DCRL: DIRECT CONTRASTIVE REWARD LEARNING FROM CARDINAL TRAJECTORY SCORES

Anonymous authors
Paper under double-blind review

ABSTRACT

Trajectory-level scores evaluate complete behaviors but not individual decisions. We study learning a standalone dense Markov reward from fixed, possibly suboptimal trajectories with cardinal scores, without transition rewards, pairwise preference labels, or additional environment interaction. We present *Direct Contrastive Reward Learning (DCRL)*, which matches similar states across trajectories to derive two local reward constraints. Contrastive ordering compares different actions using trajectory-score differences; smoothness assigns similar rewards to similar state-action pairs. DCRL combines these constraints with trajectory-score fitting, then freezes the learned reward for any downstream reinforcement-learning algorithm that accepts scalar rewards. The ordering is assumed only as a statistical tendency across state-matched comparisons, allowing individual violations. Using 10 suboptimal demonstrations per task on several Gymnasium MuJoCo benchmarks, DCRL outperforms T-REX, R4, and Frozen RRD (a fixed-data version of RRD) in final mean SAC return on every task. It achieves 3.6–10.1 times the return of the best supplied demonstration and reaches 80.0–90.5% of true-reward SAC performance on three tasks. Ablations show that contrastive ordering improves on trajectory-score fitting alone and that smoothness provides a further gain.

1 INTRODUCTION

Learning a dense reward remains a central challenge in reinforcement learning (RL). In many settings, evaluating a complete behavior is easier than assigning reliable feedback to every decision within it. We study reward learning from a fixed set of possibly suboptimal trajectories, each assigned one cardinal score. The goal is to learn a standalone dense proxy reward $f_\phi(s, a)$ for downstream reinforcement learning. Unlike an ordinal ranking, a cardinal score indicates both which trajectory is better and how far apart their observed qualities are.

A trajectory score evaluates the behavior as a whole but does not specify the rewards of its individual state-action pairs. Consequently, many transition-level reward assignments can fit the same trajectory scores while favoring different actions during policy training. Trajectory-score fitting alone therefore cannot determine how reward should be distributed among the decisions within each trajectory.

Among fixed-data methods, T-REX (Brown et al., 2019) and R4 (Kharyal et al., 2026) are closest to DCRL's standalone-reward setting, but neither uses cardinal score gaps. T-REX learns from rankings; R4 learns from ordered rating classes. In our comparison, each demonstration defines one R4 class, preserving only its score ordering. RRD (Ren et al., 2022) normally learns from returns collected during policy training; our *Frozen RRD* baseline uses the same fixed demonstrations and freezes the reward before SAC. PARTED (Xu et al., 2024) begins with fixed trajectories but couples reward learning to its offline policy optimizer. DCRL uses cardinal scores to produce a standalone dense reward for any reinforcement-learning algorithm that accepts scalar rewards.

The key idea is to compare actions taken in similar states. Because a Markov reward depends on both state and action, comparisons between actions are more informative when their states are closely matched. DCRL uses these matches to derive two local constraints. Contrastive ordering encourages actions from higher-scoring trajectories to receive higher rewards when the matched actions differ, while smoothness encourages similar state-action pairs to receive similar rewards. For contrastive ordering, the trajectory-score difference determines the preferred direction and helps set the weight of the comparison.

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.

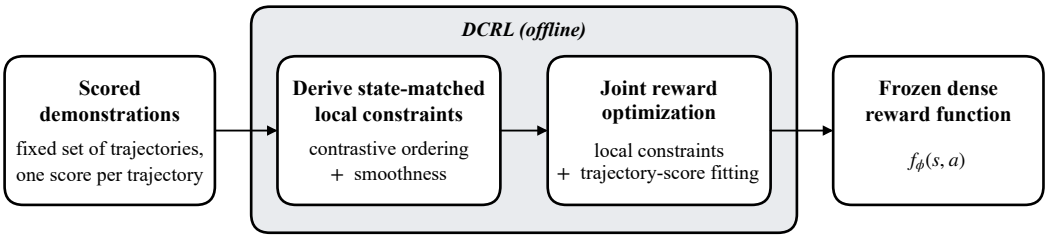

Figure 1: DCRL overview.

These comparisons provide statistical evidence rather than correct labels for every transition. A higher-scoring trajectory may contain a poor action, and a lower-scoring trajectory may contain a good one. DCRL assumes only that, across state-matched comparisons, actions from higher-scoring trajectories tend to have higher rewards. Individual comparisons may violate this ordering.

Figure 1 summarizes the complete pipeline. DCRL combines the derived local constraints with trajectory-score fitting, which matches each trajectory's average predicted reward to its observed score. Reward learning uses only the fixed scored trajectories. The resulting dense reward is then frozen and supplied to a downstream reinforcement-learning algorithm. The local constraints reduce the ambiguity left by trajectory-score fitting, but they do not uniquely identify the true reward or guarantee accurate predictions outside the demonstrated states.

Contributions. We present (1) a novel method for deriving local reward constraints from cardinal-scored trajectories through state-matched contrastive ordering and smoothness; (2) a joint learning objective that combines these constraints with trajectory-score fitting to learn a standalone dense reward from fixed demonstrations; and (3) an evaluation showing that DCRL outperforms T-REX, R4, and Frozen RRD on every evaluated Gymnasium MuJoCo task. DCRL achieves 3.6–10.1 times the return of the best supplied demonstration, while the ablations show that contrastive ordering improves over trajectory-score fitting alone and smoothness provides a further gain.

2 Related Work

Learning from fixed trajectory data. T-REX (Brown et al., 2019) learns a standalone reward from fixed, ranked, potentially suboptimal demonstrations. This closely matches DCRL's input and output, but T-REX uses only their ordering, whereas DCRL uses cardinal score gaps in state-matched comparisons. R4 (Kharyal et al., 2026) learns a standalone reward from ordered rating classes. It samples one trajectory per class, differentiably ranks the predicted returns, and fits the ranks to the class indices. In our comparison, each score-ordered demonstration forms one class, preserving the trajectory ordering but not the cardinal score gaps.

Early et al. (2022) learn non-Markovian rewards from trajectory-return labels and append the reward model's LSTM hidden state to the policy's observation. Preference Transformer (Kim et al., 2023) uses recent state-action history to compute non-Markovian rewards from pairwise preferences; its experiments then train a Markovian offline policy. DCRL computes each reward from the current state and action and does not change the policy's observations.

A further distinction is whether a method produces a standalone reward or a complete policy. PARTED (Xu et al., 2024) decomposes trajectory returns into per-step proxy rewards within its pessimistic offline policy-learning algorithm, while Contrastive Preference Learning (Hejna et al., 2024) learns a policy directly from preferences without an explicit reward. DCRL instead produces a standalone scalar reward for any downstream reinforcement-learning algorithm that accepts such rewards.

Learning with additional environment interaction. D-REX (Brown et al., 2020) and SSRR (Chen et al., 2021) train an initial policy from demonstrations and run perturbed versions to generate additional trajectories for reward learning. RRD (Ren et al., 2022) and GP-LRR (Xiao & Yu, 2026) learn dense proxy rewards while an agent gathers trajectories for which only episodic returns are observed. These published procedures use additional interaction, unlike DCRL. Because RRD's

loss can also be trained on fixed trajectory-return pairs, we include a fixed-data adaptation called *Frozen RRD*; Section 5 gives the experimental definition. Together, T-REX, R4, and Frozen RRD compare ranking, rating, and return supervision using the same demonstrations and downstream SAC procedure.

Preference-based reward learning uses pairwise judgments between trajectory segments (Christiano et al., 2017; Ibarz et al., 2018). Sadigh et al. (2017) select informative comparisons, while Bıyık et al. (2020) also consider how easily a person can answer them. PEBBLE relabels collected experience as its reward changes and uses unsupervised pretraining (Lee et al., 2021). SURF starts with limited teacher-labeled comparisons, pseudo-labels additional pairs only when its reward model is confident, and augments pairs by temporal cropping (Park et al., 2022). Knox et al. (2024) show that the model of human choice affects the recovered reward. These methods rely on pairwise preference labels; DCRL uses fixed trajectories with cardinal scores.

3 Problem Setup

3.1 Formal setting

Let $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ denote the state and action spaces. The observable input to DCRL is a fixed dataset $\mathcal{D} = \{(\tau_i, z_i)\}_{i=1}^{N}$. For each trajectory i, we observe an ordered sequence of state-action pairs $\tau_i = \{(s_{i,t}, a_{i,t})\}_{t=1}^{T_i}$ and one scalar quality score $z_i \in \mathbb{R}$ assigned to the trajectory as a whole; no reward is observed at any individual transition. The dataset therefore provides $n = \sum_{i=1}^{N} T_i$ observed state-action pairs but only N trajectory-level scores. Let $\mathbf{z} = (z_1, \ldots, z_N)$ denote the vector of observed scores.

Throughout, i and j index trajectories, while t and t' index state-action observations in trajectories i and j, respectively. We use "transition (i, t)" as shorthand for the observed pair $(s_{i,t}, a_{i,t})$; t' is an index in the second trajectory, not a next-state variable. DCRL assumes that the scores are cardinal and expressed on a common scale: $z_i > z_j$ indicates that trajectory i has higher quality than trajectory j, while $|z_i - z_j|$ measures the observed quality difference. The score values constrain trajectory-level predictions, while the signs and magnitudes of score differences determine the direction and relative weight of comparisons between observed transitions. DCRL learns the reward function $f_\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. For trajectory i, its length-normalized predicted score is

$$\hat{z}_i(f_\phi) = \frac{1}{T_i} \sum_{t=1}^{T_i} f_\phi(s_{i,t}, a_{i,t}). \tag{1}$$

Length normalization prevents the predicted score from increasing solely with trajectory length. It defines how DCRL predicts the supplied score, rather than assuming that the score represents an average of true transition rewards. Once learned, $f_\phi(s, a)$ is frozen and used as the per-step reward by a downstream reinforcement-learning algorithm. We use SAC (Haarnoja et al., 2018) in the experiments, although the learned reward can be used with any algorithm that accepts a scalar Markov reward.

3.2 Trajectory-Score Fitting and Transition-Level Ambiguity

Each trajectory score constrains the average reward predicted across that trajectory, but does not specify the reward of any individual transition. Different reward functions can therefore match the same trajectory averages while assigning different rewards to individual state-action pairs and potentially favoring different actions in downstream RL. DCRL retains trajectory-score fitting and supplements it with local reward constraints inferred from the same scores.

4 The DCRL Method

4.1 Core principle

DCRL constructs these local reward constraints by comparing actions observed at similar states in different trajectories. State matching is important because a Markov reward depends on both state

and action; when the state inputs are similar, differences between the predicted rewards provide more information about the actions. The matched transitions need not occur at the same timestep. When the corresponding trajectories have different scores, DCRL uses the score difference as statistical evidence for ordering the rewards of the observed actions. Because each score evaluates an entire trajectory rather than a single transition, this ordering need not hold for every matched pair. DCRL therefore relies on the following statistical assumption.

Assumption 1 (Statistical state-matched reward ordering)**.** *Across state-matched comparisons, actions from higher-scoring trajectories tend to have higher rewards, although individual comparisons may violate this ordering.*

Because a trajectory score summarizes many transitions, a higher-scoring trajectory may still contain an action with lower reward than one observed at a similar state in a lower-scoring trajectory. DCRL therefore treats the resulting comparisons as noisy local reward constraints. State similarity and the trajectory-score gap determine the strength of the score-based evidence. State matching aims to reduce differences attributable to the state, assuming that the matching representation preserves reward-relevant similarities. A larger score gap indicates a larger observed difference in trajectory quality. Action similarity then determines how this evidence is used: differences between the actions make a contrastive ordering constraint informative, whereas nearly identical actions support state-action smoothness. DCRL bounds the effect of the score gap with a $\tanh$ factor so that a small number of extreme gaps cannot dominate training. The following sections define the representation used for state matching, formalize how transitions are matched, and specify the resulting learning objective.

Symbol	Meaning
$\psi(s)$	state representation used for cross-trajectory matching
$\rho_{i,t}$	density-adaptive local radius of transition (i,t)
$\rho_{i,t;\,j,t'}$	symmetric radius for transition pair $(i,t;\,j,t')$
$m_{i,t;\,j,t'}$	state-similarity weight for the matched transition pair
$\kappa_{i,t;\,j,t'}$	action-similarity weight for the matched transition pair
$w_{i,t;\,j,t'}$	contrastive ordering weight
Q	robust scale for trajectory-score differences
$\mathcal{P}$	retained set of state-matched cross-trajectory transition pairs

Table 1: Core notation used in the DCRL construction. Full notation is provided in Appendix B.

4.2 State representation for matching

DCRL compares states across trajectories, so it needs a distance measure that remains meaningful when state coordinates have different scales and are correlated. We first standardize each coordinate using statistics computed from all demonstration states, preventing coordinates with larger numerical ranges from dominating the distance. We then apply PCA to reduce the number of dimensions used for neighbor search while retaining the main variation in the demonstrations. Before computing distances, we normalize each projected state:

$$\psi(s) = \frac{\Pi_d \tilde{s}}{\|\Pi_d \tilde{s}\|_2 + \epsilon}. \tag{2}$$

Here $\tilde{s}$ is the standardized state, Π_d is the PCA projection with output dimension d, and ϵ is a normalization constant. The normalization prevents the length of the projected vector from dominating state similarity. The standardization statistics and PCA projection are computed once from the fixed demonstration set. The resulting representation is used only for state matching; the reward model continues to operate on the original state-action inputs. Appendix A gives the dimension rule; Appendix D.1 lists its values.

4.3 State-matched transition-pair construction

DCRL retains a compact set of cross-trajectory transition pairs for reward training. For each anchor transition, it considers candidates from trajectories with different scores, prioritizes those with more similar states and larger score differences, and retains the top k. For each retained pair, action

similarity controls the balance between the two losses: pairs with similar actions are weighted more strongly toward matching their predicted rewards, whereas pairs with different actions are weighted more strongly toward contrastive ordering.

State similarity. State matching uses the representation ψ defined in Section 4.2. Demonstration coverage may vary substantially across this space: states are tightly packed in frequently visited regions and farther apart in sparsely visited regions. A single state radius would therefore blur distinctions in dense regions or suppress nearly all transition pairs in sparse regions.

Let k be the maximum number of candidates retained for each anchor transition and set $q = \min\{\lceil k/2 \rceil, n-1\}$. This choice ties the density estimate to the candidate budget without introducing a separate neighborhood-size hyperparameter. For transition (i,t), let $\mathrm{dist}_{i,t}^{(q)}$ denote its distance to the q-th nearest demonstration transition in ψ-space, excluding itself. The local state radius is $\rho_{i,t} = \max\{\mathrm{dist}_{i,t}^{(q)}, \epsilon_\rho\}$, with $\epsilon_\rho = 10^{-3}$. The neighbor search includes all demonstration transitions, including those from trajectory i, because it estimates local state coverage; retained transition pairs are restricted to different trajectories. The positive floor prevents duplicate or nearly identical embeddings from producing a zero radius.

For a candidate transition pair $(i,t;\, j,t')$, the two local radii are combined symmetrically as $(\rho_{i,t;\,j,t'})^2 = \frac{1}{2}\left(\rho_{i,t}^2 + \rho_{j,t'}^2\right)$. The corresponding state-similarity weight is

$$m_{i,t;\,j,t'} = \exp\left(-\frac{\|\psi(s_{i,t}) - \psi(s_{j,t'})\|_2^2}{2\left(\rho_{i,t;\,j,t'}\right)^2}\right). \tag{3}$$

Thus, $m_{i,t;\,j,t'}$ measures state distance relative to the local coverage around both transitions. For each anchor (i,t), let $\mathcal{C}_{i,t}$ contain the transitions (j,t') satisfying $j \neq i$ and $z_j \neq z_i$. DCRL ranks the candidates in $\mathcal{C}_{i,t}$ by

$$m_{i,t;\,j,t'} \tanh\left(\frac{|z_i - z_j|}{Q}\right), \qquad Q = \max\left\{\mathrm{IQR}(\mathbf{z}), 10^{-2}\right\},$$

and retains the top $\min\{k, |\mathcal{C}_{i,t}|\}$ transition pairs in $\mathcal{P}_{\mathrm{raw}}$. The IQR scales score differences by the central spread of the observed scores and is less sensitive to isolated extreme values; the lower bound prevents a near-zero scale. The state term favors comparable transitions, while the score-gap term favors transition pairs carrying stronger ordering information.

Action similarity. After the state-matched transition pairs are selected, actions are standardized coordinate-wise as $\tilde{a}_\ell = (a_\ell - \mu_\ell^a)/(\sigma_\ell^a + \epsilon)$ for $\ell = 1, \ldots, d_a$, using statistics computed over all demonstration actions. This prevents coordinates with larger numerical ranges from dominating action distance.

DCRL uses one global action radius so that the same standardized action gap is treated the same everywhere. A local radius would change its similarity score from one state region to another. The global radius is computed from the action distances of the retained transition pairs:

$$\rho_a = \max\left\{\mathrm{Quantile}_{\alpha_a}\left(\{\|\tilde{a}_{i,t} - \tilde{a}_{j,t'}\|_2 : (i,t;\,j,t') \in \mathcal{P}_{\mathrm{raw}}\}\right),\, 0.1\sqrt{d_a}\right\}.$$

Here, α_a is a lower quantile level: after the retained action distances are sorted, the quantile is the distance below which approximately an α_a fraction of them falls. Using a lower quantile keeps action similarity selective, so the transition pairs with the most similar actions receive the strongest smoothness weight while larger action gaps retain stronger contrastive weight. The value of α_a is listed in Appendix D.1. The floor $0.1\sqrt{d_a}$ corresponds to an RMS standardized action difference of 0.1 per coordinate, so the minimum radius scales with action dimension and cannot collapse to zero. The resulting action-similarity weight is

$$\kappa_{i,t;\,j,t'} = \exp\left(-\frac{\|\tilde{a}_{i,t} - \tilde{a}_{j,t'}\|_2^2}{2\rho_a^2}\right). \tag{4}$$

Similar actions produce $\kappa_{i,t;\,j,t'}$ near one, whereas different actions produce values near zero. Section 4.4 weights smoothness by $m_{i,t;\,j,t'}\kappa_{i,t;\,j,t'}$ and contrastive ordering by $m_{i,t;\,j,t'}(1 - \kappa_{i,t;\,j,t'})$, together with the score-gap factor. Intermediate action similarities contribute continuously to both penalties. The final transition-pair set $\mathcal{P}$ and its weights are computed once before reward training.

4.4 LEARNING OBJECTIVE

DCRL combines three complementary terms. The trajectory-score fitting term matches each demonstration's average predicted reward to its observed score. The contrastive term orders predicted rewards for selected transitions from differently scored trajectories, while the state-action smoothness term penalizes differences between predictions for transition pairs whose states and actions are both similar. We define the full-data objective below; Algorithm 1 optimizes its minibatch estimate.

Trajectory-score fitting loss. For any reward function g, let $\hat{z}_i(g)$ denote its length-normalized trajectory prediction, defined as in Equation (1). A squared fitting loss can give unusually large trajectory-level residuals excessive influence. DCRL therefore applies the unit-threshold Huber loss to the score-normalized residual:

$$H_1(v) = \begin{cases} \frac{1}{2}v^2, & |v| \le 1, \\ |v| - \frac{1}{2}, & |v| > 1, \end{cases} \qquad \delta = \max\{\mathrm{std}(\mathbf{z}), 10^{-2}\}. \tag{5}$$

For a raw residual x, the normalized loss $H_1(x/\delta)$ is quadratic when $|x| \le \delta$ and linear beyond that range. The scale δ sets this transition relative to the observed score variation and normalizes the trajectory-score fitting term, while the lower bound prevents division by a near-zero score scale. DCRL fits the length-normalized prediction $\hat{z}_i(f_\phi)$ to the observed score z_i using

$$\mathcal{L}_{\mathrm{MLE}}(\phi) = \frac{1}{N}\sum_{i=1}^{N} H_1\left(\frac{z_i - \hat{z}_i(f_\phi)}{\delta}\right). \tag{6}$$

Up to constants independent of ϕ, the Huber term corresponds to a robust likelihood with a Gaussian core and Laplace tails (Huber, 1964). This loss anchors each trajectory's average prediction but does not determine how reward is distributed among its individual transitions.

State-matched contrastive ordering loss. For a retained transition pair $(i,t;\, j,t')$, define $\Delta f_{i,t;\,j,t'} = f_\phi(s_{i,t}, a_{i,t}) - f_\phi(s_{j,t'}, a_{j,t'})$ and $y_{ij} = \mathrm{sgn}(z_i - z_j)$. Equal-score trajectories are excluded during transition-pair construction, so $y_{ij} \in \{-1, 1\}$. The signed margin $y_{ij}\Delta f_{i,t;\,j,t'}$ is positive when the transition from the higher-scoring trajectory receives the higher predicted reward and negative when the predicted ordering is reversed.

DCRL uses the standard Bradley–Terry log-sigmoid loss (Bradley & Terry, 1952):

$$\mathcal{L}_{\mathrm{contrast}}(\phi) = \frac{1}{|\mathcal{P}|}\sum_{(i,t;\,j,t')\in\mathcal{P}} w_{i,t;\,j,t'}\left[-\log\varsigma(y_{ij}\Delta f_{i,t;\,j,t'})\right], \tag{7}$$

where $\varsigma(x) = 1/(1+e^{-x})$. The sigmoid gives the probability assigned to the correct ordering, and the leading minus sign converts its non-positive log-probability into a non-negative loss. The loss is large when the ordering is wrong and decreases smoothly as the correctly ordered margin increases, enforcing an ordering without prescribing a fixed reward gap.

Each comparison is weighted by

$$w_{i,t;\,j,t'} = \underbrace{m_{i,t;\,j,t'}}_{\substack{\text{state}\\ \text{similarity}}}\underbrace{(1-\kappa_{i,t;\,j,t'})}_{\substack{\text{action}\\ \text{difference}}}\underbrace{\tanh(|z_i - z_j|/Q)}_{\text{score separation}}. \tag{8}$$

Equation (8) assigns greater weight to pairs with similar states, dissimilar actions, and a large trajectory-score gap. Because the factors are multiplied, a transition pair is downweighted when any condition is weak. The $\tanh$ term increases with score separation while preventing unusually large gaps from dominating. Thus, $w_{i,t;\,j,t'}$ controls the weight assigned to the inferred ordering, not the desired magnitude of the reward difference.

State-action smoothness loss. The trajectory-score loss constrains only the average reward of each demonstration, so different transition-level assignments can fit the same score. The contrastive ordering loss reduces this ambiguity when the two actions differ, but it is intentionally downweighted when the actions are similar. DCRL therefore penalizes differences between the predicted rewards of similar state-action pairs:

$$\mathcal{L}_{\mathrm{smooth}}(\phi) = \frac{1}{|\mathcal{P}|}\sum_{(i,t;\,j,t')\in\mathcal{P}} m_{i,t;\,j,t'}\kappa_{i,t;\,j,t'}\left(\Delta f_{i,t;\,j,t'}\right)^2. \tag{9}$$

The product $m_{i,t;\,j,t'}\kappa_{i,t;\,j,t'}$ gives greatest weight to pairs whose states and actions are both similar. No score-gap factor appears because this term does not impose an ordering; score gaps have already influenced pair retention. Pairs with different actions are handled primarily by the contrastive ordering loss. Algorithm 1 applies both terms during training.

Composite objective. Combining the three terms gives

$$\mathcal{L}(\phi) = \mathcal{L}_{\mathrm{MLE}}(\phi) + \lambda_c\,\mathcal{L}_{\mathrm{contrast}}(\phi) + \lambda_s\,\mathcal{L}_{\mathrm{smooth}}(\phi). \tag{10}$$

Here, $\lambda_c, \lambda_s \geq 0$ weight the two pairwise penalties. Equation (10) is a *penalized maximum-likelihood objective*: Huber score fitting anchors each trajectory's average prediction, while state-matched ordering and smoothness constrain how reward is distributed across demonstrated decisions. Algorithm 1 in Appendix A summarizes pair construction, reward learning, and downstream use without introducing any operation beyond the objective above.

5 Empirical Results

Experimental setup. We evaluate DCRL on four continuous-control tasks from Gymnasium MuJoCo: Hopper-v5, Ant-v5, Walker2d-v5, and HalfCheetah-v5. Ant-v5 uses the 27-dimensional observation without external contact forces (`cfrc_ext`). We collect demonstrations by running SAC policies saved during early stages of training and select 10 suboptimal trajectories per task. Each trajectory is scored by its total undiscounted episode return. Table 2 reports their return ranges. We compare DCRL with T-REX (Brown et al., 2019), RRD (Ren et al., 2022), and R4 (Kharyal et al., 2026) using the same demonstrations. DCRL and RRD use the trajectory scores directly, while T-REX uses their induced ranking and R4 converts the same ranking into ordinal rating classes, one class per demonstration. All four reward models are trained only on these demonstrations and then frozen before downstream RL training. We call RRD under this procedure *Frozen RRD*, and we run R4 in its full-ranking setting. We omit T-REX's hand-specified control-cost term so that it receives no additional component of the environment reward. Appendix C describes the baseline implementations.

Evaluation. We test whether SAC trained with DCRL's learned reward achieves higher true episode returns than SAC trained with T-REX, Frozen RRD, or R4, and whether it outperforms the supplied demonstrations. SAC (Haarnoja et al., 2018) serves as a representative downstream RL algorithm, with identical policy-training hyperparameters across methods. We also include SAC trained with the true environment reward as a performance reference. We compare DCRL's predicted episode returns with the true returns and examine the contributions of the contrastive and smoothness terms to SAC performance. Results report means and standard deviations across five random seeds. In Figures 2–4, curves show exponentially smoothed means, and shaded bands indicate ± 1 standard deviation. Appendix D reports the DCRL hyperparameters, SAC settings, and evaluation protocol.

Environment	Hopper-v5	Ant-v5	Walker2d-v5	HalfCheetah-v5
Return range	$[192.4,\ 792.0]$	$[-51.0,\ 506.7]$	$[101.1,\ 890.4]$	$[-180.9,\ 1431.4]$

Table 2: Undiscounted episodic-return range of the 10 demonstrations supplied to every method.

	Hopper-v5	Ant-v5	Walker2d-v5	HalfCheetah-v5
True-reward SAC	3621 ± 124	5818 ± 1366	5176 ± 457	15134 ± 1217
DCRL	$\mathbf{3276 \pm 266}$	$\mathbf{5096 \pm 1033}$	$\mathbf{3177 \pm 938}$	$\mathbf{12110 \pm 821}$
Frozen RRD	2059 ± 563	3303 ± 904	1263 ± 403	11443 ± 962
R4	1428 ± 545	881 ± 48	1051 ± 197	-1 ± 493
T-REX	2427 ± 782	278 ± 142	348 ± 433	1196 ± 1136
Best demonstration	792	507	890	1431

Table 3: Final true return after 3M SAC steps (last-10k mean $\pm$ SD over five seeds). Bold marks the best learned-reward mean; the final row reports the best demonstration.

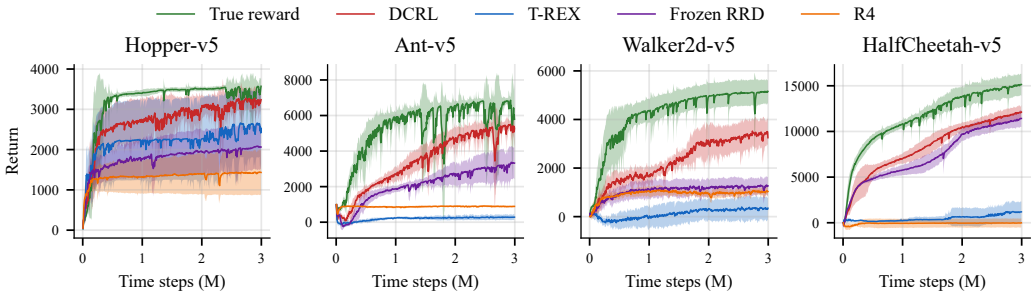

Figure 2: Frozen-reward comparison, with true-reward SAC as a reference.

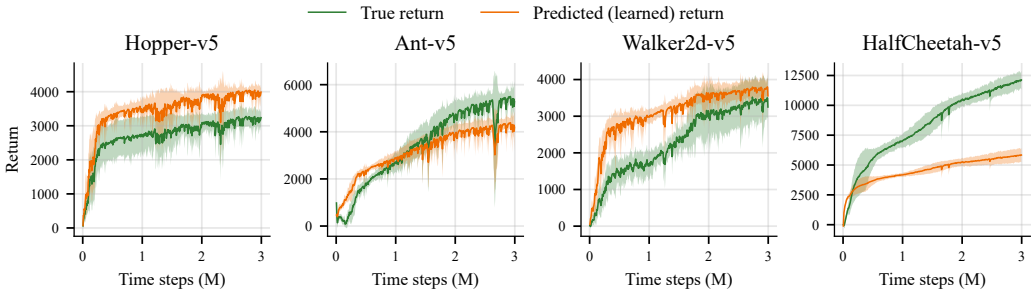

Figure 3: DCRL evaluation returns: true versus predicted.

Comparison with learned-reward baselines. DCRL outperforms T-REX, Frozen RRD, and R4 in final mean return on all four tasks when their learned rewards are evaluated using the same SAC training procedure (Table 3). Frozen RRD is the strongest baseline on Ant, Walker2d, and HalfCheetah, and T-REX is the strongest on Hopper. Compared with Frozen RRD, DCRL achieves approximately 1.6 times the return on Hopper, 1.5 times on Ant, and 2.5 times on Walker2d. The difference is smaller on HalfCheetah, where DCRL reaches 12,110 compared with 11,443 for Frozen RRD—approximately 6% higher, with overlapping standard-deviation bands. Compared with R4, DCRL achieves approximately 2.3 times the return on Hopper, 5.8 times on Ant, and 3.0 times on Walker2d, while R4 finishes near zero return on HalfCheetah. Figure 2 shows that the methods differ in how much further performance improves after the initial stages of training. On Hopper, T-REX improves rapidly at first, but its progress slows while DCRL continues to improve. On Ant and Walker2d, the gap between DCRL and Frozen RRD widens during training, whereas T-REX and R4 make limited progress. HalfCheetah presents a closer comparison: both DCRL and Frozen RRD continue improving and achieve substantially higher returns than T-REX and R4.

Training with the true environment reward provides an upper reference that no learned reward matches. DCRL comes closest to it on all four tasks, reaching 90.5% of the reference return on Hopper, 87.6% on Ant, and 80.0% on HalfCheetah, where the next-best method reaches 67.0%, 56.8%, and 75.6%. Walker2d leaves the largest gap for every method: DCRL reaches 61.4% of the reference return and the next-best method 24.4%.

Improvement beyond the demonstrations. DCRL achieves returns far above the best supplied demonstration on every task: its final mean return is 4.1 times the best demonstration return on Hopper, 10.1 times on Ant, 3.6 times on Walker2d, and 8.5 times on HalfCheetah (Table 3). These improvements exceed those achieved by T-REX, Frozen RRD, or R4 on all four tasks, with the largest relative advantage over Frozen RRD on Walker2d and the smallest on HalfCheetah. R4 exceeds the best demonstration on Hopper, Ant, and Walker2d, but not on HalfCheetah. The results demonstrate that DCRL learns a reward from suboptimal demonstrations that supports substantial further improvement during SAC training, without receiving scores for newly collected trajectories.

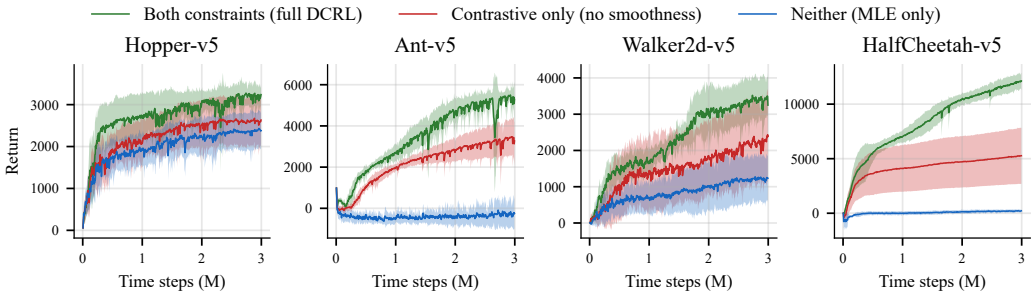

Figure 4: Objective ablation of the DCRL loss.

	Hopper-v5	Ant-v5	Walker2d-v5	HalfCheetah-v5
Spearman ρ	0.988 ± 0.004	0.984 ± 0.003	0.985 ± 0.010	0.999 ± 0.001
$\hat{R}/R$ in the final window	1.23	0.79	1.10	0.48

Table 4: Agreement between DCRL's learned return $\hat{R}$ and true return R on evaluation episodes. Spearman ρ is mean $\pm$ SD across five seeds; $\hat{R}/R$ is measured over the last 10,000 steps.

Agreement between learned and true returns. We next examine whether DCRL's learned reward reflects the improvement measured by the true environment reward. Figure 3 compares the sum of learned rewards with the true return on the same evaluation episodes. Both generally increase as training progresses on all four tasks. Mean Spearman correlations range from 0.984 to 0.999 (Table 4), indicating strong agreement in how the two returns rank the evaluated episodes. We compute these correlations using at most 10 episodes from each of 10 equal-width true-return intervals per seed, reducing the influence of the many similar episodes collected late in training.

DCRL's learned returns track improvements in true return despite differences in scale. On HalfCheetah, the mean rank correlation reaches 0.999, even though the final learned return is only 48% of the true return. This shows that the learned reward can distinguish better-performing episodes without matching their numerical returns. The correlation measures agreement across the sampled performance range, rather than among high-return episodes alone.

Contribution of the local reward constraints. Figure 4 compares trajectory-score fitting alone, score fitting with contrastive ordering, and full DCRL. Adding contrastive ordering improves the final mean return on every task. The gains are especially pronounced on Ant and HalfCheetah, where trajectory-score fitting alone produces little improvement. Adding smoothness further increases the final mean return on all four tasks, with full DCRL achieving the highest performance in each comparison. These results support combining both local constraints with trajectory-score fitting. The smoothness comparison measures its benefit when added to contrastive ordering; its effect without contrastive ordering is not evaluated.

6 Conclusion

DCRL learns a standalone dense Markov reward from fixed cardinal-scored trajectories by combining state-matched contrastive ordering and smoothness constraints with trajectory-score fitting. Reward learning requires neither transition-level feedback nor additional environment interaction. With the same demonstrations and downstream SAC procedure, DCRL outperforms T-REX, R4, and Frozen RRD in final mean return on all four Gymnasium MuJoCo tasks, achieving 3.6–10.1 times the return of the best supplied demonstration. Ablations show that contrastive ordering improves performance beyond score fitting alone and that smoothness yields further gains. Overall, DCRL shows that state-matched reward constraints can strengthen learning from trajectory-level scores and support downstream performance substantially beyond the supplied demonstrations.

AI USE STATEMENT

Generative AI assisted with literature review, code implementation, and polishing the paper's presentation. The authors reviewed all AI-assisted material and take full responsibility for the paper, including its code, claims, equations, and results.

REFERENCES

Erdem Bıyık, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1177–1190, 2020.

Ralph A. Bradley and Milton E. Terry. Rank analysis of incomplete block designs. *Biometrika*, 39 (3/4):324–345, 1952.

Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 783–792, 2019.

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 330–359, 2020.

Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Proceedings of the Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1262–1277, 2021.

Paul Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4299–4307, 2017.

Joseph Early, Tom Bewley, Christine Evers, and Sarvapali Ramchurn. Non-Markovian reward modelling from trajectory labels via interpretable multiple instance learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27652–27663, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018.

Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W. Bradley Knox, and Dorsa Sadigh. Contrastive preference learning: Learning from human feedback without reinforcement learning. In *International Conference on Learning Representations*, 2024.

Peter J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1): 73–101, 1964.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*, volume 31, pp. 8011–8023, 2018.

Chaitanya Kharyal, Calarina Muslimani, and Matthew E. Taylor. Reward learning through ranking mean squared error. In *Proceedings of the International Conference on Machine Learning*, 2026.

Changyeon Kim, Jongjin Park, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. Preference transformer: Modeling human preferences using transformers for RL. In *International Conference on Learning Representations*, 2023.

W. Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, and Alessandro Allievi. Models of human preference for learning reward functions. *Transactions on Machine Learning Research*, 2024.

Kimin Lee, Laura M. Smith, and Pieter Abbeel. PEBBLE: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *Proceedings of the International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6152–6163, 2021.

Jongjin Park, Younggyo Seo, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. SURF: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. In *International Conference on Learning Representations*, 2022.

Zhizhou Ren, Ruihan Guo, Yuan Zhou, and Jian Peng. Learning long-term reward redistribution via randomized return decomposition. In *International Conference on Learning Representations*, 2022.

Dorsa Sadigh, Anca D. Dragan, Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems XIII*, 2017.

Minheng Xiao and Xian Yu. Reward redistribution via Gaussian process likelihood estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(32):27073–27080, 2026.

Tengyu Xu, Yue Wang, Shaofeng Zou, and Yingbin Liang. Provably efficient offline reinforcement learning with trajectory-wise reward. *IEEE Transactions on Information Theory*, 70(9):6481–6518, 2024.

A FULL DCRL ALGORITHM

Algorithm 1 learns a reward from the fixed scored demonstrations. The matching representations are used only to construct transition pairs and weights; the reward model takes the original state-action inputs. Downstream policy training is shown separately and does not update the learned reward. Appendix D.1 lists the values used in the experiments.

Matching representation. State and action coordinates are standardized using their respective demonstration means and standard deviations: $\tilde{s}_\ell = (s_\ell - \mu^s_\ell)/(\sigma^s_\ell + \epsilon)$ and $\tilde{a}_\ell = (a_\ell - \mu^a_\ell)/(\sigma^a_\ell + \epsilon)$, with the same ϵ as in Equation (2). PCA is fitted to the standardized demonstration states. Let r be the number of available PCA components and $d_{\eta_{\mathrm{PCA}}}$ the smallest number retaining at least the fraction η_{PCA} of the total variance. Given a minimum dimension $d_{\min}$, the PCA output dimension is

$$d = \min\{r, \max\{d_{\min}, d_{\eta_{\mathrm{PCA}}}\}\}, \tag{11}$$

so all available components are retained when fewer than $d_{\min}$ exist. These transformations are used only for matching.

Algorithm 1 DCRL: transition-pair construction, reward learning, and downstream use.

Require: $\{(\tau_i, z_i)\}_{i=1}^N$ with at least one eligible cross-trajectory pair; integer $k \geq 1$, α_a, η_{PCA}, $d_{\min}$
Require: $\lambda_c, \lambda_s \geq 0$; transition-pair batch size B; trajectory batch size B_τ; reward-training steps T_{train}; AdamW learning rate η and weight decay λ_{wd}

$n \leftarrow \sum_i T_i$; $\epsilon \leftarrow 10^{-8}$; $\delta \leftarrow \max(\mathrm{std}(\mathbf{z}), 10^{-2})$; $Q \leftarrow \max(\mathrm{IQR}(\mathbf{z}), 10^{-2})$
Standardize demonstration states and actions coordinate-wise to obtain $\tilde{s}$ and $\tilde{a}$
Compute PCA and choose d by Equation (11)
$\psi(s) \leftarrow \Pi_d \tilde{s}/(\|\Pi_d \tilde{s}\|_2 + \epsilon)$ for all demonstration states
$q \leftarrow \min\{\lceil k/2 \rceil, n-1\}$; compute $\rho_{i,t} \leftarrow \max(\mathrm{dist}^{(q)}_{i,t}, 10^{-3})$ for every transition
Pass 1: state-matched candidates
$\mathcal{P}_{\mathrm{raw}} \leftarrow \emptyset$
for each anchor (i,t) **do**
  for each candidate (j,t') with $j \neq i$ and $z_j \neq z_i$ **do**
    $\left(\rho_{i,t;\,j,t'}\right)^2 \leftarrow (\rho^2_{i,t} + \rho^2_{j,t'})/2$
    $m \leftarrow \exp(-\|\psi(s_{i,t}) - \psi(s_{j,t'})\|_2^2/(2\left(\rho_{i,t;\,j,t'}\right)^2))$
    $c \leftarrow m \tanh(|z_i - z_j|/Q)$
  end for
  Add the top k candidate pairs by c (or all if fewer than k) to $\mathcal{P}_{\mathrm{raw}}$, storing their m values
end for
$\rho_a \leftarrow \max\big(\mathrm{Quantile}_{\alpha_a}\{\|\tilde{a}_{i,t} - \tilde{a}_{j,t'}\|_2 : (i,t;\,j,t') \in \mathcal{P}_{\mathrm{raw}}\}, 0.1\sqrt{d_a}\big)$
Pass 2: action weights
$\mathcal{P} \leftarrow \emptyset$
for each $(i,t;\,j,t';m) \in \mathcal{P}_{\mathrm{raw}}$ **do**
  $\kappa \leftarrow \exp(-\|\tilde{a}_{i,t} - \tilde{a}_{j,t'}\|_2^2/(2\rho_a^2))$
  $w \leftarrow m(1-\kappa)\tanh(|z_i - z_j|/Q)$; $y \leftarrow \mathrm{sgn}(z_i - z_j)$
  Add $(i,t;\,j,t';m,\kappa,w,y)$ to $\mathcal{P}$
end for
Initialize f_ϕ
for step $= 1, \ldots, T_{\mathrm{train}}$ **do**
  Draw a pair minibatch $\mathcal{B}$ of B pairs i.i.d. from $\mathcal{P}$, and a trajectory minibatch $\mathcal{B}_\tau$ of B_τ trajectories without replacement
  Compute the mean trajectory-score loss over $\mathcal{B}_\tau$ using Equation (6), including its residual scale δ
  Compute the contrastive and smoothness losses as means over $\mathcal{B}$ using Equations (7) and (9)
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{MLE}} + \lambda_c \mathcal{L}_{\mathrm{contrast}} + \lambda_s \mathcal{L}_{\mathrm{smooth}}$; update ϕ with AdamW
end for
Freeze f_ϕ
Output of reward learning: frozen reward function f_ϕ
Optional downstream use (SAC in the experiments)
for each downstream RL interaction **do**
  Use $r_t^{\mathrm{DCRL}} \leftarrow f_\phi(s_t, a_t)$ as the per-step reward and update the chosen RL agent
end for
return the downstream policy π; f_ϕ remains fixed

B FULL METHOD NOTATION

Symbol	Meaning	Source
N, T_i, n	trajectories, trajectory length, total transitions	Sec. 3.1
τ_i, z_i	trajectory i and its observed cardinal score	Sec. 3.1
f_ϕ, $\hat{z}_i$	learned reward and predicted trajectory score	Eq. (1)
d_s, d_a	state and action dimensions	Sec. 3.1
$\mathbf{z}$	vector of observed trajectory scores	Sec. 3.1
$\tilde{s}$, $\tilde{a}$	standardized state, standardized action	Secs. 4.2, 4.3
Π_d, d	PCA projection, output dimension	Sec. 4.2, Eq. (11)
η_{PCA}, $d_{\min}$	retained variance threshold, minimum PCA dimension	Eq. (11)
ϵ, ϵ_ρ	normalization constant, state-radius floor	Eq. (2), Sec. 4.3
$\psi(s)$	PCA-reduced, L_2-normalized state embedding	Eq. (2)
$\rho_{i,t}$	per-transition local radius (density-adaptive)	Sec. 4.3
q, $\mathrm{dist}_{i,t}^{(q)}$	neighbor rank and corresponding embedding distance	Sec. 4.3
$\rho_{i,t;\,j,t'}$	symmetrized transition-pair radius	Sec. 4.3
ρ_a	global action-matching radius	Sec. 4.3
$\alpha_a \in (0,1)$	action-distance quantile level	Sec. 4.3, App. D.1
$m_{i,t;\,j,t'} \in (0,1]$	state-similarity weight	Eq. (3)
$\kappa_{i,t;\,j,t'} \in (0,1]$	action-similarity weight	Eq. (4)
$w_{i,t;\,j,t'}$	contrastive transition-pair weight	Eq. (8)
y_{ij}, $\Delta f_{i,t;\,j,t'}$	score-induced ordering, predicted reward difference	Sec. 4.4
$\mathcal{C}_{i,t}$, $\mathcal{P}_{\mathrm{raw}}$	eligible candidates, pairs before action weighting	Sec. 4.3
$\mathcal{P}$	state-matched transition-pair set, built once offline	Alg. 1
δ	Huber residual scale $\max(\mathrm{std}(\mathbf{z}), 10^{-2})$	Eq. (5)
Q	score scale $\max(\mathrm{IQR}(\mathbf{z}), 10^{-2})$	Sec. 4.3
λ_c, λ_s	contrastive, smoothness coefficients	Eq. (10)
η, λ_{wd}	AdamW learning rate and weight decay	App. D.1
k	maximum candidates retained per anchor	Sec. 4.3
$\mathcal{B}$, $\mathcal{B}_\tau$	transition-pair minibatch, trajectory minibatch	Alg. 1
B, B_τ, T_{train}	transition-pair and trajectory minibatch sizes, training steps	App. D.1
H_1	unit-threshold Huber loss	Eq. (5)
ς	logistic sigmoid	Sec. 4.4

Table 5: Notation used in the DCRL method and training procedure.

C Baseline Implementation Details

T-REX. T-REX receives the same 10 demonstrations and uses only the ranking induced by their scores. We follow its official implementation with the MuJoCo settings reported by Brown et al. (2019). Five reward models are trained independently, each on its own draw of 5,000 pairs of trajectory segments of up to 50 steps, and each model predicts a reward from the state alone. Every model has two hidden layers of 256 ReLU units and is trained for 10,000 updates with Adam at a learning rate of 10^{-4}, batch size 64, and an L_2 penalty of 0.01. We omit the hand-specified control-cost term used in the original MuJoCo experiments, because it supplies part of the environment reward that the other methods do not receive. The reward passed to SAC is the official one: each model's output is divided by its own running standard deviation, clipped to ± 10, and averaged over the five models. The network parameters remain fixed during SAC training, while reward-normalization statistics continue to update from collected transitions, following the official implementation. The original experiments train the policy with PPO, so our numbers come from training SAC with this reward, not from reproducing T-REX's published returns.

Frozen RRD. We apply RRD's reward-learning step to the same fixed data. The reward model predicts a reward from the state, the action, and the change in state, and has two hidden layers of 256 ReLU units. Following the default MuJoCo configuration of Ren et al. (2022), each update samples 64 timesteps from each of four trajectories and fits the mean of the sampled rewards to the demonstration's average per-step score, using Adam at a learning rate of 3×10^{-4}. RRD sets no number of reward updates of its own, since it trains the reward alongside the policy throughout; we train for 20,000 updates, having confirmed that the reward model converges well within that budget. The reward is then frozen and used by the same SAC code as every other method; trajectories collected during SAC never enter the data and never update it. The name *Frozen RRD* marks this difference from the original algorithm.

R4. R4 receives the same 10 demonstrations. Its supervision is an ordinal rating class per trajectory. We use its full-ranking setting, in which every trajectory forms its own class, so the 10 demonstrations take classes 0 to 9 in order of increasing score. We follow the official implementation. Each update predicts a reward $r(s, a)$ for every transition, sums these rewards into a predicted return per trajectory, and fits the ordering of the predicted returns to the ordering of the classes through a differentiable ranking operator, minimizing the squared difference between the two orderings. The reward model is the *large* network of their repository, which has two hidden layers of 100 ReLU units. It is trained for 1,000 updates with Adam at a learning rate of 3×10^{-4} and a ranking regularization of 1.0, matching the model and number of updates that Kharyal et al. (2026) list for their offline HalfCheetah experiment. Each update uses all 10 demonstrations. Trajectories of unequal length are padded, and padded steps are excluded when computing predicted returns. The reward is then frozen and passed to the same SAC code as every other method, without clipping or rescaling. The original R4 experiments use different demonstration data, so our numbers come from training SAC on our 10 demonstrations, not from reproducing R4's published returns.

D EXPERIMENTAL CONFIGURATION

This appendix lists the settings used in the experiments: DCRL reward learning (Appendix D.1), downstream SAC training shared by all methods (Appendix D.2), and the evaluation protocol (Appendix D.3). Baseline reward models use the settings described in Appendix C.

D.1 DCRL REWARD LEARNING

Table 6 lists the DCRL settings. The upper block gives the values set for the experiments; the lower block gives quantities computed from the fixed demonstrations by the stated formulas, so they are not tuned. The matching representation, including the rule for the PCA dimension d, is defined in Appendix A.

Parameter	Value	Role
Set for the experiments		
η_{PCA} (retained variance)	0.90	matching representation, Eq. (11)
$d_{\min}$ (minimum PCA dimension)	4	matching representation, Eq. (11)
ϵ (normalization constant)	10^{-8}	standardization and Eq. (2)
ϵ_ρ (state-radius floor)	10^{-3}	local state radius, Sec. 4.3
k (maximum candidates per anchor)	2	pair selection, Sec. 4.3
α_a (action-distance quantile)	0.20	action radius ρ_a, Sec. 4.3
λ_c, λ_s (contrastive, smoothness)	1.0	loss weights, Eq. (10)
L (hidden layers)	2	reward network
H (units per hidden layer)	128	reward network
Dropout p	0.1	reward network
η (learning rate)	2×10^{-4}	AdamW optimizer
λ_{wd} (weight decay)	10^{-4}	AdamW optimizer
T_{train} (reward training steps)	2,000	reward training
B (pair minibatch size)	512	reward training
B_τ (trajectories per step)	$\min\{8, N\}$	reward training
Computed from the demonstrations		
d (PCA output dimension)	$\min\{r, \max\{d_{\min}, d_{\eta_{\mathrm{PCA}}}\}\}$	Eq. (11)
q (neighbor rank)	$\min\{\lceil k/2 \rceil, n-1\}$	Sec. 4.3
$\rho_{i,t}$ (local state radius)	$\max(\mathrm{dist}_{i,t}^{(q)}, \epsilon_\rho)$	per transition, Sec. 4.3
$\rho_{i,t;\,j,t'}$ (pair radius)	$\sqrt{(\rho_{i,t}^2 + \rho_{j,t'}^2)/2}$	per transition pair, Eq. (3)
ρ_a (action-matching radius)	α_a-quantile, floor $0.1\sqrt{d_a}$	from retained pairs, Sec. 4.3
δ (Huber residual scale)	$\max(\mathrm{std}(\mathbf{z}), 10^{-2})$	Eq. (5)
Q (score-gap scale)	$\max(\mathrm{IQR}(\mathbf{z}), 10^{-2})$	Sec. 4.3

Table 6: DCRL reward-learning settings. Upper block: values set for the experiments. Lower block: quantities computed from the fixed demonstrations.

D.2 DOWNSTREAM SAC TRAINING

SAC uses the same configuration with every learned reward and with the true environment reward (Table 7). The actor and both critics are MLPs, and the entropy temperature is fixed without automatic tuning.

Setting	Value
Actor and critic networks	two hidden layers of 256 ReLU units
Optimizer and learning rate	Adam, 3×10^{-4}
Batch size	256
Discount factor γ	0.99
Entropy temperature	0.2 (fixed)
T_{warm} (uniform random-action steps)	5,000
T_{RL} (environment steps)	3,000,000

Table 7: Downstream SAC configuration, shared by all learned rewards and the true-reward reference.

D.3 EVALUATION PROTOCOL

Every 5,000 training steps, we evaluate the policy for one episode using deterministic actions and record its true return $R(\tau)$. For DCRL, we also compute the learned return $\hat{R}(\tau) = \sum_t f_\phi(s_t, a_t)$ over the same episode. Final returns average evaluations over the last 10,000 training steps within each seed; tables report the mean and standard deviation across five random seeds. Curves use exponential moving-average smoothing and show the five-seed mean and one standard deviation. We report descriptive comparisons without significance tests.