SARC: INTERPRETABLE STATE-ADAPTIVE REWARD COMPOSITION FOR SPARSE REWARD REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse-reward reinforcement learning suffers from severe credit assignment challenges, as agents often receive no informative feedback within an episode. Reward decomposition and shaping methods alleviate this problem but often introduce redundant information or reward hacking. Recent latent-reward approaches provide structured, multi-dimensional signals, but their composition does not necessarily expose how individual factors influence policy behavior across states. We propose State-Adaptive Reward Composition (SARC), which composes the action reward as a linear combination of the environment reward and multi-dimensional latent rewards. The composition weights are produced by a state-dependent intention function trained through bilevel optimization: the upper level optimizes the intention function to maximize environment return, while the lower level optimizes the policy to maximize the composed return. The learned intention weights explicitly and interpretably reveal the agent's task priorities at each state. We further introduce stabilization techniques for this bilevel training. Across eight sparse-reward continuous-control tasks, SARC achieves strong performance against competitive baselines. Intervention experiments show that the learned weights causally steer behavior, while ablations characterize the contributions of the main components.

1 INTRODUCTION

Sparse reward reinforcement learning (RL) refers to a learning paradigm where agents receive nonzero feedback only upon achieving goals or triggering critical events. This paradigm finds extensive applications in competitive games (Vinyals et al., 2019; Silver et al., 2017), robotic control (Andrychowicz et al., 2020), and large language model (LLM) alignment (Ouyang et al., 2022; Guo et al., 2025). However, the sparse reward structure poses severe credit assignment challenges (Sutton & Barto, 1998), as agents may receive no effective feedback signal throughout an entire episode, leading to high variance in value function estimation and slow convergence (Arjona-Medina et al., 2019).

Reward shaping and return decomposition are common approaches to address the sparse reward problem (Yu et al., 2025). Reward shaping methods guide agent learning by augmenting environmental rewards with auxiliary signals. For instance, intrinsic motivation-based methods (Bellemare et al., 2016; Pathak et al., 2017; Burda et al., 2018) and behavioral metric-based approaches (Wang et al., 2023; 2025) generate auxiliary reward signals that effectively improve exploration. Nevertheless, auxiliary rewards can sometimes incentivize agents to maximize the bonus rather than pursue the task objective (Villalobos-Arias et al., 2025). Return decomposition methods (Ren et al., 2021; Lin et al., 2024) redistribute episodic rewards across timesteps, but are vulnerable to redundant information and local optima (Qu et al., 2025). Moreover, these methods still underperform reward functions designed with domain knowledge. Analysis in Go-Explore (Ecoffet et al., 2021) and DeepSeek-R1 (Guo et al., 2025) confirms that domain knowledge remains crucial for complex RL tasks.

LLMs encode rich domain knowledge through pretraining on massive corpora (Brown et al., 2020), establishing themselves as powerful tools for reward function design (Yu et al., 2025; Ma et al., 2023; Xie et al., 2024). However, designing precise reward functions requires fine-grained specification

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.

of numerical relationships, which remains challenging for current LLMs. LaRe (Qu et al., 2025) offers a paradigm better aligned with LLM capabilities. Instead of directly designing complete reward functions, it generates latent reward functions that evaluate the task from multiple semantic perspectives such as progress, efficiency, and stability. While static weight combination of latent rewards is a common reward shaping practice in the RL community, it requires extensive tuning and lacks adaptability across states. LaRe addresses this by using neural networks to learn the reward composition, achieving promising results in credit assignment. However, its learned decoder does not explicitly expose the contribution of each latent factor to policy behavior at a given state. The inherent interpretability of latent rewards has not been fully exploited to guide policy learning (Milani et al., 2024).

Inspired by the latent reward paradigm, we propose State-Adaptive Reward Composition (SARC), which models action rewards as a linear combination of environment rewards and multi-dimensional latent rewards, with composition weights generated by a state-dependent intention function. Our key insight is that the optimal intention weights for different latent reward dimensions should vary with state. For instance, agents far from the goal should prioritize progress, while those approaching the goal should emphasize precision and stability. SARC employs a bilevel optimization framework: the upper level optimizes the intention function to maximize environment returns, while the lower level trains policies to maximize composed returns under given intentions. The intention weights at each state explicitly represent the agent's task priorities, enabling interpretable decision-making while supporting effective policy learning under sparse feedback. Since dynamic intention optimization poses additional training challenges, we further introduce several stabilization techniques.

The main contributions of this paper are summarized as follows. First, we propose SARC, which combines component-wise critics with a state-dependent intention function. The same intention weights compose latent values and condition the policy, providing an explicit interface between reward priorities and behavior. We introduce stabilization techniques to address training challenges arising from dynamic intention optimization. Second, the intention weights explicitly represent the agent's task priorities at each state, providing interpretable decision-making with human-understandable semantic explanations. Third, experiments on eight continuous-control tasks demonstrate strong performance against competitive baselines, while ablations characterize the contributions of the stabilization mechanisms and latent reward dimensions.

2 Related Work

Sparse Reward Reinforcement Learning. Sparse rewards pose two central challenges: discovering rewarding behaviors and learning efficiently from limited reward feedback. Intrinsic motivation methods encourage exploration through state novelty (Bellemare et al., 2016), prediction errors (Pathak et al., 2017; Burda et al., 2018), or behavioral metrics (Wang et al., 2023; 2025). Complementarily, hierarchical RL (Vezhnevets et al., 2017) structures exploration through sub-goal decomposition, while curriculum learning (Bengio et al., 2009; Narvekar et al., 2020) organizes training through progressively challenging tasks. For reward utilization, return decomposition methods (Arjona-Medina et al., 2019; Ren et al., 2021; Lin et al., 2024) redistribute episodic rewards across timesteps. Hindsight Experience Replay (Andrychowicz et al., 2017) reuses unsuccessful trajectories through goal relabeling in goal-conditioned tasks, while Prioritized Experience Replay (Schaul et al., 2015) prioritizes transitions using TD errors. These methods primarily address exploration and reward utilization, with less emphasis on coordinating semantically distinct auxiliary rewards. SARC constructs a structured reward representation across multiple semantic dimensions, integrating domain knowledge with environmental feedback through interpretable, state-dependent composition.

LLM- and VLM-Assisted Reward Design. Manual reward design requires domain expertise and can introduce specification errors (Booth et al., 2023; Knox et al., 2023), motivating the use of pretrained models for reward construction (Yu et al., 2025). Existing approaches include generating executable reward code from task descriptions (Ma et al., 2023; Xie et al., 2024; Yu et al., 2023), deriving reward signals from pretrained vision–language representations (Rocamonde et al., 2024; Sontakke et al., 2023), and learning intrinsic rewards from LLM-generated preferences (Klissarov et al., 2023). More closely related, LaRe (Qu et al., 2025) constructs semantically meaningful latent reward components and learns a neural decoder that maps them to scalar reward estimates. SARC

builds on this structured reward representation while making the allocation across factors explicit through state-dependent intention weights.

Bilevel Reward Optimization. Bilevel reward optimization adapts auxiliary rewards according to the task performance of policies trained with them. BiPaRS (Hu et al., 2020) learns adaptive shaping weights, BARFI (Gupta et al., 2023) optimizes combinations of primary and auxiliary rewards, and MORSE (Xie et al., 2025) learns state-dependent mixtures of heuristic rewards with reward-space exploration. Compared with these approaches, SARC learns separate critics for named reward components and uses a shared, simplex-valued intention both to compose their values and to condition the policy, making reward priorities inspectable and allowing direct interventions on the trained policy. SARC implements these updates with TD3, reusing past transitions for policy and intention learning.

3 PRELIMINARY

MDP and Reward Shaping. We consider environments modeled as Markov Decision Processes (MDPs), defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ is the state transition distribution, $r(s,a)$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. The agent aims to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative reward $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, \pi(s_t))\right]$. In sparse reward settings, non-zero feedback is rare and may be entirely absent within an episode, posing severe credit assignment challenges.

Reward shaping addresses this challenge by introducing auxiliary reward signals $\tilde{r} = r + f$ to guide policy learning, but the quality of hand-designed shaping rewards f varies. Hu et al. (2020) proposed parameterized reward shaping, which learns a weight function $z_\upsilon(s,a)$ to control the utilization of the shaping reward:

$$\tilde{r}(s,a) = r(s,a) + z_\upsilon(s,a) f(s,a). \tag{1}$$

The weight function is trained by bilevel optimization: the upper level optimizes z_υ to maximize the cumulative environment reward, while the lower level optimizes the policy π_θ to maximize the composed reward $\tilde{r}$.

Latent Reward. Latent reward decomposes task objectives into multi-dimensional performance metrics, where each dimension captures a distinct aspect of task performance. Formally, a function $g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$ maps state-action pairs to K-dimensional latent rewards $\mathbf{r}_{lr} = [r_{lr}^{(1)}, \ldots, r_{lr}^{(K)}]^\top$. This function can be implemented through code generated by LLMs or manually designed by domain experts.

4 METHOD

Section 4.1 formalizes the composition model and the bilevel optimization; Section 4.2 introduces one stabilization mechanism for each of three challenges; Section 4.3 assembles the practical algorithm. Appendix A records the dual-discount discrepancy of the frozen composition score. Figure 1 gives an overview.

4.1 STATE-ADAPTIVE REWARD COMPOSITION FRAMEWORK

Latent Reward Composition Model. Building upon the latent reward representation introduced in Section 3, we propose a state-dependent *intention function* $\phi_\psi : \mathcal{S} \to \Delta_K$, where $\Delta_K = \{\mathbf{w} \in \mathbb{R}^K : w_i \geq 0, \sum_{i=1}^K w_i = 1\}$ denotes the probability simplex. The composed reward is defined as

$$R_{\text{comp}}(s,a) = \phi_\psi(s)^\top \mathbf{r}_{\text{lr}}(s,a) + r_{\text{real}}(s,a), \tag{2}$$

where $r_{\text{real}}(s,a)$ denotes the sparse environment reward. The i-th component of $\phi_\psi(s)$ represents the agent's attention to the i-th latent reward dimension at state s, constituting the agent's *intention representation*.

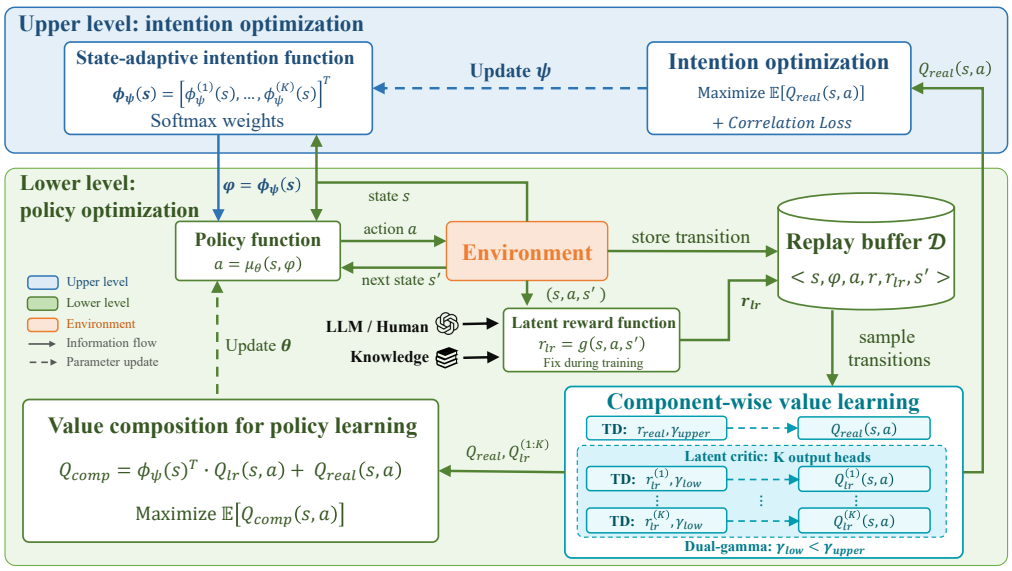

Figure 1: Overview of SARC. LLM or human knowledge provides a fixed set of latent reward functions, and SARC learns how to use them in different states through bilevel optimization. At the lower level, a state-adaptive intention vector conditions the policy and weights the latent values, which are combined with the real-reward value to guide policy optimization. At the upper level, the intention function is optimized using the real-reward value and correlation guidance, learning which latent signals to emphasize in each state. The critics are trained from replayed transitions, with $\gamma_{\text{low}} < \gamma_{\text{upper}}$ separating shorter-horizon latent guidance from longer-horizon task evaluation. Solid arrows denote data flow; dashed arrows denote parameter updates.

Bilevel Optimization Framework. Following the bilevel formulation in Section 3, SARC is formulated as

$$\begin{aligned} &\max_{\psi} J_{\text{real}}(\pi_\theta^*(\psi)) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_{\text{real}}(s_t, a_t) \mid \pi_\theta^*(\psi)\right], \\ &\text{s.t. } \pi_\theta^*(\psi) = \arg\max_{\theta} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R_{\text{comp}}(s_t, a_t)\right]. \end{aligned} \tag{3}$$

Compared with Hu et al. (2020), who learn a scalar weight for a single shaping reward, SARC learns a priority profile over K semantic dimensions and jointly optimizes both levels via reinforcement learning. The simplex constraint on the weights makes them directly interpretable as task priorities.

Value Decomposition. To jointly optimize both levels, we need the Q-function $Q_{\text{comp}}(s,a) = \mathbb{E}[\sum_t \gamma^t R_{\text{comp}}(s_t,a_t) \mid s_0{=}s, a_0{=}a]$. Learning a single critic for Q_{comp} is unstable inside the bilevel loop: every update of ψ changes the composed reward for all stored transitions, forcing the critic to chase a moving target. We instead decompose Q_{comp} into per-component critics:

$$Q_{\text{comp}}(s,a) \triangleq \phi_\psi(s)^\top \mathbf{Q}_{\text{lr}}(s,a) + Q_{\text{real}}(s,a), \tag{4}$$

where $\mathbf{Q}_{\text{lr}}(s,a) = [Q_{\text{lr}}^{(1)}(s,a), \ldots, Q_{\text{lr}}^{(K)}(s,a)]^\top$. Each $Q_{\text{lr}}^{(i)}$ estimates the cumulative value of $r_{\text{lr}}^{(i)}$ alone, so it remains stable when ψ changes. This decomposition introduces a bias because ϕ_ψ varies across states along a trajectory; we address this bias with a dedicated mechanism below. Appendix A compares the two routes in detail.

4.2 Stabilization Mechanisms for SARC

Solving Equation (3) in practice must overcome three challenges. First, the Q-decomposition biases *value estimation* because ϕ_ψ varies along trajectories. Second, the coupled *optimization* of the two

levels suffers from ambiguous reward attribution. Third, *data collection* stalls when feedback is entirely absent. We introduce one mechanism for each.

Dual-Gamma Discounting against Estimation Bias. The decomposition in Equation (4) freezes the intention at the query state, implicitly assuming that $\phi_\psi(s)$ stays constant along a trajectory. Since ϕ_ψ varies with state, this introduces a systematic bias:

$$\text{Bias} = \mathbb{E}\left[\sum_{t=1}^{T-1} \gamma^t \left(\phi_\psi(s_t) - \phi_\psi(s_0)\right)^\top \mathbf{r}_{\text{lr}}(s_t, a_t)\right]. \tag{5}$$

The bias grows with the cumulative intention drift along the trajectory. The discount factor directly controls this drift. We therefore use two different discount factors, a scheme we call *dual-gamma discounting*. The lower level uses a smaller γ_{low}, shortening the effective horizon to $T_{\text{eff}} = (1 - \gamma_{\text{low}})^{-1}$ and limiting bias accumulation. The upper level keeps a larger γ_{upper} so that intentions are still judged by long-term environment returns.

Correlation Guidance against Attribution Ambiguity. When the lower-level policy obtains rewards through exploration despite suboptimal upper-level weights, the upper level wrongly reinforces those weights while the policy learns to compensate, trapping the system in local optima (Levy et al., 2017). To break this coupling, we introduce *correlation guidance*, which supervises the intention function with a signal independent of Q-function estimation:

$$\begin{aligned} \mathcal{L}_{\text{corr}}(\psi) &= \sum_{i=1}^{K} \max(\rho_i, 0) \cdot \mathbb{E}_{s\sim\mathcal{D}}[\phi_\psi^{(i)}(s)], \\ \rho_i &= \text{Spearman}\left(\{R_{\text{lr},l}^{(i)}\}_{l=1}^{L}, \{R_{\text{real},l}\}_{l=1}^{L}\right), \end{aligned} \tag{6}$$

where $R_{\text{lr},l}^{(i)} = \sum_t r_{\text{lr}}^{(i)}(s_t^l, a_t^l)$ and $R_{\text{real},l}$ are the i-th latent return and the environment return of the l-th of the L buffered episodes; ρ_i denotes the Spearman rank correlation coefficient between them. The guidance favors positively correlated dimensions; undefined correlations, including those of the zero-compensation dimension, are set to zero.

Consistent Exploration against Zero Feedback. Standard exploration perturbs actions independently at each timestep, which rarely produces the coherent behaviors needed to discover sparse rewards (Wang et al., 2025). We instead explore in the space of intentions: at the start of each episode, we sample a bias vector $\phi_{\text{bias}} \sim \mathcal{N}(0, \sigma^2 I)$ and keep it fixed throughout. The actual intention at each timestep is

$$\tilde{\phi}(s) = \text{softmax}(\phi_a(s) + \phi_{\text{bias}}), \tag{7}$$

where $\phi_a(s)$ is the pre-softmax output of the intention network, keeping a consistent exploration direction in the space of task priorities throughout each episode.

4.3 Optimization Objectives and Algorithm

We adopt TD3 (Fujimoto et al., 2018) as the base framework, whose off-policy nature alleviates sample-efficiency issues under sparse rewards (Andrychowicz et al., 2017). For clarity we present the objectives with a single critic per value head; twin-critic and target-smoothing details are given in Appendix D.

Value Function Estimation. Following the decomposition in Section 4.1, we estimate one Q-function per latent dimension and one for the real reward. Since intentions sum to one, we reserve the last latent dimension as a zero-compensation dimension, $r_{\text{lr}}^{(K)} \equiv 0$, which lets the agent shift weight away from latent rewards when they cannot express optimal behavior. Each latent Q-function is trained by minimizing the TD error

$$\begin{aligned} \mathcal{L}_{\text{lr}}^{(i)}(\omega) &= \mathbb{E}_{(s,a,\mathbf{r}_{\text{lr}},s')\sim\mathcal{D}}\left[\left(Q_{\text{lr}}^{(i),\omega}(s,a) - y_{\text{lr}}^{(i)}\right)^2\right], \\ y_{\text{lr}}^{(i)} &= r_{\text{lr}}^{(i)} + \gamma_{\text{low}} Q_{\text{lr}}^{(i),\bar{\omega}}(s', \tilde{a}'), \end{aligned} \tag{8}$$

where $\bar{\omega}$ denotes the target network parameters and $\tilde{a}'$ is the smoothed target action. The real-reward critic Q_{real}^{ξ} is trained with γ_{upper}, instantiating the dual-gamma discounting of Section 4.2.

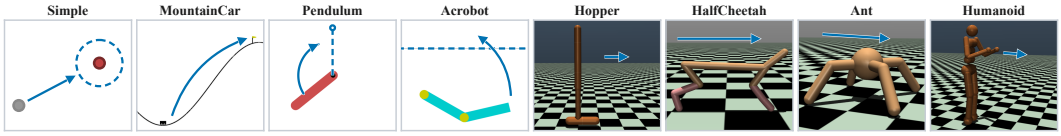

Figure 2: The eight sparse-reward continuous-control tasks. Blue marks the objective: a dashed goal state where the task defines one, and an arrow for the direction the agent must drive the system. Full specifications are in Appendix C.

Policy and Intention Optimization. The lower-level policy μ_θ maximizes the composed Q-value, while the upper-level intention ϕ_ψ maximizes the environment Q-value with correlation guidance:

$$J_\theta = \mathbb{E}_{s\sim\mathcal{D}}\left[Q_{\text{comp}}(s, \mu_\theta(s, \phi_\psi(s)))\right], \tag{9}$$

$$J_\psi = \mathbb{E}_{s\sim\mathcal{D}}\left[Q^{\xi}_{\text{real}}(s, \mu_\theta(s, \phi_\psi(s)))\right] + \lambda_{\text{corr}}\mathcal{L}_{\text{corr}}, \tag{10}$$

where $Q_{\text{comp}}(s,a) = \phi_\psi(s)^\top \mathbf{Q}^{\omega}_{\text{lr}}(s,a) + Q^{\xi}_{\text{real}}(s,a)$.

Algorithm Procedure. Each iteration updates the critics and policy, while the intention is updated at a delayed frequency; target networks use soft updates. The exploration bias is sampled at the start of each episode, and the correlation coefficients $\{\rho_i\}_{i=1}^{K}$ are periodically refreshed from recent episodes. The complete procedure is given in Appendix D.1.

5 Experiments

We evaluate SARC to answer four key questions: **(RQ1)** Can SARC learn more effective policies under sparse rewards compared to other baselines? **(RQ2)** Do the learned intention weights align with task structure and causally steer behavior? **(RQ3)** How does each stabilization mechanism contribute to overall performance? **(RQ4)** How should effective latent rewards be designed by humans or LLMs?

5.1 Experimental Setup

Environments. We use eight continuous-control tasks (Figure 2): four low-dimensional classic tasks, Simple, MountainCar, Pendulum, and Acrobot, covering open-ended exploration, extreme sparsity, and delayed reward, and four high-dimensional MuJoCo locomotion tasks. All tasks share one sparsification protocol: the original dense objective is replaced by a per-step binary reward, where $r_{\text{real}} = 1$ if and only if the state satisfies a task-specific success condition, such as reaching a goal region or meeting thresholds on height, speed, and posture. Full task specifications are in Appendix C.

Latent Rewards. For each task, the latent reward encoder g is generated by an LLM from a natural-language task description, following Qu et al. (2025). Each dimension captures a distinct, human-readable aspect, such as *Position Progress*, *Speed*, and *Momentum* for MountainCar; the zero-compensation dimension of Section 4.3 is appended by the framework. Prompts and the resulting functions are in Appendices B and C.

Baselines. We compare SARC against three groups of methods. The first group includes standard continuous-control algorithms: SAC (Haarnoja et al., 2018), TD3 (Fujimoto et al., 2018), TQC (Kuznetsov et al., 2020), and DDPG (Lillicrap et al., 2015). The second group includes intrinsic-motivation methods: RND (Burda et al., 2018) and ICM (Pathak et al., 2017). The third group includes LLM-empowered reward design methods: Eureka (Ma et al., 2023) and LaRe (Qu et al., 2025). The exploration-based and reward-shaping methods are implemented on top of TD3.

Evaluation Protocol. Every method is evaluated at intervals of 1% of the training steps with 30–50 episodes per checkpoint. The reported metric is the sparse environment return. Final performance is the mean sparse return over the last five checkpoints, averaged across 5 seeds; curves show the

Table 1: Final performance (mean ± standard deviation over 5 seeds) across eight sparse-reward continuous control tasks. **Bold**: best; underline: runner-up; ties at the reported precision share the same marking.

Method	Simple	MountainCar	Pendulum	Acrobot	Hopper	HalfCheetah	Ant	Humanoid
SAC	**20.8±0.1**	0.0±0.0	18.1±0.6	0.0±0.0	953.0±25.6	586.4±478.8	749.7±229.0	896.0±65.6
TD3	18.7±2.5	0.0±0.0	9.5±5.3	0.0±0.0	701.9±383.9	0.0±0.0	8.5±4.5	0.0±0.0
TQC	20.7±0.0	0.8±0.4	19.7±0.9	0.0±0.0	889.5±165.2	983.0±1.5	779.3±25.0	648.1±228.5
DDPG	20.7±0.2	0.6±0.5	14.6±7.3	13.7±11.2	907.8±74.5	900.9±121.3	732.6±80.0	391.8±118.6
Eureka	16.5±1.1	0.0±0.0	13.9±0.9	28.2±0.1	534.4±317.4	785.9±76.5	570.1±37.6	936.9±4.2
LaRe	20.7±0.1	0.8±0.4	**20.9±0.6**	20.6±10.5	954.2±8.4	982.6±2.5	490.8±322.9	153.5±50.0
RND	0.0±0.0	0.8±0.4	0.1±0.0	26.2±0.9	454.1±193.5	0.0±0.0	11.2±0.3	331.9±15.5
ICM	0.0±0.0	0.8±0.4	0.4±0.2	25.9±2.1	759.0±381.6	0.0±0.0	28.3±13.8	579.6±462.6
SARC (Ours)	**20.8±0.1**	**1.0±0.0**	20.5±0.4	**31.1±1.2**	**972.6±6.4**	**983.4±0.4**	**910.8±51.3**	**953.3±6.1**

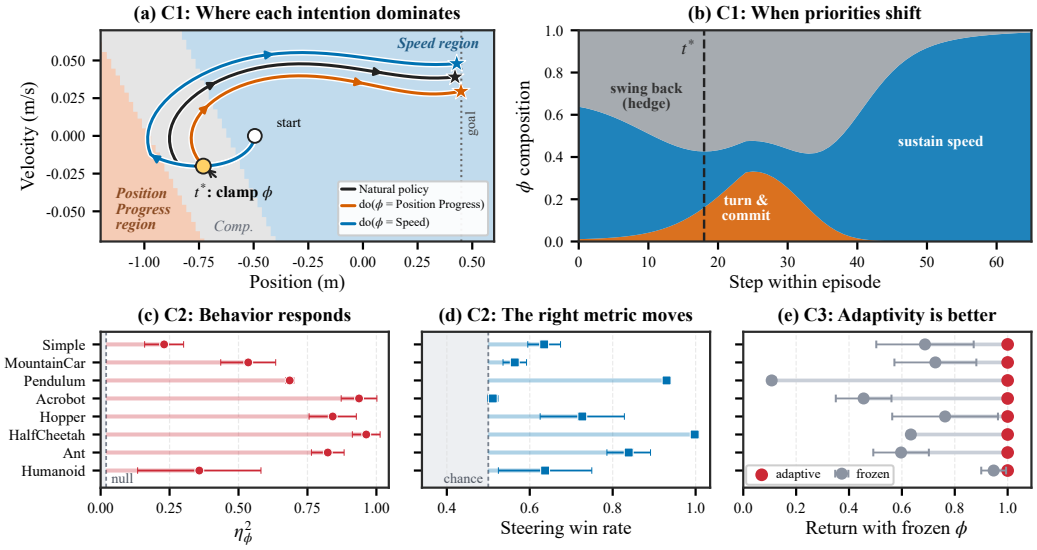

Figure 3: Interpretability of SARC. Panels **a** and **b** show what the intention says on MountainCar; **c–e** test whether it holds on all eight tasks, one row per task. **(a)** States colored by their dominant intention, with three rollouts from one start state: the natural policy and two branches clamped to a one-hot from t^* on. Stars mark goal arrival. **(b)** Intention weights along the natural episode. **(c)** Behavioral variance explained by which intention was set, measured from the shuffled null. **(d)** How often strengthening intention i raises metric i more than strengthening another one does, measured from chance. **(e)** Sparse return when the intention is frozen for whole episodes. Markers are means and whiskers span the training runs; see Appendix F.

mean with standard-error bands. Ablations additionally report a per-environment *normalized score*, the final return divided by the maximum attainable sparse return, with 95% bootstrap confidence intervals. The full protocol and hyperparameters are in Appendices C and D.

5.2 The Superiority of SARC

Undirected exploration is insufficient under sparse rewards. As shown in Table 1, the standard baselines all struggle under sparse rewards. TD3 and DDPG rely on undirected random exploration and fail on tasks that demand structured exploration such as MountainCar and Acrobot. Notably, TD3 trails DDPG under sparsity despite being stronger under dense rewards (Appendix I): its clipped double-Q pessimism suppresses the optimistic value estimates needed to reach rare successes (Ciosek et al., 2019; Matheron et al., 2020). SAC and TQC do better through maximum-entropy exploration but remain unstable on Acrobot. SARC achieves strong performance across

the eight tasks: state-adaptive composition of latent rewards supplies the structured feedback that undirected exploration alone cannot. Learning curves and an extended analysis are in Appendix E.

Comparison with LLM-based reward-shaping methods. Eureka performs moderately, indicating that directly authoring a precise shaping reward remains challenging for LLMs. LaRe is competitive on several tasks, including Pendulum and HalfCheetah, but its performance varies across the benchmark. SARC achieves higher mean returns than LaRe on most tasks; its explicit latent-reward composition and intention-space exploration provide structured guidance under sparse feedback.

Intrinsic motivation fails on open-ended exploration. RND and ICM help on exploration-driven tasks such as Acrobot but fail almost entirely in open-ended environments such as Simple and the MuJoCo tasks, which lack the novelty structure that curiosity-driven methods need.

5.3 The Interpretability of SARC

Reading intentions. The intention $\phi_\psi(s)$ is SARC's built-in explanation. Its i-th coordinate is the weight on the latent reward whose semantic label was fixed before training, and the weights lie on the simplex, so they read as priorities. Three conditions decide whether this reading can be trusted, and Figure 3 answers all three. Appendix F gives the protocol and the full-resolution results.

C1: Intentions match the task's phase structure. MountainCar has an engine too weak to climb directly, so the car must oscillate to build momentum first. The learned intentions recover this structure. Figure 3a colors each state by its dominant intention. *Speed* covers the oscillation corridor, and *Position Progress* covers the left slope where the car must swing back. Figure 3b traces one episode, in which the compensation dimension leads during the backswing, *Position Progress* fires at the turning point, and *Speed* sustains the final ascent. This reading holds on most of the tasks, where one core dimension tends to dominate and the others take over mainly at the transitions. That dimension also tends to be the one whose removal most degrades performance in RQ4, suggesting that the explanation points at a part of the reward that matters.

C2: Intentions causally steer behavior. Matching the task structure does not yet prove that the policy *uses* the intention. We therefore intervene on the intention itself and measure the resulting behavioral consequence. This intervention changes only the intention input, keeping network weights fixed while allowing actions to change. Figure 3c shows whether behavior changes when the intervention is applied, and Figure 3d asks whether the changed behavior is consistent with the intervened intention dimension. On all eight tasks, the behavioral response is far above the shuffled null, showing that the policy uses the intention input. The single rollout in Figure 3a splits into three within a few steps of the intervention. Acrobot is the one task whose steering is not selective, and the reason lies in its latent rewards rather than in the policy. Its three metrics are physically coupled: because raising the tip also raises momentum and energy, no intervention can move a single metric alone. Humanoid responds least, and C3 gives the reason: whichever of its four priorities is pushed, the behavior that serves it is close to the same walking gait, so the conditions differ less from one another than on any other task.

C3: State-adaptivity is better. Figure 3e freezes the intention at a single dimension for the entire episode, asking what a policy loses when it commits to only one priority. Almost every task incurs a loss. Pendulum retains hardly any of its return, because swing-up and stabilization call for opposite priorities. MountainCar still reaches the goal, but takes far longer. Hopper, HalfCheetah, and Ant each give up a substantial portion of their return. Humanoid is the exception, and on closer inspection the reason is that a properly functioning gait already satisfies speed, height, and balance simultaneously; fixed weights therefore incur little loss. Overall, adaptive reward design not only yields interpretability but also outperforms static weight composition across a range of tasks.

5.4 Delving into SARC

Stabilization mechanisms matter more as tasks grow complex. We ablate the three stabilization mechanisms using the per-environment normalized score (Figure 4, top), reading the comparison

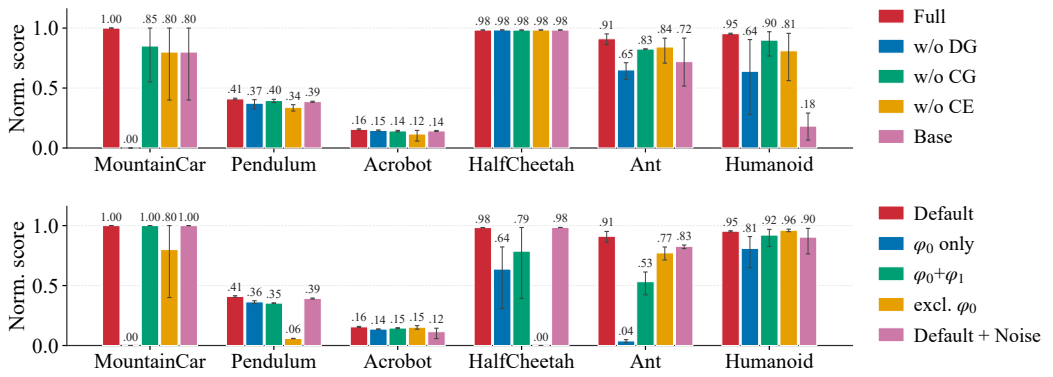

Figure 4: Ablations of SARC across six environments. Bars show the final normalized score with 95% bootstrap confidence intervals over 5 seeds. The top row reports module ablations, where *w/o DG*, *w/o CG*, and *w/o CE* remove one stabilization mechanism each and *Base* removes all three; the bottom row reports latent-dimension ablations over the full model and its reduced and extended variants, where ϕ_0 and ϕ_1 are the first two dimensions the encoder emits. The leftmost red bar is the complete model in both rows, labeled *Full* above and *Default* below. Simple and Hopper are omitted because every variant saturates there.

from MountainCar, Acrobot, Ant, and Humanoid, the tasks that have not yet reached the normalization ceiling. Removing dual-gamma discounting (*w/o DG*), correlation guidance (*w/o CG*), or consistent exploration (*w/o CE*) each degrades performance, increasingly so on harder tasks; dual-gamma discounting is the most critical, consistent with its role in bounding the value-estimation bias. *Base*, with all three removed, still retains solid performance on some tasks, indicating that even without any stabilization mechanism SARC already shows strong potential. Per-task learning curves are in Appendix G.

Core dimensions are necessary; redundant ones are tolerable. We vary the latent-reward design (Figure 4, bottom). Excluding the primary dimension (*excl.* ϕ_0) or collapsing to it alone (ϕ_0 *only*) markedly hurts the complex tasks, whereas the two leading dimensions ($\phi_0 + \phi_1$) already recover most of the performance; appending an uninformative dimension (*Default + Noise*) changes little, as the intention assigns it negligible weight. Which dimension is the core one is task-specific, so neither variant is uniformly the worse: on MountainCar, whose core dimension is ϕ_1, keeping only ϕ_0 is the more damaging choice. For authoring, cover the core objectives rather than prune: redundant dimensions cost little, a missing core one is damaging.

Sensitivity. A dual-gamma study shows that separating the two discount factors is consistently beneficial: the preferred γ_{low} rises with task complexity yet degrades when set too high; correlation guidance helps across a broad range of its weight (Appendix H).

Beyond sparse rewards. Under the original dense-reward MuJoCo setting, SARC remains competitive with SAC, TD3, TQC, and DDPG, suggesting that state-adaptive composition is not merely a sparse-reward fix but can also be useful under dense rewards (Appendix I).

6 Conclusion

This paper proposes State-Adaptive Reward Composition (SARC), a bilevel framework that composes multi-dimensional latent rewards with state-dependent intention weights. The intention function guides policy learning under sparse feedback while exposing the agent's priorities as a human-readable explanation. The future work is to extend SARC to multi-agent systems and real-world domains, such as vision-based control and physical robotics.

AI USE STATEMENT

An LLM was used to generate latent-reward encoder code from task descriptions, as described in the method and Appendix B. Generative AI tools also assisted with language polishing and literature search. The authors take responsibility for the final content of this work, including all AI-assisted text, code, and other artifacts.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Serena Booth, W Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5920–5929, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, volume 32, pp. 1785–1796, 2019.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Dhawal Gupta, Yash Chandak, Scott Jordan, Philip S. Thomas, and Bruno C. da Silva. Behavior alignment via reward function optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2297. URL https://doi.org/10.52202/075280-2297.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

W Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis) design for autonomous driving. *Artificial Intelligence*, 316:103829, 2023.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International conference on machine learning*, pp. 5556–5566. PMLR, 2020.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Haoxin Lin, Hongqiu Wu, Jiaji Zhang, Yihao Sun, Junyin Ye, and Yang Yu. Episodic return decomposition by difference of implicitly assigned sub-trajectory reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13808–13816, 2024.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Guillaume Matheron, Nicolas Perrin, and Olivier Sigaud. Understanding failures of deterministic actor-critic with continuous action spaces and sparse rewards. In *International Conference on Artificial Neural Networks*, pp. 308–320. Springer, 2020.

Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7):1–36, 2024.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Yun Qu, Yuhang Jiang, Boyuan Wang, Yixiu Mao, Cheems Wang, Chang Liu, and Xiangyang Ji. Latent reward: Llm-empowered credit assignment in episodic reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20095–20103, 2025.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Zhizhou Ren, Ruihan Guo, Yuan Zhou, and Jian Peng. Learning long-term reward redistribution via randomized return decomposition. *arXiv preprint arXiv:2111.13485*, 2021.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 28446–28463, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/7a7f6cc5dc2a84fb4edf0feb8e5cfd50-Paper-Conference.pdf.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36:55681–55693, 2023.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Leonardo Villalobos-Arias, Grant Forbes, Jianxun Wang, David L. Roberts, and Arnav Jhala. Minding motivation: The effect of intrinsic motivation on agent behaviors. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 21(1):347–356, 2025. doi: 10.1609/aiide.v21i1.36838. URL https://ojs.aaai.org/index.php/AIIDE/article/view/36838.

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

Yiming Wang, Ming Yang, Renzhi Dong, Binbin Sun, Furui Liu, and Leong Hou U. Efficient potential-based exploration in reinforcement learning using inverse dynamic bisimulation metric. *Advances in Neural Information Processing Systems*, 36:38786–38797, 2023.

Yiming Wang, Kaiyan Zhao, Yan Li, and Leong Hou U. Bile: an effective behavior-based latent exploration scheme for deep reinforcement learning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 6497–6505, 2025.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, 2024.

Yuqing Xie, Jiayu Chen, Wenhao Tang, Ya Zhang, Chao Yu, and Yu Wang. Automatic reward shaping from multi-objective human heuristics. *arXiv preprint arXiv:2512.15120*, 2025. URL https://arxiv.org/abs/2512.15120.

Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: A survey. In James Kwok (ed.), *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 10807–10816. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/1199. URL https://doi.org/10.24963/ijcai.2025/1199. Survey Track.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 374–404. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/yu23a.html.

APPENDIX

This appendix collects the theoretical analysis, LLM prompts, experimental configurations, and additional results deferred from the main paper. All cross-references link within this document. Main-paper numbering is retained; appendix equations are numbered by section, and appendix figures and tables carry an "A" prefix. The material is organized as follows.

- **Appendix A (Theoretical Analysis)** records the frozen-intention discrepancy of the dual-discount composition score for a fixed intention map and reference policy, including the discount-mismatch term relative to a common-discount composed return.
- **Appendix B (LLM Prompts for Latent Reward Generation)** documents how the multi-dimensional latent rewards were generated, expanding the *Latent Rewards* paragraph of the main paper: the prompt template, an example task description, the corresponding LLM response, and a brief LLM sensitivity analysis of latent rewards.
- **Appendix C (Environment Details)** is the authoritative definition of every task's sparsification protocol, success predicate, observation and action dimensions, and the semantics of each latent dimension ϕ_i referenced throughout the paper.
- **Appendix D (Implementation Details)** specifies the network architectures, the stabilization mechanisms, and all hyperparameters needed to reproduce our results.
- **Appendix E (Additional Comparative Results)** reports the per-task learning curves and an analysis of the baseline failure modes that complement the main results table (main paper RQ1).
- **Appendix F (Intervention Protocol and Extended Interpretability Results)** details the causal intervention pipeline behind the main paper's interpretability figure (RQ2).
- **Appendix G (Additional Ablation Results)** expands the module and latent-dimension ablations of the main paper (RQ3 and RQ4).
- **Appendix H (Sensitivity Analysis)** analyzes sensitivity to the dual-gamma discount factors and the correlation weight.
- **Appendix I (Beyond Sparse Rewards: Continuous-Reward Adaptation)** reports the behavior of SARC under the original dense rewards for reference.
- **Appendix J (Extended Related Work)** restores the citations behind the main paper's *Related Work* paragraph.

Notation. Table A1 lists the symbols that are specific to SARC, together with those of the main paper's *Method* section that this appendix refers back to.

Table A1: Core symbols of SARC used throughout this appendix.

Symbol	Description
Latent rewards and reward composition	
K, K'	Total latent dimension including the zero-compensation coordinate, and the number of informative dimensions, $K' = K - 1$
$\mathbf{r}_{\text{lr}}$, $r_{\text{lr}}^{(i)}$	Latent reward vector and its i-th entry; the framework appends $r_{\text{lr}}^{(K)} \equiv 0$ to the K' generated coordinates
r_{real}, $R_{\text{real},l}$	Per-step environment reward (binary in the sparse tasks), and the summed environment return in buffered episode l
$R_{\text{lr},l}^{(i)}$	Summed latent return of dimension i in buffered episode l in Eq. (6); implementation normalization is described in Appendix D.4
R_{comp}	Composed reward $\phi_\psi(s)^\top \mathbf{r}_{\text{lr}}(s,a) + r_{\text{real}}(s,a)$ (Eq. (2))
R_r, R_ℓ	Reward bounds $\lvert r_{\text{real},t}\rvert \le R_r$ and $\lVert \mathbf{r}_{\text{lr},t}\rVert_\infty \le R_\ell$ (Assumption 1)
MAXRETURN	Per-task reference episodic return used to normalize experimental scores; distinct from the per-step bounds R_r, R_ℓ
Intention and coordinate conventions	
Δ_K, e_i	Probability simplex $\{w \in \mathbb{R}^K : w_i \ge 0,\ \sum_i w_i = 1\}$, and its i-th vertex
$\phi(s)$, ϕ_ψ, ψ	Intention vector, parameterized intention network $\mathcal{S} \to \Delta_K$, and its parameters; Appendix A fixes the map ϕ
ϕ_i	Weight of coordinate i, also used as its label in experimental plots. Theory indexes $1, \dots, K$; code and plots index $0, \dots, K-1$. The final coordinate is compensation
$\phi_a(s)$, ϕ_{bias}	Pre-softmax intention logits and the episode-level Gaussian bias added for consistent exploration
$\tilde{\phi}$	Intention perturbed by exploration (Eq. (7)) or by a dose intervention (Eq. (F.1))
Exact reference values and frozen-intention discrepancy	
γ_ℓ, γ_u, T_{eff}	$\gamma_\ell = \gamma_{\text{low}}$, $\gamma_u = \gamma_{\text{upper}}$, $0 \le \gamma_\ell \le \gamma_u < 1$; effective latent horizon $T_{\text{eff}} = (1-\gamma_\ell)^{-1}$
π, $\mathbb{E}_{s,a}^\pi$	Fixed policy $\pi(s) = \mu_\theta(s, \phi(s))$; expectation conditional on $s_0 = s, a_0 = a$, following π thereafter
Q_r^π, q_i^π, q^π	Exact environment value discounted by γ_u, exact latent value discounted by γ_ℓ, and $q^\pi = (q_1^\pi, \dots, q_K^\pi)^\top$ (Eq. (A.1))
F_ϕ^π, G_ϕ^π	Frozen score $Q_r^\pi + \phi(s)^\top q^\pi$ and trajectory-weight functional using $\phi(s_t)$ with the same two discounts (Eq. (A.2))
$\delta_t(s,a)$	Expected intention drift $\mathbb{E}_{s,a}^\pi \lVert\phi(s_t) - \phi(s)\rVert_1$ at lag t
$B_{\text{drift}}(s,a)$	Bound $R_\ell \sum_{t\ge1} \gamma_\ell^t \delta_t(s,a)$ on $\lvert(G_\phi^\pi - F_\phi^\pi)(s,a)\rvert$ (Theorem 1)
$Q_{\text{comp},u}^\pi$	Composed return with trajectory weights and common discount γ_u; its gap from F_ϕ^π includes the discount-mismatch term (Eqs. (A.6), (A.8))
Bias	Common-discount discrepancy in main-paper Eq. (5); Appendix A treats the dual-discount comparison separately
Learned networks, objectives, and hyperparameters	
μ_θ, θ	Deterministic actor with input $[s;\phi]$, and its parameters
$\mathbf{Q}_{\text{lr}}^\omega$, Q_{real}^ξ	Learned latent and environment critics, distinct from exact reference values q^π, Q_r^π; $\bar{\omega}$ denotes target latent-critic parameters
Q_{comp}	Actor score $\phi_\psi(s)^\top \mathbf{Q}_{\text{lr}}^\omega(s,a) + Q_{\text{real}}^\xi(s,a)$ used in Eq. (9)
$J_{\text{real}}(\pi)$	Expected *discounted* environment return in the bilevel objective (Eq. (3))
J_θ, J_ψ	Policy and intention objectives, both maximized (Eqs. (9), (10))
$\mathcal{L}_{\text{lr}}^{(i)}$, $y_{\text{lr}}^{(i)}$	TD loss and target of the i-th latent critic (Eq. (8))
$\mathcal{L}_{\text{corr}}$, λ_{corr}, ρ_i	Correlation-guidance term, its coefficient, and latent/environment Spearman correlation; despite its name, $\mathcal{L}_{\text{corr}}$ is added to the maximized J_ψ
L, $\mathcal{D}$, $\mathcal{H}$	Correlation window ($L = 500$ episodes), replay buffer, and episode history
σ, σ_a, ϵ_t	Intention-bias standard deviation, action-noise scale, and sampled action noise $\epsilon_t \sim \mathcal{N}(0, \sigma_a^2 I)$ in Algorithm 1
σ_{targ}, ϵ, d, τ	Target-smoothing standard deviation; random-action probability (distinct from ϵ_t); update-delay parameter; Polyak coefficient
Interpretability read-outs	
$\text{do}(\phi)$, α	Intervention on the actor's intention input; dose α mixes e_i with $\phi_\psi(s_t)$ as in Eq. (F.1); uniform control is $\mathbf{1}/K$
η_ϕ^2, WR, $S_i(\alpha)$	Intervention partial variance share (Eq. (F.2)), paired steering win-rate, and dose–response; the WR baseline depends on tie handling

A THEORETICAL ANALYSIS

This appendix records the discrepancy introduced by freezing the intention inside the dual-discount composition score. Fix a measurable intention map $\phi : \mathcal{S} \to \Delta_K$ and the deterministic reference policy $\pi(s) = \mu_\theta(s, \phi(s))$ associated with one parameter snapshot. Conditional on $s_0 = s$ and a feasible $a_0 = a$, $\mathbb{E}^\pi_{s,a}$ follows this same policy thereafter. Write $\gamma_\ell = \gamma_{\text{low}}$ and $\gamma_u = \gamma_{\text{upper}}$, with $0 \le \gamma_\ell \le \gamma_u < 1$. The map and policy are fixed within the comparison. The argument does not treat nonstationary training, correlation guidance, episode-level intention exploration, or a separation from static weights.

Assumption 1 (Uniformly bounded rewards)**.** *For finite constants $R_r, R_\ell \ge 0$, the rewards satisfy $|r_{real,t}| \le R_r$, $\|\mathbf{r}_{lr,t}\|_\infty \le R_\ell$, and $r^{(K)}_{lr,t} = 0$ almost surely, uniformly over feasible state–action pairs and times.*

These are conditions of the comparison, not new clipping operations. A finite-horizon problem is represented with time-augmented states and zero rewards after its horizon. A time-limit truncation that is bootstrapped in an implementation need not have the same return; that gap is not treated as an exact value identity. We use $\gamma^0 = 1$, including when $\gamma = 0$.

Define the exact reference values of this fixed policy,

$$\begin{aligned} Q^\pi_r(s,a) &= \mathbb{E}^\pi_{s,a} \sum_{t=0}^{\infty} \gamma_u^t r_{\text{real},t}, \\ q^\pi_i(s,a) &= \mathbb{E}^\pi_{s,a} \sum_{t=0}^{\infty} \gamma_\ell^t r^{(i)}_{\text{lr},t}, \qquad i = 1, \dots, K, \end{aligned} \tag{A.1}$$

and write $q^\pi = (q^\pi_1, \dots, q^\pi_K)^\top$. The frozen score and the trajectory-weight comparison functional are

$$\begin{aligned} F^\pi_\phi(s,a) &= Q^\pi_r(s,a) + \phi(s)^\top q^\pi(s,a), \\ G^\pi_\phi(s,a) &= \mathbb{E}^\pi_{s,a} \sum_{t=0}^{\infty} \left[\gamma_u^t r_{\text{real},t} + \gamma_\ell^t \phi(s_t)^\top \mathbf{r}_{\text{lr},t} \right]. \end{aligned} \tag{A.2}$$

F^π_ϕ is the exact-value form of the actor score in Equation (9): each component is an action value of π, and the latent weights are the intention at the query state. G^π_ϕ uses the same two discounts but lets the latent weights follow $\phi(s_t)$ along the trajectory. Neither object is assumed to satisfy a single-discount Bellman equation for R_{comp}.

Theorem 1 (Frozen-intention discrepancy with two discounts)**.** *Let $\delta_t(s,a) = \mathbb{E}^\pi_{s,a} \|\phi(s_t) - \phi(s)\|_1$ and*

$$B_{\text{drift}}(s,a) = R_\ell \sum_{t=1}^{\infty} \gamma_\ell^t \delta_t(s,a). \tag{A.3}$$

Then

$$(G^\pi_\phi - F^\pi_\phi)(s,a) = \mathbb{E}^\pi_{s,a} \sum_{t=1}^{\infty} \gamma_\ell^t [\phi(s_t) - \phi(s)]^\top \mathbf{r}_{lr,t}, \tag{A.4}$$

and $|G^\pi_\phi - F^\pi_\phi|(s,a) \le B_{\text{drift}}(s,a)$. For fixed π, ϕ, reward functions, and transition law, each $\delta_t(s,a)$ does not depend on γ_ℓ, so this upper bound is nondecreasing in γ_ℓ.

Proof. Bounded rewards, simplex weights, and discounts below one make both series absolutely integrable, so expectation and summation may be interchanged. Because $\phi(s)$ does not depend on the future trajectory,

$$\phi(s)^\top q^\pi(s,a) = \mathbb{E}^\pi_{s,a} \sum_{t=0}^{\infty} \gamma_\ell^t \phi(s)^\top \mathbf{r}_{\text{lr},t}. \tag{A.5}$$

Subtracting this expression and Q^π_r from G^π_ϕ cancels the environmental series. The latent term at $t = 0$ vanishes because $s_0 = s$. This gives Equation (A.4). Hölder's inequality on the simplex yields $|[\phi(s_t) - \phi(s)]^\top \mathbf{r}_{\text{lr},t}| \le R_\ell \|\phi(s_t) - \phi(s)\|_1$. Taking the expectation and summing the resulting nonnegative series gives B_{drift}. The claimed monotonicity in γ_ℓ is then immediate from the series coefficients. □

For comparison with the composed return that discounts both reward streams by γ_u, define

$$Q^{\pi}_{\mathrm{comp},u}(s,a) = \mathbb{E}^{\pi}_{s,a} \sum_{t=0}^{\infty} \gamma_u^t [r_{\mathrm{real},t} + \phi(s_t)^\top \mathbf{r}_{\mathrm{lr},t}]. \tag{A.6}$$

The same cancellation yields

$$\begin{aligned}(Q^{\pi}_{\mathrm{comp},u} - F^{\pi}_{\phi})(s,a) = (G^{\pi}_{\phi} - F^{\pi}_{\phi})(s,a) \\ + \mathbb{E}^{\pi}_{s,a} \sum_{t=0}^{\infty} (\gamma_u^t - \gamma_\ell^t) \phi(s_t)^\top \mathbf{r}_{\mathrm{lr},t}.\end{aligned} \tag{A.7}$$

Since $\gamma_u \geq \gamma_\ell$ and $|\phi(s_t)^\top \mathbf{r}_{\mathrm{lr},t}| \leq R_\ell$,

$$|Q^{\pi}_{\mathrm{comp},u} - F^{\pi}_{\phi}|(s,a) \leq B_{\mathrm{drift}}(s,a) + \frac{R_\ell(\gamma_u - \gamma_\ell)}{(1-\gamma_u)(1-\gamma_\ell)}. \tag{A.8}$$

The second term is zero when the two discounts agree, and it can be positive for a constant intention map, where the drift term is zero. For fixed γ_u it increases as γ_ℓ decreases.

Remark 1 (The drift bound does not order the actual discrepancy)**.** *Equation* (A.4) *is an expectation of signed terms, so its magnitude need not move with the upper bound. Consider a deterministic three-step trajectory with two latent coordinates,* $\phi(s_0) = (0.1, 0.9)$*,* $\phi(s_1) = (0.5, 0.5)$*,* $\phi(s_2) = (0.9, 0.1)$*, informative rewards* 1 *and* -1 *at times* 1 *and* 2*, and every other reward equal to zero. The discrepancy equals* $0.4\gamma_\ell - 0.8\gamma_\ell^2$*: its magnitude is* 0 *at* $\gamma_\ell = 0.5$ *and* 0.05 *at* $\gamma_\ell = 0.25$*. Both values are compatible with any fixed* $\gamma_u \geq 0.5$*. Reducing the latent discount can therefore increase the actual discrepancy while decreasing* B_{drift}*. The comparison does not select a discount, and it does not imply that learned performance improves as* γ_ℓ *decreases.*

The critic networks plugged into Equation (9) are not identified with Q^{π}_{r} and q^{π}. Target smoothing, replay, and a bootstrap that continues through a time limit can all separate the two. Theorem 1 bounds the exact frozen score against G^{π}_{ϕ} and, through Equation (A.8), against $Q^{\pi}_{\mathrm{comp},u}$. It leaves that estimation gap unquantified.

B LLM Prompts for Latent Reward Generation

This section gives the prompt template, together with an example task description, the LLM response it produced, and a brief LLM sensitivity analysis of the latent rewards. We follow the *environment prompting* design of LaRe (Qu et al., 2025). A single task-agnostic template combines a *role instruction* with a *self-prompting* step, and is instantiated for each task by a natural-language *task description*. Each task's latent rewards were generated with DeepSeek-V4-Pro before training, and the LLM is never queried during training.

Relation to LaRe's prompts. Our prompt design reuses LaRe's three components—role instruction, self-prompting, and task information; we present them in the same three blocks below. The only changes are three constraints in the role instruction required by the SARC composition model: (i) each evaluation factor must be normalized by a characteristic physical scale and clipped to $[0, 1]$ or $[-1, 1]$, so that all dimensions live on a common scale and none dominates value estimation (Eq. (4)); (ii) the factors must *not* be aggregated or weighted—the composition is learned by the intention function, and the zero-compensation dimension is appended by the framework; (iii) each factor must be computable per step from the current observation, action, and simulator info, since SARC consumes per-step latent rewards rather than episodic returns.

B.1 Prompt Template

The template is task-agnostic; the placeholders in braces are filled from each task's documentation. It comprises the *role instruction* and, a *self-prompting* step that consolidates several candidate responses into a single final one.

Prompt Template

ROLE INSTRUCTION:
You are good at understanding tasks and writing python codes. You should fully understand the provided task and the exact observation and action form. Then, based on your understanding and the goal of the task, analyze potential positive and negative behaviours or statuses that can be reflected in the observation and action, giving priority to quantities that appear in the stated success condition. Finally, write an evaluation function that returns factors evaluating the current status from different aspects.

Note:
1. Do not use information you are not given!
2. Focus on the most relevant evaluation factors and use as few of them as possible (2-5), each capturing a distinct aspect; never return two factors that are monotone transforms of the same quantity.
3. Every factor must be computed from the current step only (observation, action, info); no episode-level statistics.
4. Normalize every factor by a characteristic scale of the underlying physical quantity and clip it to [0, 1] (higher is better) or [-1, 1] (signed progress); never return unbounded raw physical quantities.
5. Do NOT sum or weight the factors into a single scalar; a downstream algorithm learns the combination. A constant zero "compensation" factor is appended automatically; do not add it yourself.
6. The code should be generic, complete and not contain omissions! Avoid dividing by zero!

Please think step by step and adhere to the following JSON format (just replace the () with your answer):
{
Understand: (your thought about the task),
Analyze: (think step by step and analyze potential positive and negative behaviours or statuses, and name each factor in 1-3 words),
Functions: (a python function of the form 'def latent_reward(obs, action, info): ... return [a list of factor values]')
}

SELF-PROMPTING:
You have generated several evaluation functions. Please summarize them and generate a new evaluation function that incorporates the most relevant evaluation factors, subject to the constraints of the role instruction. If there are other important evaluation factors, please include them as well.

After a response is produced, we run LaRe's pre-verification: the extracted function is executed on pre-collected random state–action pairs, and execution errors or range violations are returned to the model as feedback until the function is executable and within range.

B.2 Example Task Information and Responses

We give two instantiations: a low-dimensional classic-control task (MountainCar) and a high-dimensional locomotion task (Humanoid).

Task Information — *MountainCar*

TASK DESCRIPTION:
A car is placed at the bottom of a sinusoidal valley and can apply a horizontal force. The engine is too weak to climb the right hill directly; the car must first rock left to build momentum. Goal: reach the flag on the right hilltop.

SUCCESS CONDITION:
The environment returns r = 1 at a step if and only if position >= 0.45 AND velocity >= 0; otherwise r = 0. Episodes last at most 999 steps.

STATE-ACTION FORM:
The observation is 2 dimensions:
0: position: position of the car along the x-axis, range [-1.2, 0.6];
1: velocity: velocity of the car, range [-0.07, 0.07].
The action is 1 dimension. An action represents the horizontal force applied on the car, range [-1, 1].

LLM Response

Analyze:
Positive: being further right (progress toward the flag); moving fast in either direction (energy needed to escape the valley); moving right while not yet at the goal (converting stored energy into progress).
Factors: "Position Progress", "Speed", "Momentum".

Functions:
```
def latent_reward(obs, action, info):
    pos, vel = obs                  # raw physical values
    r0 = (pos + 1.2) / 1.8          # Position Progress, [0,1]
    r1 = abs(vel) / 0.07            # Speed, [0,1]
    r2 = float(vel > 0) * (1.0 - r0)  # Momentum, [0,1]
    return [r0, r1, r2]
```

With the framework-appended compensation dimension, $K = 4$: ϕ_0 = Position Progress, ϕ_1 = Speed, ϕ_2 = Momentum, ϕ_3 = Compensation, matching Figure 3 and Table A4.

Task Information — *Humanoid*

TASK DESCRIPTION:
A 17-DoF 3D bipedal humanoid must walk forward as fast as possible while staying upright and not drifting sideways, by applying torques at its hinge joints. Goal: maintain forward locomotion in a healthy, upright posture.

SUCCESS CONDITION:
The environment returns r = 1 at a step if and only if the mass-center height is in [1.1, 2.0] m AND forward velocity v_x > 1.2 m/s AND lateral speed |v_y| < 0.3 m/s; otherwise r = 0. Episodes last at most 1000 steps.

STATE-ACTION FORM:
The observation is 348 dimensions, the first 45 of which are about all the position and velocity:
0: position: z-coordinate of the torso (centre);
1: angle: w-orientation of the torso (centre);
2-4: angle: x/y/z-orientation of the torso (centre);
5-7: angle: z-angle and y-angle of the abdomen (in lower_waist), x-angle of the abdomen (in pelvis);
8-10: angle: x/z/y-coordinate of angle between pelvis and right hip (in right_thigh);
11: angle: angle between right hip and the right shin (in right_knee);
12-15: angle: the same four angles for the left leg (in left_thigh, left_knee);
16-17: angle: coordinate-1 and coordinate-2 (multi-axis) angle between torso and right arm (in right_upper_arm);
18: angle: angle between right upper arm and right_lower_arm;
19-21: angle: the same three angles for the left arm (in left_upper_arm, left_lower_arm);
22-24: velocity: x/y/z-coordinate velocity of the torso (centre);
25-27: angular velocity: x/y/z-coordinate angular velocity of the torso (centre);
28-44: angular velocity: the angular velocity of each of the joint angles 5-21, in the same order.
45-174: inertia: mass and inertia of the 13 body parts relative to the centre of mass (13 x 10);
175-252: velocity: centre-of-mass based velocity of the 13 body parts (13 x 6: velocity x, y, z then angular velocity x, y, z);
253-269: force: actuator force generated at each of the 17 joints;
270-347: force: centre-of-mass based external contact forces on the 13 body parts (13 x 6: force x, y, z then torque x, y, z).
The action is 17 dimensions. An action represents the torques applied at the hinge joints, range [-0.4, 0.4].
The info dict provides 'x_velocity' and 'y_velocity': the forward and lateral velocity of the centre of mass in m/s.

LLM Response

Analyze:

```
Positive: moving forward fast (the primary success term); keeping the torso at a healthy walking height (
      upright posture); not drifting sideways (lateral stability); spending little control effort (energy
      efficiency, a proxy for a sustainable gait).
Factors: "Forward Locomotion", "Height Stability", "Lateral Stability", "Energy Expenditure".

Functions:
def latent_reward(obs, action, info):
   h  = obs[0]                    # torso height
   vx = info["x_velocity"]        # forward COM velocity
   vy = info["y_velocity"]        # lateral COM velocity
   c  = 0.1 * np.sum(np.square(action)) # control cost, from the action
   r0 = np.clip(vx / 3.0, -1.0, 1.0)    # Forward Locomotion, [-1,1]
   r1 = np.clip(1.0 - abs(h - 1.5) / 0.5, 0.0, 1.0)  # Height Stability, [0,1]
   r2 = np.clip(1.0 - abs(vy) / 0.3, 0.0, 1.0)      # Lateral Stability, [0,1]
   r3 = np.clip(1.0 - c / 2.0, 0.0, 1.0)            # Energy Expenditure, [0,1]
   return [r0, r1, r2, r3]
```

With the framework-appended compensation dimension, $K = 5$: ϕ_0 = Forward Locomotion, ϕ_1 = Height Stability, ϕ_2 = Lateral Stability, ϕ_3 = Energy Expenditure, ϕ_4 = Compensation.

B.3 Sensitivity to the Generator LLM

Since the latent rewards are generated once before training, we ask how much they depend on the model that produced them. The Humanoid prompt above was run unchanged through seven further LLMs under the same pipeline, and the eight resulting encoders were compared on 4000 `Humanoid-v5` states, constructed by overwriting the root degrees of freedom of random rollouts so that the sample straddles every threshold of the success predicate. For each factor we identify the input quantity it reads by perturbing one quantity at a time, and we measure by least squares how well one basis reconstructs another.

Two findings follow from Table A2. First, the bases agree closely across models, with $\bar{R}^2$ between 0.70 and 0.97. Second, every model covers the core dimensions: each devotes exactly one factor to each of the three quantities named in the success condition, and no factor mixes two. The choice of model is therefore not critical for SARC.

Table A2: Humanoid latent bases produced by seven further LLMs from the identical prompt, against the released encoder (*Main*), which was generated with DeepSeek-V4-Pro. A ✓marks a factor dedicated to that quantity, and no factor of any encoder reads more than one. h: torso height; v_x, v_y: forward and lateral velocity; tilt: torso orientation; ω: torso angular velocity; $\|a\|$: control effort. K' counts informative factors, before the zero-compensation dimension the framework appends. $\bar{R}^2$ is the accuracy of linearly reconstructing the released encoder's three core factors from that model's basis, so it measures agreement on factor *shape* rather than basis quality.

		Success condition			**Discretionary**			
Generator	K'	h	v_x	v_y	tilt	ω	$\|a\|$	$\bar{R}^2$
DeepSeek-V3	5	✓	✓	✓	✓	✓		0.97
Gemini 3.1 Pro	3	✓	✓	✓				0.70
GLM-5.2	4	✓	✓	✓	✓			0.89
GPT-5.5	4	✓	✓	✓	✓			0.87
Grok 4.20	4	✓	✓	✓	✓			0.90
Kimi K3	5	✓	✓	✓	✓		✓	0.75
Claude Opus 4.8	4	✓	✓	✓		✓		0.89
Main	4	✓	✓	✓			✓	—

C Environment Details

This appendix is the authoritative definition of all task configurations and latent dimension semantics. Table A3 previews the benchmark and Figure A1 illustrates the eight tasks; Tables A4 and A5 give the latent dimensions and the sparse reward of every task.

C.1 Benchmark Overview and Environment Versions

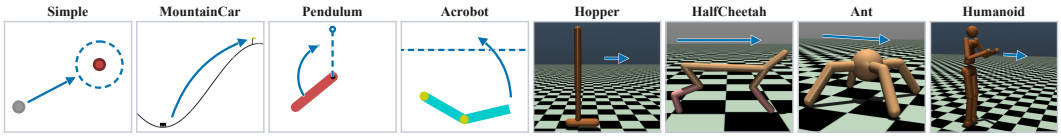

Figure A1: The eight tasks of the benchmark. Blue marks the task objective. **Simple**: a single particle must enter and stay inside a ring-shaped goal region around the landmark. **MountainCar**: an underpowered car must build momentum and crest the hill at the right-hand flag. **Pendulum**: a pendulum released from an arbitrary angle must swing up and hold the vertical, upright pose. **Acrobot**: a two-link arm actuated only at the elbow must swing its tip above the dashed target height and sustain it there. **Hopper**: a single-legged robot must hop forward without falling. **HalfCheetah**: a planar cheetah-like robot must run forward as fast as possible. **Ant**: a four-legged robot must walk forward while staying upright and on course. **Humanoid**: a bipedal humanoid must walk forward while staying upright.

Table A3: Specifications of the eight tasks. K includes the zero-compensation dimension. MaxReturn is the fixed reference used for normalized scores. Since the sparse reward is granted per step and the success predicate can in principle hold at every step, MaxReturn equals the step limit, except on MountainCar, whose episode terminates on arrival at the goal.

Task	Environment ID	Obs	Act	K	Max steps	MaxReturn
Simple	`simple` (MPE)	4	5	4	25	25
MountainCar	`MountainCarContinuous-v0`	2	1	4	999	1
Pendulum	`Pendulum-v1`	3	1	3	50	50
Acrobot	`Acrobot-v1` (continuous)	6	1	4	200	200
Hopper	`Hopper-v5`	11	3	5	1000	1000
HalfCheetah	`HalfCheetah-v5`	17	6	4	1000	1000
Ant	`Ant-v5`	105	8	6	1000	1000
Humanoid	`Humanoid-v5`	348	17	5	1000	1000

Simulator versions. Seven of the eight tasks run on Gymnasium 1.2.2 (Towers et al., 2025), with the four locomotion tasks using its MuJoCo suite on MuJoCo 3.4.0 (Todorov et al., 2012); the eighth, Simple, runs on PettingZoo 1.25.0's MPE wrapper.

C.2 Sparsification Protocol

Replacing the dense reward of standard benchmarks with a binary success signal is an established protocol for studying sparse-reward learning (Andrychowicz et al., 2017; Plappert et al., 2018); we follow it on all eight tasks with a *per-step* binary reward: at every step the environment returns $r_{\text{real}} = 1$ if the current state satisfies the task's success predicate and 0 otherwise. The success predicates are built either from each task's canonical goal, such as MountainCar's flag position or Pendulum's upright condition, or from the healthy ranges and target velocities of the Gymnasium locomotion suite (Towers et al., 2025), tightened so that standing still, creeping, and drifting sideways do not collect reward. Alongside the sparse reward, the environment exposes the K-dimensional latent reward of Appendix B; the dense shaping signal used by the Eureka baseline and the original dense reward used in Appendix I are computed from the same simulator state but are invisible to SARC. Tables A4 and A5 list the two channels every method sees under our protocol. Actions are normalized to $[-1, 1]$ and rescaled internally to the native actuator ranges.

Table A4: Reward channels on the four low-dimensional tasks. The zero-compensation dimension appended to reach the K listed is omitted here. Here, d is the distance to the landmark, $\mathbf{v}$ the velocity, and h_{tip} the tip height above the pivot ($h_{\max} = 2\,\text{m}$).

Task	Latent dimension	Formula	Sparse reward
Simple	Task Completion ϕ_0	$\text{clip}(-\lvert d^2 - 0.0475\rvert/0.025,\, -1, 0)$	$\mathbf{1}[\,0.045 < d^2 < 0.05\,]$
	Exploration Drive ϕ_1	$\text{clip}(-\lVert\mathbf{v}\rVert,\, -1, 0)$	
	State Novelty ϕ_2	$\text{clip}(-\max(0, 0.045 - d^2)/0.045,\, -1, 0)$	
MountainCar	Position Progress ϕ_0	$(\text{pos} + 1.2)/1.8$	$\mathbf{1}[\,\text{pos} \geq 0.45 \,\wedge\, \text{vel} \geq 0\,]$
	Speed ϕ_1	$\lvert\text{vel}\rvert/0.07$	
	Momentum ϕ_2	$\mathbf{1}[\text{vel} > 0] \cdot \big(1 - (\text{pos} + 1.2)/1.8\big)$	
Pendulum	Upright Angle ϕ_0	$(\cos\theta + 1)/2$	$\mathbf{1}[\,\cos\theta > 0.98 \,\wedge\, \lvert\dot\theta\rvert < 0.2\,]$
	Angular Stability ϕ_1	$1 - \lvert\dot\theta\rvert/8$	
Acrobot	Tip Height ϕ_0	$\text{clip}\big((h_{\text{tip}} + h_{\max})/(2h_{\max}),\, 0, 1\big)$	$\mathbf{1}[\,h_{\text{tip}} > 0.9\,]$
	Swing Momentum ϕ_1	$(1 - \cos\theta_1)/2$	
	Total Energy ϕ_2	$\text{clip}\big((E_{\text{pot}} + E_{\text{kin}})/4,\, 0, 1\big)$	

Table A5: Reward channels on the four MuJoCo locomotion tasks, in the format of Table A4. Here, h is the torso height and θ the torso angle; v_x and v_y are the forward and lateral velocities; $c, c_{\text{contact}} \leq 0$ are the control and contact cost terms.

Task	Latent dimension	Formula	Sparse reward
Hopper	Forward Velocity ϕ_0	$\text{clip}(v_x/2.0,\, 0, 1)$	$\mathbf{1}\begin{bmatrix} h > 0.9 \,\wedge\, \lvert\theta\rvert < 0.10 \\ \wedge\, v_x > 1.2 \end{bmatrix}$
	Height Stability ϕ_1	$\text{clip}(1 - \lvert h - 1.2\rvert/0.3,\, 0, 1)$	
	Posture Angle ϕ_2	$\text{clip}(1 - \lvert\theta\rvert/0.10,\, 0, 1)$	
	Control Efficiency ϕ_3	$\text{clip}(1 + c/2.0,\, 0, 1)$	
HalfCheetah	Running Speed ϕ_0	$\text{clip}(v_x/5.0,\, -1, 1)$	$\mathbf{1}\begin{bmatrix} v_x > 3.0 \,\wedge\, c > -0.3 \\ \wedge\, \lvert\theta\rvert < 0.35 \end{bmatrix}$
	Control Efficiency ϕ_1	$\text{clip}(1 + c/0.3,\, 0, 1)$	
	Torso Stability ϕ_2	$\text{clip}(1 - \lvert\theta\rvert/0.35,\, 0, 1)$	
Ant	Forward Velocity ϕ_0	$\text{clip}(v_x/2.5,\, -1, 1)$	$\mathbf{1}\begin{bmatrix} 0.35 \leq h \leq 0.75 \\ \wedge\, v_x > 1.5 \,\wedge\, \lvert v_y\rvert < 0.3 \end{bmatrix}$
	Lateral Stability ϕ_1	$\text{clip}(1 - \lvert v_y\rvert/0.3,\, 0, 1)$	
	Height Stability ϕ_2	$\text{clip}(1 - \lvert h - 0.55\rvert/0.2,\, 0, 1)$	
	Control Efficiency ϕ_3	$\text{clip}(1 + c/2.0,\, 0, 1)$	
	Contact Quality ϕ_4	$\text{clip}(1 + c_{\text{contact}},\, 0, 1)$	
Humanoid	Forward Locomotion ϕ_0	$\text{clip}(v_x/3.0,\, -1, 1)$	$\mathbf{1}\begin{bmatrix} 1.1 \leq h \leq 2.0 \\ \wedge\, v_x > 1.2 \,\wedge\, \lvert v_y\rvert < 0.3 \end{bmatrix}$
	Height Stability ϕ_1	$\text{clip}(1 - \lvert h - 1.5\rvert/0.5,\, 0, 1)$	
	Lateral Stability ϕ_2	$\text{clip}(1 - \lvert v_y\rvert/0.3,\, 0, 1)$	
	Energy Expenditure ϕ_3	$\text{clip}(1 + c/2.0,\, 0, 1)$	

D IMPLEMENTATION DETAILS

This appendix collects what is needed to reproduce our results: network architectures, hyperparameters, baseline implementations, and the compute environment. Every configuration reported in the paper—SARC, its variants, and all baselines—is trained with the same five random seeds, $\{1, 16, 42, 123, 2026\}$, which control network initialization, exploration noise, and environment resets; the paired start states used by the intervention protocol are drawn from a separate seed range (Appendix F.1). The accompanying code is a reference implementation of SARC and its key modules rather than the complete codebase behind every experiment reported here.

D.1 COMPLETE SARC PROCEDURE

Algorithm 1 State-Adaptive Reward Composition (SARC)

Require: Task description $\mathcal{T}$, discount factors $\gamma_{\text{low}}, \gamma_{\text{upper}}$, coefficient λ_{corr}, exploration noise σ, σ_a, policy delay d

$g \leftarrow \texttt{LLM}(\mathcal{T})$ **or** $g \leftarrow \texttt{HumanDesign}(\mathcal{T})$
Initialize policy μ_θ, intention network ϕ_ψ, critics Q^ω_{lr}, Q^ξ_{real}, and target networks
Initialize replay buffer $\mathcal{D}$, episode history $\mathcal{H}$
for each episode **do**
  Sample exploration bias $\phi_{\text{bias}} \sim \mathcal{N}(0, \sigma^2 I)$
  for each timestep t **do**
    Compute $\tilde{\phi}(s_t) = \text{softmax}(\phi_a(s_t) + \phi_{\text{bias}})$
    Execute $a_t = \mu_\theta(s_t, \tilde{\phi}(s_t)) + \epsilon_t$, $\epsilon_t \sim \mathcal{N}(0, \sigma_a^2 I)$
    Observe $s_{t+1}, r_{\text{real},t}$ and compute $\mathbf{r}_{\text{lr},t}$ by g
    Store transition in $\mathcal{D}$
    Update Q^ω_{lr} and Q^ξ_{real} by Equation (8)
    if $t \bmod d = 0$ **then**
      Update μ_θ by Equation (9)
      Update ϕ_ψ by Equation (10)
      Soft update target networks
    end if
  end for
  Update $\{\rho_i\}_{i=1}^K$ via Spearman correlation on $\mathcal{H}$
end for

D.2 SARC NETWORK ARCHITECTURE

All networks are two-hidden-layer MLPs with hidden width 128 on the classic-control tasks and 256 on the MuJoCo tasks.

- **Policy** μ_θ: input $[s; \phi]$ (state concatenated with the K-dim. intention), two ReLU hidden layers, $\tanh$ output scaled to the action range.
- **Intention network** ϕ_ψ: input s, two $\tanh$ hidden layers, linear output of K logits followed by a softmax onto Δ_K. The episode-consistent exploration bias (Eq. (7)) is added to the logits *before* the softmax.
- **Latent critic** $\mathbf{Q}^\omega_{\mathbf{lr}}$: input $[s; a]$, two ReLU hidden layers, a single vector-valued output head with K units, one per latent dimension.
- **Environmental critic** $Q^\xi_{\mathbf{real}}$: input $[s; a]$, two ReLU hidden layers, scalar output.

Every network has a target copy updated by Polyak averaging with coefficient τ.

D.3 TD3 DETAILS

TD3 mechanisms. SARC keeps TD3's three mechanisms but applies them along the intention pathway. Target policy smoothing perturbs the target intention logits before the softmax rather than

the action. The target action is then computed from the perturbed intention. The delayed update is likewise applied to the intention ϕ_ψ alone. The policy μ_θ and both critics are updated at every step.

D.4 SARC HYPERPARAMETERS

Table A6 lists all SARC hyperparameters by task group. Optimization uses Adam (Kingma & Ba, 2015). Apart from network width and buffer size, the two groups differ only in the discount factors.

Table A6: SARC hyperparameters by task group.

Hyperparameter	**Classic control**	**MuJoCo**
Upper discount γ_{upper}	0.95	0.99
Lower discount γ_{low}	0.8	0.9
Correlation weight λ_{corr}	0.1	0.1
Correlation buffer length L (episodes)	500	500
Min. episodes before first ρ_i update	50	50
ρ_i update period	every episode (EMA 0.01)	
Intention-bias std. σ (CE)	$1.0 \to 0.1$, linear over first 25%	
Action noise rate σ_a	$0.1 \to 0.01$, linear over first 25%	
Random-action prob. ϵ	$0.5 \to 0.01$, linear over first 25%	
Target smoothing	$\sigma_{\text{targ}} = 0.1$, clip 0.5	
Policy delay d	2	2
Soft update τ	0.01	0.005
Batch size	1024	1024
Learning rate (μ_θ, ϕ_ψ)	10^{-5}	10^{-5}
Learning rate (critics)	10^{-4}	10^{-4}
Replay buffer size	10^5	10^6
Hidden width	128	256

The correlation-guidance rows deserve an implementation note. The Spearman coefficients ρ_i of the correlation loss (Eq. (6)) are estimated over a sliding window of the most recent $L = 500$ episodes, using per-episode latent returns normalized by episode length. They are re-estimated after every episode once at least 50 episodes are available and smoothed with an exponential moving average of rate 0.01.

D.5 BASELINE IMPLEMENTATIONS

Provenance. All baselines are re-implemented within our codebase rather than wrapped from an external library, so that every method shares the same environment wrappers, replay buffer, evaluation protocol, and logging. The algorithmic content of each implementation follows the original papers and was cross-checked, mechanism by mechanism, against widely used reference implementations: the authors' official TD3 code (Fujimoto et al., 2018) for TD3 and all TD3-backbone methods, and Stable-Baselines3 (Raffin et al., 2021) (SB3-Contrib for TQC) for DDPG, SAC, and TQC. TQC uses the library's default quantile configuration, and SAC uses automatic temperature tuning. The intrinsic-motivation modules (RND, ICM) and the LaRe decoder follow their original papers, adapted to the TD3 backbone as described below.

Shared settings. All baselines use the same network widths, replay buffer sizes, batch size (1024), discount ($\gamma = 0.95$ classic / 0.99 MuJoCo, equal to SARC's γ_{upper}), critic learning rate (10^{-4}), evaluation protocol, and the exploration schedule of their group (Table A6) where applicable. The actor learning rate was tuned separately for the baselines and for SARC. A grid search over $\{10^{-5}, 5 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$ on the sparse tasks selected 10^{-4} for the baselines. SARC instead uses 10^{-5}, an order of magnitude smaller, because its bilevel loop requires the policy to move slowly relative to the intention. Larger values destabilize its training.

- **TD3** (Fujimoto et al., 2018): policy delay $d = 2$, target action noise 0.2 clipped at 0.5, soft update $\tau = 0.01$ (classic) / 0.005 (MuJoCo).
- **DDPG** (Lillicrap et al., 2015): single critic, no smoothing or delay; otherwise as TD3.

- **SAC** (Haarnoja et al., 2018): automatic temperature tuning with initial $\alpha = 0.2$ and temperature learning rate 10^{-4}.
- **TQC** (Kuznetsov et al., 2020): 2 quantile critics with 25 quantiles each, dropping the top 2 quantiles per critic; entropy tuning as in SAC.
- **RND** (Burda et al., 2018): TD3 backbone; a random target network and predictor (hidden width 128, embedding 64, learning rate 10^{-4}, updated every step) provide an intrinsic bonus added to the sparse reward with coefficient annealed $0.5 \to 0.01$.
- **ICM** (Pathak et al., 2017): TD3 backbone; forward and inverse models with feature dimension 64 and hidden width 128 (learning rate 10^{-4}, updated every step); intrinsic coefficient β annealed $0.2 \to 0.01$, forward-loss weight 0.2.
- **Eureka** (Ma et al., 2023): TD3 backbone trained on an LLM-designed *dense* shaping reward instead of the sparse reward. The shaping reward is written from the task description and refined over 4 evolutionary iterations of reward reflection. The sparse environment return serves as the fitness function that scores the candidates.
- **LaRe** (Qu et al., 2025): TD3 backbone. The original LaRe operates in the episodic return-decomposition setting; we adapt it to our per-step sparse setting while preserving its central mechanism—decoding a scalar proxy reward from the LLM latent encoding—as follows. LaRe receives the *same* per-step latent reward vector $\mathbf{z}_t^r = \mathbf{r}_{\text{lr}}(s_t, a_t)$ as SARC (without the compensation dimension). A reward decoder $f_\rho : \mathbb{R}^{K-1} \to \mathbb{R}$ (MLP, hidden width 64, learning rate 10^{-4}) is trained with the return-decomposition loss $\mathcal{L}_{\text{RD}} = \big(R(\tau) - \sum_t f_\rho(\mathbf{z}_t^r)\big)^2$ on minibatches of 32 stored episodes, updated once per episode, where $R(\tau)$ is the episodic sparse return. The TD3 agent is trained on the blended reward $r_t^{\text{eff}} = w\, f_\rho(\mathbf{z}_t^r) + (1 - w)\, r_{\text{real},t}$, where w ramps linearly from 0 to 1 over the first 500 decoder updates (after a minimum of 10 updates), so the proxy takes over only once the decoder fits episodic returns. This gives LaRe the same inputs, backbone, and budget as SARC, isolating the difference in *how* latent rewards are composed.

D.6 Compute and Software Environment

Hardware. All experiments were run on rented Linux cloud instances equipped with NVIDIA RTX 4090 GPUs, with multiple runs sharing each GPU. The workload is GPU-bound; host CPU and memory were not a limiting factor and varied across instances.

Software. Training used Python 3.10 and PyTorch 2.4.1 built against CUDA 12.1. The remaining stack is pinned in the `requirements.txt` shipped with the code: Gymnasium 1.2.2 (with the MuJoCo extra), MuJoCo 3.4.0, NumPy 1.26.4, SciPy 1.9.3, and pandas 2.0.3; the Simple scenario additionally requires Gym 0.26.2 and PettingZoo 1.25.0 for the legacy MPE core. All reported numbers were produced with this single environment.

Cost. Six runs share one RTX 4090, so the times below are wall-clock times under that level of contention. A run takes 0.5–2 hours on the classic-control tasks, 8–10 hours on Hopper, HalfCheetah and Ant, and 10–12 hours on Humanoid. The full study comprises roughly $1{,}200$ runs.

D.7 Code and Model Availability

The supplementary material contains the SARC implementation, the eight benchmark environments with the sparse success predicates and latent reward definitions of Appendix C, the LLM latent-reward generation and verification pipeline of Appendix B, and example trained checkpoints for the tasks. It also includes a demonstration script that rolls out a released checkpoint while plotting the intention vector $\phi(s)$ as it is recomputed at each step, so the state-adaptive composition described in Appendix F can be inspected directly. The code will be released publicly under the MIT license upon publication.

E Additional Comparative Results

The final returns of all nine methods are given in the main text (Table 1); Figure A2 complements them with the per-task learning curves. This appendix summarizes the pattern behind these results and analyzes the baseline failure modes, including the TD3–DDPG reversal noted in the main text. The pattern behind Table 1 is that SARC is the only method without a failure task, while every base-

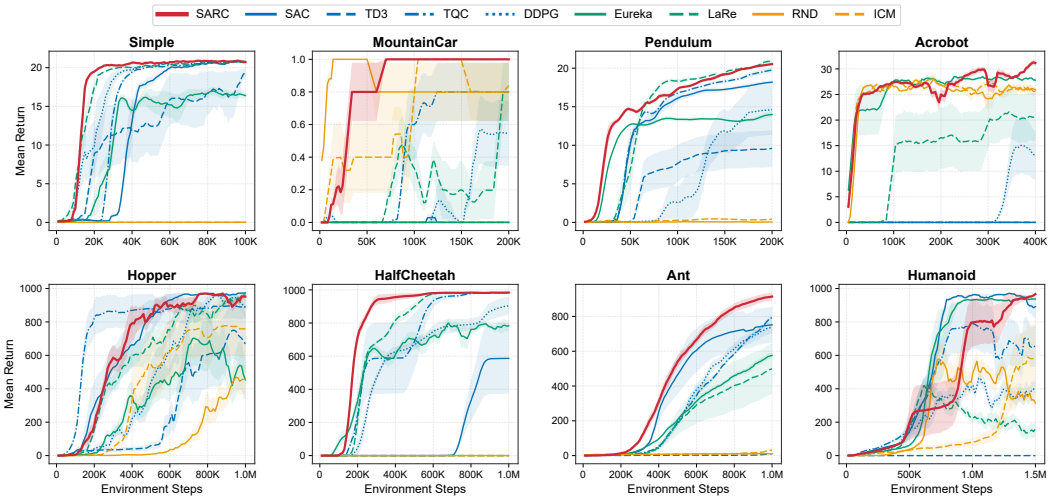

Figure A2: Average episode return for different tasks over training. Shaded regions denote the standard error of the mean (SEM) across 5 random seeds.

line collapses somewhere: SAC/TD3/TQC on the exploration-critical MountainCar and Acrobot, RND/ICM on the novelty-poor Simple and MuJoCo tasks, Eureka on tasks where its dense shaping misleads, and LaRe on the high-dimensional Humanoid task. This consistency across tasks and seeds—rather than a few strong outlier runs—is what separates SARC from the runner-up baselines.

Why TD3 trails DDPG under sparse rewards. Table 1 contains a pattern that may appear surprising: TD3 falls far below DDPG on the sparse MuJoCo tasks, although the two implementations are identical except for TD3's three stabilizers. This reversal is a documented property of pessimistic value learning under sparse rewards, not an implementation artifact. Clipped double-Q learning deliberately biases value estimates downward to counteract overestimation (Fujimoto et al., 2018). When nearly every TD target is zero, this pessimism erases precisely the optimistic value bumps around occasionally rewarded transitions that a deterministic policy needs in order to be pulled toward them—the "pessimistic underexploration" analyzed by Ciosek et al. (2019). Moreover, deterministic actor-critic can enter a deadlock under sparse rewards in which the critic's gradient field flattens and the actor stops improving even after rewarded samples enter the buffer (Matheron et al., 2020); by propagating the lower of two critics, TD3 flattens this field more aggressively than DDPG, whose single critic's overestimation bias acts as an accidental optimism bonus that repeatedly rescues it. Three observations in our results support this reading. First, the collapse disappears when rewards are dense: with the identical implementation, TD3-Dense is the best or second-best method on HalfCheetah, Ant, and Humanoid, while DDPG-Dense is the weakest on Humanoid (Table A9), so the sparse-reward gap tracks the reward regime rather than the code. Second, adding any auxiliary signal that densifies the objective revives the same TD3 backbone: RND and ICM both lift Humanoid and Acrobot off zero (Table 1). Third, SARC keeps all of TD3's stabilizers unchanged and still solves every task: the composed latent value supplies the informative learning targets that plain TD3 is starved of, so the pessimism that suppresses exploration in TD3 does no harm once latent guidance is present.

F Intervention Protocol and Extended Interpretability Results

This appendix expands the interpretability study of the main text; it does not introduce new claims but provides the protocol, the full-resolution evidence, and the robustness checks behind the results reported there. The main text validates the intention interface through three claims, and each subsection below supports one of them directly:

- **C1 (alignment—intentions align with the task's phase structure).** Supported by the cross-seed dominance analysis of Appendix F.3 (Table A7) and by the spatial/temporal intention analyses on all eight tasks in Appendix F.5.
- **C2 (causality—intentions causally steer behavior).** Supported by the intervention protocol of Appendix F.1, the full response structure of Appendix F.2, which expand the scalar η^2_ϕ and WR read-outs of panels (c) and (d) of Figure 3, and the per-dimension breakdown of Appendix F.4.
- **C3 (value—state-adaptivity is the better default).** Supported by the retention read-out of panel (e) of Figure 3 and by the full-clamping analysis of Appendix F.2, which also states what that read-out does and does not establish.

F.1 Intervention Protocol

This protocol is the source of panels (c)–(e) of Figure 3.

What the test asks. SARC's intention vector is an explanation *by construction*, in the tradition of reward-decomposition explanations (Juozapaitis et al., 2019), and our evaluation targets its *faithfulness*—whether the explanation reflects the actual decision process—in the sense of Doshi-Velez & Kim (2017). A weight vector printed next to a state proves nothing on its own: the policy might ignore it, or the semantic labels attached to the latent dimensions might not describe what the corresponding weight actually controls. Interventional, rather than purely observational, validation follows the methodology advocated for explanations in RL (Atrey et al., 2020) and mirrors test-time intervention in concept bottleneck models (Koh et al., 2020): the intention vector is the explanation interface, so we manipulate it directly and measure the behavioral consequence.

Why the intervention is well defined. The intention is not an annotation attached after the fact. The policy network receives $[s; \phi]$ as its input, so at execution time the intention is one of the two arguments from which the action is computed. Overwriting it therefore realises $\mathrm{do}(\phi)$ on the explanation interface itself while leaving the state, the environment, and all network weights untouched. If the policy had learned to ignore the intention channel, the executed behavior would be invariant under this overwrite, and every read-out below would collapse to its null value.

Which checkpoints. The pipeline reloads five trained checkpoints per task, drawn from the 5-seed runs of Appendix D. Each checkpoint contributes 100 (classic) or 50 (MuJoCo) paired start states and every condition is replayed from each of them.

Procedure. For each task and each checkpoint we execute the following steps, and all read-outs below are then averaged over checkpoints.

1. **Load.** Restore the intention network ϕ_ψ and the actor from the RQ1 checkpoint, put both in evaluation mode, and act deterministically with no exploration noise. The latent reward functions are the same generated code used during training, so the behavior metrics below are measured with the same instruments the agent was trained on.
2. **Draw paired start states.** Reset the environment with seeds $20000 + n$ for $n = 1, \ldots, N$, with $N = 100$ on the classic tasks and $N = 50$ on the MuJoCo tasks. Every condition of a given task and seed is replayed from the identical start state, which turns the comparison into a randomized-block design and removes start-state variability from every contrast we report.

3. **Roll out the conditions.** From each start state we run one full episode for each of: the natural policy $\phi = \phi_\psi(s_t)$, a uniform-intention control $\phi = \mathbf{1}/K$, and one dose intervention

$$\tilde{\phi}_t = \alpha\, e_i + (1 - \alpha)\, \phi_\psi(s_t) \tag{F.1}$$

for every non-compensation dimension i and every dose α. The classic tasks use the full grid $\alpha \in \{0.25, 0.5, 0.75, 1\}$; the MuJoCo tasks use $\alpha = 1$ only, because their episodes are 5–20$\times$ longer and the single-dose read-outs are the ones the main text reports.

4. **Execute.** At every step t we query $\phi_\psi(s_t)$, build $\tilde{\phi}_t$ by Eq. (F.1), feed $[s_t; \tilde{\phi}_t]$ to the actor, and step the environment with the resulting action. The intervention is held for the whole episode. The only exception is the counterfactual illustration in Figure 3(a), where the intervention starts at a chosen step t^* so that the natural and intervened trajectories share a common prefix and the divergence is visible.

5. **Record.** For each episode we store the episode-mean latent reward vector $Y \in \mathbb{R}^{K'}$ over the K' informative dimensions, the sparse environment return, the episode length, and whether the task was solved.

6. **Normalize.** Each metric is z-scored across all conditions and all start states of that task and seed.

Read-outs. Let $Y_{n,c,j}$ be metric j from start state n under condition c. We report four quantities.

- η^2_ϕ: the share of variance of Y explained by the intervention identity after the start-state main effect is removed. In this randomized-block design, SS_{tot} is the total sum of squares of $Y_{n,c}$ across start states n and conditions c; $\mathrm{SS}_{\text{state}}$ and SS_{int} are the portions attributable to the start-state and intervention main effects, respectively; and the remainder, $\mathrm{SS}_{\text{res}} = \mathrm{SS}_{\text{tot}} - \mathrm{SS}_{\text{state}} - \mathrm{SS}_{\text{int}}$, is residual variance. We then define

$$\eta^2_\phi = \frac{\mathrm{SS}_{\text{int}}}{\mathrm{SS}_{\text{int}} + \mathrm{SS}_{\text{res}}}, \tag{F.2}$$

averaged over metrics. The null shuffles intervention labels *within* each start state, which preserves the blocking, and stays at or below 0.02 on every task.

- The paired steering win-rate $\mathrm{WR} = P\big[Y_i \,|\, \mathrm{do}(e_i) > Y_i \,|\, \mathrm{do}(e_j)\big]$, estimated by counting, over all start states and all ordered pairs $i \neq j$, how often strengthening dimension i raises metric i above the value it takes when a different dimension is strengthened from the same start state. Chance is 0.5, and the chance level is verified with the same within-state label permutation. η^2_ϕ and WR answer two different questions: the first asks whether behavior responds to the intention at all, the second whether the response respects the labels.
- The dose–response $S_i(\alpha) = \mathbb{E}\big[Y_i(\alpha e_i)\big] - \mathbb{E}_{j \neq i}\big[Y_i(\alpha e_j)\big]$, which tests whether the policy treats the intention as a continuous priority signal rather than a discrete switch.
- Success rates and sparse-return retention under full clamping, meaning $\alpha = 1$ held for whole episodes. Retention is the clamped sparse return divided by the natural one, averaged over the one-hot targets, and it quantifies the cost of committing to a single fixed priority for an episode.

What the read-outs would show if the interface were not faithful. The three failure modes leave distinct signatures, which is what makes the test informative. If the policy ignored the intention channel, then η^2_ϕ would sit at the shuffled null, WR would sit at chance, and retention would be one on every task. If the policy used the intention but the semantic labels were arbitrary, η^2_ϕ would be large while WR stayed at chance everywhere. If state-adaptivity were cosmetic, retention would be one even where the intention visibly changes along the episode.

F.2 Intervention Response Structure

Figure A3 expands the win-rates of Figure 3(d) into full response matrices, and Figure A4 resolves the classic-task responses by dose. The matrices are diagonally dominant on most tasks and their off-diagonal signs follow the task physics. The response also grows smoothly with the dose, so the policy reads the intention as a continuous priority rather than as a switch.

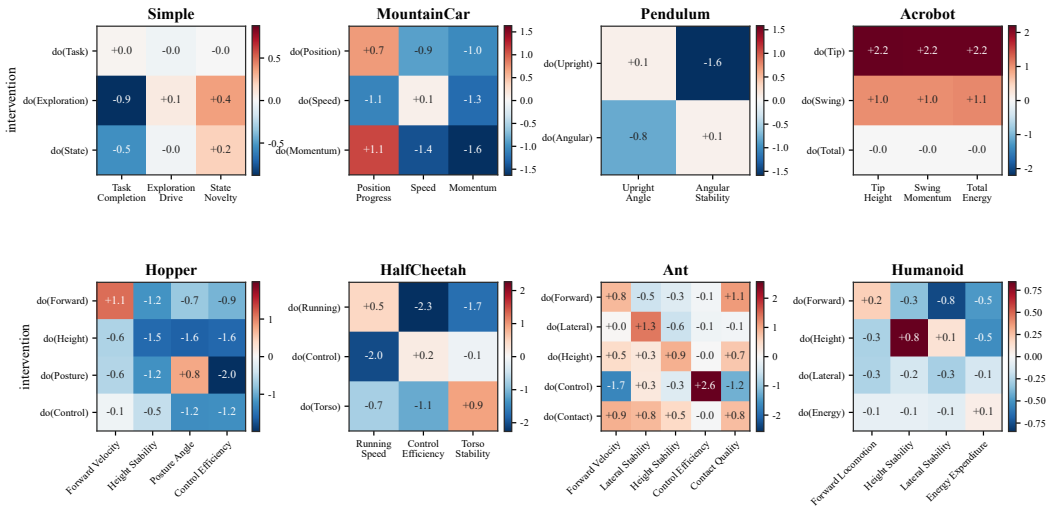

Figure A3: Intervention response matrices for all eight tasks (top row: classic tasks, 100 start states; bottom row: MuJoCo tasks, 50 start states; averaged over the training runs). Cell (i, j): z-scored change of metric j under $\mathrm{do}(e_i)$ relative to the uniform control. Color scales are per panel.

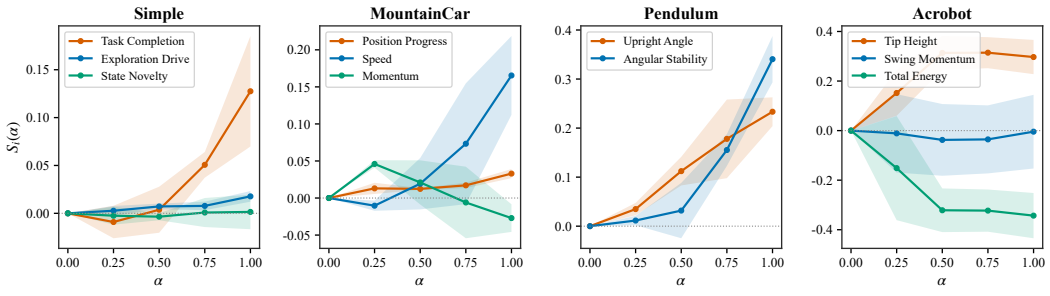

Figure A4: Dose–response curves $S_i(\alpha)$ on the classic tasks, one curve per non-compensation dimension (mean $\pm$ std over the training runs). The MuJoCo protocol uses $\alpha = 1$ only and is therefore not shown.

One exception recurs across tasks. Maintenance-type dimensions such as Hopper's *Height Stability* carry negative diagonal entries, because the natural gait already holds them near saturation and an episode-long clamp only disturbs the gait that sustains them.

Two conditions bound how the retention read-out of panel (e) should be read. It is averaged over the one-hot targets, so it mixes the cost of giving up state-dependence with the cost of committing to an antagonistic priority. It also acts on the input of an already-trained agent, so it describes the run-time role of the intention channel. It does not establish a training-time advantage over separately trained static-weight policies; the fixed-policy analysis in Appendix A does not address that comparison.

F.3 Cross-Seed Consistency of Core Dimensions

Dimension *indices* are permutation-arbitrary across training runs, so we compare a more fundamental property: which dimension dominates the state space. Table A7 reports an 80×80 grid scan of each classic task's physical state space, using the pipeline of Figure 3(a).

Table A7: Core intention dimensions across seeds (80×80 state-grid scan). Dominance % is the share of the scanned state space in which the listed dimension carries the largest weight, with the range spanning the independent training seeds; Mean act. is its mean activation over the grid.

Environment	Core dimension	Dominance % (min–max)	Mean act.
Simple	Task Completion (ϕ_0)	86.8 (75.9–100.0)	0.86
MountainCar	Speed (ϕ_1)	83.8 (67.4–100.0)	0.82
Pendulum	Upright Angle (ϕ_0)	68.7 (65.0–71.2)	0.69
Acrobot	Tip Height (ϕ_0)	99.7 (99.1–100.0)	0.99

Every seed of a task converges on the same core dimension, and in each case it is the quantity a human designer would nominate as the task's primary progress variable.

F.4 Per-Dimension Steering on Humanoid

Humanoid has the least load-bearing intention of the eight tasks at convergence. Table A8 resolves its steering by target dimension.

Table A8: Per-dimension steering on Humanoid (50 paired start states per seed, averaged over seeds). Per-dim WR restricts the win-rate to pairs whose target metric is the row dimension (chance $= 0.5$). Headroom: gap between the natural policy's metric and the best value attainable across interventions, in pooled-std units.

Dimension	Per-dim WR	Headroom (σ)
Forward Locomotion	0.65 ± 0.20	0.73
Height Stability	0.80 ± 0.18	1.10
Lateral Stability	0.45 ± 0.35	1.02
Energy Expenditure	0.66 ± 0.17	0.95

Steerability depends on the type of dimension. *Height Stability*, *Forward Locomotion*, and *Energy Expenditure* steer reliably, whereas *Lateral Stability* carries the maintenance signature above and cannot be clamped without disturbing the gait that produces it. High retention on this task indicates a small average cost of the tested one-hot clamps. It does not establish equivalence to separately trained static-weight policies. These read-outs describe converged policies: Humanoid's priorities are still moving while the gait is being acquired (Figure A5).

F.5 Intention Analyses on All Eight Tasks

This section extends the qualitative intention analyses of the main text to the MuJoCo group and to training time. All panels use trained SARC agents, and dimension labels follow Appendix C.

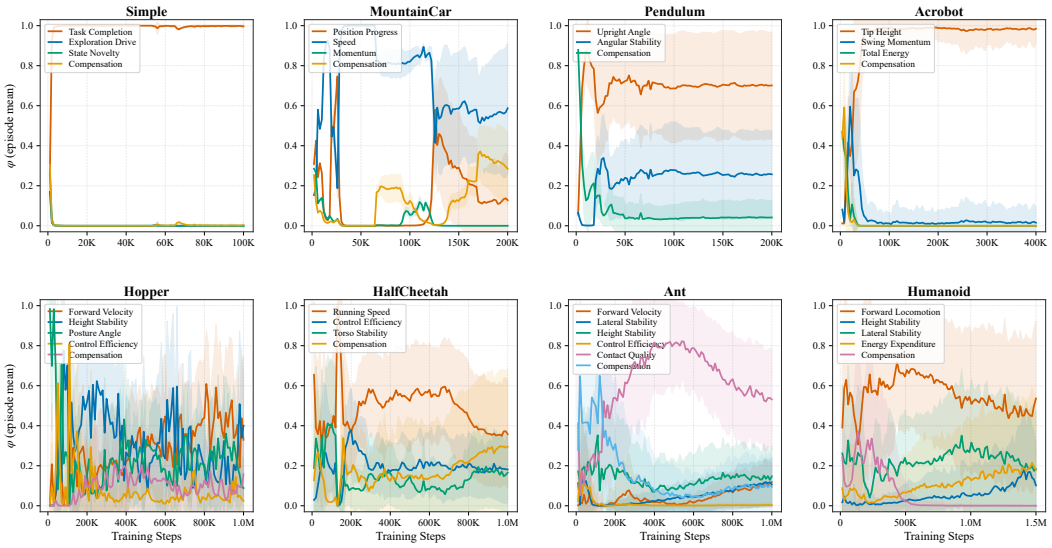

Figure A5: Evolution of the episode-mean intention weights over training for all eight tasks (shaded: within-episode std).

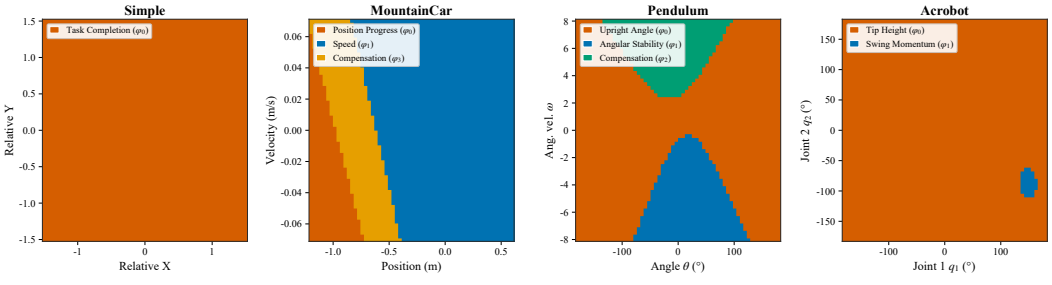

Figure A6: Dominant intention dimension over the physical state space of the four classic tasks (single representative seed; cross-seed statistics in Table A7).

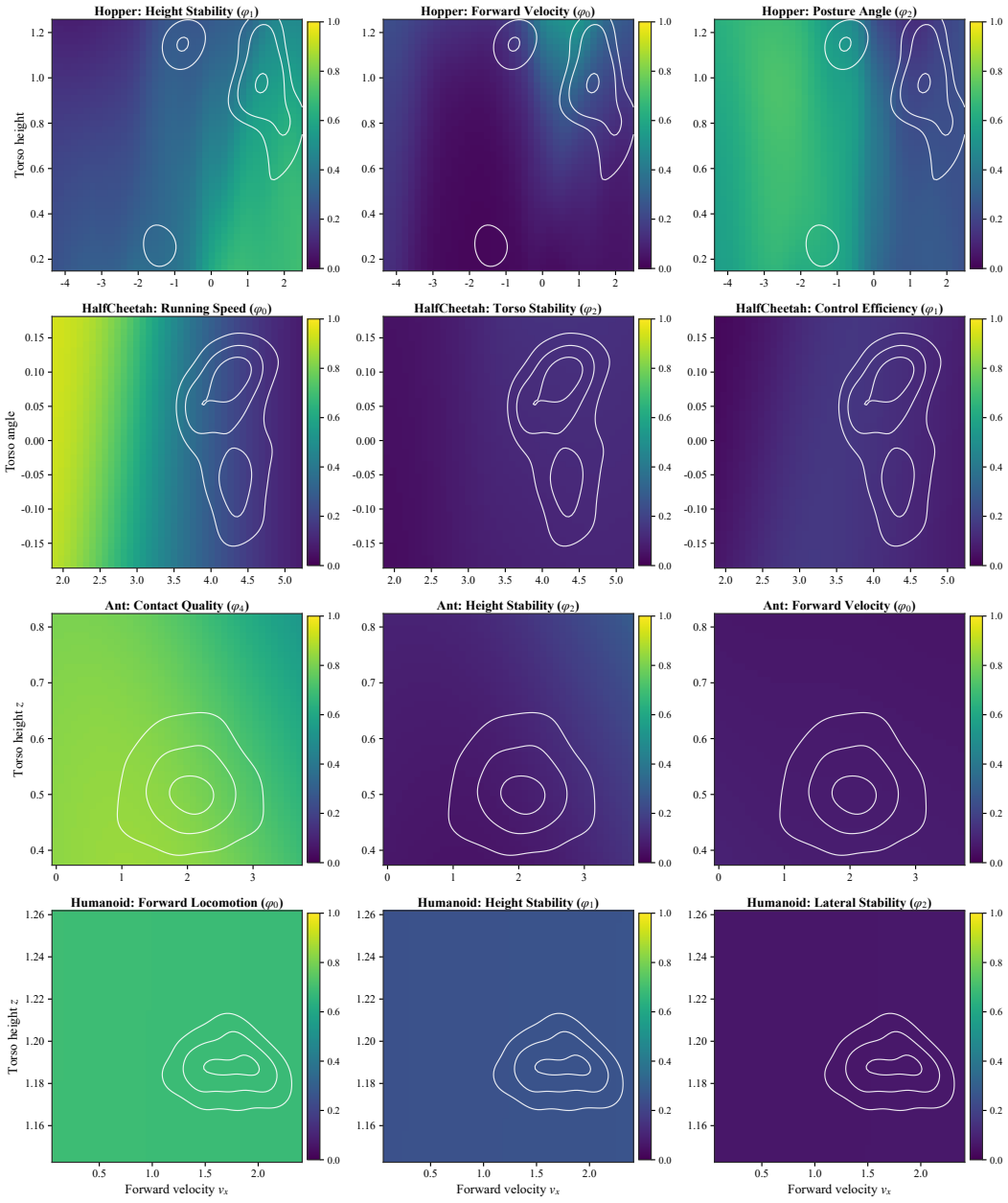

Figure A7: Per-dimension intention activation fields for the MuJoCo tasks (three representative dimensions per task), computed as partial-dependence maps over (forward velocity, torso height/angle); the remaining observation dimensions are marginalized over states visited by the trained policy. White contours mark the visited-state density.

Training-time evolution. Figure A5 traces the episode-mean intention over training. On the classic tasks a single dimension emerges early while the within-episode variability first grows and then settles, so state-adaptivity is learned rather than present at initialization. The MuJoCo intentions stay mixed, as their conjunctive success predicates require.

Spatial structure. Figure A6 colors each classic task's state space by the locally dominant dimension. The partitions follow the task physics: a progress dimension rules most of the space, and a stability or momentum dimension takes over in the region that calls for it.

High-dimensional tasks. MuJoCo state spaces cannot be colored directly, so Figure A7 shows partial-dependence maps over the two most task-relevant physical coordinates. The fields are smooth: stability dimensions activate in the low-velocity recovery situations, and speed dimensions dominate inside the visited high-speed region. The intentions therefore align with the physical phase structure on the high-dimensional tasks as well.

G Additional Ablation Results

This appendix provides the per-task learning curves behind the normalized-score summary of Figure 4. The two subsections answer how much each stabilization mechanism contributes (main paper RQ3) and how the latent dimensions should be chosen (RQ4).

G.1 Module Ablation

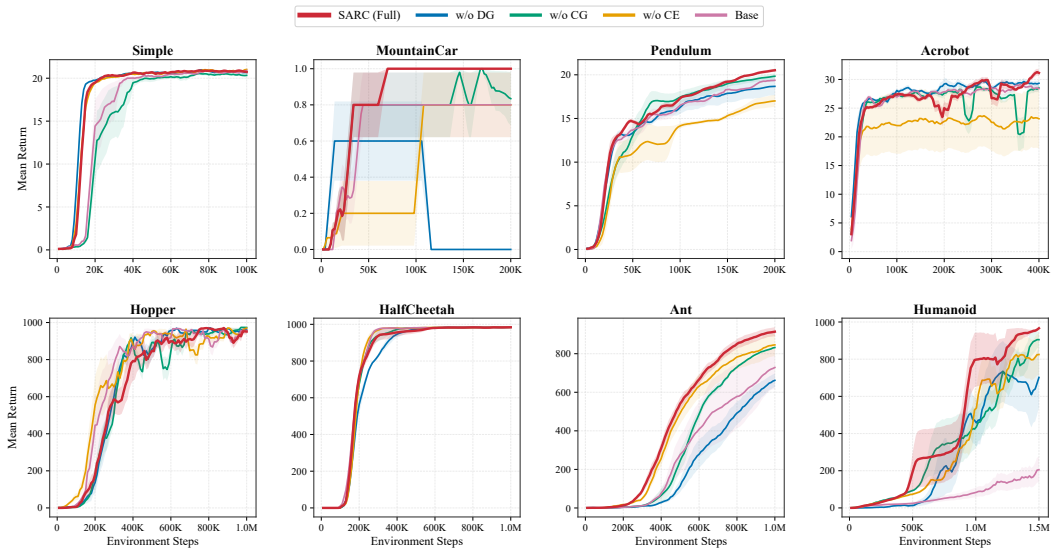

Figure A8: Module ablation: per-task learning curves for SARC (Full), w/o DG, w/o CG, w/o CE, and Base (all three mechanisms removed). Mean ± SEM over 5 seeds.

Figure A8 compares SARC (Full) against removing dual-gamma discounting (*w/o DG*: $\gamma_{\text{low}} = \gamma_{\text{upper}}$), correlation guidance (*w/o CG*: $\lambda_{\text{corr}} = 0$), episode-consistent exploration (*w/o CE*: per-step intention noise instead of a fixed episode bias), and *Base* with all three removed. The curves reproduce the ordering reported in the main text and add one qualification: how much a mechanism matters follows the temporal placement of the reward rather than the state dimension, and w/o DG does its worst damage where reward arrives only at the end of a long trajectory.

The mechanisms also interact rather than compose additively. On MountainCar, Base outperforms w/o DG, so removing correlation guidance and consistent exploration recovers a task that removing dual-gamma alone destroys. CG and CE concentrate the intention on a single dimension, which helps only when the lower level evaluates that dimension over a short horizon; dual-gamma discounting is therefore critical *given* the other two, which is how the full model is configured.

G.2 Latent-Dimension Design

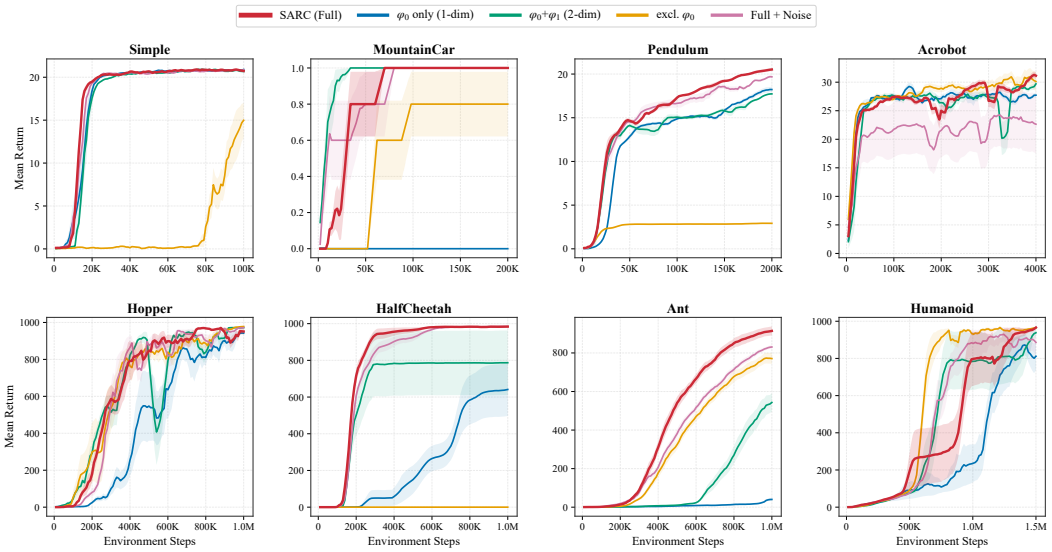

Figure A9: Latent-dimension design: per-task learning curves for SARC (Full), ϕ_0 only, ϕ_0+ϕ_1, excl. ϕ_0, and Full+Noise. *Full* here is the complete dimension set, labeled *Default* in the main paper's Figure 4. Mean ± SEM over 5 seeds.

Figure A9 varies the informative dimension set to ask which dimensions a latent reward has to cover. The curves show that what matters is coverage of the core dimension rather than the dimension count, and that which index carries the core differs by task (Table A7). Appending a pure-noise dimension changes little, because the intention function assigns it negligible weight, exactly as it does the compensation dimension. That tolerance is the robustness one wants when delegating reward design to an LLM, which tends to over-generate rather than omit.

H Sensitivity Analysis

This appendix examines the robustness of SARC to its two method-specific hyperparameters: the lower-level discount γ_{low} (dual-gamma discounting) and the correlation guidance weight λ_{corr}.

H.1 Dual-Gamma Lower Discount γ_{low}

Figure A10 sweeps γ_{low} per task group, with the w/o-DG ablation ($\gamma_{\text{low}} = \gamma_{\text{upper}}$) as the degenerate endpoint. Every setting strictly below γ_{upper} matches or beats that endpoint on the difficult tasks, and the classic tasks peak at a shorter effective horizon than the MuJoCo tasks. Theorem 1 bounds the frozen-intention drift discrepancy for a fixed policy; reducing the latent discount decreases that upper bound but increases the discount-mismatch term in Eq. (A.8) at fixed upper discount. Remark 1 shows that the bound does not order the actual discrepancy or predict the best-performing discount. The plateau around the two defaults is an empirical observation.

H.2 Correlation Guidance Weight λ_{corr}

Figure A11 sweeps the correlation guidance weight. Removing the term degrades the difficult tasks, so an anchor independent of the value estimates is needed to break the credit-assignment ambiguity of the bilevel loop. Over-weighting it pulls the intention toward globally correlated dimensions and costs some state-adaptivity. The default sits on the plateau between these two regimes.

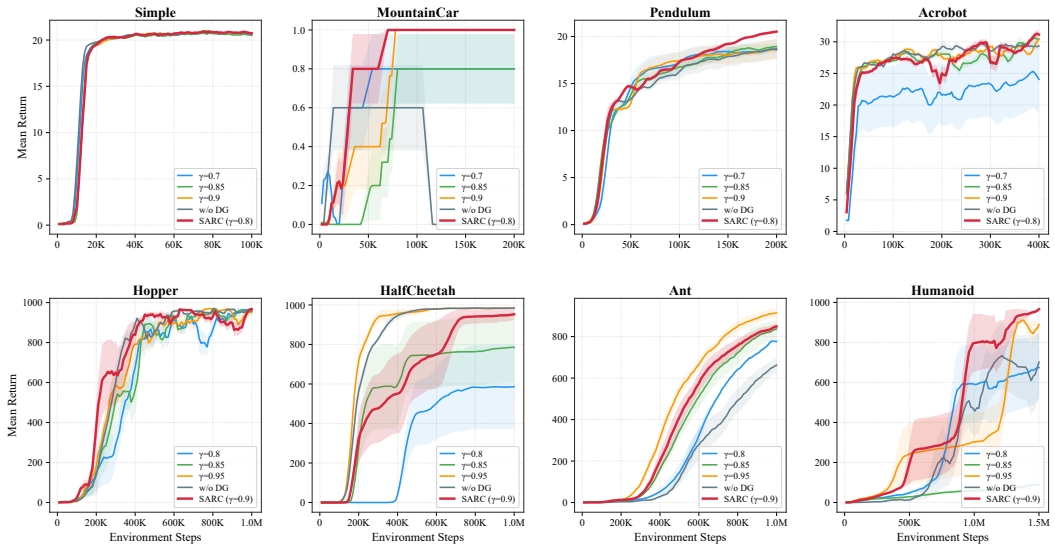

Figure A10: Sensitivity to γ_{low}. Grids differ by task group (classic: $\{0.7, 0.8, 0.85, 0.9\}$ plus w/o DG, i.e. $\gamma_{\text{low}} = \gamma_{\text{upper}} = 0.95$; MuJoCo: $\{0.8, 0.85, 0.9, 0.95\}$ plus w/o DG $= 0.99$). Mean $\pm$ SEM over 5 seeds.

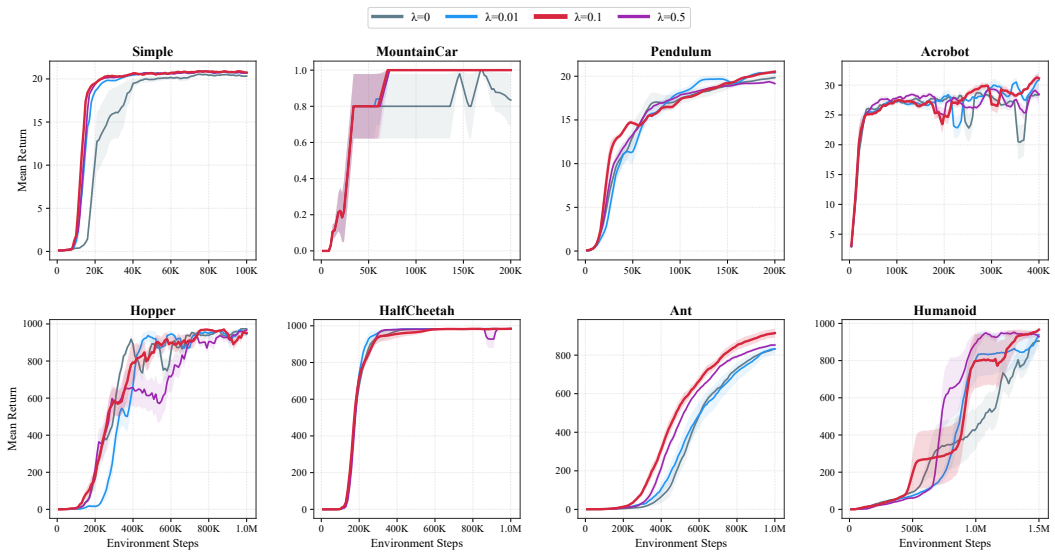

Figure A11: Sensitivity to $\lambda_{\text{corr}} \in \{0, 0.01, 0.1, 0.5\}$; $\lambda_{\text{corr}} = 0$ coincides with the w/o-CG ablation. Mean $\pm$ SEM over 5 seeds.

I BEYOND SPARSE REWARDS: CONTINUOUS-REWARD ADAPTATION

SARC is designed for sparse rewards, but nothing in the framework requires sparsity. This appendix evaluates SARC under the original *dense* rewards of the four MuJoCo tasks.

Setup. SARC-Dense differs from SARC in one respect: the environmental reward channel r_{real} carries the standard Gymnasium dense reward instead of the per-step binary success signal. SAC-Dense, TD3-Dense, TQC-Dense, and DDPG-Dense are the standard algorithms of Appendix D.5 trained on the same dense reward. Budgets, evaluation, and the 5-seed protocol match the sparse experiments; the metric is the dense episodic return.

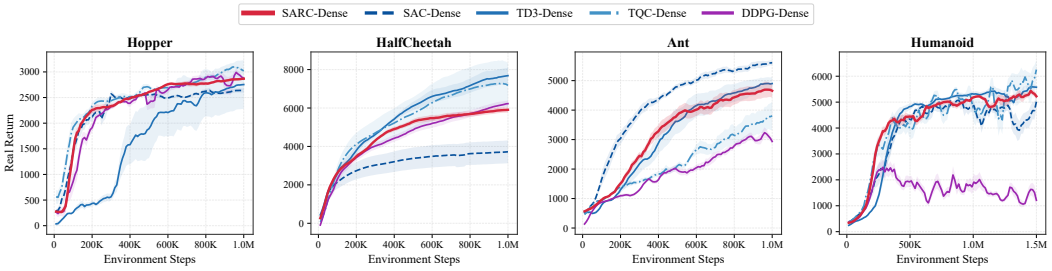

Figure A12: Dense-reward validation on the four MuJoCo tasks. Mean ± SEM over 5 seeds; the metric is the original dense return.

Table A9: Final dense return (mean ± std over 5 seeds). **Bold**: best per task.

Method	Hopper	HalfCheetah	Ant	Humanoid
SARC-Dense	2860 ± 52	5895 ± 318	4693 ± 559	5358 ± 91
SAC-Dense	2639 ± 90	3707 ± 1328	$\mathbf{5590 \pm 191}$	4673 ± 377
TD3-Dense	2745 ± 1059	$\mathbf{7653 \pm 911}$	4902 ± 477	5587 ± 174
TQC-Dense	$\mathbf{3063 \pm 36}$	7254 ± 2643	3782 ± 930	$\mathbf{5825 \pm 777}$
DDPG-Dense	2933 ± 86	6189 ± 381	3121 ± 129	1620 ± 368

Results. Figure A12 and Table A9 show that SARC-Dense is competitive on every task. An unmodified SARC also remains applicable when the reward is dense.

J EXTENDED RELATED WORK

The main paper compresses four lines of work into one paragraph. This appendix restores the citations and states, for each line, the single difference that matters for SARC.

J.1 SPARSE-REWARD RL: EXPLORATION AND UTILIZATION

Work on sparse rewards splits by which half of the problem it attacks. *Exploration* methods make a rewarding state easier to reach: intrinsic motivation supplies a surrogate signal from state novelty or pseudo-counts (Bellemare et al., 2016), from the prediction error of a learned or random target network (Pathak et al., 2017; Burda et al., 2018), or from behavioral and bisimulation-style metrics (Wang et al., 2023; 2025), while hierarchical RL (Vezhnevets et al., 2017) and curriculum learning (Bengio et al., 2009; Narvekar et al., 2020) shorten the effective horizon or reorder the tasks. *Utilization* methods extract more signal from reward once it has been observed, redistributing an episodic return over the timesteps that earned it (Arjona-Medina et al., 2019; Ren et al., 2021; Lin et al., 2024) or re-using transitions through hindsight relabeling (Andrychowicz et al., 2017) and prioritized replay (Schaul et al., 2015).

Both halves output a *scalar*, which can assert that a transition was valuable but not which aspect of the task it served, so it supports neither a semantic read-out nor a re-prioritization among competing

objectives. The two lines nevertheless remain complementary to ours: a latent reward supplies direction, an intrinsic bonus supplies coverage.

J.2 LLM-Empowered Reward Design

Hand-designed rewards require expertise and hide specification errors that surface only after training (Booth et al., 2023; Knox et al., 2023); see Yu et al. (2025) for a survey. LLMs have been enlisted as the designer along two routes. The first asks for the reward function itself as executable code, optionally refined against training feedback (Ma et al., 2023; Xie et al., 2024; Yu et al., 2023), but a usable function must commit to numerical trade-offs that a task description does not determine. The second asks the language or vision-language model for a judgment instead: similarity to a language goal (Rocamonde et al., 2024), video-language alignment (Sontakke et al., 2023), or preferences distilled into a reward model (Klissarov et al., 2023). That plays to the model's strengths, but the output is again a single number.

LaRe (Qu et al., 2025) is the proposal SARC builds on and the only one emitting a *vector*: an LLM writes an encoder scoring each transition on several named aspects, a request that fixes no cross-aspect trade-off, and a neural network then maps the vector to a scalar reward. SARC keeps the representation and replaces that network with a state-dependent distribution over the named dimensions, which is what makes the composition readable and lets correlation guidance (Eq. (6)) supply an episode-level signal where a black-box composition has nothing to fit.

J.3 Learned Reward Shaping

The closest prior work is Hu et al. (2020), who learn *when* to trust a given shaping reward: a scalar weight function gates a single shaping term, trained bilevelly with the upper level scored by the environment return. SARC generalizes that gate to a simplex-constrained vector over K named signals, optimized jointly with the policy by off-policy RL. Two of our three stabilization mechanisms exist because of the generalization: with K terms the upper level must decide which signal deserves credit (correlation guidance), and a state-dependent composition makes the lower-level reward non-stationary in a way a scalar gate does not (dual-gamma discounting). Appendix A quantifies the frozen-intention discrepancy for a fixed policy and intention map; it does not establish a regret separation from static-weight methods.

J.4 Interpretability in Reinforcement Learning

Explanations of RL policies are usually post-hoc—saliency maps, surrogate models, feature attributions—and the recurring objection is that such an explanation may be persuasive without being faithful to the computation it describes (Atrey et al., 2020; Milani et al., 2024); faithfulness in the sense of Doshi-Velez & Kim (2017) is what the intervention protocol of Appendix F.1 tests. The alternative is to build the interface into the model, as in reward-decomposition explanations (Juozapaitis et al., 2019), which split the value function along reward components, or concept bottleneck models (Koh et al., 2020), which route predictions through editable named concepts. SARC belongs to this family with one structural difference: its composition weights are learned and are an *input* to the policy rather than a summary of its output. Overwriting the intention is therefore an intervention on the mechanism, which is what makes the clamping and steering experiments possible.