DECOMPOSED CONTRASTIVE CRITICS FOR CONTINUAL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual reinforcement learning is commonly studied with dense, task-specific rewards, although many control problems are more naturally specified by goals and sparse feedback on task completion. We study this setting on a sequence of goal-conditioned manipulation tasks and ask how contrastive reinforcement learning can transfer knowledge across task boundaries. Because contrastive critics must capture both reusable goal-reaching structure and task-specific action preferences, transfer requires the former to persist across tasks while the latter remains adaptable. We introduce *Decomposed Contrastive Critics* (DCC), an algorithm that decomposes the critic into a persistent shared state-action representation and a task-specific representation re-initialized at each task boundary, coupled with a success replay buffer and an action-sensitive critic gate that deploys targeted policy supervision when the critic cannot differentiate actions. On the ten-task Continual World V2 Sawyer sequence, we compare against nine contrastive transfer baselines and two sparse goal-conditioned SAC baselines. DCC improves both continual adaptation and task success by a substantial margin, successfully solving all ten manipulation tasks in the curriculum.

1 INTRODUCTION

Continual reinforcement learning aims to sequentially adapt to new tasks via forward transfer, while simultaneously maintaining past knowledge to prevent catastrophic forgetting. To date, most existing continual RL benchmarks and algorithms are built around dense, hand-engineered rewards (Wołczyk et al., 2021; 2022; Kessler et al., 2022; Hu et al., 2025). In practice, however, we often want agents that continuously learn new independent tasks with each task specified only by a goal, with no per-step shaping signal. For example, a robot may be asked to do a sequence of independent tasks, such as reach a location, move an object, open a drawer, or place an item in a target configuration, with the desired behavior for each task specified by a goal rather than by dense rewards. If a continual learning agent requires a new reward function to be engineered for every new task, then the system has not fully solved the problem of autonomous skill acquisition.

Although sparse-reward and goal-conditioned RL methods have been studied extensively in single-task settings (Andrychowicz et al., 2017; Eysenbach et al., 2022b; Liu et al., 2025), to our knowledge they have not been studied for continual RL. This leads to an important unexplored question: *How should an agent transfer knowledge across sparse, goal-specified tasks?* We focus on forward transfer in this setting, where the evaluation target is how quickly and how well each new task is learned.

Goal-conditioned contrastive RL provides a natural starting point for this problem. Under a binary goal-reaching reward, the optimal value can be interpreted as a discounted probability of reaching the goal (Eysenbach et al., 2022b). Contrastive methods learn this quantity by training a critic to distinguish reached goals from negative goals using hindsight-relabelled experience (Andrychowicz et al., 2017; van den Oord et al., 2018; Eysenbach et al., 2022b). The resulting critic acts as an implicit goal-conditioned reward model. Instead of reading a manually shaped reward, it scores whether a state-action pair is aligned with reaching a goal. This makes contrastive GCRL especially appealing for sparse-reward continual learning.

The key challenge is how to transfer what the critic has learned about the environment across task boundaries. Resetting the critic discards reusable physical knowledge, while full parameter persistence causes representational collapse and loss of plasticity. Building on contrastive GCRL, we in-

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.

troduce *Decomposed Contrastive Critics* (DCC), a continual RL algorithm structured around three complementary components. Specifically, DCC combines (1) a decomposed state-action encoder that pairs a shared representation carried across tasks with a task-specific component re-initialized at each task boundary, (2) a success replay buffer that stores successful rollouts to provide direct policy supervision, and (3) an action-sensitive critic gate that deploys this supervision only when the contrastive critic cannot differentiate candidate actions.

We evaluate DCC on the ten-task Continual World Sawyer manipulation sequence. Empirical comparisons show that contrastive methods provide a much stronger foundation for sparse continual control than goal-conditioned SAC, which fails across multiple environments. DCC further improves upon all nine contrastive transfer baselines, achieving a 61% relative increase in curriculum AUC over the strongest baseline and reaching an average best success rate of 98%. Furthermore, DCC successfully solves the most challenging environments for contrastive RL variants in the sequence, such as `handle-press-side` where success increases from 24% in the baseline to 100%, and `window-close` where success rises from 37% to 100%. In-depth ablation experiments confirm that each component addresses a distinct failure mode of the others, ensuring steady forward transfer while preserving policy exploration on multi-stage tasks.

The primary contributions of this paper are to (1) formulate the problem of sparse-reward continual reinforcement learning where each task is specified purely by a goal, (2) design and benchmark a systematic suite of sparse continual baselines spanning goal-conditioned SAC, parameter-space adaptation, and contrastive transfer variants, (3) introduce Decomposed Contrastive Critics (DCC), an algorithm that substantially outperforms prior methods and solves all ten tasks in the manipulation sequence, and (4) provide an empirical and theoretical investigation into how action-informative states govern task difficulty in contrastive reinforcement learning.

2 Problem Setup

We formalize the sparse-reward continual reinforcement learning setting studied in this paper. An agent learns a fixed sequence of K tasks defined over a shared state space $\mathcal{S}$ and action space $\mathcal{A}$. Each task $k \in \{1, \dots, K\}$ is an episodic Markov Decision Process $\mathcal{M}^{(k)} = (\mathcal{S}, \mathcal{A}, p^{(k)}, r^{(k)}, \rho_0^{(k)}, H, \gamma)$, where $p^{(k)}(s' \mid s, a)$ denotes the transition kernel, $\rho_0^{(k)}$ the initial-state distribution, H the episode horizon, and $\gamma \in [0, 1)$ the discount factor. Each task is specified by a goal $g^{(k)} \in \mathcal{S}$ and a success region $\mathcal{G}^{(k)} \subseteq \mathcal{S}$ containing $g^{(k)}$. The environment provides only a sparse terminal reward indicating task completion, $r_t^{(k)} = \mathbf{1}\{t = H \text{ and } s_H \in \mathcal{G}^{(k)}\}$. Tasks are presented in a fixed order $\mathcal{M}^{(1)}, \dots, \mathcal{M}^{(K)}$. Across task boundaries, the agent may retain parameters, replay buffers, and learned representations accumulated on earlier tasks.

3 Methodology

3.1 Contrastive goal-conditioned reinforcement learning

Our method builds on goal-conditioned contrastive reinforcement learning (CRL) (Eysenbach et al., 2022b; Liu et al., 2025; Bortkiewicz et al., 2025). CRL represents a state-action pair (s, a) and a goal g using two encoders, ϕ and ψ, and defines their compatibility by

$$f(s, a, g) = \mathtt{sim}\big(\phi(s, a), \psi(g)\big), \tag{1}$$

where we use cosine similarity following prior work. The critic is trained with hindsight experience replay (HER) (Andrychowicz et al., 2017) and an InfoNCE objective. For a positive goal g^+ sampled from a future state in the same trajectory and negative goals $\{g_j^-\}_{j=1}^M$ sampled from the in-batch distribution,

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp\big(f(s, a, g^+)/\tau\big)}{\exp\big(f(s, a, g^+)/\tau\big) + \sum_{j=1}^M \exp\big(f(s, a, g_j^-)/\tau\big)}, \tag{2}$$

where τ is the temperature. The resulting critic score is related to future-state occupancy and provides the value-like signal used for policy improvement (Eysenbach et al., 2022b; Bastankhah et al.,

2026). Following the entropy-regularized variant used in Liu et al. (2025), we train the actor with

$$J_{\mathrm{CRL}}(\theta) = \mathbb{E}_{(s,g)\sim\mathcal{D},\, a'\sim\pi_\theta(\cdot|s,g)} \left[f(s, a', g) - \alpha \log \pi_\theta(a' \mid s, g) \right], \tag{3}$$

where α is an adaptive entropy coefficient.

3.2 Decomposed contrastive critic

We now introduce our method, *Decomposed Contrastive Critic* (DCC). DCC modifies contrastive RL in two ways. First, it decomposes the state-action critic into a shared branch and a task-specific branch. Second, it maintains two replay buffers with different roles. The standard replay buffer $\mathcal{D}$ is used for contrastive critic training with HER and for the standard contrastive actor update.

A second buffer $\mathcal{D}_{\mathrm{succ}}$ stores successful trajectories generated online while training. The critic is trained only from $\mathcal{D}$ with the contrastive objective, while the actor is trained from the contrastive objective and, when appropriate, an additional likelihood term on $\mathcal{D}_{\mathrm{succ}}$. This additional term is gated by the critic itself: it is strongest when the critic provides little action discrimination and weakest when the critic already gives a meaningful action ranking. Before the first successful trajectory is observed, $\mathcal{D}_{\mathrm{succ}}$ is empty and the regularization term contributes no update.

Critic decomposition. In continual reinforcement learning, transfer is often largely mediated by the critic (Wołczyk et al., 2022; Hu et al., 2025). This is particularly relevant to contrastive RL, where the learned contrastive representation serves both as a representation of goal-reaching structure and as a value function used for policy improvement (Eysenbach et al., 2022b; Liu et al., 2025; Bastankhah et al., 2026). In continual goal-conditioned RL, these two roles naturally correspond to two forms of knowledge. Features that capture reusable environment structure should be retained and transferred across tasks, while features that determine task-specific action preferences should remain adaptable as the task changes. Motivated by this distinction, we separate the state-action encoder into a shared branch and a task-specific branch. At task k, their representations are combined as

$$z^{(k)}_{s,a} = \phi_{\mathrm{shared}}(s, a) + \phi^{(k)}_{\mathrm{task}}(s, a), \tag{4}$$

where ϕ_{shared} is retained across tasks and $\phi^{(k)}_{\mathrm{task}}$ is re-initialised at each task boundary.

The goal representation is produced by a shared goal encoder,

$$z_g = \psi(g), \tag{5}$$

and the contrastive score at task k is

$$f^{(k)}(s, a, g) = \mathtt{sim}\Big(\phi_{\mathrm{shared}}(s, a) + \phi^{(k)}_{\mathrm{task}}(s, a), \psi(g)\Big). \tag{6}$$

In our set-up, each task is associated with only one goal during rollout. However, this does not mean that the goal encoder is trained on only one goal state. Since the critic is trained using HER, future states from the same trajectory are relabelled as goals during training. Thus, the goal encoder receives gradients from a distribution of future states, while the behaviour policy is conditioned on the single commanded goal defined by the environment. We keep the goal encoder shared across the task sequence without re-initialization at each task boundary.

Actor regularization. Contrastive RL is commonly formulated as a reward-free method (Eysenbach et al., 2022b; Liu et al., 2025; Bastankhah et al., 2026), following the information-maximization view underlying contrastive representation learning (Eysenbach et al., 2022b). In the sparse-reward setting considered here, however, the terminal reward $r^{(k)}_H \in \{0, 1\}$ provides one additional bit of task information. This bit identifies trajectories that have already completed the commanded task. DCC stores those trajectories in $\mathcal{D}_{\mathrm{succ}}$ and uses them to provide a selective policy regularization signal to the actor.

The regularization should matter only when the contrastive critic is not already informative for action selection. We therefore gate it by comparing the critic's sensitivity to the action and to the state. For a state-goal pair (s, g) from $\mathcal{D}_{\mathrm{succ}}$, we estimate action sensitivity by varying a set of probe actions while holding s and g fixed, giving $\sigma_{\mathrm{act}}(s, g) = \mathrm{Std}_j\, f^{(k)}(s, a^{(j)}, g)$, where $a^{(j)} \sim \mathrm{Unif}(\mathcal{A})$.

We estimate local state sensitivity by sampling a policy action $\tilde{a} \sim \pi_\theta(\cdot \mid s, g)$, holding $\tilde{a}$ and g fixed, and perturbing the state locally as $s^{(j)} = s + \varepsilon\widehat{\sigma} \odot \xi^{(j)}$, with $\xi^{(j)} \sim \mathcal{N}(0, I)$, yielding $\sigma_{\text{state}}(s, g) = \text{Std}_j f^{(k)}(s^{(j)}, \tilde{a}, g)$. Here, $\widehat{\sigma}$ is the coordinate-wise state standard deviation in the success minibatch.

We then define the critic gate as

$$w(s, g) = \frac{\sigma_{\text{state}}(s, g)}{\sigma_{\text{state}}(s, g) + \sigma_{\text{act}}(s, g)} \in [0, 1], \tag{7}$$

with $w = 1$ when both sensitivities vanish and gradients stopped through w; the probe settings are given in Table 2. When the critic is insensitive to action, σ_{act} is small and the gate approaches one; when the critic already separates candidate actions, the gate decreases and leaves the contrastive objective in charge. We augment the standard contrastive actor objective in Eq. 3 with a critic-gated supervised regularization term,

$$J_{\text{actor}}(\theta) = J_{\text{CRL}}(\theta) + \lambda_{\text{succ}}\mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{succ}}}\left[w(s, g) \log \pi_\theta(a \mid s, g)\right], \tag{8}$$

where λ_{succ} controls the strength of the supervised regularizer.

From the expectation-maximization view of policy search, reinforcement learning alternates between forming an improved, return-weighted distribution over behavior and fitting the policy toward that distribution (Dayan & Hinton, 1997; Peters & Schaal, 2007). In online RL, this target distribution evolves because the behavior policy determines which trajectories are collected. For the binary success setting considered here, we derive this connection formally in Appendix B. The supervised term in Eq. 8 introduces a weak bias toward actions observed in successful trajectories, increasing their probability while leaving the contrastive objective to drive goal-directed policy improvement and the entropy term to maintain policy stochasticity.

The contrastive critic is trained with HER on trajectories generated by the current behaviour policy, so shifting the policy also shifts the replay distribution seen by the critic. The supervised regularizer therefore influences critic learning indirectly by changing the distribution of future states on which the contrastive objective is trained. Algorithm 1 in Appendix A summarizes the full training procedure.

4 Experiments

Benchmark. We evaluate DCC on the ten Sawyer manipulation tasks in the Continual World V2 task order (Wołczyk et al., 2021), built from Meta-World (Yu et al., 2019). We run directly on the underlying Sawyer environments rather than using the dense-reward Continual World wrappers, consistent with the sparse-reward, goal-conditioned setting defined in Section 2. The complete task sequence, environment details, and training configuration are provided in Appendix C.

Baselines. We compare DCC against two families of sparse-reward goal-conditioned baselines. The first family is goal-conditioned SAC with hindsight experience replay (HER) (Haarnoja et al., 2018; Andrychowicz et al., 2017). We evaluate two natural continual variants. In *reset/reset* (R/R), both the actor and critic are re-initialised at each task boundary; in *persistent/persistent* (P/P), both are carried across the full task sequence. Implementation details are provided in Appendix E.1.

The second family is contrastive goal-conditioned RL. Its two direct continual extensions are again R/R and P/P, corresponding to learning each task from scratch or retaining the complete contrastive agent across tasks. We additionally consider parameter-space adaptation. A substantial line of meta-learning research encodes transferable knowledge in model parameters by learning parameter configurations that support rapid adaptation to new tasks (Finn et al., 2017; Nichol et al., 2018). In this view, transfer is performed through the parameters from which the new task is learned, whereas DCC introduces task-specific adaptation directly in the learned critic representation. To make this distinction experimentally testable, we instantiate parameter-space decomposition using Continual Knowledge Adaptation (CKA) (Hu et al., 2025). CKA retains shared parameters while composing them with task-dependent parameter directions, providing a direct comparison with DCC over where task-specific adaptation is introduced. Its algorithm and implementation details are given in Appendix E.3. The actor and critic can therefore each independently use one of three

Table 1: Performance across the ten-task Sawyer sequence. R, P, and C denote reset, persistent, and CKA parameter transfer; actor transfer is listed first. Selected task columns report AUC / best success.

	Overall		Selected tasks			
Method	AUC	Success	T4	T5	T7	T8
Sparse SAC (−1/0 penalty reward)						
SAC R/R	0.160 ± 0.003	0.420 ± 0.005	0.621/1.00	0.110/0.48	0.000/0.00	0.297/0.77
SAC P/P	0.132 ± 0.015	0.427 ± 0.011	0.218/0.98	0.109/0.48	0.000/0.00	0.313/0.78
Sparse contrastive GCRL baselines						
CRL P/P	0.390 ± 0.007	0.787 ± 0.018	0.462/1.00	0.011/0.20	0.138/0.66	0.033/0.37
CRL P/R	0.435 ± 0.002	0.823 ± 0.008	0.581/1.00	0.021/0.24	0.357/0.94	0.052/0.40
CRL R/R	0.434 ± 0.002	0.840 ± 0.003	0.473/1.00	0.022/0.24	0.412/0.98	0.055/0.37
CRL R/C	0.447 ± 0.004	0.840 ± 0.002	0.512/1.00	0.009/0.20	0.470/0.98	0.062/0.48
Decomposed contrastive GCRL ablations						
Critic-only DCC	0.479 ± 0.009	0.904 ± 0.012	0.568/1.00	0.181/0.64	0.471/0.97	0.144/0.75
Ungated DCC	0.624 ± 0.029	0.896 ± 0.019	0.218/0.80	0.664/0.99	0.105/0.43	**0.955/1.00**
Full DCC	$\mathbf{0.722 \pm 0.022}$	$\mathbf{0.979 \pm 0.014}$	0.616/0.90	**0.698/1.00**	**0.608/1.00**	0.926/1.00

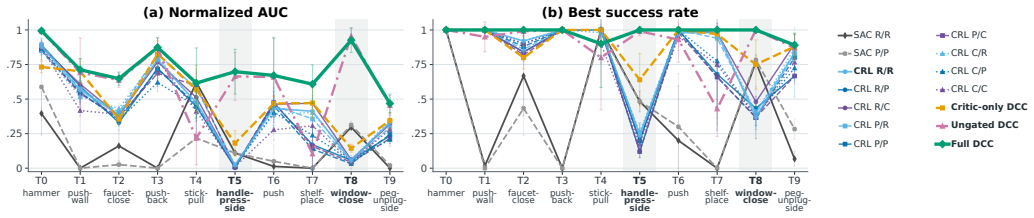

Figure 1: Per-task normalized AUC (left) and best success rate (right) across the ten-task Continual World Sawyer curriculum for the Sparse SAC baselines, the nine Contrastive RL baselines, and DCC ablation variants. Line styles indicate actor transfer, while colors denote critic transfer mode. Error bars show ± 1 standard deviation across seeds.

transfer modes: reset (R), persistent (P), or CKA-based adaptation (C), yielding a 3×3 grid of nine contrastive baselines. The complete grid is described in Appendix E.2, and Appendix H further compares parameter-space and representation-space adaptation.

To deconstruct the mechanism of DCC, we evaluate its constituent components through three ablation variants: (1) *critic-only DCC*, which isolates representation decomposition by training the policy exclusively with the contrastive objective; (2) *ungated DCC*, which introduces the success replay buffer and adds behavior cloning on successful trajectories without critic gating; and (3) *full DCC*, which combines critic decomposition with critic-gated behavior cloning. In total, our evaluation spans fourteen methods across two sparse SAC baselines, nine contrastive transfer baselines, and three DCC variants.

Evaluation protocol. All methods are evaluated across ten random seeds. For task k, seed r, and training step t, let $\mathrm{SR}_{k,r}(t) \in [0, 1]$ denote the evaluation success rate, i.e., the fraction of evaluation episodes that successfully complete task k at checkpoint t. We report two metrics: normalized area under the learning curve, $\mathrm{AUC}_{k,r} = \frac{1}{T_k} \int_0^{T_k} \mathrm{SR}_{k,r}(t)\, dt$, which captures learning efficiency over the training budget, and best success, $\mathrm{Best}_{k,r} = \max_{0 \le t \le T_k} \mathrm{SR}_{k,r}(t)$, which measures the highest evaluator success rate reached during training. For curriculum-level summaries, we average each metric across all $K = 10$ tasks per seed, and report the mean and standard error across seeds ($N = 10$). Full per-task breakdowns for all methods are documented in Appendix F.2.

4.1 Performance and Comparative Analysis

Table 1 reports curriculum-level performance across all ten manipulation tasks, measuring cumulative learning efficiency through normalized AUC and peak performance through best success rate, alongside detailed breakdowns on representative environments. Figure 1 shows the corresponding per-task AUC and best success for all fourteen methods.

DCC solves the full curriculum. Full DCC reaches a curriculum AUC of 0.72 and an average best success of 98%, compared with 0.45 and 84% for CRL R/C, the strongest contrastive baseline (Table 1). It is the only method with a best success of at least 89% on every task, and it reaches 100%

on Tasks 5 and 8, where no contrastive baseline exceeds 48%. We trace this advantage through three questions, asking (1) which value formulation suits sparse continual control, (2) what should transfer across tasks, and (3) where task-specific adaptation should enter.

Contrastive goal conditioning provides a strong, stable foundation for sparse continual reinforcement learning. The first fundamental observation is that goal-conditioned contrastive RL serves as an exceptionally capable solver in sparse continual environments where traditional value-based methods struggle. As shown in Table 1, both sparse SAC variants reach a curriculum AUC of at most 0.16, with near-zero success on `push-wall`, `push-back`, and `shelf-place`, whereas every contrastive baseline exceeds 0.37 AUC and 0.77 best success (Appendix F.2). Without dense reward shaping, temporal-difference methods must propagate credit through long chains of uninformative negative penalties. Contrastive learning sidesteps this problem by re-framing policy evaluation as future-state reachability classification rather than Bellman error minimization, so that even CRL R/R, which learns every task from scratch, reaches 84% best success.

Representation decomposition turns negative transfer into positive transfer. While contrastive learning eases the single-task credit assignment problem, transferring its representations across a sequence of tasks introduces a new dilemma. In standard contrastive RL, persisting all network weights across task boundaries yields negative transfer, causing overall AUC to drop from 0.43 down to 0.39, accompanied by feature-rank collapse and order-of-magnitude parameter-norm growth, the usual signatures of plasticity loss (Appendix I).

These observations suggest that the central challenge for contrastive transfer is not whether all parameters should be preserved or discarded wholesale, but rather where each new task should be allowed to reshape the geometry used for learning. DCC answers this question by confining task-specific change to a re-initialized branch of the critic (Eq. 4). As shown in Table 1, critic-only DCC alone raises curriculum AUC to 0.48 and best success to 90%, above every baseline.

Continual adaptation in representation space, rather than parameter space, yields effective contrastive supervision. Having established the utility of decomposing the critic, a natural architectural question arises: where should this decomposition occur? CKA adds task-specific vectors to shared weights in parameter space, whereas DCC adds a task-specific branch in the critic's representation space. This matters for contrastive critics, whose values depend on the cosine similarity between state-action and goal embeddings. When task vectors are combined in parameter space, the resulting weight shifts can act as an almost uniform translation across the state-action embedding manifold (Appendix H). To first order, such an offset shifts every candidate action's score by the same amount, leaving the action ranking, and hence the actor's gradient, unchanged.

DCC's task-specific encoder, by contrast, is a nonlinear function of both state and action and can therefore reshape the action ranking directly. As reported in Table 1, this architectural advantage is most pronounced on the hardest, transfer-sensitive environments. On Task 5 (`handle-press-side`) and Task 8 (`window-close`), critic-only DCC raises best success over the strongest CKA baseline, CRL R/C, from 20% to 64% and from 48% to 75%, respectively. Across the entire curriculum, CRL R/C barely improves on training from scratch, with an AUC of 0.45 compared with 0.43 for CRL R/R and an identical best success of 84%, consistent with the invariance argument above. Yet decomposing the critic representation alone, while improving over vanilla contrastive RL on Tasks 5 and 8, still leaves both tasks largely unsolved, a limitation we address in Section 4.2.

4.2 Action Information and Success Replay

The curriculum results in Table 1 and Figure 2 reveal a complementary interplay among the three components of DCC. First, critic-only DCC leaves Task 5 (`handle-press-side`) and Task 8 (`window-close`) largely unsolved despite improving transfer elsewhere. Second, adding a success replay buffer with an ungated behavior cloning update directly resolves Tasks 5 and 8, but induces severe performance drops on exploratory, multi-stage environments such as Task 4 (`stick-pull`) and Task 7 (`shelf-place`). Finally, integrating our critic gate in full DCC reconciles these two mechanisms, recovering performance on Tasks 4 and 7 while keeping Tasks 5 and 8 solved, and matching or exceeding the AUC of critic-only DCC on every task.

In this section, we examine why contrastive critics fail on specific manipulation tasks, and how the interplay between success replay and action-sensitive gating resolves these failure modes. Figure 2 visualizes the per-task AUC across these ablation variants, and Figure 3 details our empirical measurements of action sensitivity.

A contrastive critic must simultaneously solve two distinct problems: predicting state reachability and distinguishing actions. The difficulty observed on Tasks 5 and 8 is particularly intriguing because these tasks are not intrinsically hard, since sparse SAC, the weakest method overall, outperforms every contrastive baseline on both. What, then, makes a task hard for a contrastive critic specifically? A contrastive critic is trained to answer a prediction problem. Given a state-action pair, it should assign a high score to states that are likely to occur in its future. The actor uses the same score for a different purpose. At a fixed state and goal, it needs the critic to tell different candidate actions apart. These two requirements are closely related, yet they place fundamentally different demands on the critic. This distinction becomes clear from the density ratio that an ideal contrastive critic represents,

$$\log \frac{p(g \mid s, a)}{p(g)} = \log \frac{p(g \mid s)}{p(g)} + \log \frac{p(g \mid s, a)}{p(g \mid s)}. \tag{9}$$

The first term measures how much the current state already predicts the goal, independent of the action. The second term captures how much additional information is provided by the action once the state is known. For policy improvement, only this second term can distinguish one action from another at the same state. If the critic assigns nearly identical values to different actions at a fixed state, the actor receives no directional gradient, even if the critic predicts goal-reaching with high overall accuracy.

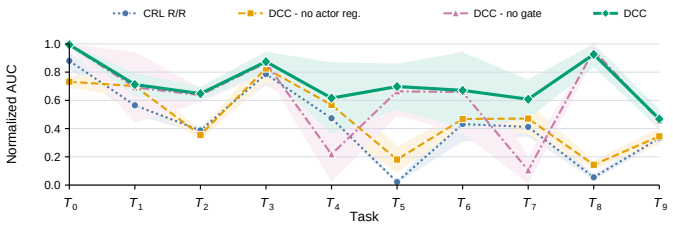

Figure 2: Per-task AUC ablation of DCC. We compare Contrastive RL with reset actor and critic, critic-only DCC, ungated DCC, and full DCC. Lines show means and ribbons show one standard deviation across seeds. Complete ablation and baseline results are reported in Appendix F.2.

Task difficulty for contrastive RL depends on the prevalence of action-informative states rather than horizon length. Conventional intuition might suggest that tasks requiring prolonged multi-stage manipulation, such as `stick-pull` (Task 4) and `shelf-place` (Task 7), would be the most difficult in sparse-reward settings, while tasks involving a single contact event, like `handle-press-side` (Task 5) and `window-close` (Task 8), should be straightforward. In our results, this intuition holds for sparse SAC, but the exact opposite holds for contrastive RL. Sparse SAC never solves `shelf-place` yet reaches 48% and 77% best success on the two contact tasks, whereas CRL R/R reaches 98% on `shelf-place` but only 24% and 37% on Tasks 5 and 8. Informally, a state is action-informative when changing the action meaningfully changes the distribution over future outcomes, a notion we formalize in Appendix G. If changing the action changes the future throughout a large portion of the state distribution, then the critic repeatedly sees examples in which understanding the action is necessary for predicting what happens next.

Continuous pushing is a simple example. Once the hand is in contact, the action moves the object, and the object's position is exactly what distinguishes one future from another. In `push` (Task 6), the hand is in contact with the object for 86% of the episode (Figure 3A), so most transitions require the critic to model the action. The InfoNCE objective is therefore forced to learn fine-grained action rankings, and the critic stays action-sensitive throughout the episode (Figure 3B).

In other tasks, the outcome may depend on the action only during a brief contact window, such as a press, and much less on the actions taken before or after it. When such states occupy little probability mass in replay, the critic can predict future states accurately while seeing few examples that force it to distinguish the actions that matter for control.

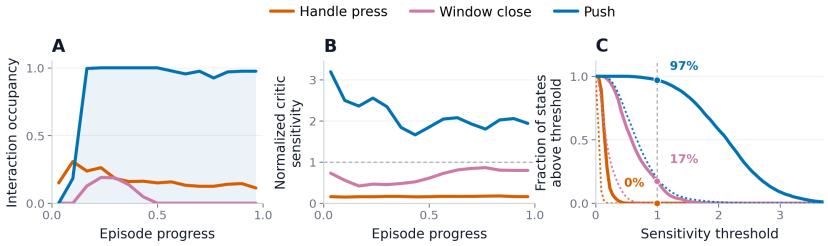

Figure 3: Action sensitivity and interaction occupancy on Tasks 5, 8, and 6. **A.** Interaction-region occupancy. **B.** Normalized critic action sensitivity. **C.** Fraction of states above sensitivity threshold ϵ. Task 6 shows sustained interaction and action sensitivity, whereas on Tasks 5 and 8 both are rare.

Our empirical analysis on Tasks 5 and 8 is consistent with this picture. On these tasks, the hand occupies the interaction region for only 17% and 5% of the episode (Figure 3A). Almost no states on Task 5 and fewer than 20% on Task 8 are action-sensitive, versus 97% on Task 6 (Figure 3C). The critic accurately recognizes when the handle is pressed downward, yet evaluates pressing actions and hovering actions with nearly identical scores. The policy therefore reaches the handle but gets no gradient to press it, and diagnostic probes localize this gap to the contact boundary (Appendix G.7 and Table 7).

Replaying verified task completions provides direct policy supervision where contrastive action gradients vanish. When an exploratory rollout occasionally reaches the goal, the environment provides a positive terminal reward indicating task completion. Rather than forcing the contrastive objective to resolve fine-grained action differences from the narrow fraction of contact transitions in replay, a direct remedy is to store these successful rollouts in a separate success replay buffer $\mathcal{D}_{\text{succ}}$ and apply behavior cloning on them. This motivates the second component of our algorithm. Adding this update to critic-only DCC defines our *ungated DCC* variant. *Ungated DCC* resolves both tasks. Relative to critic-only DCC, best success rises from 64% to 99% on Task 5 and from 75% to 100% on Task 8. The larger change is in the normalized AUC, which rises from 0.18 to 0.66 and from 0.14 to 0.96, showing that success replay mainly makes success persist once it is found.

Critic gating is necessary to prevent premature policy convergence on exploratory tasks. Unconditional behavior cloning, however, can harm tasks that require multi-stage coordination, such as Task 4 (`stick-pull`) and Task 7 (`shelf-place`). Relative to critic-only DCC, ungated DCC lowers AUC from 0.57 to 0.22 on Task 4 and from 0.47 to 0.11 on Task 7, where best success falls from 97% to 43% (Table 1 and Figure 2). We attribute this to the exploratory demands of multi-stage manipulation. In `shelf-place`, for example, the robot must grasp an object, transport it through clutter, and position it stably on an elevated surface. Early successes on such tasks are often inefficient, erratic, and fragile, and cloning them unconditionally concentrates the policy around these suboptimal traces before the contrastive objective can refine them.

In principle, behavior cloning should be deployed if and only if two operational criteria are met. First, the agent must have discovered at least one successful trajectory, ensuring that the success buffer contains valid experience. Second, the critic must exhibit poor action discrimination at the current state, failing to provide directional gradients despite the availability of goal-reaching rollouts. The critic gate (Eq. 7) applies the second criterion to states from successful trajectories, comparing how much the critic's score varies across candidate actions with how much it varies under small perturbations of the state, so that behavior cloning receives a weight near one where the critic cannot rank actions and near zero where it already does. With the gate, AUC on Tasks 4 and 7 rises to 0.62 and 0.61, above critic-only DCC, while the gains on the contact-sparse tasks are retained almost entirely. The same quantity, how much the critic's score depends on the action, thus explains where contrastive RL fails, why success replay repairs it, and where that repair must be withheld.

5 Related Work

A common approach in continual learning is to combine shared parameters with task-specific components. Prior methods use separate columns, pruned subnetworks, attention masks, low-rank adapters, or regularisation around earlier solutions (Rusu et al., 2016; Mallya & Lazebnik, 2018;

Serrà et al., 2018; Kirkpatrick et al., 2017; Liang & Li, 2024; Zhou et al., 2024). DCC applies this idea inside a contrastive critic: its state-action encoder combines a shared representation with a task-specific residual that is reinitialised at each task boundary. This design is also related to work on plasticity loss in RL, where resets and selective reinitialisation are used to restore adaptability (Nikishin et al., 2022; Sokar et al., 2023; Elsayed & Mahmood, 2024; Farias & Jozefiak, 2025).

Value functions can share knowledge across goals, rewards, and tasks. Universal value-function approximators, successor features, and zero-shot value-learning methods learn reusable representations for related control problems (Schaul et al., 2015; Barreto et al., 2017; Touati & Ollivier, 2021; Park et al., 2024b;a). Contrastive goal-conditioned RL instead learns a goal-reaching critic by classifying future states in hindsight-relabelled trajectories (Andrychowicz et al., 2017; van den Oord et al., 2018; Eysenbach et al., 2022b; Liu et al., 2025; Bastankhah et al., 2026). This supplies a learning signal without dense reward shaping, making it a natural starting point for sparse-reward continual RL.

Behavioral supervision has also been combined with contrastive RL. Offline contrastive RL adds a goal-conditioned behavioral-cloning term to keep the actor close to actions in the dataset (Zheng et al., 2024). More generally, self-imitation methods reuse the agent's own good or successful trajectories (Oh et al., 2018; Li et al., 2022), although imitating achieved outcomes without correction can preserve suboptimal behavior (Eysenbach et al., 2022a). Selective supervision has also been studied in demonstration-based RL. Nair et al. (2018) maintain a separate demonstration buffer and apply behavioral cloning only when the critic judges the demonstrated action to be better than the policy action.

Our finding that task difficulty is determined by the availability of action-informative states, rather than horizon length alone, provides a complementary view of recent work on action chunking (Li et al., 2025; Yang et al., 2026; Korniak et al., 2026). Action chunking extends the action horizon, making the influence of decisions on future states more visible to the critic. Our analysis considers the same problem from the state-distribution side by asking where, and how often, actions meaningfully change the future outcome. Both views suggest that effective control depends on exposing the critic to enough information about the consequences of actions.

Sparse-reward continual contrastive RL is both an algorithmic problem and a representation-learning problem. As task sequences grow, agents need enough capacity to acquire new skills, reuse previous knowledge, and generalise across related goals (Khetarpal et al., 2022). Recent work shows that contrastive GCRL can benefit from larger representations and scalable implementations, and goal-conditioned benchmarks now extend beyond manipulation to navigation, locomotion, and long-horizon control (Wang et al., 2025; Bortkiewicz et al., 2025; Park et al., 2025; Radosavovic et al., 2024; Park et al., 2024b).

6 Conclusion

In this paper we formulated sparse-reward continual RL, where each task is specified by a goal and new tasks are learned by transferring knowledge from previous tasks. We introduced DCC, which combines a decomposed contrastive critic with selective policy regularization from successful experience. On the ten-task Continual World V2 Sawyer benchmark, DCC substantially improves learning efficiency over sparse SAC and contrastive transfer baselines while solving all ten tasks. Overall, our results show that DCC is a promising approach for sparse-reward continual reinforcement learning.

AI USE STATEMENT

In preparing this paper, we used generative AI tools to discuss mathematical ideas, identify relevant literature, discuss implementation details and experimental diagnostics, improve the writing, format the manuscript in LaTeX, and assist in producing figures from our experimental results. The authors reviewed AI-assisted mathematical suggestions, checked literature references against the cited papers and implementation details against the code, and verified the figures against the underlying results. The authors take responsibility for the final text, claims, analyses, and figures.

ETHICS STATEMENT

This work evaluates reinforcement-learning methods on a simulated robot manipulation benchmark. The experiments involve no human participants, personal data, or physical robots.

REPRODUCIBILITY STATEMENT

The appendix provides training pseudocode (Appendix A), the benchmark and evaluation protocol (Appendix C), architecture and optimization settings (Appendix D), baseline definitions (Appendix E), complete per-task results (Appendix F), and representation metrics (Appendix I).

REFERENCES

George Andriopoulos, Zixuan Dong, Li Guo, Zifan Zhao, and Keith W. Ross. The prevalence of neural collapse in neural multivariate regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Mahsa Bastankhah, Grace Liu, Dilip Arumugam, Thomas L. Griffiths, and Benjamin Eysenbach. Demystifying the mechanisms behind emergent exploration in goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2026. Poster.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned reinforcement learning algorithms and research. In *International Conference on Learning Representations (ICLR)*, 2025. Spotlight.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Peter Dayan and Geoffrey E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997. doi: 10.1162/neco.1997.9.2.271.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Mohamed Elsayed and A. Rupam Mahmood. Addressing loss of plasticity and catastrophic forgetting in continual learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Benjamin Eysenbach, Soumith Udatha, Sergey Levine, and Ruslan Salakhutdinov. Imitating past successes can be very suboptimal. In *Advances in Neural Information Processing Systems*, volume 35, 2022a. URL `https://arxiv.org/abs/2206.03378`.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Vivek F. Farias and Adam Daniel Jozefiak. Self-normalized resets for plasticity in continual learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 2017.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

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 (ICML)*, 2018.

Tom Hennigan, Trevor Cai, Tamara Norman, Lena Martens, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL `http://github.com/deepmind/dm-haiku`.

Jinwu Hu, Zihao Lian, Zhiquan Wen, Chenghao Li, Guohao Chen, Xutao Wen, Bin Xiao, and Mingkui Tan. Continual knowledge adaptation for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J. Roberts. Same state, different task: Continual reinforcement learning without interference. In *AAAI Conference on Artificial Intelligence*, 2022.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017.

Michal Korniak, Kamil Dybek, Benjamin Eysenbach, Marco Bagatella, and Michał Bortkiewicz. Three steps at a time: Learning representations from action sequences in contrastive rl, 2026.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. doi: 10.52202/085713-1856. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/50348e8f9aef984abe0ea1ec2a326f78-Abstract-Conference.html.

Yunfei Li, Tian Gao, Jiaqi Yang, Huazhe Xu, and Yi Wu. Phasic self-imitative reduction for sparse-reward goal-conditioned reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12765–12781, 2022. URL https://proceedings.mlr.press/v162/li22g.html.

Yan-Shuo Liang and Wu-Jun Li. InfLoRA: Interference-free low-rank adaptation for continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive RL without rewards, demonstrations, or subgoals. In *International Conference on Learning Representations (ICLR)*, 2025.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pp. 23190–23211. PMLR, 2023.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan Pascanu, and Caglar Gulcehre. No representation, no trust: Connecting representation, collapse, and trust issues in PPO. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299, 2018. doi: 10.1109/ICRA.2018.8463162. URL https://arxiv.org/abs/1709.10089.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3878–3887, 2018. URL https://proceedings.mlr.press/v80/oh18b.html.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *International Conference on Machine Learning (ICML)*, 2024a.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *International Conference on Learning Representations (ICLR)*, 2024b.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 745–750, 2007. doi: 10.1145/1273496.1273590.

Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint*, 2016.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning (ICML)*, 2015.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 2018.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzciński, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. Best Paper Award.

Maciej Wołczyk, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Maciej Wołczyk, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Disentangling transfer in continual reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jiarui Yang, Bin Zhu, Jingjing Chen, and Yu-Gang Jiang. Actor-critic for continuous action chunks: A reinforcement learning framework for long-horizon robotic manipulation with sparse reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18692–18700, 2026. doi: 10.1609/aaai.v40i22.38937. URL https://ojs.aaai.org/index.php/AAAI/article/view/38937.

Yuzhe Yang, Guo Zhang, Zhi Xu, and Dina Katabi. Harnessing structures for value-based planning and reinforcement learning. In *International Conference on Learning Representations*, 2020.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Chongyi Zheng, Benjamin Eysenbach, Homer Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive RL: Techniques for robotic goal reaching from offline data. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2306.03346.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. In *International Joint Conference on Artificial Intelligence (IJCAI) Survey Track*, pp. 8363–8371, 2024.

A TRAINING ALGORITHM

Algorithm 1 gives the full training procedure for Decomposed Contrastive Critics. At each task boundary, the task-specific encoder, actor, standard replay buffer, and success buffer are re-initialized, while the shared state-action encoder and goal encoder are carried forward across the task sequence. The success buffer is inactive until the current policy generates a trajectory satisfying the environment success predicate.

Algorithm 1 Decomposed Contrastive Critics with success replay and critic gate

Input: tasks $\mathcal{M}^{(0:K-1)}$ and $T_k, \lambda_{\text{succ}}, N_a, N_s, \varepsilon$
Init: shared state-action encoder ϕ_{shared} and goal encoder ψ
for $k = 0, \dots, K-1$ **do**
  Reset $\phi_{\text{task}}^{(k)}$, π_θ, $\mathcal{D}_k$, and $\mathcal{D}_{\text{succ}}$
  while $\text{env_steps}_k < T_k$ **do**
    Collect $\zeta = \{(s_t, a_t, r_t)\}_{t=0}^{H-1}$ using $\pi_\theta(\cdot \mid s_t, g^{(k)})$
    $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \zeta$
    if $r_H^{(k)} = 1$ **then**
      Insert $\{(s_t, a_t, g^{(k)})\}_{t=0}^{H-1}$ into the ring buffer $\mathcal{D}_{\text{succ}}$
    end if
    for $i = 1, \dots, N_{\text{sgd}}$ **do**
      Sample a hindsight-relabeled batch $\mathcal{B} \sim \mathcal{D}_k$
      $z_{s,a}^{(k)} \leftarrow \phi_{\text{shared}}(s, a) + \phi_{\text{task}}^{(k)}(s, a)$; $z_g \leftarrow \psi(g)$
      Minimize $\mathcal{L}_{\text{InfoNCE}}$ in Eq. 2 over the critic encoders
      Compute $J_{\text{CRL}}(\theta)$ on $\mathcal{B}$
      if $\mathcal{D}_{\text{succ}} \neq \varnothing$ **then**
        Sample $\mathcal{B}_{\text{succ}} \sim \text{Unif}(\mathcal{D}_{\text{succ}})$
        $\widehat{\sigma} \leftarrow \text{CoordStd}(\{s : (s, a, g^{(k)}) \in \mathcal{B}_{\text{succ}}\})$
        for all $(s, a, g^{(k)}) \in \mathcal{B}_{\text{succ}}$ **do**
          Sample $a^{(1:N_a)} \overset{\text{iid}}{\sim} \text{Unif}(\mathcal{A})$
          $\sigma_{\text{act}}(s, g^{(k)}) \leftarrow \text{Std}_j\big[f^{(k)}(s, a^{(j)}, g^{(k)})\big]$
          Sample $\tilde{a} \sim \pi_\theta(\cdot \mid s, g^{(k)})$ and $\xi^{(1:N_s)} \overset{\text{iid}}{\sim} \mathcal{N}(0, I)$
          $\sigma_{\text{state}}(s, g^{(k)}) \leftarrow \text{Std}_j\big[f^{(k)}(s + \varepsilon\widehat{\sigma} \odot \xi^{(j)}, \tilde{a}, g^{(k)})\big]$
          $w(s, g^{(k)}) \leftarrow \text{sg}[\sigma_{\text{state}}/(\sigma_{\text{state}} + \sigma_{\text{act}})]$
        end for
        $J_{\text{succ}}(\theta) \leftarrow |\mathcal{B}_{\text{succ}}|^{-1} \sum_{(s,a,g^{(k)}) \in \mathcal{B}_{\text{succ}}} w(s, g^{(k)}) \log \pi_\theta(a \mid s, g^{(k)})$
      else
        $J_{\text{succ}}(\theta) \leftarrow 0$
      end if
      Update θ by ascending $J_{\text{CRL}}(\theta) + \lambda_{\text{succ}} J_{\text{succ}}(\theta)$
    end for
  end while
  Carry forward ϕ_{shared} and ψ
end for

B AN EXPECTATION-MAXIMIZATION VIEW OF BEHAVIOR CLONING ON SUCCESSFUL TRAJECTORIES

The actor regularizer in Eq. 8 has a simple interpretation through the expectation-maximization view of policy search (Dayan & Hinton, 1997; Peters & Schaal, 2007). When return is binary, the return-weighted trajectory distribution reduces exactly to the distribution of successful trajectories generated by the policy itself.

B.1 Binary success defines an improved trajectory distribution

Fix a commanded goal g, and let $r_H(\zeta, g) \in \{0, 1\}$ indicate whether trajectory ζ successfully achieves that goal. Under policy π_θ, define the probability of success as

$$Z_\theta(g) = \mathbb{E}_{\zeta \sim p_\theta(\zeta|g)} \left[r_H(\zeta, g) \right] = P_\theta(r_H = 1 \mid g). \tag{10}$$

Consider a policy π_{θ_n} with $Z_{\theta_n}(g) > 0$. The return-weighted trajectory distribution used by EM policy search is

$$q_n(\zeta \mid g) = \frac{r_H(\zeta, g)\, p_{\theta_n}(\zeta \mid g)}{Z_{\theta_n}(g)}. \tag{11}$$

Since $r_H(\zeta, g)$ is binary,

$$q_n(\zeta \mid g) = p_{\theta_n}(\zeta \mid g, r_H = 1). \tag{12}$$

The improved distribution is therefore exactly the success-conditioned trajectory distribution of the current policy.

B.2 Fitting successful trajectories improves the success objective

For any distribution q supported on successful trajectories, define

$$\mathcal{F}(\theta; q) = \mathbb{E}_{\zeta \sim q} \left[\log p_\theta(\zeta \mid g) - \log q(\zeta) \right]. \tag{13}$$

Since $p_\theta(\zeta \mid g, r_H = 1) = r_H(\zeta, g)\, p_\theta(\zeta \mid g)/Z_\theta(g)$ and $r_H = 1$ on the support of q,

$$\log Z_\theta(g) = \mathcal{F}(\theta; q) + D_{\mathrm{KL}}(q \,\|\, p_\theta(\cdot \mid g, r_H = 1)) \;\geq\; \mathcal{F}(\theta; q). \tag{14}$$

By Eq. 12, the divergence vanishes at $\theta = \theta_n$ for $q = q_n$, so

$$\mathcal{F}(\theta_n; q_n) = \log Z_{\theta_n}(g), \tag{15}$$

and the bound is tight at the policy that generated q_n.

Since q_n is fixed during the policy update, its entropy is independent of θ. Improving the bound therefore reduces to

$$\max_\theta \mathbb{E}_{\zeta \sim q_n} \left[\log p_\theta(\zeta \mid g) \right]. \tag{16}$$

Assuming the environment dynamics are independent of the policy parameters,

$$\log p_\theta(\zeta \mid g) = \text{const.} + \sum_{t=0}^{H-1} \log \pi_\theta(a_t \mid s_t, g). \tag{17}$$

The policy-improvement step therefore becomes

$$\max_\theta \mathbb{E}_{\zeta \sim q_n} \left[\sum_{t=0}^{H-1} \log \pi_\theta(a_t \mid s_t, g) \right]. \tag{18}$$

For binary return, the M-step of EM policy search is therefore maximum-likelihood fitting to trajectories that the policy itself has discovered to be successful. The same equivalence follows directly from the policy-gradient identity. At $\theta = \theta_n$,

$$\nabla_\theta \log Z_\theta(g) = \mathbb{E}_{\zeta \sim q_n} \left[\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t, g) \right]. \tag{19}$$

Thus, under the current policy, maximum-likelihood fitting to the success-conditioned trajectory distribution follows exactly the gradient of log success probability.

Moreover, because the lower bound is tight at θ_n, any generalized-EM update satisfying

$$\mathcal{F}(\theta_{n+1}; q_n) \geq \mathcal{F}(\theta_n; q_n) \tag{20}$$

also satisfies

$$Z_{\theta_{n+1}}(g) \geq Z_{\theta_n}(g). \tag{21}$$

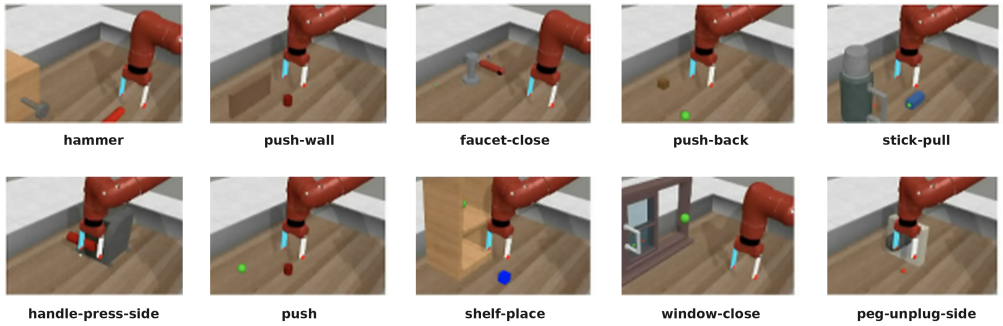

Figure 4: Ten-task Sawyer manipulation sequence used in our experiments, following the Continual World V2 task order.

B.3 Relation to our actor objective

Our implementation stores successful trajectories discovered online in $\mathcal{D}_{\text{succ}}$ and uses

$$\mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{succ}}}\left[\log \pi_\theta(a \mid s, g^{(k)})\right] \tag{22}$$

as a weak regularizer rather than replacing the contrastive policy objective. The success buffer therefore provides a replay-based empirical approximation to the success-conditioned distribution in Eq. 12.

Because the buffer also contains trajectories generated by earlier policy iterations, the exact gradient identity in Eq. 19 and the monotonic-improvement guarantee in Eq. 21 apply to the idealized on-policy update rather than to each finite replay update. We use this result to motivate the direction of the regularizer while retaining the contrastive objective and entropy regularization as the primary policy-learning objective.

C Benchmark and evaluation protocol

C.1 Benchmark

We use the ten Sawyer manipulation environments from Meta-World (Yu et al., 2019) and follow the task order introduced by Continual World V2 (Wołczyk et al., 2021). The task sequence is

`hammer`, `push-wall`, `faucet-close`, `push-back`, `stick-pull`,
`handle-press-side`, `push`, `shelf-place`, `window-close`, `peg-unplug-side`.

We index the tasks $k = 0, \ldots, 9$ in this order and refer to them as T0–T9 in figures and tables. For every task k, the training budget is $T_k = 8 \times 10^6$ environment steps. The episode horizon is $H = 500$, and the action space is the Meta-World default $[-1, 1]^4$.

C.2 Observation and goal representation

All reported experiments use the task-ID-free setting (`use_task_id=false`). Each transition is represented as

$$[\, s\,(11),\ g\,(11)\,],$$

where s is the current state and g is the goal input to the goal-conditioned policy and critic. The state vector s contains Sawyer proprioceptive and object features, including end-effector position, gripper state, and task-specific object pose. We zero-pad all task observations to a unified dimension of 11, which is the maximum state dimension across the ten tasks.

We do not append an explicit task identifier. In our setting, each task is associated with a commanded goal during rollout, and task boundaries are reflected by changes in the commanded goal and the corresponding state distribution. Thus, the agent receives task information through the goal-conditioned input rather than through an additional one-hot task label.

The goal vector g has the same dimension and layout as s. During training, goals are constructed by hindsight relabeling. For each transition, we sample a future index from the same trajectory using a geometrically discounted distribution over the remainder of the episode, and use the observation at that index as the relabeled goal. With probability 0.5, this future-state goal is replaced by a uniformly sampled state from the batch to provide contrastive negatives.

Our implementation also supports an optional task-ID wrapper, which would represent transitions as

$$[\, s\,(11),\ \mathbf{1}_k\,(K),\ g\,(11),\ \mathbf{1}_k\,(K)\,], \qquad K = 10.$$

This option is not used in the experiments reported in this paper.

D Training and implementation details

This section lists the architectural, optimization, and infrastructure details used for the experiments in the main paper. Unless otherwise specified, the same settings are used across the contrastive baselines and the proposed decomposed critic.

D.1 Network architecture and optimization

Encoders. The state-action encoder and goal encoder are pre-norm residual MLPs with LayerNorm, Swish activations, and a linear projection at the residual stem. Both encoders use depth 4, hidden width 1024, and embedding dimension $d = 64$ (Wang et al., 2025).

Decomposed critic modules. The task-specific encoder ϕ_{task} is a residual MLP with hidden width 256 and depth 4. Its output has the same embedding dimension $d = 64$ as the shared state-action encoder, so the two representations can be added before computing the contrastive score.

Actor. The actor is a $\tanh$-Gaussian policy using the same backbone family as the contrastive actor. We use an adaptive entropy coefficient with target entropy -2.0 (Haarnoja et al., 2018).

Optimization and replay. All actor and critic modules are trained with Adam using learning rate 3×10^{-4}. The batch size is 1024, the discount is $\gamma = 0.99$, and each task uses a replay buffer with capacity 10^6 transitions. Training begins after a warm start of 10^4 transitions. We perform one gradient update per environment step and use Polyak averaging for target-network updates.

D.2 Hyperparameters

Table 2 summarizes the main hyperparameters used throughout the experiments.

D.3 Infrastructure and hardware

The implementation is written in JAX (Bradbury et al., 2018) and Haiku (Hennigan et al., 2020), using Brax (Freeman et al., 2021) and JaxGCRL (Bortkiewicz et al., 2025). Experiments were run on NVIDIA A100 80GB and H100 GPUs across two SLURM-managed clusters. We use SLURM job arrays to co-schedule multiple processes per GPU.

E Baseline details

E.1 Goal-conditioned SAC

The goal-conditioned SAC baselines, SAC (R/R) and SAC (P/P), serve as non-contrastive reference points for the same sparse-reward continual RL setting studied in this paper. We adapt JAXGCRL's SAC implementation (Bortkiewicz et al., 2025) to our problem setup, and keep every implementation choice that is not strictly SAC-specific identical to our contrastive methods to make the comparison as tight as possible. Algorithm 2 describes how the goal-conditioned SAC is trained with hindsight experience replay (HER) (Andrychowicz et al., 2017). We highlight a few design choices.

Table 2: Training hyperparameters used in the main experiments. Entries for the success buffer and critic gate apply to full DCC; critic-only DCC sets $\lambda_{\text{succ}} = 0$.

Category	Hyperparameter	Value
Backbone	Hidden width	1024
	Depth	4
	Activation	Swish
	Normalization	LayerNorm (pre-norm)
	Embedding dimension d	64
Optimization	Optimizer	Adam
	Learning rate	3×10^{-4}
	Batch size	1024
	Discount γ	0.99
	Updates per environment step	1
Replay	Capacity per task	10^6
	Warm-start transitions	10^4
	HER future probability	0.5
Entropy	Entropy coefficient	adaptive
	Target entropy	-2.0
Decomposed critic	ϕ_{task} width/depth	256 / 4
	`use_task_id`	false
Success buffer	Regularization weight λ_{succ}	0.1
	Capacity	4096 transitions
	Success minibatch size	64
	Sampling	uniform
	Inclusion rule	$r_H^{(k)} = 1$
	Stored transitions	entire successful trajectory
Critic gate	Action probes N_a	8
	Action-probe distribution	$\text{Unif}([-1, 1]^4)$
	Local state probes N_s	8
	State-perturbation scale ε	0.1
	State-coordinate scale	success-minibatch std.
	Numerical tolerance δ	10^{-6}
	Gradient through gate	stopped
CKA	Knowledge-pool capacity	5
	Merge rule	max-cos, mean-merge
	`adapt_heads_only`	true
Continual loop	Number of tasks K	10
	Environment steps per task	8×10^6
	Episode horizon	500
	Evaluation cadence	every 10^5 steps

Training. First, we use the standard step-penalty sparse reward $r \in \{-1, 0\}$ rather than the $\{0, +1\}$ variant (Line 9). The reward is 0 when the achieved goal of the next state lies within the goal proximity threshold τ_{reach} of the relabeled goal g, and -1 otherwise. This design is the convention most commonly paired with sparse TD-learning and provides a non-trivial learning signal on every transition (Andrychowicz et al., 2017; Park et al., 2023; 2025). In our experiments, we found this implementation improves SAC performance across tasks.

Second, the HER discount γ_{HER} folds two terminal conditions into a single multiplicative mask (Line 9). It zeros the bootstrap whenever the relabeled transition reaches the goal ($r = 0$) and whenever the environment itself emits a done signal ($d = 1$). Treating goal-reaching as terminal keeps Q bounded under the sparse reward and is the standard sparse-goal trick.

Finally, online data is collected at the trajectory level (Line 5). Each rollout is stored as a complete episode in replay buffer $\mathcal{D}$ together with its per-step done signals. At each gradient step, we sample trajectories from $\mathcal{D}$ and, for each timestep t, draw a future index $t' > t$ with probability proportional to $\gamma^{t'-t}$. Then, the achieved goal $\text{m}(s_{t'})$ serves as the relabeled goal g.

Algorithm 2 Goal-conditioned SAC with HER for the Single Task k

Input: initial policy parameters ϕ, Q-function parameters θ_1, θ_2, entropy temperature α, empty replay buffer $\mathcal{D}$, task-commanded goal $g^{(k)}$, goal proximity threshold τ_{reach}, achieved-goal mapping $\mathrm{m} : \mathcal{S} \to \mathcal{G}$, discount factor γ, Polyak coefficient ρ
Set target parameters $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$
repeat
  Collect a rollout trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ by sampling $a_t \sim \pi_\phi(\cdot \mid s_t, g^{(k)})$
  Store the trajectory τ (with per-step done signals d_t) in $\mathcal{D}$
  if it's time to update **then**
    for j in range(however many updates) **do**
      Sample from trajectory buffer $\mathcal{D}$ and apply HER by geometrically sampling in-trajectory future states to obtain relabeled goal g
      Compute sparse reward and HER discount:

$$r \;=\; \mathbf{1}[\,\|\mathrm{m}(s') - g\|_2 < \tau_{\text{reach}}\,] - 1, \quad \gamma_{\text{HER}} \;=\; \big(1 - \mathbf{1}[r = 0]\big)(1 - d)\,\gamma$$

      Form mini-batch $B = \{(s, a, r, s', \gamma_{\text{HER}}, g)\}$
      Compute targets for the Q-functions:

$$y(r, s', \gamma_{\text{HER}}, g) \;= r + \gamma_{\text{HER}} \Big(\min_{i=1,2} Q_{\bar{\theta}_i}(s', \tilde{a}', g) - \alpha \log \pi_\phi(\tilde{a}' \mid s', g) \Big)$$
$$, \tilde{a}' \;\sim\; \pi_\phi(\cdot \mid s', g)$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\theta_i} \frac{1}{|B|} \sum_{(s,a,r,s',\gamma_{\text{HER}},g) \in B} \big(Q_{\theta_i}(s, a, g) - y(r, s', \gamma_{\text{HER}}, g) \big)^2 \qquad \text{for } i = 1, 2$$

      Update policy by one step of gradient ascent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,g) \in B} \Big(\min_{i=1,2} Q_{\theta_i}(s, \tilde{a}, g) - \alpha \log \pi_\phi(\tilde{a} \mid s, g) \Big), \quad \tilde{a} \sim \pi_\phi(\cdot \mid s, g)$$

      Update target networks with $\bar{\theta}_i \leftarrow \rho\, \bar{\theta}_i + (1 - \rho)\, \theta_i$, for $i = 1, 2$
    end for
  end if
until convergence

Network architecture. The SAC actor network shares the same backbone as the CRL actor backbone, with the contrastive head replaced by a standard `NormalTanhDistribution` output. Each Q network independently takes the concatenation $[s, g, a]$ through a `ResidualMLP` of width 1024 and depth 4, using the same LayerNorm, Swish activations, and residual connections as the actor body and the contrastive methods.

Hyperparameters. Table 3 reports the hyperparameters for the sparse goal-conditioned SAC baseline. To keep a tight comparison with the contrastive methods, all non-SAC-specific settings (backbone, optimization, replay/HER, continual loop) are inherited from Table 2. The SAC-specific choices are set to the values most commonly used in sparse SAC implementations.

Continual transfer. We report two cross-task modes that mirror the contrastive R/R and P/P. In SAC (R/R), the SAC agent resets at every task boundary, while in SAC (P/P), the same SAC agent is carried forward across the entire ten-task curriculum. No knowledge pool or parameter-space mixture is used in either SAC mode. Appendix E.2 provides more details about the transfer modes studied in this paper.

E.2 Contrastive GCRL transfer variants

The contrastive transfer grid pairs one of three transfer modes on each of the actor and the critic, yielding nine baseline cells: $\{\text{reset}, \text{persistent}, \text{CKA}\} \times \{\text{reset}, \text{persistent}, \text{CKA}\}$. All nine cells share

Table 3: Hyperparameters for the goal-conditioned SAC baseline. Settings shared with the contrastive methods (Table 2) are reproduced for self-containment.

Category	Hyperparameter	Value
Backbone	Hidden width	1024
	Depth (residual blocks)	4
	Activation	Swish
	Norm	LayerNorm (pre-norm)
Optimization	Optimizer	Adam
	Learning rate (actor, critic)	3×10^{-4}
	Batch size	1024
	Discount γ	0.99
	Updates per env step	1
Replay / HER	Capacity (per task)	10^6
	Warm-start transitions	10^4
Entropy	Coefficient	adaptive
	Target entropy	-2.0
SAC-specific	Reward definition	$\{-1, 0\}$
	Goal proximity threshold τ_{reach}	0.05
	Polyak coefficient ρ	0.995
Continual loop	Tasks K	10
	Env steps per task	8×10^6
	Episode length T	500
	Evaluation cadence	every 10^5 env steps

the same contrastive critic architecture, the same hindsight-relabeled replay buffer, and the same per-task budget. They differ only in how the actor parameters and the critic parameters are propagated across task boundaries. We denote each cell as *actor-mode*/*critic-mode*, with R, P, and C standing for reset, persistent, and CKA respectively.

Reset (R). The network is re-initialized at every task boundary; the per-task replay buffer is discarded along with the parameters and the optimizer state. This is the natural baseline that ignores transfer entirely and treats each task as an independent sparse-reward goal-conditioned problem.

Persistent (P). The parameters, target parameters, and optimizer state are carried forward unchanged across task boundaries; only the replay buffer is reset. Persistent transfer is the strongest form of weight-level reuse: any reachability structure learned on earlier tasks is available to the next task by default, at the cost of accumulating drift if the new task's gradients pull the network away from earlier solutions.

CKA (C). The network is decomposed as θ_{base} plus a softmax-weighted mixture of past task vectors plus a fresh per-task residual v_k, following the recipe of Hu et al. (2025). The trainable bundle $(v_k, \boldsymbol{\beta}, s)$ is updated jointly inside the inner gradient loop; the pool of past residuals is updated at task boundaries with a cosine-similarity merge rule. The full description is in Appendix E.3 and a list of CKA-specific hyperparameters is in Table 4.

The main body reports curriculum averages for the complete nine-cell contrastive grid. Tables 5 and 6 provide the corresponding per-task results, including the completed critic-only DCC ablation.

E.3 CKA PARAMETER-MIXTURE VARIANTS

We include the CKA parameter-mixture of Hu et al. (2025) as a baseline because it is the canonical continual-RL recipe for composing a shared backbone with a per-task contribution, and so it serves as a direct point of comparison against the representation-space decomposition of Section 3.

Table 4: Hyperparameters specific to the CKA parameter-mixture baseline (Appendix E.3). All other hyperparameters, including backbone, optimizer, batch size, replay, HER, and per-task budget, are inherited from the underlying contrastive cell and are not repeated here. The same values are used on the actor and on the critic.

Category	Hyperparameter	Value
Pool	Capacity $K_{\max}$	5
	Merge rule	max-cosine, mean-merge
Mixture	Logits init. $\boldsymbol{\beta}(k)$	$\mathcal{N}(0, 0.01^2 I)$
	Scale init. s	1.0
Per-task residual	Residual init. v_k	$\mathbf{0}$
	Adapted parameters	heads only (`adapt_heads_only`=true)
Optimization	Bundle $(v_k, \boldsymbol{\beta}, s)$	single Adam
	Learning rate	3×10^{-4}
	Updates per env step	1
Task-0 initialization	Pool	empty
	Base parameters θ_{base}	θ_0 after task-0 training

Composition. Let θ_{base} denote the parameters of a network frozen at the end of task 0 (the actor in actor-CKA cells; the critic in critic-CKA cells), and let $\{v_1, \ldots, v_n\}$ be a finite pool of parameter residuals collected from past tasks. At task k the working parameters are

$$\theta_k = \theta_{\text{base}} + \sum_{j=1}^{n} \alpha_j(k)\, v_j + v_k, \tag{23}$$

where v_k is a per-task residual learned from scratch and $\boldsymbol{\alpha}(k) \in \Delta^{n-1}$ is a softmax-normalized mixture over the pool slots:

$$\alpha_j(k) = \frac{\exp\big(s\, \beta_{k,j}\big)}{\sum_{i=1}^{n} \exp\big(s\, \beta_{k,i}\big)}. \tag{24}$$

Here $\boldsymbol{\beta}(k) \in \mathbb{R}^n$ are per-task mixture logits and $s \in \mathbb{R}_{>0}$ is a scalar temperature; both are trained jointly with v_k.

The per-task trainable bundle is therefore $(v_k, \boldsymbol{\beta}(k), s)$, and the entire bundle is updated in the inner gradient loop by a single Adam optimizer, with gradients flowing through Equation equation 23 into the actor or critic loss. The base parameters θ_{base} and the pool $\{v_j\}$ are constants of this optimization.

Pool management. The pool has a fixed capacity $K_{\max}$ (we use $K_{\max} = 5$). At a task boundary the learned residual v_k is appended to the pool. If the pool then exceeds capacity, the two slots most similar in *cosine distance* are merged by elementwise averaging into a single slot, which keeps the pool at $K_{\max}$ entries while compressing redundant directions. Similarity is computed on the flattened parameter vectors,

$$\text{sim}(v_i, v_j) = \frac{\langle v_i, v_j \rangle}{\|v_i\|_2 \, \|v_j\|_2}, \tag{25}$$

where the flattening concatenates every leaf of the pytree. Merging is the only inter-task interaction with the pool; the selection of which pair to merge is the only host-side operation.

Initialization and pool seeding. At task 0, the learned residual v_0 is folded into the base parameters ($\theta_{\text{base}} \leftarrow \theta_{\text{base}} + v_0$) and the pool starts empty; the mixture term in Equation equation 23 is then zero by convention until the first non-trivial entry is appended at the boundary $k = 0 \to 1$. At each subsequent task k, v_k is initialized fresh, $\boldsymbol{\beta}(k)$ is initialized by Gaussian noise of standard deviation 0.01, and s is initialized to 1.

Algorithm 3 summarizes the per-task procedure as it is implemented in our codebase. The same procedure applies symmetrically to the actor and the critic; in our experiments we run it on both networks. The actor and the critic each maintain their own base parameters θ_{base}, pool $\{v_j\}$, per-task residual v_k, mixture logits $\boldsymbol{\beta}(k)$, and scale s.

Algorithm 3 CKA parameter mixture at task k (baseline of Hu et al. (2025))

input task $\mathcal{M}^{(k)}$ for $k \geq 1$; base parameters θ_{base}; pool $\{v_j\}_{j=1}^{n}$; capacity $K_{\max}$
init $v_k \leftarrow 0$; $\boldsymbol{\beta}(k) \sim \mathcal{N}(\mathbf{0}, 0.01^2 I)$; $s \leftarrow 1$
init Adam optimizer over the bundle $(v_k, \boldsymbol{\beta}(k), s)$
for each environment step inside task k **do**
    Collect a transition; sample a mini-batch from $\mathcal{D}_k$
    $\boldsymbol{\alpha}(k) \leftarrow \mathrm{softmax}\big(s\,\boldsymbol{\beta}(k)\big)$
    $\theta_k \leftarrow \theta_{\text{base}} + \sum_{j=1}^{n} \alpha_j(k)\, v_j + v_k$
    Compute the actor or critic loss using θ_k
    Update $(v_k, \boldsymbol{\beta}(k), s)$ by one Adam step on the gradient of the loss w.r.t. the bundle
end for
$\{v_j\} \leftarrow \{v_j\} \cup \{v_k\}$
if $|\{v_j\}| > K_{\max}$ **then**
    $(i^*, j^*) \leftarrow \arg\max_{i \neq j} \mathrm{sim}(v_i, v_j)$ ▷ cosine similarity, Eq. 25
    $v_{i^*} \leftarrow (v_{i^*} + v_{j^*})/2$; remove v_{j^*} from the pool
end if

F VARIANT OVERVIEW AND FULL RESULTS

F.1 VARIANT NOTATION

We use R, P, and C to denote reset, persistent, and CKA-mixture transfer, respectively. For contrastive GCRL variants, the notation actor-mode/critic-mode specifies how the actor and critic are handled at task boundaries. For example, R/R resets both actor and critic at each new task, P/P carries both forward, and C/R applies the CKA mixture to the actor while resetting the critic. The SAC baselines use the same R/R and P/P notation, but only have actor and critic transfer modes and do not use CKA or the decomposed critic.

Detailed implementation choices for SAC, CKA, and the contrastive transfer grid are given in Appendix E.

F.2 PER-TASK GRID ACROSS ALL VARIANTS

Tables 5 and 6 report per-task normalized AUC and best success across the ten Sawyer tasks. They include the two sparse goal-conditioned SAC baselines, the nine contrastive RL baselines, two DCC ablations, and DCC. Figure 1 in the main text visualizes the same per-task profiles.

Table 5: Per-task normalized AUC for all methods. Values are means ± standard deviations across seeds; the best mean in each row is bold.

Task	**Sparse SAC**		**Contrastive RL**									**DCC Ablations**		**DCC**
			Actor Reset			Actor Persistent			Actor CKA			Critic-only	Ungated	Full
	Actor Reset Critic Reset	Actor Persistent Critic Persistent	Critic Reset	Critic Persistent	Critic CKA	Critic Reset	Critic Persistent	Critic CKA	Critic Reset	Critic Persistent	Critic CKA	DCC	DCC	DCC
T_0 Hammer	0.396 ± 0.156	0.588 ± 0.243	0.880 ± 0.060	0.891 ± 0.038	0.876 ± 0.036	0.894 ± 0.025	0.850 ± 0.067	0.855 ± 0.072	0.850 ± 0.032	0.865 ± 0.025	0.872 ± 0.031	0.732 ± 0.041	0.992 ± 0.015	**0.994** ± 0.012
T_1 Push-wall	0.000 ± 0.000	0.000 ± 0.000	0.565 ± 0.069	0.593 ± 0.114	0.608 ± 0.088	0.573 ± 0.134	0.548 ± 0.075	0.537 ± 0.162	0.518 ± 0.041	0.551 ± 0.090	0.417 ± 0.158	0.701 ± 0.087	0.692 ± 0.250	**0.712** ± 0.082
T_2 Faucet-close	0.161 ± 0.113	0.026 ± 0.032	0.387 ± 0.019	0.329 ± 0.047	0.380 ± 0.023	0.405 ± 0.018	0.353 ± 0.052	0.372 ± 0.034	0.432 ± 0.035	0.348 ± 0.040	0.389 ± 0.034	0.355 ± 0.034	0.637 ± 0.052	**0.648** ± 0.048
T_3 Push-back	0.000 ± 0.000	0.000 ± 0.000	0.786 ± 0.076	0.723 ± 0.132	0.769 ± 0.052	0.822 ± 0.064	0.716 ± 0.066	0.687 ± 0.093	0.807 ± 0.062	0.621 ± 0.066	0.781 ± 0.037	0.826 ± 0.096	0.866 ± 0.075	**0.874** ± 0.070
T_4 Stick-pull	**0.621** ± 0.095	0.218 ± 0.245	0.473 ± 0.042	0.447 ± 0.187	0.512 ± 0.111	0.581 ± 0.060	0.462 ± 0.121	0.597 ± 0.093	0.453 ± 0.063	0.446 ± 0.107	0.411 ± 0.168	0.568 ± 0.176	0.218 ± 0.192	0.616 ± 0.254
T_5 Handle-press	0.110 ± 0.013	0.109 ± 0.012	0.022 ± 0.007	0.003 ± 0.002	0.009 ± 0.004	0.021 ± 0.006	0.011 ± 0.009	0.008 ± 0.002	0.026 ± 0.006	0.013 ± 0.002	0.011 ± 0.003	0.181 ± 0.091	0.664 ± 0.174	**0.698** ± 0.162
T_6 Push	0.013 ± 0.029	0.050 ± 0.118	0.432 ± 0.125	0.472 ± 0.139	0.463 ± 0.064	0.452 ± 0.058	0.451 ± 0.078	0.476 ± 0.109	0.389 ± 0.095	0.403 ± 0.037	0.278 ± 0.077	0.468 ± 0.053	0.659 ± 0.288	**0.671** ± 0.270
T_7 Shelf-place	0.000 ± 0.000	0.000 ± 0.000	0.412 ± 0.068	0.170 ± 0.194	0.470 ± 0.030	0.357 ± 0.200	0.138 ± 0.146	0.158 ± 0.151	0.403 ± 0.073	0.239 ± 0.202	0.307 ± 0.221	0.471 ± 0.110	0.105 ± 0.097	**0.608** ± 0.139
T_8 Window-close	0.297 ± 0.026	0.313 ± 0.027	0.055 ± 0.020	0.058 ± 0.033	0.062 ± 0.015	0.052 ± 0.011	0.033 ± 0.040	0.043 ± 0.024	0.059 ± 0.015	0.052 ± 0.025	0.049 ± 0.017	0.144 ± 0.029	**0.955** ± 0.057	0.926 ± 0.090
T_9 Peg-unplug	0.001 ± 0.001	0.020 ± 0.011	0.332 ± 0.030	0.234 ± 0.058	0.318 ± 0.082	0.295 ± 0.025	0.245 ± 0.052	0.214 ± 0.035	0.346 ± 0.091	0.208 ± 0.051	0.280 ± 0.023	0.345 ± 0.057	0.455 ± 0.073	**0.468** ± 0.068
Average	0.160 ± 0.010	0.132 ± 0.049	0.434 ± 0.005	0.408 ± 0.021	0.447 ± 0.014	0.435 ± 0.007	0.390 ± 0.022	0.392 ± 0.010	0.424 ± 0.013	0.375 ± 0.031	0.396 ± 0.026	0.479 ± 0.029	0.624 ± 0.091	**0.722** ± 0.071

Table 6: Per-task best success for all methods. Values are means ± standard deviations across seeds; the best mean in each row is bold.

Task	**Sparse SAC**		**Contrastive RL**									**DCC Ablations**		**DCC**
			Actor Reset			Actor Persistent			Actor CKA			Critic-only	Ungated	Full
	Actor Reset Critic Reset	Actor Persistent Critic Persistent	Critic Reset	Critic Persistent	Critic CKA	Critic Reset	Critic Persistent	Critic CKA	Critic Reset	Critic Persistent	Critic CKA	DCC	DCC	DCC
T_0 Hammer	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000
T_1 Push-wall	0.017 ± 0.041	0.000 ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	0.950 ± 0.108	**1.000** ± 0.000
T_2 Faucet-close	0.667 ± 0.288	0.433 ± 0.197	0.920 ± 0.084	0.840 ± 0.055	0.840 ± 0.055	0.860 ± 0.089	0.880 ± 0.084	0.820 ± 0.045	0.920 ± 0.045	0.900 ± 0.082	0.900 ± 0.082	0.800 ± 0.082	0.990 ± 0.032	**1.000** ± 0.000
T_3 Push-back	0.000 ± 0.000	0.000 ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000
T_4 Stick-pull	**1.000** ± 0.000	0.983 ± 0.041	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	0.980 ± 0.045	**1.000** ± 0.000	0.900 ± 0.141	**1.000** ± 0.000	0.800 ± 0.377	0.900 ± 0.316
T_5 Handle-press	0.483 ± 0.075	0.483 ± 0.075	0.240 ± 0.089	0.120 ± 0.045	0.200 ± 0.000	0.240 ± 0.055	0.200 ± 0.122	0.120 ± 0.045	0.260 ± 0.055	0.200 ± 0.082	0.175 ± 0.050	0.640 ± 0.190	0.990 ± 0.032	**1.000** ± 0.000
T_6 Push	0.200 ± 0.395	0.300 ± 0.385	0.980 ± 0.045	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	**1.000** ± 0.000	0.980 ± 0.045	**1.000** ± 0.000	**1.000** ± 0.000	0.950 ± 0.100	**1.000** ± 0.000	0.930 ± 0.164	**1.000** ± 0.000
T_7 Shelf-place	0.000 ± 0.000	0.000 ± 0.000	0.980 ± 0.045	0.680 ± 0.319	0.980 ± 0.045	0.940 ± 0.089	0.660 ± 0.385	0.660 ± 0.391	0.980 ± 0.045	0.775 ± 0.320	0.750 ± 0.436	0.970 ± 0.048	0.430 ± 0.200	**1.000** ± 0.000
T_8 Window-close	0.767 ± 0.052	0.783 ± 0.041	0.367 ± 0.058	0.433 ± 0.058	0.480 ± 0.110	0.400 ± 0.100	0.367 ± 0.153	0.367 ± 0.153	0.400 ± 0.000	0.375 ± 0.096	0.367 ± 0.058	0.750 ± 0.178	**1.000** ± 0.000	**1.000** ± 0.000
T_9 Peg-unplug	0.067 ± 0.052	0.283 ± 0.075	0.833 ± 0.058	0.667 ± 0.153	**0.900** ± 0.071	0.833 ± 0.058	0.800 ± 0.100	0.667 ± 0.058	0.867 ± 0.153	0.725 ± 0.096	0.767 ± 0.058	0.880 ± 0.092	0.870 ± 0.095	0.890 ± 0.087
Average	0.420 ± 0.017	0.427 ± 0.034	0.840 ± 0.010	0.773 ± 0.038	0.840 ± 0.007	0.823 ± 0.025	0.787 ± 0.058	0.773 ± 0.031	0.840 ± 0.010	0.798 ± 0.034	0.803 ± 0.023	0.904 ± 0.037	0.896 ± 0.061	**0.979** ± 0.045

G ACTION-INFORMATIVE STATES AND CONTRASTIVE CONTROL

Contrastive RL trains a critic to predict future states from state-action pairs. For control, however, predicting the future is not sufficient by itself: the critic must also distinguish how different actions change that future. This section characterizes where such action-dependent information appears in the replay distribution.

Let F denote a future state sampled from the same trajectory as (S, A), as in hindsight relabeling, and let $\mu(s)$ denote the state marginal of the replay distribution. Importantly, F is an observed future state and need not equal the goal originally commanded to the behavior policy.

G.1 SEPARATING STATE AND ACTION INFORMATION

Rather than assuming a particular parameterization of the learned InfoNCE score, we first consider the population density ratio associated with the contrastive prediction problem,

$$r(s, a, f) = \log \frac{p(f \mid s, a)}{p(f)}. \tag{26}$$

Introducing the state-conditioned future distribution gives the exact decomposition

$$r(s, a, f) = \log \frac{p(f \mid s)}{p(f)} + \log \frac{p(f \mid s, a)}{p(f \mid s)}. \tag{27}$$

For fixed s and f, the first term is independent of the action. All variation that can distinguish actions therefore lies in the second term. Averaging Eq. 27 over the experience distribution recovers the mutual-information chain rule,

$$I((S, A); F) = I(S; F) + I(A; F \mid S), \tag{28}$$

where

$$I(A; F \mid S = s) = \mathbb{E}_{A,F|S=s}\left[\log \frac{p(F \mid s, A)}{p(F \mid s)}\right]. \tag{29}$$

Thus, $I(A; F \mid S = s)$ measures the additional information about the future provided by the action once the current state is already known.

G.2 ACTION-INFORMATIVE STATES

We now formalize the states at which actions can meaningfully alter the future distribution. Define

$$\Delta(s) = \sup_{a,a'} D_{\mathrm{TV}}\left(p(F \mid s, a), p(F \mid s, a')\right), \tag{30}$$

where D_{TV} denotes total-variation distance. For a threshold $\epsilon \geq 0$, define the set of *action-informative states*

$$\mathcal{S}^{\epsilon}_{\mathrm{act}} = \{s : \Delta(s) > \epsilon\}, \qquad \delta_{\epsilon} = \mu(\mathcal{S}^{\epsilon}_{\mathrm{act}}). \tag{31}$$

The quantity δ_{ϵ} is the replay probability mass assigned to states at which changing the action changes the future-state distribution by more than ϵ.

Proposition 1. If $s \notin \mathcal{S}^{0}_{\mathrm{act}}$, then

$$p(F \mid s, a) = p(F \mid s) \qquad \text{for every action } a, \tag{32}$$

and therefore

$$I(A; F \mid S = s) = 0. \tag{33}$$

Consequently,

$$I(A; F \mid S) = \int_{\mathcal{S}^{0}_{\mathrm{act}}} I(A; F \mid S = s)\, d\mu(s). \tag{34}$$

Proof. For $s \notin \mathcal{S}^{0}_{\mathrm{act}}$, all conditional distributions $p(F \mid s, a)$ are identical across actions. Their mixture under the behavior policy is therefore the same distribution, $p(F \mid s)$. Hence the action-conditional term in Eq. 27 is zero, which implies Eq. 33. Integrating over the state marginal gives Eq. 34. □

This result isolates the part of replay that can provide action-dependent predictive supervision. If states are sampled independently from μ in a minibatch of size B, and

$$N_\epsilon = \sum_{i=1}^{B} \mathbf{1}\left[S_i \in \mathcal{S}^{\epsilon}_{\mathrm{act}}\right], \tag{35}$$

then

$$\mathbb{E}[N_\epsilon] = B\delta_\epsilon, \qquad \Pr(N_\epsilon = 0) = (1-\delta_\epsilon)^B. \tag{36}$$

When δ_ϵ is small, only a small fraction of replay states can expose strong action-dependent changes in future occupancy. This is an upper bound on the available opportunity for action discrimination, since visiting an action-informative state does not guarantee that the sampled actions themselves expose the relevant contrast.

G.3 The goal-specific view

The same idea can be stated directly in terms of the commanded goal. Let

$$\rho^\pi(g \mid s, a) \tag{37}$$

denote the discounted future occupancy of goal g. Define the goal-specific action gap

$$\Gamma_g(s) = \sup_{a,a'} \left|\rho^\pi(g \mid s, a) - \rho^\pi(g \mid s, a')\right|. \tag{38}$$

If $\Gamma_g(s) = 0$, then every action has the same discounted probability of reaching g from s. Even an exact occupancy critic therefore provides no preference among actions at that state.

G.4 High contrastive accuracy does not imply action discrimination

Low replay mass of action-informative states also explains why aggregate contrastive accuracy can be misleading. The following simple construction shows that a critic can identify the correct future with arbitrarily high categorical accuracy while ignoring the action entirely.

Proposition 2. Let S be uniform on $\{1, \dots, M\}$, let A be uniform on $\{0, 1\}$, and let the deterministic future outcome be

$$F = (S, A). \tag{39}$$

For each state s, let $e_s \in \mathbb{R}^M$ be its one-hot vector and define the cosine-similarity critic

$$f((s,a),(u,b)) = e_s^\top e_u. \tag{40}$$

The critic ignores both a and b. Nevertheless, with one positive candidate and $N-1$ independently sampled negatives, its categorical matching accuracy is at least

$$\left(1 - \frac{1}{M}\right)^{N-1}. \tag{41}$$

Hence the accuracy approaches one as $M \to \infty$, although the critic assigns identical scores to the action that reaches (s, b) and the action that reaches $(s, 1-b)$.

Proof. The positive candidate always has the same state index as the anchor and therefore receives score 1. Any negative with a different state index receives score 0. The positive is therefore the unique highest-scoring candidate whenever none of the $N-1$ negatives shares the anchor state. Since each negative matches the anchor state with probability $1/M$, this event occurs with probability $(1-1/M)^{N-1}$. □

For an InfoNCE temperature τ, on such collision-free batches the loss of this action-independent critic is

$$\log\left(1 + (N-1)e^{-1/\tau}\right). \tag{42}$$

Thus, strong future-matching performance can be obtained primarily from state information even when the action determines which future within that state is actually reached.

G.5 Implications for contrastive control

The two results above describe complementary aspects of the same failure mode. First, when $\delta_\epsilon = \mu(\mathcal{S}^\epsilon_{\text{act}})$ is small, the replay distribution rarely presents states at which actions induce substantially different futures. Second, high aggregate contrastive accuracy does not rule out this failure, because the prediction problem can often be solved using state information alone.

G.6 Action-sensitivity measurements

Figure 3 compares how strongly actions affect the critic across Tasks 5, 8, and 6. The measurements are computed from on-policy rollouts of DCC critics trained without the critic gate.

For each state s, we evaluate the critic on a set of alternative actions $\mathcal{A}_{\text{probe}}(s)$ and define the critic action range

$$\Delta_f(s, g) = \max_{a \in \mathcal{A}_{\text{probe}}(s)} f(s, a, g) - \min_{a \in \mathcal{A}_{\text{probe}}(s)} f(s, a, g). \tag{43}$$

The candidate set contains the policy action together with zero, axis-aligned perturbations, shuffled actions, uniformly sampled actions, and Gaussian perturbations around the policy action.

Raw critic scores have different scales across tasks, so we normalize the action range by the variation of the policy-action score within the same task. Let

$$\kappa_f = \text{Std}_s \left[f(s, a_\pi(s), g) \right]. \tag{44}$$

The normalized critic sensitivity used in Figure 3 is then

$$\widetilde{\Delta}_f(s, g) = \frac{\Delta_f(s, g)}{\kappa_f}. \tag{45}$$

We summarize how frequently large action sensitivity appears through

$$\widehat{\delta}(\epsilon) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1} \left[\widetilde{\Delta}_f(s_i, g) > \epsilon \right], \tag{46}$$

which is the empirical fraction of sampled states whose normalized critic action range exceeds a common threshold ϵ. The dotted curves in Figure 3 repeat this calculation using only local Gaussian perturbations around the policy action.

We additionally measure interaction occupancy using the distance between the robot hand and the manipulated object or mechanism. A state is counted as lying in the interaction region when this distance is below $9\,\text{cm}$, using the same threshold for all three tasks. The reported occupancy is the fraction of rollout states satisfying this condition.

G.7 Task 5 as an example of sparse action information

Task 5 can be viewed as a concrete example of a task in which action-informative states occupy little mass. The agent reaches the commanded goal during early training, but the successful behavior is not retained. This makes Task 5 useful for examining what the critic has learned after a successful press has already been observed.

At approximately 100k steps, the critic already distinguishes the completed-success state from the hover state under the commanded goal, with a score gap of approximately $+7.8$. At the same hover state, however, the press-biased action is only rank 16.4 among 34 candidate actions, scores 0.13 below the policy action, and has local gradient alignment -0.21 with the press direction. Action-induced score variation is only 0.075 of the variation across states. The matched Task 6 (`push`) probe shows the opposite pattern, with the push action near rank one, a score advantage of $+2.04$, gradient alignment $+0.86$, and action-to-state score variation of 1.95.

A feature-shuffling probe further shows how the contrastive matching problem can remain well supported by state information. On Task 5, shuffling mechanism xy reduces 256-way HER retrieval accuracy by 0.17, whereas shuffling the action reduces it by only 0.02. The handle xy coordinate is fixed within an episode and shared by a state and its HER future, making it a useful retrieval cue even though it does not identify the pressing action.

Table 7: Local action sensitivity on Task 5 and Task 6 (`push`). The state and goal are held fixed while the action is varied.

Diagnostic	Task 5 (100k)	Task 6 (`push`, 250k)
Correct-action score − policy-action score	-0.13	$+2.04$
Correct-action rank among 34 actions	16.4	1.04
$\cos(\nabla_a f$, correct-action direction)	-0.21	$+0.86$
$\mathrm{std}_a(f)/\mathrm{std}_s(f)$	0.075	1.95

A natural alternative to behavior cloning is to give the successful experience to the critic rather than to the actor. We tested this in a single-seed intervention. At approximately 250.5k steps, 50,100 successful transitions were injected into a replay buffer of roughly 250k transitions, making successful material about 20% of replay. HER success mass increased from approximately 0.03 to 0.15, yet evaluator success subsequently fell from 20% to 10% and then to 0%, while HER success mass decayed to approximately 0.07. More successful futures in replay therefore did not give the critic the action ranking it lacked.

Together, these measurements locate the Task 5 bottleneck in the critic's action ranking rather than in the scarcity of successful data. The critic learns strong state-level matching but separates actions only weakly at the state where pressing matters, and adding successful futures to replay does not restore that local action signal. The gated behavior cloning update in Eq. 8 avoids this bottleneck because it uses a recorded successful action as a direct training target for the policy instead of asking the critic to rank it.

H Parameter-space mixtures versus representation-space adaptation

CKA and DCC both keep a shared component and introduce task-dependent adaptation. They look conceptually similar. However, Table 6 shows that CKA underperforms the critic-only DCC ablation on the hard tasks. Relative to C/R, the ablation improves best success from 0.26 to 0.44 on `handle-press-side` and from 0.40 to 0.52 on `window-close`. We examined why two methods with a similar high-level motivation behave differently in contrastive continual RL.

The key difference is that the two methods act in different spaces. CKA mixes task directions in parameter space,

$$\theta'_k = \theta_{\text{base}} + \sum_j \alpha_{k,j} v_j + v_k, \tag{47}$$

whereas DCC introduces a task-specific state-action encoder directly in representation space. For a contrastive critic, this distinction matters. A parameter-space mixture can locally behave like a nearly state-independent shift in the state-action embedding,

$$z'_{s,a} \approx z_{s,a} + c. \tag{48}$$

Such a shift has limited effect on the objectives that use $z_{s,a}$. In the actor update, adding the same scalar $c^\top \psi(g)$ to all candidate actions does not change the action ranking. In the InfoNCE loss, a shared shift of the logits is also largely removed by the softmax normalization. Thus, part of the critic-side CKA update can lie close to a direction that the contrastive objective does not use effectively. In contrast, the DCC task-specific encoder $\phi^{(k)}_{\text{task}}(s, a)$ depends on the current state and action, so it can change the geometry of the critic representation used for goal-reaching.

The empirical evidence is consistent with this interpretation. We log the end-of-task mixture norm

$$s_\alpha = \sum_j |\alpha_{k,j}| \tag{49}$$

and compare it with end-of-task success across 24 task-seed pairs from a CKA run. As shown in Figure 5, the actor-side mixture norm is only weakly and non-significantly correlated with success ($r = +0.27$, $p = 0.20$). On the critic side, however, the correlation is strongly negative ($r = -0.54$, $p = 0.007$). On the hard tasks, the same pattern appears when comparing the mixture norm with the InfoNCE logsumexp statistic, with Spearman correlations $\rho = -0.62$ on $k = 5$ and $\rho = -0.43$ on $k = 8$.

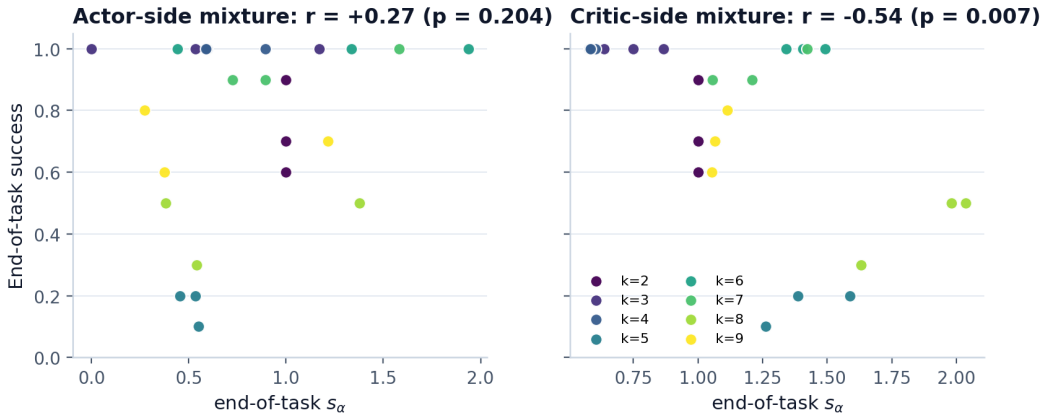

Figure 5: End-of-task mixture norm $s_\alpha = \sum_j |\alpha_{k,j}|$ versus end-of-task success across 24 task-seed pairs from a CKA run, color-coded by task index. The actor-side mixture has a weak, non-significant positive correlation with success ($r = +0.27$, $p = 0.20$), while the critic-side mixture is negatively correlated with success ($r = -0.54$, $p = 0.007$).

This suggests that for the CKA variants the issue is not simply a lack of parameter directions. The mean off-diagonal cosine similarity is around 0.15–0.20 on the actor side and near zero on the critic side, which means the direction pool remains non-degenerate. Instead, the problem is that the critic-side mixture coefficient decays toward zero across the CKA runs we track, and larger critic-side mixture norms correlate with worse success. In other words, CKA provides task-dependent directions in parameter space, but the contrastive critic does not appear to use them as an effective goal-reaching adaptation. This supports the main design choice of DCC: for contrastive critics, task-dependent adaptation is more direct when it changes the state-action representation itself.

I Representation metrics and additional representation studies

I.1 Metric definitions

Let f_θ be a network parameterized by θ, and let $\Phi \in \mathbb{R}^{B\times D}$ denote the matrix whose rows $h_i^\top$ are the penultimate-layer activations of f_θ on a batch $\{x_i\}_{i=1}^B$. For the actor, x_i is the policy input; for the critic, x_i is the state-action concatenation (s_i, a_i). Let $W \in \mathbb{R}^{D\times n}$ denote the final linear layer that maps features to the n-dimensional output of f_θ, and write $\bar{h} = \frac{1}{B}\sum_i h_i$.

Effective rank. Let $\sigma_1 \geq \cdots \geq \sigma_r \geq 0$ be the singular values of the centered feature matrix $\Phi - \mathbf{1}\bar{h}^\top$. Following Yang et al. (2020); Moalla et al. (2024), the approximate effective rank is the smallest k such that

$$\sum_{i=1}^{k} \sigma_i^2 \Big/ \sum_{j=1}^{r} \sigma_j^2 \;\geq\; \beta,$$

with $\beta = 0.99$ in our experiments. A higher effective rank indicates that features span more independent directions.

Weight norm. We report the parameter ℓ_2 norm $\|\theta\|_2 = \sqrt{\sum_i \theta_i^2}$. Monotonic growth in $\|\theta\|_2$ is commonly used as a proxy for parameter drift and loss of plasticity (Nikishin et al., 2022; Lyle et al., 2023).

Feature entropy. We normalize the i-th row of the absolute feature matrix $|\Phi|$ into a probability distribution $p_{i,j} = |\Phi_{i,j}|/(\sum_k |\Phi_{i,k}| + \varepsilon)$ and report

$$\mathcal{H}_{\text{feat}}(\Phi) = \frac{1}{B}\sum_{i=1}^{B}\left(-\sum_j p_{i,j}\log(p_{i,j}+\varepsilon)\right).$$

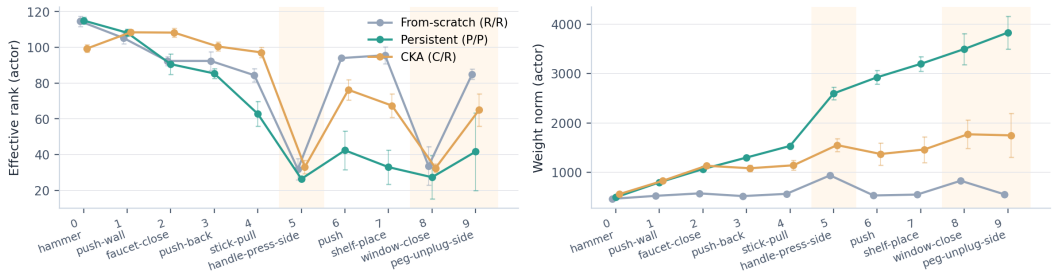

Figure 6: End-of-task actor representation across the main contrastive baseline cells. **Left**: effective rank of the actor penultimate-layer features. **Right**: weight norm of the actor network. The hard tasks coincide with rank collapse, while the persistent baseline shows cumulative weight-norm growth across the curriculum.

Higher entropy indicates a more dispersed, less collapsed representation. Separately, Appendix I.3 reports the entropy of the tanh-Gaussian policy, averaged over the same evaluation batch.

Neural regression collapse (NRC1, NRC2). Following Andriopoulos et al. (2024), we measure two complementary forms of collapse on the penultimate-layer features. Let $\tilde{h}_i = (h_i - \bar{h})/\|h_i - \bar{h}\|_2$ denote the centered and unit-normalized features; let Φ_{PCA_n} be the $D \times n$ matrix whose columns are the top n principal components of the centered feature matrix $\Phi - \mathbf{1}\bar{h}^\top$; and let $\mathrm{proj}(v \mid C)$ denote the projection of v onto the column space of C. NRC1 measures collapse of features onto the n-dimensional principal subspace,

$$\mathrm{NRC1} = \frac{1}{B}\sum_{i=1}^{B}\left\|\tilde{h}_i - \mathrm{proj}(\tilde{h}_i \mid \Phi_{\mathrm{PCA}_n})\right\|_2^2,$$

while NRC2 measures self-duality between features and the column space of the final layer weight,

$$\mathrm{NRC2} = \frac{1}{B}\sum_{i=1}^{B}\left\|\tilde{h}_i - \mathrm{proj}(\tilde{h}_i \mid W)\right\|_2^2.$$

Both quantities lie in $[0, 1]$, with smaller values indicating stronger feature collapse. For the contrastive critic, we use $n = \dim(\mathcal{A})$ on the state-action encoder when computing its NRC1 metric.

I.2 Representation dynamics across the curriculum

We use these diagnostics to examine why carrying the full contrastive agent across tasks can be less effective than resetting it. Figure 6 provides a compact end-of-task view of the actor representation across the main contrastive baseline cells. On the hard tasks, especially $k = 5$ and $k = 8$, actor effective rank drops sharply, while the persistent baseline maintains the lowest rank across much of the later curriculum. Its actor weight norm also grows steadily across the task sequence. Together, these trends are consistent with cumulative loss of representational flexibility under persistence.

Figure 7 gives the corresponding within-task trajectories for both the actor representation and the critic state-action representation. We sample each metric approximately 31 times per task and average across seeds.

Across cells, the hard tasks $k = 5$ and $k = 8$ have consistently lower effective rank than the easier tasks. The persistent cell also shows evidence of cumulative degradation: later tasks begin from a lower rank than the corresponding tasks in the reset cell, even before the within-task training dynamics unfold.

I.3 Additional diagnostics

Weight-norm drift. Figure 8 reports actor and critic weight norms at the end of each task. The persistent cell shows monotonic actor weight-norm growth from approximately 490 at $k = 0$ to approximately 3830 at $k = 9$, an $8\times$ increase across the curriculum. The critic follows a similar trend.

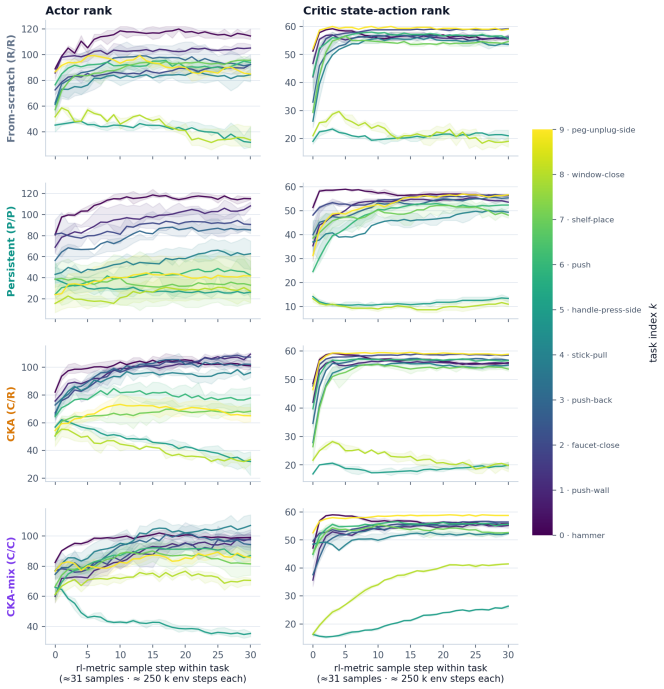

Figure 7: Effective-rank evolution within each task. Rows correspond to baseline cells; columns correspond to the actor representation and the critic state-action representation. Bold curves show the mean across seeds, with interquartile ranges shaded.

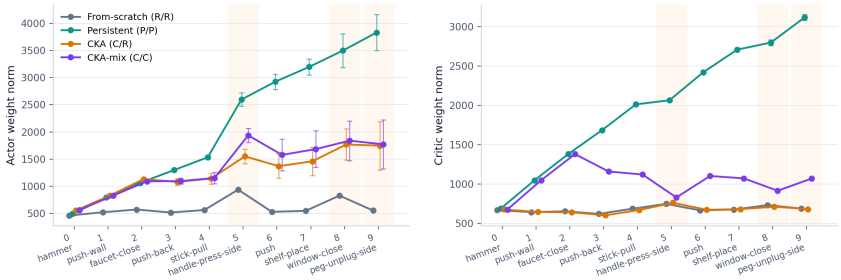

Figure 8: Actor and critic weight norm at the end of each task.

In contrast, the from-scratch cell remains bounded in $[460, 940]$, and the CKA cells lie between the two pure transfer modes. This pattern is consistent with the loss-of-plasticity behavior observed in online RL (Nikishin et al., 2022; Lyle et al., 2022; Dohare et al., 2024).

Neural Regression Collapse. Figure 9 reports NRC1 and NRC2 on the critic state-action representation. NRC1 drops on hard tasks for every cell, with the largest drop in the persistent cell. On $k = 8$, the persistent cell reaches approximately 0.07, compared with approximately 0.42 for the from-scratch cell, suggesting that its critic representation becomes largely collapsed and underutilizes the available feature directions in the latent space. NRC2 also drops for R/R and C/R variants on the two hardest tasks. For P/P and C/C, NRC2 instead rises alongside their growing weight norms, including the final-layer norm.

Actor policy entropy. Figure 10 reports the actor entropy at the end of each task. The persistent cell decreases monotonically from approximately 4.80 at $k = 0$ to approximately 4.21 at $k = 9$, while the from-scratch and CKA cells remain in the range $[4.45, 5.05]$. This suggests that persistence also reduces policy stochasticity over the task sequence. Under sparse rewards, such early commitment to a narrower action distribution can make later recovery more difficult.

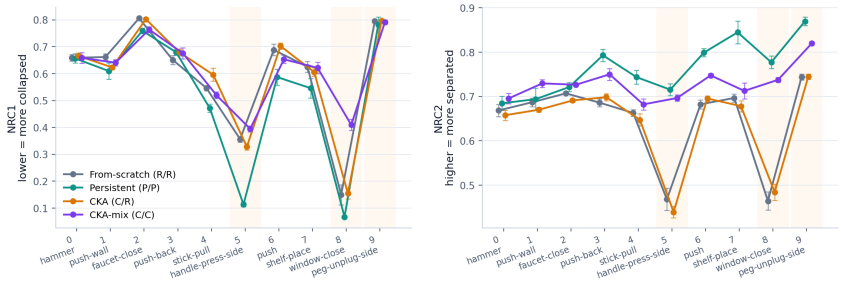

Figure 9: Neural collapse metrics on the critic state-action representation.

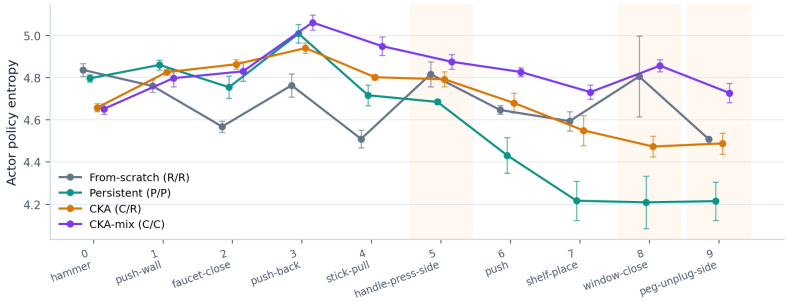

Figure 10: Actor policy entropy at the end of each task.