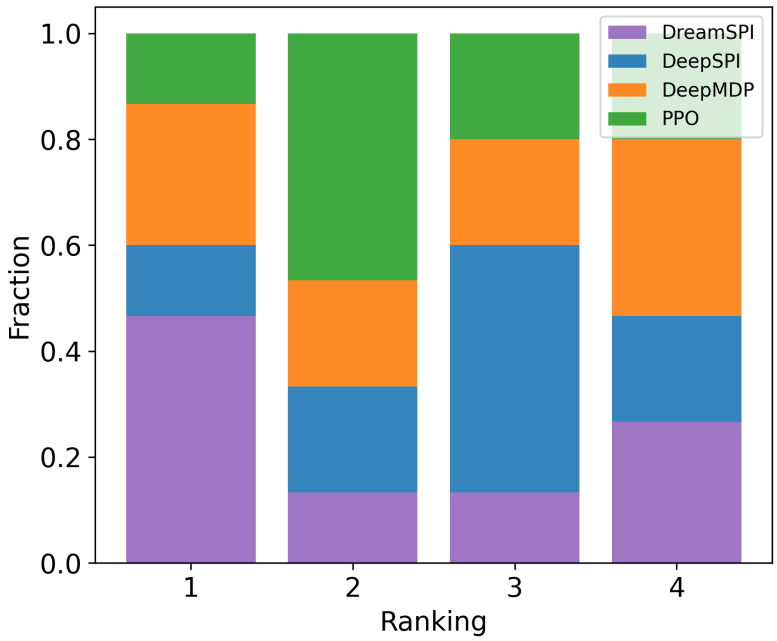


Figure 21: Aggregate Octax-15 ranks; lower is better. `DreamSPI` also leads by this measure, although several games remain close.

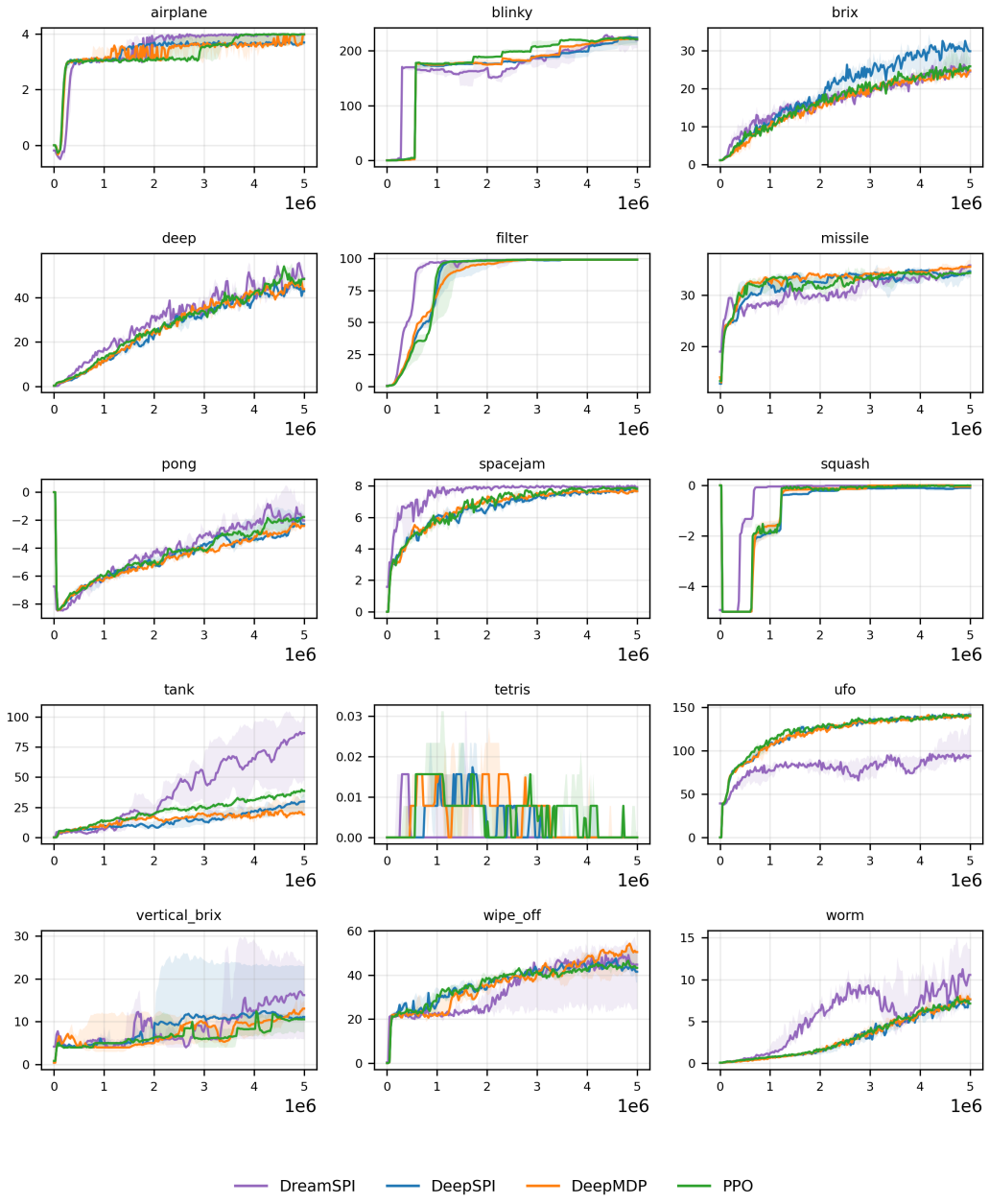


Figure 22: Per-environment Octax-15 learning curves (median and IQR).

L.6 Atari-5 Ablation Diagnostics

We test design choices on Atari-5 to reduce training cost, then evaluate the selected configuration on all 57 ALE games. Figures 23–26 add final aggregate metrics and full-width learning curves for the mechanism, optimizer, and representation-interface ablations shown in the main text.

L.6.1 Representation-Interface Ablations

These ablations compare the representations given to the actor and predicted by the model. Categorical (simple) flattens the encoder output, projects it to $K \times L$ logits, and converts these to one-hot variables with straight-through gradients (Bengio et al., 2013). It removes both tokenization and the learned embedding projection. Categorical (+softmoe) first processes the CNN tokens with `SoftMoE` (Sokar

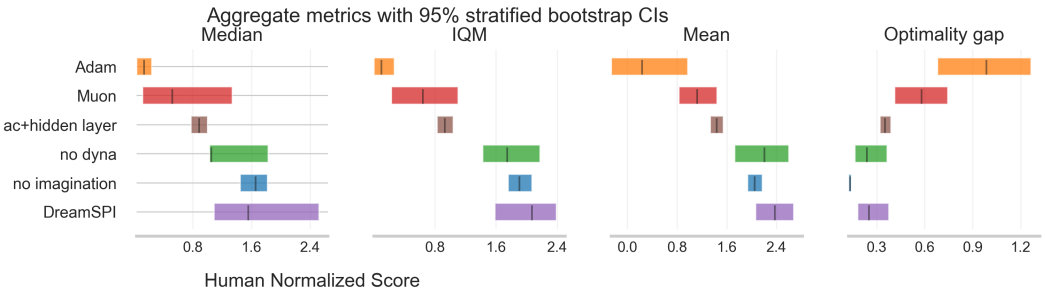


Figure 23: Final aggregate metrics for mechanism and optimizer ablations on Atari-5.

et al., 2025), then projects them to categorical logits and one-hots. Both variants give the actor one-hot variables.

Hadamard/hyper-tanh embeddings (Kooi et al., 2024) retain tokens but bound the encoder and codebook outputs through a hyperbolic-tangent product geometry, limiting their diameter. Simplicial embeddings (Obando-Ceron et al., 2026) normalize coordinates onto a simplex, giving bounded, nonnegative features that sum to one.

The `DeepSPI`-style variant Delgrange et al. (2025) flattens the CNN output into a continuous representation for the actor and critic. It uses Lipschitz-constrained reward and transition heads and predicts transitions with a mixture of diagonal Gaussians. The results favor giving the actor continuous token embeddings while restricting the transition model's predictions to the finite codebook.

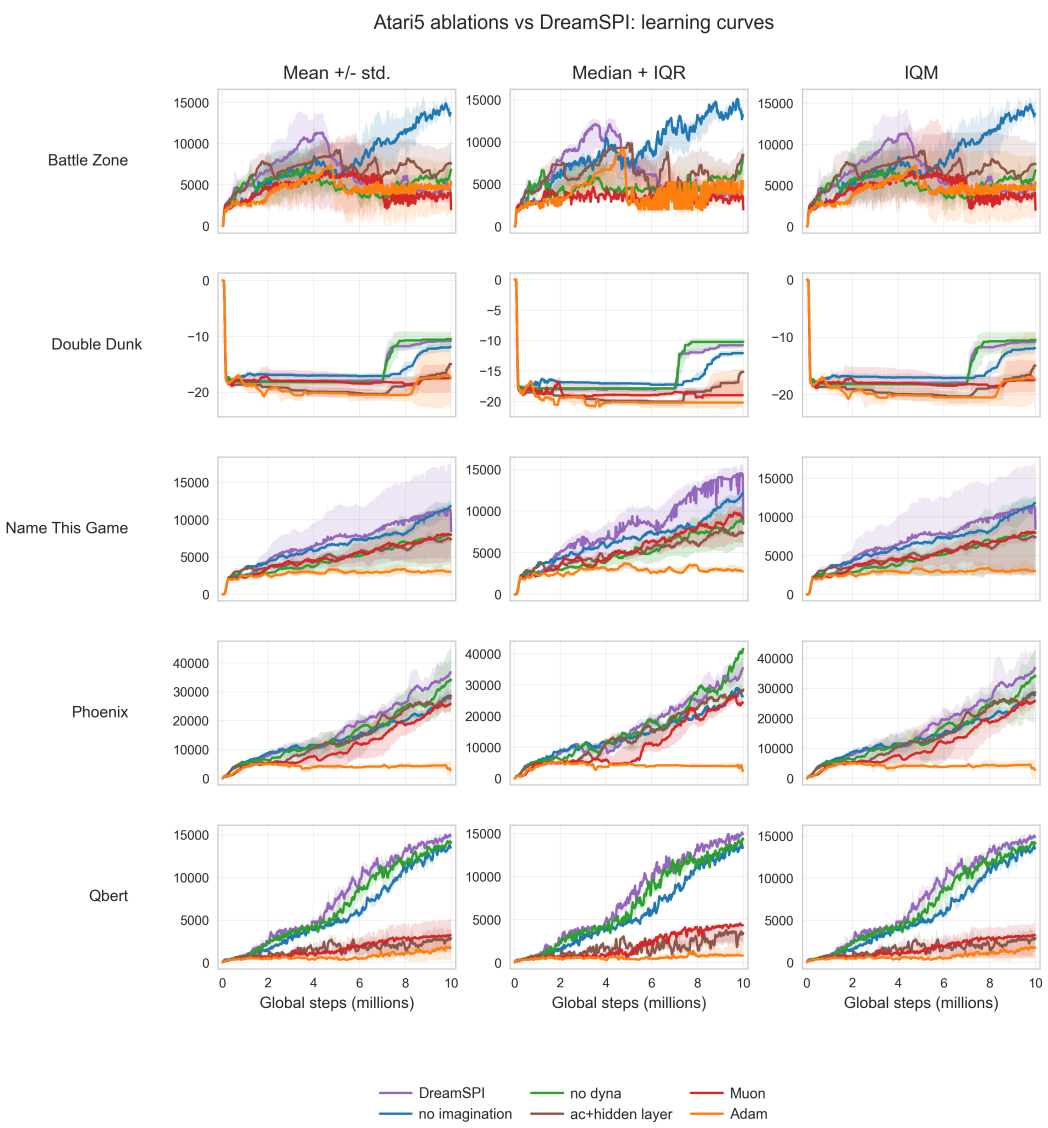


Figure 24: Full-width aggregate learning curves for mechanism and optimizer ablations on Atari-5.

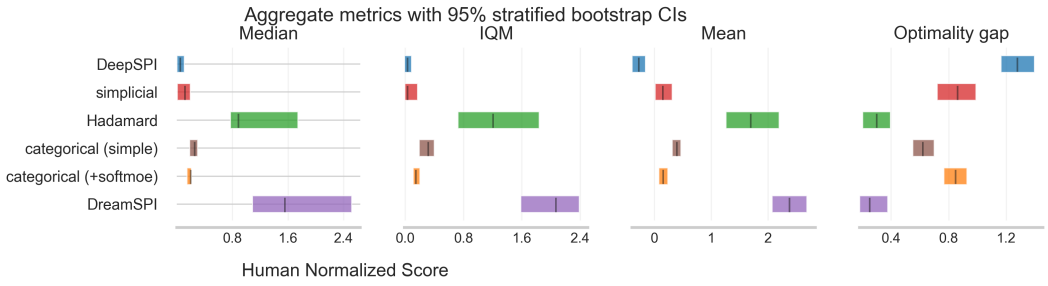


Figure 25: Final aggregate metrics for representation ablations on Atari-5. Continuous token embeddings with a finite prediction codebook outperform the tested categorical (simple), categorical (+softmoe), Hadamard, simplicial, and `DeepSPI`-style alternatives.

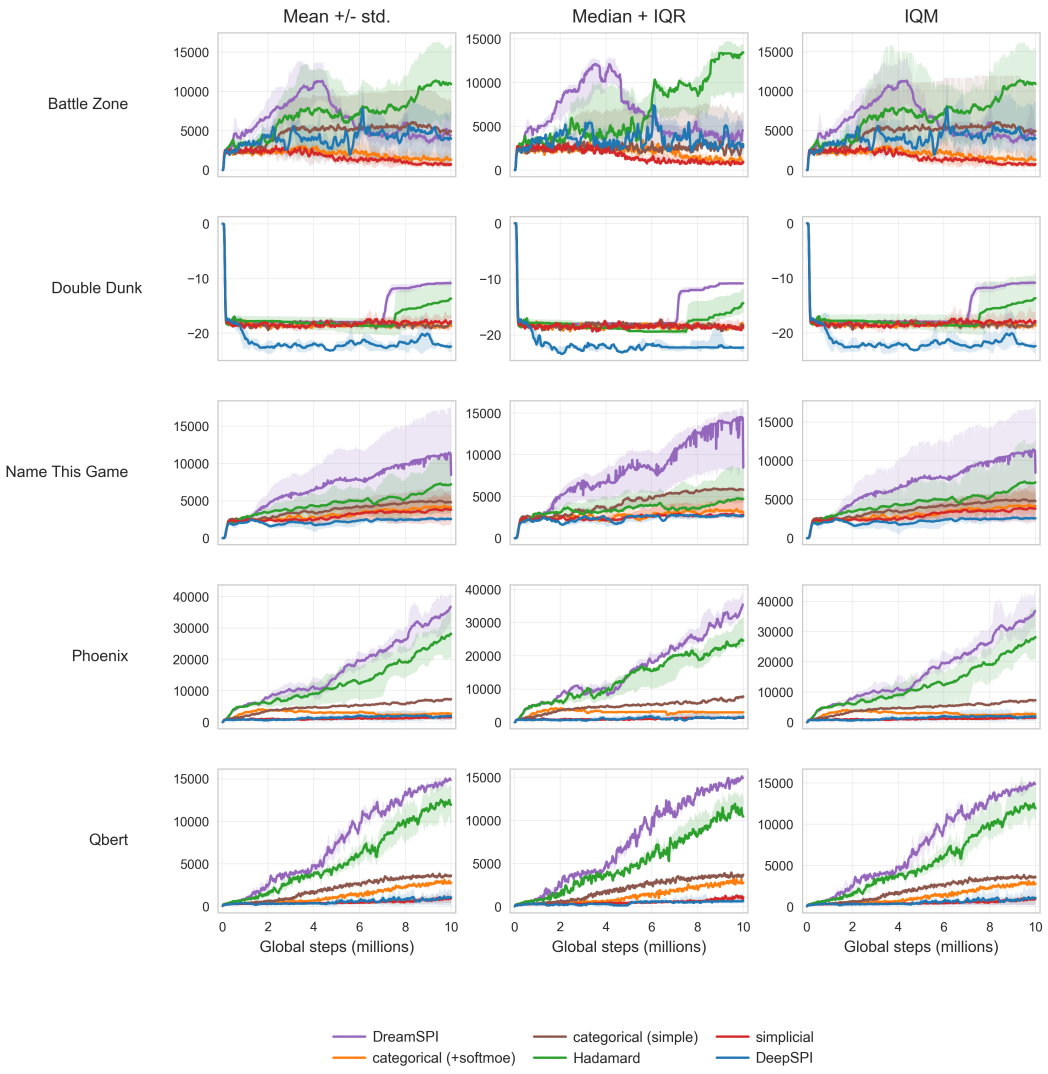


Figure 26: Full-width aggregate learning curves for representation-interface ablations on Atari-5.

M Auxiliary Targets and Regularity Ablations

M.1 Reconstruction, target encoders, and multi-step prediction

Reconstruction, target networks, and self-predictive losses are natural candidates for `DreamSPI` because the algorithm already learns a next-state predictor. The reconstruction and encoder-target ablations are reported in the main text (Fig. 8a and Fig. 8b); we give the interpretation here and the per-environment curves in Figs. 28–29.

The reconstruction ablation uses a `DreamerV3`-style symlog image target (Hafner et al., 2025). Symlog compresses large values and is nearly linear around zero, reducing the influence of large magnitudes in a squared loss. A deconvolutional head D_ψ reconstructs the most recent frame from $z_t = \phi(o_t)$. We normalize the frame to $[0, 1]$ and apply $\mathrm{symlog}(x) = \mathrm{sign}(x)\log(1 + |x|)$. With $\ell_2(x, y) = \frac{1}{2}(x - y)^2$, the per-sample loss is

$$\mathcal{L}_{\mathrm{rec}}(z_t, o_t) = \sum_{h,w} \frac{1}{C} \sum_{c=1}^{C} \ell_2\big(D_\psi(z_t)_{c,h,w}, \mathrm{symlog}\big(o^{\mathrm{last}}_{t,c,h,w}/255\big)\big)\,,$$

then averaged over the batch and weighted by the reconstruction coefficient. The ablation adds observation reconstruction to latent prediction. In generative world-model agents, reconstruction helps learn model states for imagination. The SPI bound instead depends on reward and transition errors in the policy's representation. The performance drop in Fig. 8a suggests that reconstruction can favor visual details that matter little for control. Figure 28 shows the per-game effects: reconstruction helps on *BattleZone*, removes the main `DreamSPI` advantage on *NameThisGame*, *Qbert*, and *DoubleDunk*, and finishes below on *Phoenix*. Its benefit for control thus varies across games.

Stop-gradient and EMA target encoders address changing prediction targets. In JEPA-style one-step prediction (Assran et al., 2023; Bardes et al., 2024), stop-gradient blocks differentiation through the target, while EMA uses a slowly updated target encoder. Our EMA variants (Figs. 8b and 29) use $\bar{\vartheta} \leftarrow \tau\bar{\vartheta} + (1-\tau)\vartheta$ with $\tau = 0.985$. Before the Atari-5 ablation, we tested $\tau \in [0.9, 0.995]$. The better runs clustered around 0.985–0.99; we chose 0.985 to track the online encoder most closely within that range.[5] In `DreamSPI`, EMA targets use an older representation than the one used by the actor. Stop-gradient with the current encoder leaves the numerical prediction error unchanged, but prevents this error from updating the encoder through the next-state target. Figure 8b shows no gain from these targets. Both variants perform worse on all five Atari-5 games, with particularly large losses on *DoubleDunk* and *Qbert* (Fig. 29). The SPI bound uses prediction errors computed with the actor's encoder. Our ablations support retaining gradients through both predictions and next-state targets.

`SPR`-style losses (Schwarzer et al., 2021) train multi-step latent predictions to match encoded future observations, using stop-gradient or EMA targets. This consistency objective supplies temporal supervision beyond the next frame without reconstructing pixels and can improve sample efficiency. We test a five-step `SPR`-style target in the same tokenized latent space. The loss masks terminal crossings and averages valid one- through five-step prediction pairs. Note that `DreamSPI` in Fig. 27 corresponds to `SPR` with horizon 1 and no stop-gradient or EMA. On nonterminal pairs, this is the standard transition target: apply the categorical model once to $z_t = \phi(o_t)$ and match $\phi(o_{t+1})$.

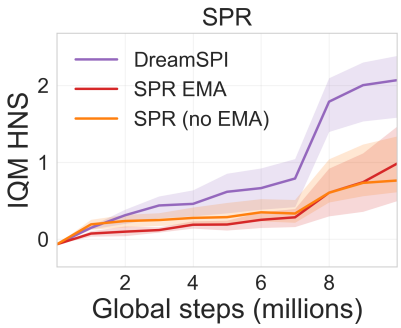


Figure 27: Atari-5 IQM-HNS sample efficiency for SPR-style auxiliary targets.

Figure 27 shows no gain from multi-step prediction. The variants improve only *BattleZone* and perform substantially worse on *NameThisGame*, *Phoenix*, *Qbert*, and *DoubleDunk*; adding EMA further reduces performance. The auxiliary loss changes the model in which the actor and critic train: it adds longer-horizon representation consistency to the one-step errors used by SPI. It may also use capacity to predict factors unrelated to return. These results favor the current one-step transition target paired with reward prediction and local policy updates. Multi-step targets may help if aligned with the same local errors, but the auxiliary loss tested here does not.

[5] Off-policy agents with replay often use $\tau = 0.995$. Our on-policy setting uses fewer updates per environment step, so we tune τ for this update frequency.

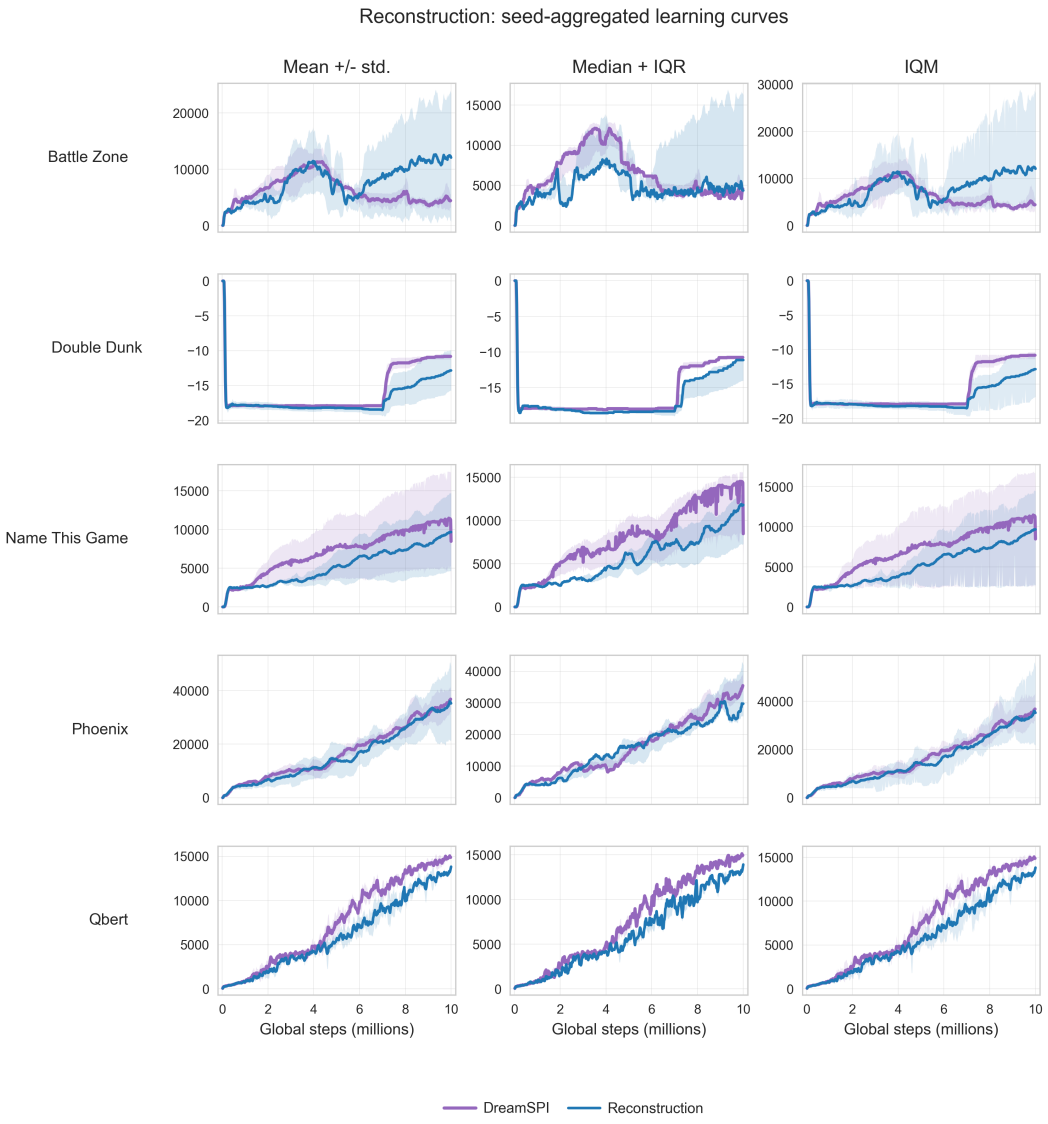


Figure 28: Per-game learning curves, aggregated across seeds, for the Atari-5 reconstruction ablation. Reconstruction improves return on *BattleZone* but reduces it on the other four games.

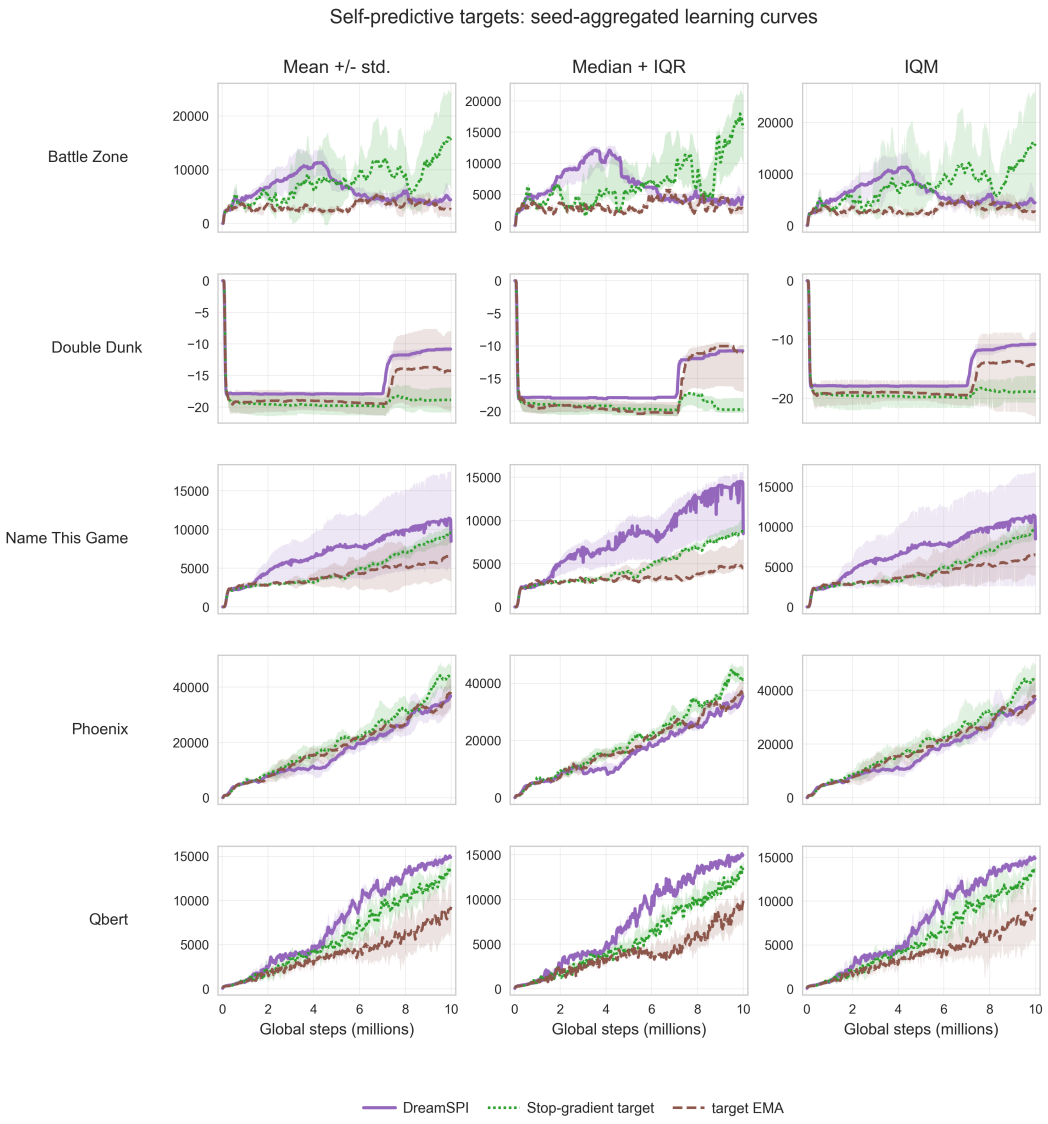


Figure 29: Per-game learning curves, aggregated across seeds, for stop-gradient and EMA targets on Atari-5.

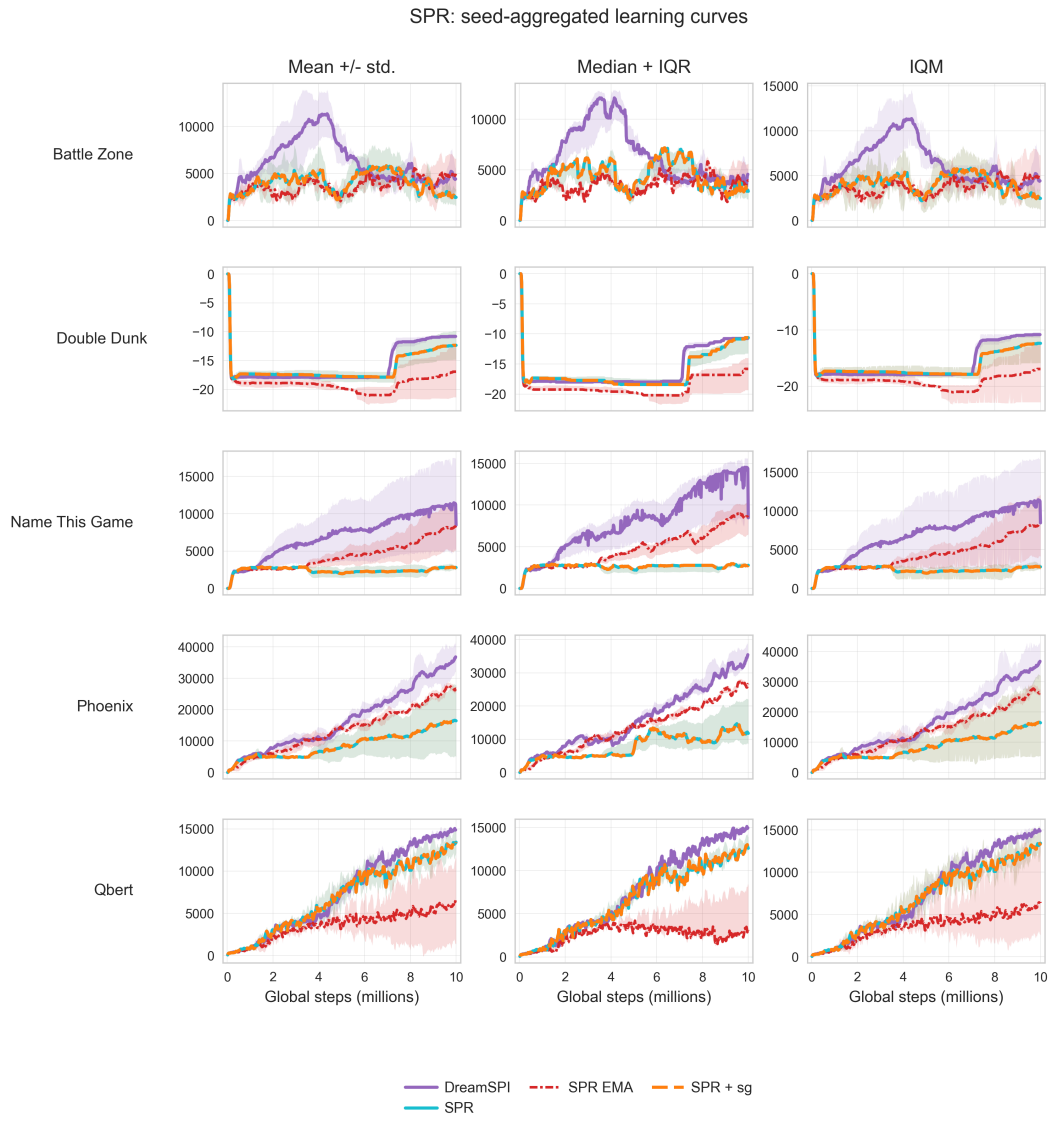


Figure 30: Per-environment seed-aggregated learning curves for `SPR`-style multi-step prediction losses on Atari-5. The multi-step self-predictive targets help *BattleZone* but reduce performance on the other Atari-5 games, so this auxiliary objective does not improve control on most tested games.

M.2 Lipschitz Transition and Reward Networks

The deep SPI bound depends on how much reward and transition predictions change between nearby latent states. A Lipschitz transition head would bound this change for each action, including between an encoder output and its nearest codebook point (Appendix J). We test whether explicitly constraining these networks improves learning. The variants use spectral or `Björck` normalization with GroupSort, MaxMin, or Mish activations, following Anil et al. (2019). We also test `WGAN`-style gradient penalties (Gulrajani et al., 2017) and direct gradient penalties as in `DeepMDP` (Gelada et al., 2019).

In Fig. 31, `DreamSPI` denotes the selected token-embedding/codebook model without these additional constraints. *GP* and `WGAN-GP` add direct or interpolation-based gradient penalties, respectively, with the coefficient in the label: $\lambda_{\text{GP}} \in \{0.01, 0.1, 1, 10\}$. *Spectral* and `Bjorck` replace the reward and transition heads with spectrally or `Björck`-normalized networks. For each, *1L* and *2L* denote

one or two 512-unit hidden layers; *GSort*, *MaxMin*, and *Mish* specify the activation. We test all six depth–activation combinations for each normalization.

The best regularized variant, a direct gradient penalty with weight 1, reaches Atari-5 IQM 1.658, below `DreamSPI`'s 2.068. The best `WGAN-GP` and spectral variants reach 1.512 and 1.462; `Björck` variants score lower. None of these additions improves on the selected codebook model and optimizer.

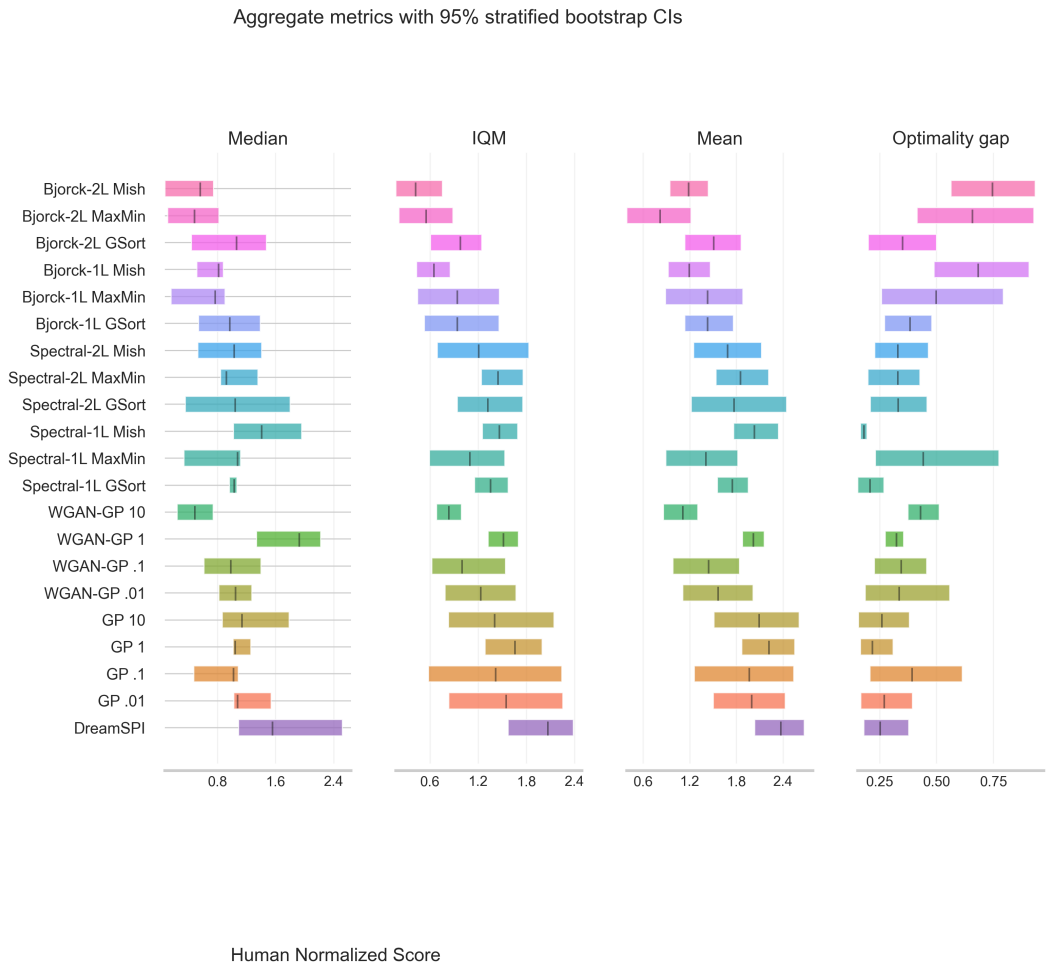


Figure 31: Final aggregate metrics for Lipschitz-network and gradient-penalty ablations on Atari-5. None of the tested regularizers improves on the selected `DreamSPI` model.

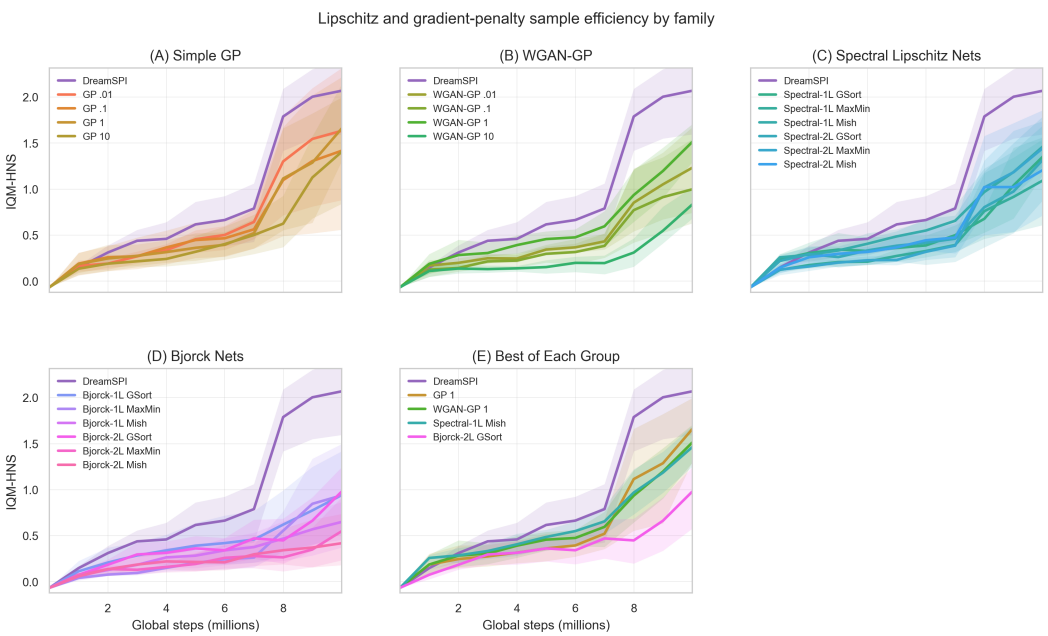


Figure 32: IQM-HNS sample efficiency for Lipschitz-network and gradient-penalty ablations on Atari-5. Each panel compares `DreamSPI` with one family of regularizers: simple gradient penalties, `WGAN-GP` penalties, spectral Lipschitz heads, `Björck` heads, and the best variant from each family.

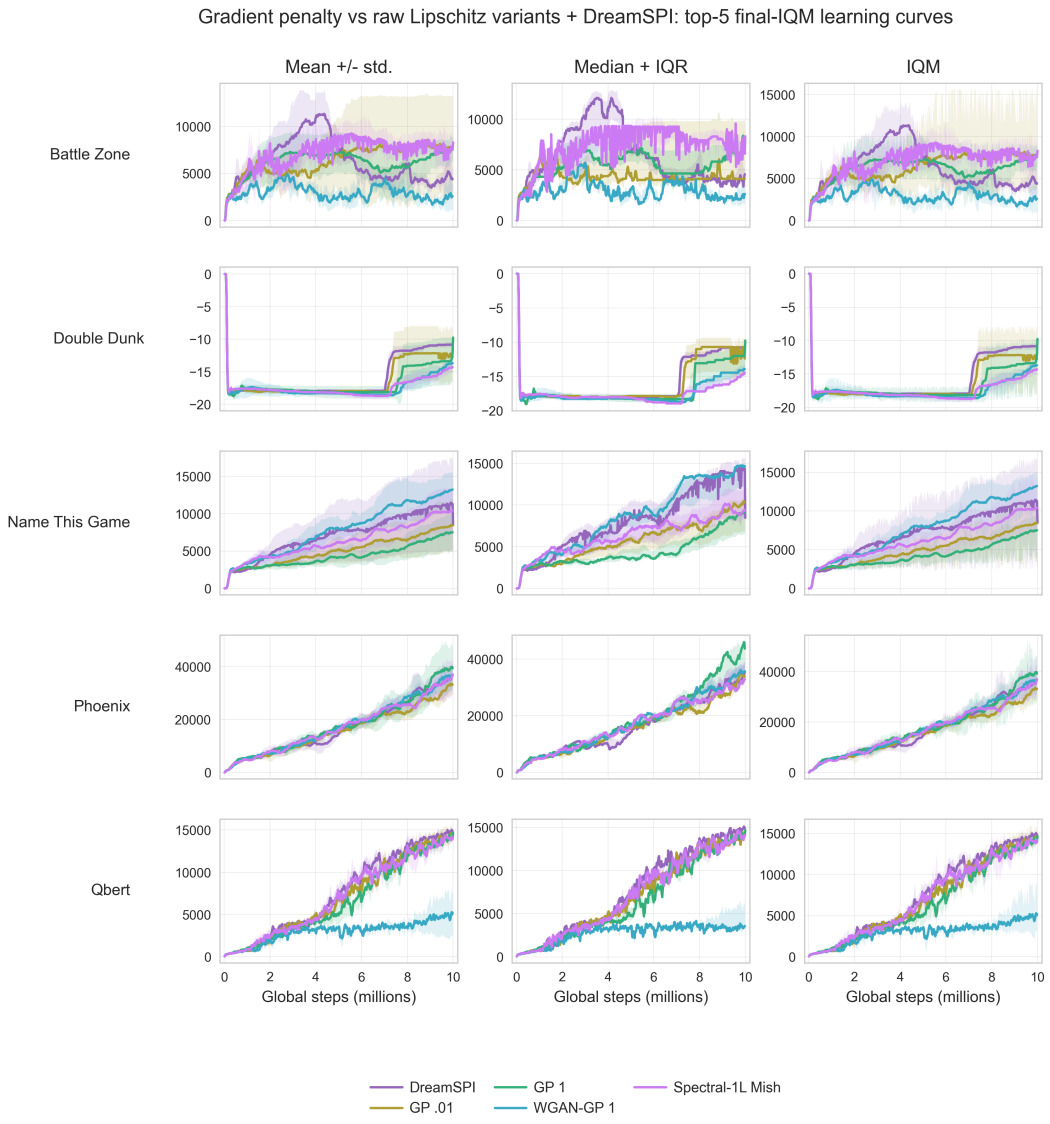


Figure 33: Aggregate learning curves for the top Lipschitz and gradient-penalty variants.

Eq. (6) bounds how much predicted rewards and next-state distributions change as the latent state changes. Spectral and `Björck` normalization constrain the network weights to limit changes in its outputs. Gradient penalties instead penalize large derivatives at sampled states or points between them.

To connect these network constraints to SPI, we must relate changes in output scores (logits) to changes in next-state distributions. The proposition below shows that a Lipschitz bound on the logits gives a Wasserstein bound on the predicted distribution when the codebook is fixed.

Proposition 1. *Let $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ be fixed with diameter $\Delta_X = \max_{i,j} \|x_i - x_j\|_2$, and let $(\mathcal{S}, d_{\mathcal{S}})$ be a metric space. Suppose $f : \mathcal{S} \to \mathbb{R}^n$ is L-Lipschitz with respect to the Euclidean norm. For*

$$p(s) = \mathrm{softmax}(f(s)), \qquad D_s = \sum_{i=1}^{n} p_i(s)\delta_{x_i},$$

the map $s \mapsto D_s$ is Lipschitz from $\mathcal{S}$ to probability measures over X equipped with 1-Wasserstein distance:

$$W_1(D_s, D_t) \le \frac{\Delta_X}{2} L\, d_{\mathcal{S}}(s, t).$$

Proof. For probability vectors $p, q \in \Delta^{n-1}$ supported on the same finite set X, the amount of mass that must be moved is at most $\frac{1}{2}\|p-q\|_1$, and every unit of mass moves distance at most Δ_X. Therefore

$$W_1\left(\sum_i p_i\delta_{x_i}, \sum_i q_i\delta_{x_i}\right) \le \frac{\Delta_X}{2}\|p-q\|_1 .$$

It remains to bound the softmax probabilities. Let σ denote softmax and let $a, b \in \mathbb{R}^n$. For $h = a - b$ and $z(\tau) = b + \tau h$,

$$\sigma(a) - \sigma(b) = \int_0^1 J_\sigma(z(\tau))h\, d\tau.$$

Writing $p = \sigma(z)$, the ith component of $J_\sigma(z)h$ is $p_i(h_i - \sum_j p_j h_j)$. Hence $\|J_\sigma(z)h\|_1$ is the mean absolute deviation of the values h_i under p. By Cauchy–Schwarz it is at most $\sqrt{\mathrm{Var}_p(h_i)} \le \sqrt{\mathbb{E}_p h_i^2} \le \|h\|_\infty$. Thus $\|\sigma(a)-\sigma(b)\|_1 \le \|a-b\|_\infty \le \|a-b\|_2$. Applying this with $a = f(s)$ and $b = f(t)$ yields $\|p(s)-p(t)\|_1 \le L d_{\mathcal{S}}(s,t)$, which gives the result. □

We now apply the proposition to `DreamSPI` to bound the constants in Eq. (6). We first combine its K independent categorical predictions, then average over the actor's actions. The actor matters because different action probabilities can change predicted rewards and next-state distributions, even if the model's predictions for each action stay the same.

For a fixed action, suppose the ith logit vector is L_i-Lipschitz in z. The proof above bounds the total variation between its distributions at z and z' by $L_i\|z - z'\|/2$. Independently couple each variable's two outcomes so that their mismatch probability equals their total variation. The full codes differ only if at least one variable differs, so a union bound limits the total variation of the product distributions by the sum of the individual bounds. Decoding onto a fixed codebook of diameter D gives

$$W_1\big(\bar{P}(\cdot \mid z, a), \bar{P}(\cdot \mid z', a)\big) \le \frac{D}{2}\left(\sum_{i=1}^K L_i\right)\|z - z'\| .$$

The policy-induced kernel averages over actions, so its regularity also depends on how the actor's action probabilities change. If K_P and K_R bound the transition and reward Lipschitz constants uniformly over actions, $|\bar{R}(z,a)| \le \bar{R}_{\max}$, and $\mathrm{TV}(\bar{\pi}(\cdot \mid z), \bar{\pi}(\cdot \mid z')) \le L_\pi\|z - z'\|$, then separating the change in model predictions from the change in action probabilities gives

$$K_{\bar{P}}^{\bar{\pi}} \le K_P + DL_\pi, \qquad K_{\bar{R}}^{\bar{\pi}} \le K_R + 2\bar{R}_{\max}L_\pi.$$

For our softmax readout, the same probability bound gives $L_\pi \le \|W_\pi\|_{\mathrm{op}}/2$, with the operator norm from Theorem 3. The policy-induced constants therefore account for changes in both model predictions and action probabilities. These results hold with the codebook fixed. Comparing different codebooks also requires accounting for how far their embeddings move; separation is not needed for this fixed-support proposition. The spectral/`Björck` heads and gradient penalties tested here do not improve on the selected architecture.

N Empirical SPI Diagnostics

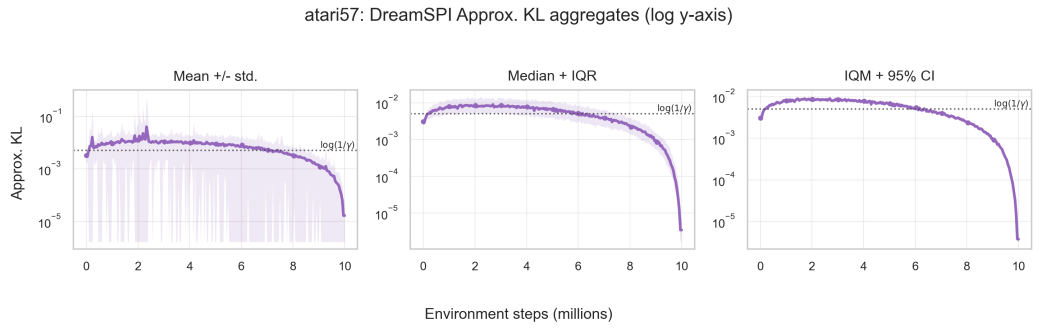
The SPI bound depends on model prediction errors and the size of policy changes. We therefore track two training diagnostics for `DreamSPI` on ALE and Octax: approximate KL measures average changes in action probabilities, while reward and transition losses show how well the model fits the current policy's data.

N.1 Policy-Locality Proxy

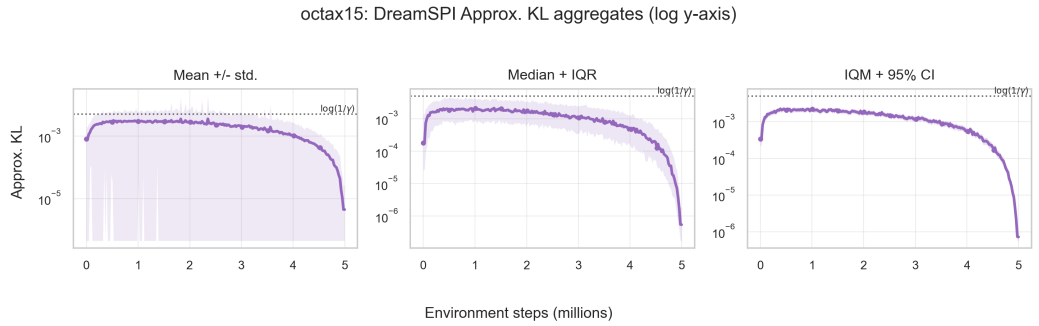
`DreamSPI` uses clipped actor updates as a practical proxy for the pointwise neighborhood in Eq. (2). Figure 34 reports the corresponding logged approximate KL on real environment updates, computed as the sample mean of the nonnegative quantity $(\rho-1)-\log\rho$. For fixed policies and unclipped ratios, its expectation under π_{old} equals $D_{\mathrm{KL}}(\pi_{\mathrm{old}}\|\pi_{\mathrm{new}})$, with $\rho = \pi_{\mathrm{new}}(a \mid s)/\pi_{\mathrm{old}}(a \mid s)$ (Schulman, 2020). The dotted line marks $\log(1/\gamma)$ for $\gamma = 0.995$ as a visual reference; it is not a KL threshold from the SPI theorem. At any state, ratios in $[1-\epsilon, 1+\epsilon]$ imply $D_{\mathrm{KL}}(\pi_{\mathrm{old}}\|\pi_{\mathrm{new}}) \le -\frac{1}{2}\log(1-\epsilon^2)$,

by convexity of $-\log\rho$ and $\mathbb{E}_{\pi_{\text{old}}}\rho = 1$. The condition $\epsilon < 1/\gamma - 1 \simeq 5.0 \cdot 10^{-3}$ keeps the return-bound denominator positive for every policy in the neighborhood. Recall that our configuration uses a clipping radius of 0.1 for joint real-data updates and 0.01 for imagination, with reference-penalty coefficient $\beta_{\text{KL}} = 1.02 \cdot 10^{-4}$ (Appendix F). Clipping and the reference penalty discourage policy changes, but even a smaller clipping radius would not enforce the neighborhood at every state and action.

For Octax, about 90% of logged approximate KL values are below $\log(1/\gamma)$, and the median and IQM curves remain below the reference for most of training. ALE's median and IQM approximate KL rise above the reference during the first half of training, then fall below it later; by 7.5M environment steps, roughly three quarters of the selected ALE points are below the reference, and the final logged updates are essentially at zero. Note that these curves measure average policy changes during training and do not show whether the policy satisfies the SPI ratio bounds at every state and action, or isolate the effect of allowing larger ratios.



(a) ALE-57.



(b) Octax-15.

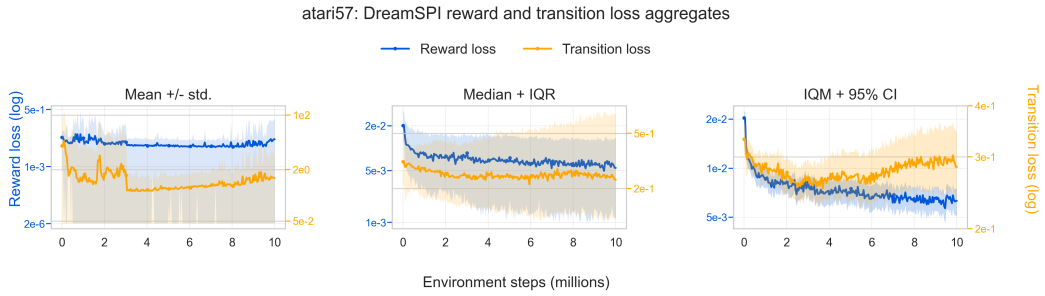
Figure 34: Approximate-KL locality diagnostics for selected `DreamSPI` runs. Shaded bands report standard deviation, IQR, or 95% bootstrap confidence intervals depending on the panel. The dotted reference is $\log(1/\gamma)$ with $\gamma = 0.995$. Lower values indicate smaller average movement from the behavior policy used for the update.

N.2 Local Reward and Transition Losses

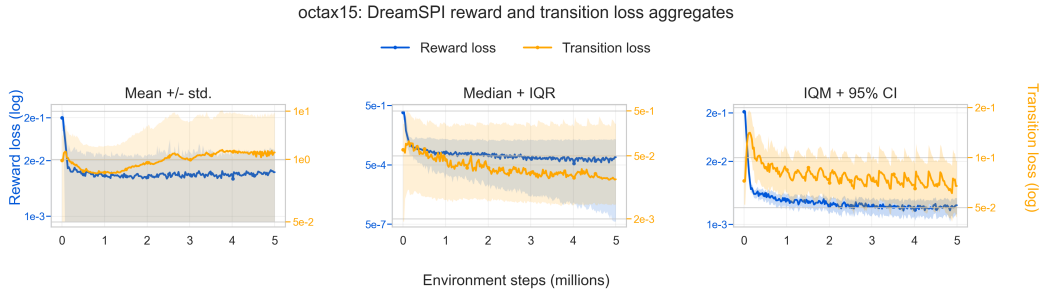
The transfer penalty ζ uses the absolute reward error and Wasserstein transition error in Eq. (7). Figure 35 shows the smooth-L1 training surrogates; the logged reward term also includes termination prediction. These curves describe model fitting, but their values cannot be substituted directly into ζ. The blue reward curve and orange transition curve use separate log axes because their scales differ.

On Octax, both losses fall quickly after the initial model fit, and their IQMs decrease through most of training. The final median reward and transition losses are about 9.9×10^{-4} and 1.5×10^{-2}, respectively. Model losses decrease on the current policy's data while approximate KL remains low. On ALE, the median and IQM reward losses settle around 10^{-3}–10^{-2}. Transition loss falls early, then its IQM rises slightly late in training.

Gelada et al. (2019) show that reward and transition losses can conflict in `DeepMDP`. An encoder can reduce transition loss by making different states look alike. Reward prediction instead requires retaining differences that affect rewards. Their harder settings needed additional supervision to balance these losses. Delgrange et al. (2025) did not observe this conflict in `DeepSPI`, which learns from the current policy's data collected in parallel environments rather than replay. `DreamSPI` appears closer to `DeepSPI`: transition loss does not approach zero as in the `DeepMDP` collapse case, reward loss stays low, and both Octax losses decrease. We interpret the slight late increase on ALE as a shift in the balance between the two losses under more difficult dynamics. The low reward loss and final returns suggest that the model remains useful for control.



(a) ALE-57.



(b) Octax-15.

Figure 35: Local model-error diagnostics for selected `DreamSPI` runs. Blue curves report the reward loss on the left log axis; orange curves report the transition loss on the right log axis.

O Grouped Environment Diagnostics

The ALE aggregate does not show which games benefit from `DreamSPI`. Following the benchmark-level caution of Castro (2025), we examine per-game curves to ask where local model-based updates help.

After each real-data update, `DreamSPI` generates short rollouts and updates only the actor–critic readout, keeping the encoder and model fixed. We expect this to help when games repeatedly present short-horizon action consequences and enough reward feedback to identify useful features. It may help less with exploration, memory, route planning, or precise timing.

We group games using their mechanics, learning curves, and final per-game IQM gaps against the strongest baseline. These groups help interpret the ALE results. Wins count games where `DreamSPI` has the highest final per-game IQM. The reported IQM, median, and mean summarize per-game HNS gaps. Figure 36 shows learning curves for the same groups.

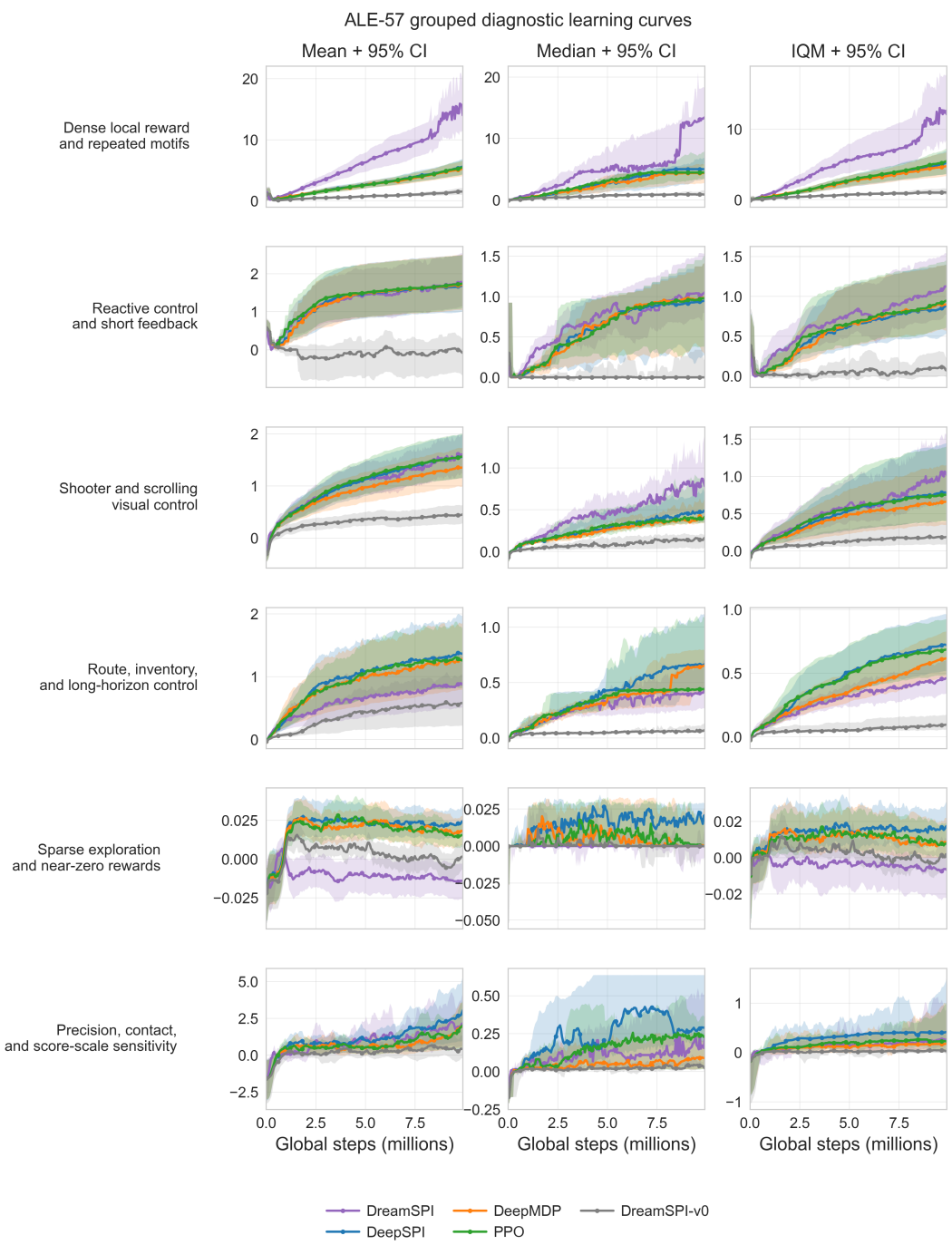


Figure 36: Learning curves for the ALE game groups in Appendix O, covering all 57 games. Each row shows one group; columns give the mean, median, and IQM. All curves include 95% bootstrap confidence intervals.

Frequent rewards. *Games: Atlantis*, *DemonAttack*, *UpNDown*, *Breakout*, *Phoenix*, *DoubleDunk*, and *Gopher*. *Metrics:* wins $7/7$; IQM $+7.18$, median $+3.72$, mean $+8.41$. `DreamSPI`'s median HNS rises from 0.77 at 1M steps to 4.68 at 5M and 13.50 at 10M. Frequent rewards show which actions help. Repeated action–outcome patterns may also let the actor reuse its model in short imagined rollouts before the policy changes.

Reactive control with quick feedback. *Games: IceHockey*, *Enduro*, *Boxing*, *FishingDerby*, *Tennis*, *Robotank*, *Freeway*, and *Pong*. *Metrics:* wins $5/8$; IQM $+0.04$, median $+0.04$, mean $+0.03$. Median HNS rises from 0.22 to 0.83 to 0.96 at 1M, 5M, and 10M steps. `DreamSPI` gains on *IceHockey*, *Enduro*, and *Boxing*, but nearly ties or loses slightly on *Robotank*, *Freeway*, and *Pong*. Quick reward feedback already helps the model-free baselines learn good policies, which may explain the modest gains from imagination.

Shooters and scrolling games. *Games: Asterix*, *ChopperCommand*, *WizardOfWor*, *BeamRider*, *YarsRevenge*, *Assault*, *StarGunner*, *KungFuMaster*, *Krull*, *Alien*, *Defender*, *Kangaroo*, *Seaquest*, *SpaceInvaders*, *TimePilot*, *Zaxxon*, and *CrazyClimber*. *Metrics:* wins $9/17$; IQM $+0.03$, median $+0.01$, mean -0.10. Median HNS rises from 0.12 to 0.41 to 0.92. `DreamSPI` gains on *Asterix*, *ChopperCommand*, *WizardOfWor*, *BeamRider*, and *YarsRevenge*, but loses on *Assault*, *StarGunner*, *KungFuMaster*, and *Krull*. Short rollouts may miss details needed to choose targets, time actions, and track rapid changes in the scene.

Route planning, resources, and long action sequences. *Games: Amidar*, *BankHeist*, *Berzerk*, *Frostbite*, *Hero*, *Jamesbond*, *MsPacman*, *Qbert*, *Riverraid*, *RoadRunner*, and *Tutankham*. *Metrics:* wins $2/11$; IQM -0.26, median -0.07, mean -0.48. Median HNS reaches 0.48 at 10M steps. `DreamSPI` is close to the best baseline on *Riverraid* and *Hero*, but falls behind on *RoadRunner*, *Jamesbond*, *BankHeist*, and *Tutankham*. Decisions here can depend on maps and resource use, with consequences that only appear after long action sequences. Short imagined rollouts may end before those consequences become clear.

Sparse rewards and exploration. *Games: Adventure*, *MontezumaRevenge*, *PrivateEye*, *Pitfall*, *Venture*, *Gravitar*, and *Solaris*. *Metrics:* wins $1/7$; IQM -0.03, median -0.04, mean -0.03. Median HNS stays near 0.00 throughout training; *Adventure* and *MontezumaRevenge* are zero-score ties. `DreamSPI` adds model-based policy updates, but no directed exploration, intrinsic rewards, search over remembered states, or curriculum of starting states.

Precise timing and control. *Games: VideoPinball*, *Skiing*, *BattleZone*, *Centipede*, *NameThisGame*, *Bowling*, and *Asteroids*. *Metrics:* wins $4/7$; IQM -0.26, median $+0.03$, mean -0.88. Results on *Centipede*, *NameThisGame*, *Bowling*, and *Asteroids* are wins or near ties. Larger losses on *VideoPinball*, *Skiing*, and *BattleZone* pull the mean below zero even though the median gap is positive. These games demand accurate timing and estimates of position, speed, and direction. Even an otherwise plausible prediction can lead to the wrong action: a pinball hits a different target, a skier misses a gate, or a shot misses its mark.

The largest gains come from games with frequent rewards. `DreamSPI`'s final IQM exceeds the best baseline by more than 18 HNS on both *Atlantis* and *DemonAttack*. It also improves on *UpNDown*, *Breakout*, *Phoenix*, *DoubleDunk*, and *Gopher*.

Reward prediction encourages the encoder to distinguish states with different immediate rewards. Predicting transitions between these states lets imagined rollouts show how actions lead to rewards. These local losses (Eq. (7)) appear in both the bound on abstraction quality (Eq. (8)) and the SPI penalty ζ (Eq. (9)). Without reward feedback, accurately predicting latent states need not help the actor choose useful actions.

The weaker groups suggest two limits. In sparse-reward games, the agent must first discover rewarding states; otherwise, the reward loss gives little guidance on which predictions matter for control. Exploration driven by prediction errors (Pathak et al., 2017) could help find these states. Precision-control games pose a different problem. In *VideoPinball* and *Skiing*, success depends on contact timing, velocity, heading, score scale, or delayed physical effects. Four stacked frames may omit information needed to predict these effects. The MDP assumption can then fail in our implementation, as discussed by Castro (2025): the theory assumes a latent MDP with bounded model error, while the observations may be only partially informative. Action selection may then require histories or belief states (Kaelbling et al., 1998).

These results suggest that short imagined rollouts are most useful when real trajectories provide enough reward feedback to learn which actions help. Frequent rewards alone do not ensure large gains, however: several reactive-control games improve only modestly. In sparse-reward games, the

agent must still discover rewarding states. Route planning may need longer rollouts, while precise control may require information missing from the frame stack. This post-hoc analysis motivates controlled tests of directed exploration, encoders with memory, and longer rollouts. Measuring model errors alongside return would help determine which changes improve prediction and which improve the policy.