TOWARDS ADAPTIVE TEMPORAL CREDIT ASSIGNMENT VIA BIDIRECTIONAL FRACTIONAL OPERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

In continuous control and sequential decision-making tasks, agents may receive only an episodic return, with no intermediate rewards. Learning from such feedback requires connecting decisions to the terminal outcome. However, recurrent compression can weaken early contextual information, while fixed exponential weighting can attenuate distant learning signals. Moreover, history encoding and credit propagation may require different temporal scales. To address these challenges, we propose Fractional Credit Assignment (FCA), a framework based on two oppositely directed Caputo fractional operators. First, the left-sided operator introduces nonlocal history encoding through a power-law kernel. This allows early trajectory information to directly contribute to later representations. Second, the right-sided operator provides power-law temporal weights for backward credit assignment. Terminal-to-history attention combines content matching with these weights to redistribute the episodic return across steps. The fractional orders α_1 and α_2 are learned independently, allowing the two temporal scales to adapt to each task. We evaluate FCA under episodic-return-only feedback on seven MuJoCo tasks and four DeepMind Control Suite tasks. The latter use fixed 1,000-step episodes. FCA achieves the highest final average return on nine of the eleven tasks. On Swimmer, Ant, and Walker2d, it improves average returns over the strongest baseline in the same feedback setting by 90.7%, 53.3%, and 42.8%, respectively. It also outperforms SAC trained with dense rewards on five MuJoCo tasks. Ablation studies support the benefits of nonlocal history encoding and independently learned temporal scales for long-horizon credit assignment. The code is available at https://anonymous.4open.science/r/FCA-FBEC.

1 INTRODUCTION

In complex continuous control and sequential decision-making tasks, reinforcement learning agents often face the challenge of *episodic-return-only feedback*. Under this setting, the environment provides no intermediate rewards. At the end of each episode, it provides a single return that assesses the entire trajectory. However, policy optimization algorithms, such as policy gradient and actor–critic methods, rely on step-level learning signals to guide action updates. The core challenge is therefore *credit assignment*: how to decompose a single terminal return into meaningful step-level proxy rewards. This decomposition should preserve the total return while providing informative signals for policy optimization.

Accurately decomposing a terminal return is inherently ill-posed. Credit assignment involves two interdependent stages: *forward history representation* and *backward credit attribution*. On the one hand, understanding history is a prerequisite for reliable credit assignment. The terminal outcome reflects the cumulative effects of actions across the trajectory. Without step-level supervision, the model must retain information about delayed causal effects over long horizons to provide reliable historical context for attribution. On the other hand, credit assignment gives history representation a concrete learning objective. Even an accurate history representation cannot uniquely determine how a single return should be decomposed. The model therefore needs an explicit mechanism that uses the terminal representation as a reference. This mechanism should translate the content relevance and temporal distance of past actions into step-level credit. It can then reduce ambiguity in return decomposition and guide policy updates.

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.

These two stages raise three challenges for long-horizon credit assignment:

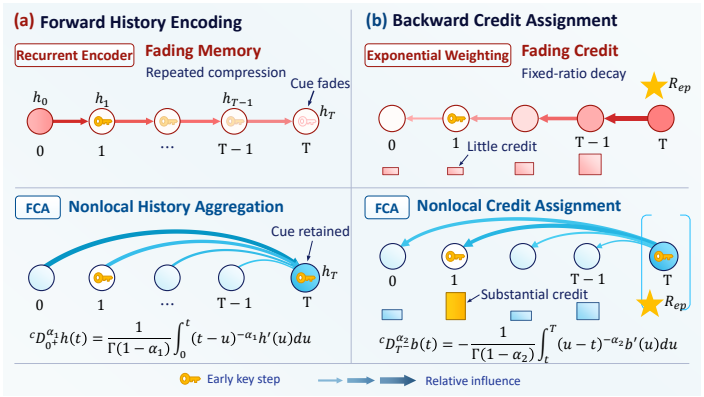


Figure 1: Comparison of forward history encoding (a) and backward credit assignment (b).

Markovian Trajectory Representation Bottleneck. Traditional recurrent encoders maintain history through local state updates, represented abstractly as $h_t = f(h_{t-1}, x_t)$ (Elman, 1990; Hochreiter & Schmidhuber, 1997). Such encoders are used for return prediction in RUDDER and sub-trajectory reward estimation in Diaster (Arjona-Medina et al., 2019a; Lin et al., 2024b). Information from early decisions must pass through successive updates before reaching the terminal representation h_T. As the trajectory length T grows, preserving relevant long-range dependencies through these recursions can become difficult (Pascanu et al., 2013). This creates a potential bottleneck for retaining the historical context needed for credit assignment. The upper panel of Figure 1(a) illustrates how early cues can fade through recurrent compression.

Fixed Exponential Weighting of Temporal Credit. Classical trace-based estimators use exponential weighting to aggregate learning signals over time. For example, generalized advantage estimation (GAE) weights future temporal-difference residuals by $(\gamma\lambda)^{\ell}$ (Schulman et al., 2016). For fixed $0 < \gamma\lambda < 1$, these weights decrease exponentially with temporal distance. Consequently, distant learning signals can be strongly attenuated before influencing earlier decisions. This motivates more flexible temporal weighting for long-horizon credit assignment. The upper panel of Figure 1(b) illustrates the resulting attenuation of credit to distant steps.

Limited Explicit Control of Two Temporal Scales. Forward memory and backward attribution describe two distinct processes. The former determines how far into the past information is retained. The latter determines how far credit extends backward from an outcome. RUDDER and Diaster derive redistributed rewards from differences in predicted returns or sub-trajectory rewards (Arjona-Medina et al., 2019a; Lin et al., 2024b). These designs do not explicitly parameterize memory retention and credit propagation through separate, learnable temporal kernels. This motivates separate controls for the two temporal scales.

To overcome these limitations, we propose the *Fractional Credit Assignment* (FCA) framework. Our design is motivated by the nonlocality of fractional-order derivatives. These operators characterize dynamics through responses accumulated across an entire temporal interval and weighted by a power-law kernel. This provides access to information beyond adjacent time steps. Based on this intuition, FCA introduces a pair of oppositely directed Caputo fractional operators for credit assignment. Its core components are as follows:

Nonlocal Forward History Encoding with a Left-Sided Caputo Operator. FCA uses the left-sided Caputo operator ${}^{\mathrm{C}}D^{\alpha_1}_{0+}h(t)$ to redefine the hidden-state dynamics. This allows semantic information from early trajectory segments to directly contribute to subsequent representations across intervening steps. The terminal representation h_T can thus retain information from distant history. This is illustrated by the retained early cue in the lower panel of Figure 1.

Terminal-Driven Nonlocal Credit Assignment with a Right-Sided Caputo Operator. FCA uses the right-sided Caputo operator ${}^{\mathrm{C}}D^{\alpha_2}_{T-}b(t)$ to trace credit backward from the terminal time T. It computes content matching scores using the terminal representation h_T as the query and the historical representations h_k as keys. These scores are combined with the power-law temporal kernel derived from the operator to obtain normalized step-level credit weights.

Decoupled Optimization of Two Temporal Scales. FCA introduces two independently learnable fractional orders, α_1 and α_2. The order α_1 controls the effective memory span of forward history

encoding. The order α_2 independently controls how far credit extends backward in time. Optimizing these orders allows FCA to adapt both temporal scales to the structure of each task.

We evaluate FCA under episodic-return-only feedback on seven MuJoCo tasks and four DeepMind Control Suite (DMC) tasks. FCA achieves the highest final average return on nine of the eleven tasks. Compared with the strongest baseline in the same feedback setting, FCA improves average returns by 90.7% on Swimmer, 53.3% on Ant, and 42.8% on Walker2d. It also outperforms SAC trained with the original dense rewards on five MuJoCo tasks. Across all four DMC tasks, each with a fixed horizon of 1,000 steps, FCA consistently achieves the best performance. It improves over the strongest competing baseline, RBT, by 2.5%–11.2%. Ablation studies show substantial performance drops when the fractional encoder is replaced with a GRU of comparable parameter count. Performance also drops substantially when the fractional orders are tied or fixed. These results support the benefits of nonlocal trajectory modeling and decoupled temporal scales.

2 Related Work

The studies most relevant to our work focus on return decomposition and credit assignment under episodic returns or other forms of coarse-grained feedback. RUDDER (Arjona-Medina et al., 2019b) uses an LSTM to predict future returns and redistributes delayed rewards according to changes between successive return predictions. RRD (Ren et al., 2022a) estimates step-level proxy rewards from randomly sampled subsets of trajectory steps. Diaster (Lin et al., 2024b) decomposes episodic returns through complementary sub-trajectories and derives step-level rewards from their return relations. RBT (Tang et al., 2025b) combines trajectory representations with bidirectional attention to redistribute coarse-grained rewards across time steps. GP-LRR (Xiao & Yu, 2026b) models correlations among state–action rewards with a Gaussian process and infers latent step-level rewards from trajectory-level returns. LaRe (Qu et al., 2025) uses prior knowledge from large language models to construct latent reward representations for fine-grained credit assignment.

Other studies investigate long-term credit assignment from complementary perspectives. QCA and HQCA (Mesnard et al., 2023) estimate decision contributions from quantiles of the return distribution. COCOA (Meulemans et al., 2023) uses counterfactual contribution analysis to measure how historical actions affect future outcomes. Preference Transformer (Kim et al., 2023) learns non-Markovian rewards from preference feedback, while Hindsight PRIOR (Verma & Metcalf, 2024) exploits state importance along a trajectory to assist reward learning. FracTrace (Mousavi et al., 2026) introduces fractional-order eligibility traces to extend credit propagation over long temporal horizons.

These methods improve credit estimation through return prediction, sub-trajectory relations, attention, reward correlations, semantic priors, and counterfactual contributions. However, in long-horizon trajectories, credit depends on both the temporal distance to the terminal outcome and the relevance between historical and terminal information. Our method therefore separately learns how to preserve historical information and how to allocate credit according to temporal distance. The final stepwise credit is determined by terminal–history content matching under a soft temporal weighting.

3 Problem formulation

We consider an episodic process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \rho_0, R)$. Here, $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, P is the transition distribution, and ρ_0 is the initial-state distribution. Starting from $s_0 \sim \rho_0$, actions $a_k \sim \pi(\cdot \mid s_k)$ and transitions $s_{k+1} \sim P(\cdot \mid s_k, a_k)$ produce a finite trajectory $\tau = (s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T)$. The agent receives only the episodic return $R(\tau)$ at its end, where $R : \mathcal{T} \to \mathbb{R}$ is defined on finite trajectories and $T \geq 1$ counts decision steps. To maximize expected episodic return, we seek stepwise credits that conserve this return:

$$\sum_{k=0}^{T-1} \widetilde{r}_k = R(\tau). \tag{1}$$

For $T > 1$, this constraint does not uniquely determine the allocation. FCA combines temporal weights and trajectory content to select an allocation, then distills the credits into a state–action reward for policy learning.

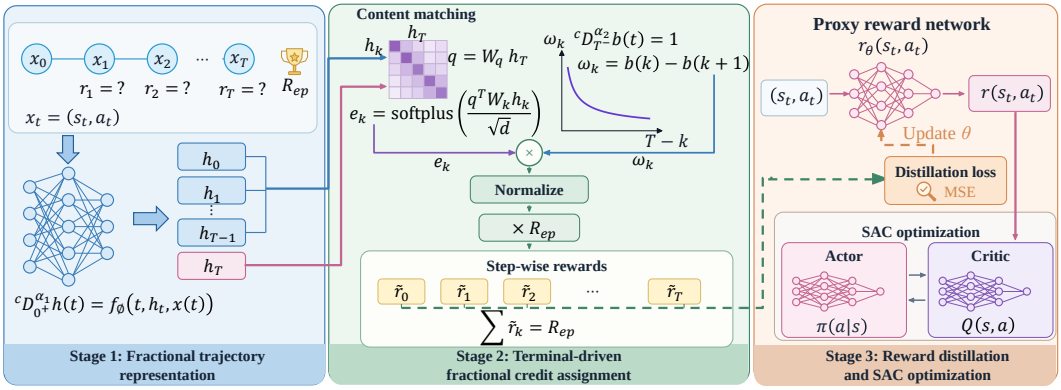


Figure 2: Overview of the FCA framework: fractional trajectory representation, terminal-driven fractional credit assignment, and reward distillation for SAC optimization.

4 Fractional Credit Assignment

As shown in Fig. 2, the FCA framework consists of three core stages.

Fractional trajectory representation. Given a state–action sequence, the left-sided Caputo fractional encoder aggregates historical interactions. It produces historical hidden states and the terminal representation h_T.

Terminal-driven fractional credit assignment. The terminal representation is matched with historical states to obtain content scores e_k. A right-sided Caputo equation defines the cumulative temporal mass $b(t)$. Its interval differences give the temporal weights ω_k. The products of e_k and ω_k are normalized and scaled by the return R_{ep}. The resulting step-wise rewards $\tilde{r}_k$ sum to R_{ep}.

Reward distillation and SAC optimization. The step-wise rewards $\tilde{r}_k$ serve as stop-gradient targets for training the proxy reward network $r_\theta(s, a)$. This network takes state–action pairs as input and predicts rewards. These predictions are used for value estimation and policy updates in SAC.

4.1 Fractional Trajectory Representation

Let $x_k = (s_k, a_k)$ denote the input at decision step k. We extend the sequence to continuous time by setting $x(t) = x_k$ on $[k, k+1)$. The hidden state $h : [0, T] \to \mathbb{R}^{d_h}$ evolves according to the Caputo fractional neural differential equation (Diethelm, 2010)

$$ {}^{C}D^{\alpha_1}_{0+} h(t) = f_\phi(x(t), h(t)), \qquad h(0) = h_0, \tag{2}$$

where ${}^{C}D^{\alpha_1}_{0+}$ is the left-sided Caputo derivative of order $\alpha_1 \in (0, 1)$. The neural network f_ϕ parameterizes the hidden-state dynamics. For an absolutely continuous solution with integrable dynamics, Equation (2) admits the equivalent Volterra integral form

$$ h(t) = h_0 + \frac{1}{\Gamma(\alpha_1)} \int_0^t (t-u)^{\alpha_1 - 1} f_\phi(x(u), h(u))\, du, \tag{3}$$

where Γ is the Gamma function. We discretize $[0, T]$ on the unit grid and denote the numerical approximation to $h(k)$ by h_k. The following lemma specifies the resulting update.

Lemma 4.1 (Trajectory encoder discretization) *Let $f_j = f_\phi(x_j, h_j)$ and approximate the dynamics by f_j on each interval $[j, j+1)$. Then the discrete approximation to Equation (3) satisfies*

$$ \begin{aligned} h_{k+1} &= h_0 + \frac{1}{\Gamma(1+\alpha_1)} \sum_{j=0}^{k} w_{k,j}(\alpha_1) f_j, \\ w_{k,j}(\alpha_1) &= (k+1-j)^{\alpha_1} - (k-j)^{\alpha_1}, \qquad 0 \le j \le k < T. \end{aligned} \tag{4}$$

Proof. Split the integral in Equation (3) at the grid points and substitute the piecewise constant dynamics. Each interval contributes $f_j w_{k,j}(\alpha_1)/\alpha_1$ before multiplication by $1/\Gamma(\alpha_1)$. The identity $\alpha_1\Gamma(\alpha_1) = \Gamma(1+\alpha_1)$ gives the stated update. □

The recurrence aggregates the dynamics responses $f_0, \ldots, f_k$ with positive power-law weights. We learn ϕ and α_1 jointly through this recurrence. The state h_{k+1} encodes the trajectory through decision k, and h_T is the complete-trajectory representation. Appendix A.1 gives the operator definition and the full derivation.

4.2 A TERMINAL-DRIVEN FRACTIONAL CREDIT OPERATOR

We model backward cumulative temporal mass $b(t)$ from the trajectory endpoint T using the right-sided Caputo terminal value problem (Diethelm, 2010)

$$ {}^{C}D^{\alpha_2}_{T-} b(t) = 1, \qquad b(T) = 0, \qquad \alpha_2 \in (0,1), \tag{5}$$

where α_2 is learned independently of the encoder order α_1. The boundary condition fixes zero remaining mass at T.

Lemma 4.2 (Caputo-induced temporal kernel) *Equation* (5) *has a unique absolutely continuous solution on* $[0,T]$*. Its solution and the induced temporal density are*

$$ b(t) = \frac{(T-t)^{\alpha_2}}{\Gamma(1+\alpha_2)}, \qquad \kappa_{\alpha_2}(T,t) := -b'(t) = \frac{(T-t)^{\alpha_2-1}}{\Gamma(\alpha_2)}, \quad 0 \le t < T. \tag{6}$$

Thus, $b(t) = \int_t^T \kappa_{\alpha_2}(T,u)\,du$ is the cumulative mass of the Caputo-induced temporal kernel. Following the kernel construction of Kang et al. (2025), we extend this kernel with a content-dependent attention factor. The terminal representation $h(T)$ queries historical representations through

$$ q = W_q h(T), \qquad e(t) = \mathrm{softplus}\left(\frac{q^\top W_{\mathrm{key}} h(t)}{\sqrt{d}}\right). \tag{7}$$

Here, W_q and W_{key} project into a shared d-dimensional space, and $\mathrm{softplus}(z) = \log(1+\exp z)$ ensures $e(t) > 0$. The temporal kernel is generalized to

$$ K_\tau(T,t) = \kappa_{\alpha_2}(T,t)e(t) = \frac{(T-t)^{\alpha_2-1}}{\Gamma(\alpha_2)} e(t). \tag{8}$$

This retains the fractional power-law factor and introduces dependence on both $h(T)$ and $h(t)$. Replacing κ_{α_2} by K_τ in the cumulative integral gives

$$ B_\tau(t) = \int_t^T K_\tau(T,u)\,du, \qquad B_\tau(T) = 0. \tag{9}$$

For $e(t) \equiv 1$, this reduces to the Caputo solution $B_\tau(t) = b(t)$. We normalize by the total mass $B_\tau(0)$. The credit for decision k is the normalized decrement of B_τ over $[k, k+1]$, scaled by the observed return:

$$ \begin{aligned} \widetilde{r}_k &= R(\tau)\frac{B_\tau(k) - B_\tau(k+1)}{B_\tau(0)} \\ &= R(\tau)\frac{\int_k^{k+1}(T-t)^{\alpha_2-1}e(t)\,dt}{\int_0^T (T-t)^{\alpha_2-1}e(t)\,dt}, \qquad k = 0, \ldots, T-1. \end{aligned} \tag{10}$$

The factor $1/\Gamma(\alpha_2)$ cancels under normalization.

Lemma 4.3 (Return conservation) *Let* $\alpha_2 \in (0,1)$ *and let* $e : [0,T] \to (0,\infty)$ *be measurable and bounded. Then* $0 < B_\tau(0) < \infty$*, and Equation* (10) *satisfies* $\sum_{k=0}^{T-1} \widetilde{r}_k = R(\tau)$ *for every finite* $R(\tau)$*.*

Appendix A.2 proves Lemmas 4.2 and 4.3 and gives the discrete interval weights.

4.3 Learning objectives for credit assignment

A return head g_ψ supervises the terminal representation:

$$\mathcal{L}_{\text{ret}} = \big(g_\psi(h_T) - R(\tau)\big)^2. \tag{11}$$

This trains the encoder and return head, but supplies no gradients to α_2, W_q, or W_{key}. To learn the allocation, an independent head u predicts prefix returns from the shared encoder. For $T > 1$, we choose a fixed midpoint k_c and use

$$\mathcal{L}_{\text{sub}} = \left(\sum_{k=0}^{k_c-1} \widetilde{r}_k - \text{sg}(u(h_{k_c}))\right)^2 + \big(u(h_T) - R(\tau)\big)^2. \tag{12}$$

Here, sg denotes stop-gradient. The first term trains the encoder and credit operator through the prefix credit sum, using a stopped-gradient soft target. The second anchors u to the observed return and updates both u and the encoder. Separate head parameters are intended to limit interference between prefix targets and representation learning. A single cut constrains the prefix total; the kernel and content scores determine its internal allocation. On complete-trajectory replay $\mathcal{D}_{\text{ep}}$, we jointly minimize

$$\mathcal{L}_{\text{FCA}} = \mathbb{E}_{\tau\sim\mathcal{D}_{\text{ep}}}[\mathcal{L}_{\text{ret}} + \lambda_{\text{sub}}\mathcal{L}_{\text{sub}}], \tag{13}$$

where $\lambda_{\text{sub}} \geq 0$ weights subtrajectory consistency. For $T = 1$, the sole decision receives the entire return and $\mathcal{L}_{\text{sub}}$ is omitted.

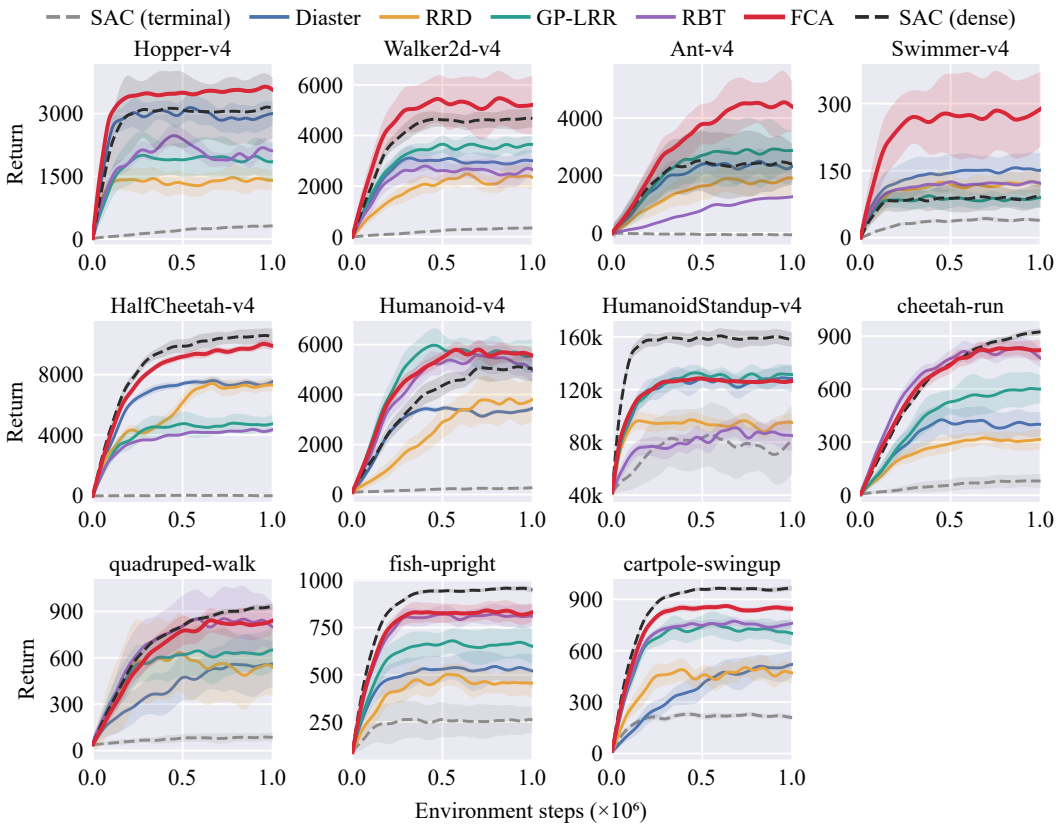


Figure 3: Return profiles on seven MuJoCo tasks.

4.4 Reward distillation and policy optimization

Credits depend on complete trajectories, so the same state–action pair can receive different labels across episodes. We distill them into a state–action reward network r_θ. Unpacking replay trajectories

Algorithm 1 Fractional Credit Assignment (FCA)

Require: Interaction budget N; warm-up steps N_0.
Ensure: Policy π_Ω.

```
Initialize Θ, θ, Ω; D_ep, D_tr ← ∅; n ← 0.
while n < N do
    (τ, R(τ)) ← Rollout(π_Ω; n, N_0); n ← n + |τ|.
    D_ep ← D_ep ∪ {(τ, R(τ))}; D_tr ← D_tr ∪ Transitions(τ).
    if n ≥ N_0 then
        for j = 1, ..., 8 do
            B ~ D_ep.
            Θ ← OptStep(Θ, ∇_Θ L_FCA(B)).
        end for
        for j = 1, ..., 8 do
            B ~ D_ep; y ← sg(FCA_Θ(B)).
            θ ← OptStep(θ, ∇_θ L_dis(B, y)).
        end for
        for each scheduled SAC update do
            B_tr ~ D_tr.
            Ω ← SACStep(Ω, B_tr; r_θ).
        end for
    end if
end while
```

gives the stepwise distribution $\mathcal{D}_{\text{step}}$ and loss

$$\mathcal{L}_{\text{dis}} = \mathbb{E}_{(s_k,a_k,\widetilde{r}_k)\sim\mathcal{D}_{\text{step}}}\left[\left(r_\theta(s_k,a_k) - \text{sg}(\widetilde{r}_k)\right)^2\right]. \tag{14}$$

Only θ is updated by this loss. Under the conditions in Appendix A.3, its population minimizer is the conditional mean credit. This does not guarantee return conservation along each individual trajectory after distillation. SAC (Haarnoja et al., 2018) then optimizes

$$J_{\text{proxy}}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=0}^{T-1}\gamma^k\left(r_\theta(s_k,a_k) + \beta\mathcal{H}(\pi(\cdot \mid s_k))\right)\right], \tag{15}$$

where γ is the discount, β is the entropy temperature, and $\mathcal{H}$ is policy entropy. Credit conservation alone does not imply policy invariance after distillation, discounting, and entropy regularization.

Algorithm 1 uses eight credit updates, then eight distillation updates with refreshed labels, followed by scheduled SAC updates averaging one per environment step. Θ contains the encoder, both orders, content projections, and both prediction heads; Ω contains the SAC parameters. Appendix B.1 defines the routine notation and training settings. Encoding costs $O(T^2)$ and credit allocation costs $O(T)$ at fixed network width. At execution, only the policy and current observation are needed.

5 Experiments

We evaluate return-based policy learning and the contributions of FCA's components. All methods use PyTorch and a single NVIDIA A800 GPU.

5.1 Experimental setup

We use seven Gymnasium v4 MuJoCo tasks: Hopper, Walker2d, Ant, Swimmer, HalfCheetah, Humanoid, and HumanoidStandup. A terminal-feedback wrapper sets intermediate rewards to zero and reveals their original sum as $R(\tau)$ at episode end. Original rewards are otherwise used only for independent evaluation, never as training inputs under terminal-only supervision. Observations are standard state vectors without reward components. We also test four fixed-horizon DMC tasks described below.

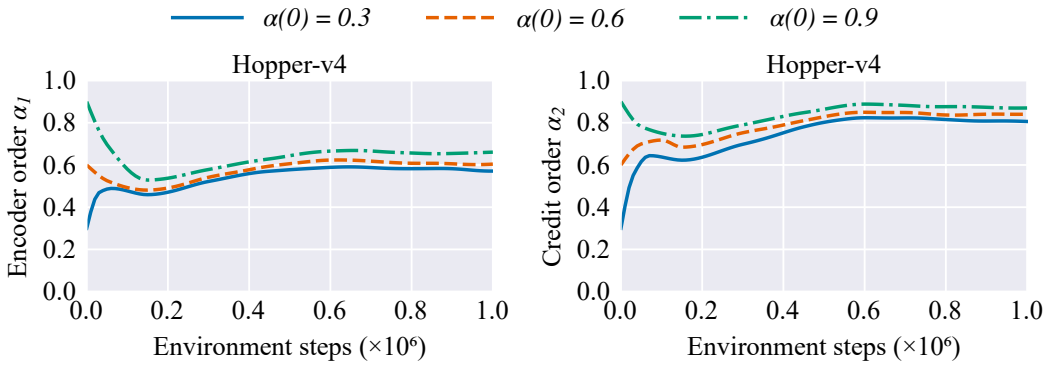


Figure 4: Encoder order α_1 and credit order α_2 for three initializations on Hopper.

SAC (terminal) (Haarnoja et al., 2018) learns directly from wrapped terminal rewards, while SAC (dense) provides a reference with original stepwise supervision. We compare four return decomposition methods under the same terminal feedback, interaction budget, and network scale. Diaster (Lin et al., 2024a) fits a subtrajectory reward model by matching the summed predictions of complementary segments to the episodic return. It trains a state–action reward model on differences between successive prefix predictions for policy optimization. RRD (Ren et al., 2022b) reconstructs episodic returns from scaled sums of predicted rewards at uniformly sampled time steps. Its surrogate objective reduces computation and adds a sampling-induced variance penalty to return regression. GP-LRR (Xiao & Yu, 2026a) models correlated step rewards with a Gaussian process over state–action pairs. It fits leave-one-out reward targets by likelihood maximization and uses the learned mean as the policy reward. RBT (Tang et al., 2025a) combines a causal Transformer with bidirectional attention to redistribute rewards within each bag. It learns from bag-return and next-state prediction losses, then relabels replay rewards for policy optimization.

We train each method for 1M interaction steps per task using five random seeds. Every 10k steps, we evaluate five episodes with deterministic actions. We report final episodic returns as the mean and sample standard deviation across seeds. Further evaluation and training settings are given in Appendix B.1.

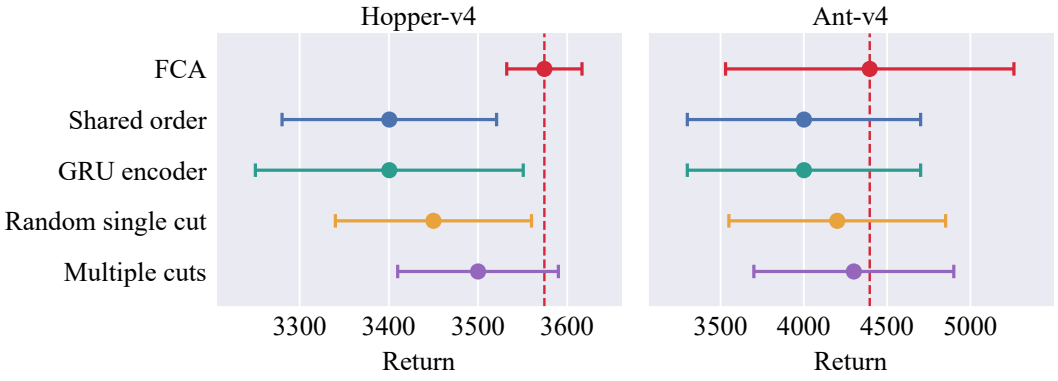


Figure 5: Component ablations on Hopper and Ant.

5.2 Main results on MuJoCo

FCA achieves the highest final mean return among terminal-feedback methods on five of seven MuJoCo tasks (Figure 3). The largest relative gains over the strongest competing terminal-feedback baseline occur on Swimmer, Ant, and Walker2d. Their final mean returns increase from 151.1 to 288.1, 2867.3 to 4395.7, and 3650.4 to 5211.3, respectively. FCA also exceeds SAC (terminal) on every task. There are two exceptions to its leading rank. GP-LRR obtains a slightly higher mean

on Humanoid (5580.5 versus 5565.9). On HumanoidStandup, both GP-LRR and Diaster exceed FCA. Compared with SAC (dense), FCA has a higher final mean on five tasks, while the dense-reward reference leads on HalfCheetah and HumanoidStandup. These results support the use of FCA for learning from episodic returns across several control tasks. They also show that its advantage depends on the task and does not extend to every baseline comparison.

Early failures in Hopper, Walker2d, Ant, and Humanoid produce short trajectories whose lengths change with the policy. We therefore test cheetah-run, quadruped-walk, fish-upright, and cartpole-swingup in DMC, using state observations and fixed 1000-step episodes without early termination. Feedback, baselines, budgets, and evaluation follow Section 5.1. FCA has the highest final mean return among terminal-feedback methods on all four DMC tasks (Figure 3). RBT is the strongest competing terminal-feedback method on each task. FCA's relative gains over RBT range from 2.5% on fish-upright to 11.2% on cartpole-swingup. The improvements extend the MuJoCo findings to fixed 1000-step episodes. The mean gaps to RBT are also modest on fish-upright and quadruped-walk relative to the reported variation across seeds.

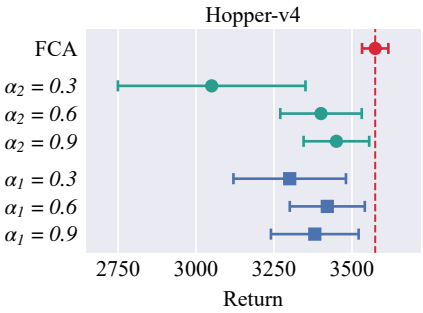


Figure 6: Final returns in the order ablation on Hopper.

5.3 Ablation studies

We vary one component at a time on Hopper and Ant, keeping the five seeds and all other settings unchanged. On Hopper, we fix one order at 0.3, 0.6, or 0.9 while learning the other. Content matching, within-trajectory normalization, training objectives, and distillation remain unchanged. Figure 6 compares final returns. Learning both orders gives the highest mean return on Hopper (Figure 6). Fixing the credit order at 0.3 reduces the return from 3574.4 to 3050.1, the largest drop among the tested fixed-order variants. Figure 4 shows the encoder and credit orders for three initializations.

Figure 5 tests a shared order, a parameter-matched GRU encoder, and alternative prefix cut points. The default fixed midpoint is compared with one random cut and multiple cuts. Sharing the orders or using a parameter-matched GRU lowers the mean return on both Hopper and Ant. The fixed midpoint also gives a higher mean than either alternative cut-point strategy on both tasks. Multiple cut points improve over a single random cut point, but remain below the default. These comparisons favor separate learnable orders and the default midpoint within the configurations tested here.

6 Conclusion

In this paper, we propose Fractional Credit Assignment (FCA) for long-horizon reinforcement learning under episodic-return-only feedback. FCA introduces two oppositely directed Caputo fractional operators. The left-sided operator enables nonlocal history encoding, while the right-sided operator provides power-law temporal weights for backward credit assignment. Terminal-to-history attention combines these weights with content relevance to redistribute the episodic return. The two fractional orders, α_1 and α_2, are learned independently. FCA thus separately learns how to remember history and how to assign credit according to temporal distance. We systematically evaluate FCA on seven MuJoCo tasks and four DeepMind Control Suite tasks. FCA achieves the highest final average return on nine of the eleven tasks. It also outperforms SAC trained with the original dense rewards on five MuJoCo tasks. Its leading performance across all four 1,000-step DMC tasks further supports its effectiveness over long horizons. Ablation studies show substantial performance drops when the fractional encoder is replaced with a GRU of comparable parameter count. Tying or fixing the fractional orders also degrades performance. These findings support the benefits of nonlocal history encoding and independently learned temporal scales. Together, they demonstrate the value of separately adapting memory retention and credit propagation when learning from episodic returns.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language polishing, translation, and presentation, including improving the grammar, clarity, and fluency of the manuscript. Generative AI tools were not used to execute experiments, generate reported experimental results, or make scientific conclusions. All methodological choices, implementations, experimental analyses, and scientific claims were determined and verified by the authors. We reviewed all AI-assisted content and take full responsibility for the final manuscript, including its text, results, claims, and artifacts.

ETHICS STATEMENT

All experiments are conducted in controlled benchmark environments and do not involve human subjects, private user data, or personally identifiable information. The proposed method is evaluated on standard reinforcement learning benchmarks, including continuous-control environments. We do not identify additional ethical risks beyond those commonly associated with reinforcement learning research.

REPRODUCIBILITY STATEMENT

The paper specifies the episodic-return-only setting, the trajectory representation procedure, the left- and right-sided fractional operators, the terminal-driven credit assignment mechanism, the proxy reward learning procedure, and the subsequent policy optimization. We also report the benchmark environments, environment versions, model configurations, hyperparameters, training budgets, random seeds, and evaluation protocols used in the experiments. The implementation includes the fractional trajectory encoder, temporal credit weighting, terminal-to-history matching, reward distillation, and SAC optimization. Code, configurations, and implementation details are available at `https://anonymous.4open.science/r/FCA-FBEC/`.

REFERENCES

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

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, pp. 13566–13577, 2019b.

Kai Diethelm. *The Analysis of Fractional Differential Equations: An Application-Oriented Exposition Using Differential Operators of Caputo Type*, volume 2004 of *Lecture Notes in Mathematics*. Springer Berlin, Heidelberg, 2010. doi: 10.1007/978-3-642-14574-2. URL `https://link.springer.com/book/10.1007/978-3-642-14574-2`.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

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 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL `https://proceedings.mlr.press/v80/haarnoja18b.html`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Qiyu Kang, Wenjun Cui, Xuhao Li, Yuxin Ma, Xueyang Fu, Wee Peng Tay, Yidong Li, and Zheng-Jun Zha. Neural fractional attention differential equations. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1125. URL `https://papers.neurips.cc/paper_files/paper/2025/hash/3040bf93152e90d5719379818a8664b4-Abstract-Conference.html`.

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.

Haoxin Lin, Hongqiu Wu, Jiaji Zhang, Yihao Sun, Junyin Ye, and Yang Yu. Episodic return decomposition by difference of implicitly assigned sub-trajectory reward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(12):13808–13816, 2024a. doi: 10.1609/aaai.v38i12.29287. URL https://ojs.aaai.org/index.php/AAAI/article/view/29287.

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

Thomas Mesnard, Wenqi Chen, Alaa Saade, Yunhao Tang, Mark Rowland, Theophane Weber, Clare Lyle, Audrunas Gruslys, Michal Valko, Will Dabney, Georg Ostrovski, Eric Moulines, and Remi Munos. Quantile credit assignment. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24517–24531, 2023.

Alexander Meulemans, Simon Schug, Seijin Kobayashi, Nathaniel D. Daw, and Gregory Wayne. Would i have gotten that reward? long-term credit assignment by counterfactual contribution analysis. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yashar Mousavi, Rashin Mousavi, Arash Mousavi, Mahsa Tavasoli, Ibrahim Beklan Kucukdemiral, Afef Fekih, and Umit Cali. FracTrace: Long-horizon credit assignment in deep reinforcement learning via fractional eligibility traces. *Neurocomputing*, 706:134924, 2026. doi: 10.1016/j.neucom.2026.134924.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318. PMLR, 2013.

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

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

Zhizhou Ren, Ruihan Guo, Yuan Zhou, and Jian Peng. Learning long-term reward redistribution via randomized return decomposition. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=lpkGn3k2YdD.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

Yuting Tang, Xin-Qiang Cai, Yao-Xiang Ding, Qiyu Wu, Guoqing Liu, and Masashi Sugiyama. Reinforcement learning from bagged reward. *Transactions on Machine Learning Research*, 2025a. URL https://openreview.net/forum?id=bXUipBbZDA.

Yuting Tang, Xin-Qiang Cai, Yao-Xiang Ding, Qiyu Wu, Guoqing Liu, and Masashi Sugiyama. Reinforcement learning from bagged reward. *Transactions on Machine Learning Research*, 2025b.

Mudit Verma and Katherine Metcalf. Hindsight PRIORs for reward learning from human preferences. In *International Conference on Learning Representations*, 2024.

Minheng Xiao and Xian Yu. Reward redistribution via Gaussian process likelihood estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(32):27073–27080, 2026a. doi: 10.1609/aaai.v40i32.39921. URL https://ojs.aaai.org/index.php/AAAI/article/view/39921.

Minheng Xiao and Xian Yu. Reward redistribution via gaussian process likelihood estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27073–27080, 2026b.

A Operator definitions, solutions, and discretization

A.1 Left-sided Caputo operator and trajectory encoder discretization

The left-sided Caputo operator in Equation (2) is defined as (Diethelm, 2010)

$$ {}^{C}D_{0+}^{\alpha_1} h(t) = \frac{1}{\Gamma(1-\alpha_1)} \int_0^t (t-u)^{-\alpha_1} h'(u)\, du, \qquad t \in (0, T], $$

where h is absolutely continuous on $[0, T]$.

Derivation of Equation (3)**.** Apply the left-sided Riemann–Liouville integral of order α_1 to both sides of Equation (2). Its inverse relation with the Caputo derivative, together with $h(0) = h_0$, yields the Volterra integral equation in Equation (3). For $\alpha_1 \in (0, 1)$, past dynamics responses receive weights $(t-u)^{\alpha_1 - 1}$ that decay as a power law in the time lag. At the boundary $\alpha_1 = 1$, this equation reduces to the integral form of an ordinary differential equation.

Proof of Lemma 4.1. We use a piecewise constant approximation on the unit grid $\{0, 1, \ldots, T\}$. Let $h_j \approx h(j)$, and set $f_j = f_\phi(x(j), h_j)$ on $[j, j+1)$. Splitting the integral in Equation (3) across decision intervals gives

$$ h_{k+1} = h_0 + \frac{1}{\Gamma(\alpha_1)} \sum_{j=0}^{k} f_j \int_j^{j+1} (k+1-u)^{\alpha_1 - 1}\, du. $$

Each interval integral evaluates to

$$ \int_j^{j+1} (k+1-u)^{\alpha_1 - 1}\, du = \frac{(k+1-j)^{\alpha_1} - (k-j)^{\alpha_1}}{\alpha_1} = \frac{w_{k,j}(\alpha_1)}{\alpha_1}. $$

Substituting this expression and using $\Gamma(1+\alpha_1) = \alpha_1 \Gamma(\alpha_1)$ gives Equation (4). □

For each k, Equation (4) sums all preceding dynamics responses. Encoding a trajectory of length T therefore costs $O(T^2)$. The terminal representation is h_T.

A.2 Fractional attention kernel, return conservation, and interval weights

Proof of Lemma 4.2. Let $\alpha = \alpha_2 \in (0, 1)$. By definition, the right-sided Caputo derivative is the right-sided Riemann–Liouville integral applied to $-b'$:

$$ {}^{C}D_{T-}^{\alpha} b(t) = I_{T-}^{1-\alpha}(-b')(t) = -\frac{1}{\Gamma(1-\alpha)} \int_t^T (s-t)^{-\alpha} b'(s)\, ds. $$

Equation (5) is therefore equivalent to $I_{T-}^{1-\alpha}(-b') = 1$. Apply I_{T-}^{α} to both sides and use the semi-group property $I_{T-}^{\alpha} I_{T-}^{1-\alpha} = I_{T-}^{1}$. This gives

$$ I_{T-}^{1}(-b')(t) = I_{T-}^{\alpha} 1(t). $$

Since $b(T) = 0$, the left-hand side becomes

$$ \int_t^T (-b'(s))\, ds = b(t) - b(T) = b(t). $$

Direct integration of the right-hand side gives

$$ I_{T-}^{\alpha} 1(t) = \frac{1}{\Gamma(\alpha)} \int_t^T (s-t)^{\alpha - 1}\, ds = \frac{(T-t)^{\alpha}}{\alpha \Gamma(\alpha)} = \frac{(T-t)^{\alpha}}{\Gamma(1+\alpha)}. $$

The last equality uses $\Gamma(1+\alpha) = \alpha \Gamma(\alpha)$. Hence,

$$ b(t) = \frac{(T-t)^{\alpha_2}}{\Gamma(1+\alpha_2)}. $$

Differentiating with respect to t yields

$$b'(t) = -\frac{\alpha_2(T-t)^{\alpha_2-1}}{\Gamma(1+\alpha_2)} = -\frac{(T-t)^{\alpha_2-1}}{\Gamma(\alpha_2)}.$$

Thus, $-b'(t) = (T-t)^{\alpha_2-1}/\Gamma(\alpha_2)$. Since $\alpha_2 \in (0,1)$, this density is strictly positive on $[0,T)$ and strictly decreases as a function of the lag $T-t$. Integrating over $[k,k+1)$, or directly taking the difference, gives

$$b(k) - b(k+1) = \frac{(T-k)^{\alpha_2} - (T-k-1)^{\alpha_2}}{\Gamma(1+\alpha_2)}.$$

For uniqueness, suppose $\widetilde{b}$ is another solution. Then $\delta = b - \widetilde{b}$ satisfies ${}^{C}D^{\alpha}_{T-}\delta = 0$ and $\delta(T) = 0$. The null space of the right-sided Caputo operator consists of constant functions. The terminal condition forces this constant to be zero, so the solution is unique. □

From the Caputo solution to the attention integral. Since $b(T) = 0$ and $-b'(t) = \kappa_{\alpha_2}(T,t)$, the baseline solution satisfies

$$b(t) = \int_t^T \kappa_{\alpha_2}(T,u)\,du.$$

Replacing this cumulative integral kernel by $K_\tau(T,u) = \kappa_{\alpha_2}(T,u)e(u)$ defines B_τ in Equation (9). For measurable, positive, bounded e, B_τ is absolutely continuous and

$$-B'_\tau(t) = K_\tau(T,t) \quad \text{almost everywhere,} \qquad B_\tau(k) - B_\tau(k+1) = \int_k^{k+1} K_\tau(T,t)\,dt.$$

The attention extension acts on the cumulative integral kernel induced by the Caputo solution. For $e \equiv 1$, it recovers $B_\tau = b$; for general e, it defines a content-dependent extension rather than a solution of the unchanged unit-forcing equation.

Proof of Lemma 4.3. Define the normalization constant and normalized density by

$$Z_\tau = B_\tau(0) = \int_0^T K_\tau(T,u)\,du, \qquad p_\tau(t) = \frac{K_\tau(T,t)}{Z_\tau}. \tag{16}$$

For $\alpha_2 \in (0,1)$, the temporal kernel $\kappa_{\alpha_2}(T,t)$ is positive on $[0,T)$ and has an integrable singularity at $t = T$. Its integral is $T^{\alpha_2}/\Gamma(1+\alpha_2)$. If $0 < e(t) \le M$ on $[0,T]$ and e is measurable, then

$$0 < Z_\tau = \int_0^T \kappa_{\alpha_2}(T,t)e(t)\,dt \le \frac{MT^{\alpha_2}}{\Gamma(1+\alpha_2)} < \infty.$$

Thus, p_τ in Equation (16) is a well-defined density with unit integral. The decision intervals partition $[0,T]$ up to their endpoints, so

$$\sum_{k=0}^{T-1} \widetilde{r}_k = R(\tau) \sum_{k=0}^{T-1} \int_k^{k+1} p_\tau(t)\,dt = R(\tau) \int_0^T p_\tau(t)\,dt = R(\tau).$$

The identity holds for positive, zero, and negative returns. □

Interval weights. Under a piecewise constant approximation $e(t) = e_k$ on $[k,k+1)$, define

$$\omega_k = \int_k^{k+1} \kappa_{\alpha_2}(T,t)\,dt = \frac{(T-k)^{\alpha_2} - (T-k-1)^{\alpha_2}}{\Gamma(1+\alpha_2)}.$$

Equation (10) then reduces to

$$\widetilde{r}_k = R(\tau)\frac{\omega_k e_k}{\sum_{j=0}^{T-1} \omega_j e_j}.$$

The product weights are normalized once across all decision intervals. Computing all interval weights and their normalization costs $O(T)$.

When the content matching score is constant, the relative credits depend only on $b(k) - b(k+1)$. As $\alpha_2 \to 0^+$, the increments for all earlier intervals approach zero, and the allocation concentrates on the final step. As $\alpha_2 \to 1^-$, the interval increments become equal, and the allocation approaches a uniform distribution. A single continuous parameter thus connects uniform allocation and concentration on the final step.

A.3 THE CONDITIONAL MEAN LEARNED BY REWARD DISTILLATION

Consider Equation (14) as a squared-loss risk under a fixed distribution $\mathcal{D}_{\text{step}}$, conditional on (s, a). Let $\widetilde{r}$ denote the credit label. Assume $\mathbb{E}[\widetilde{r}^2] < \infty$ and that the reward network can represent the map $(s, a) \mapsto \mathbb{E}[\widetilde{r} \mid s, a]$. Then

$$\mathbb{E}\left[(r_\theta(s,a) - \widetilde{r})^2\right] = \mathbb{E}\left[\left(r_\theta(s,a) - \mathbb{E}[\widetilde{r} \mid s,a]\right)^2\right] + \mathbb{E}\left[\left(\mathbb{E}[\widetilde{r} \mid s,a] - \widetilde{r}\right)^2\right].$$

The second term does not depend on θ. The population-optimal prediction is therefore $r_{\theta^*}(s, a) = \mathbb{E}[\widetilde{r} \mid s, a]$, the conditional mean credit given the state and action. This derivation holds only for a fixed stepwise sample distribution. It does not imply that $\sum_k r_\theta(s_k, a_k) = R(\tau)$ along every individual trajectory. □

B EXPERIMENTAL DETAILS

B.1 TRAINING CONFIGURATION

All SAC-based methods share the tasks, seeds, interaction budgets, and evaluation protocol in Section 5.1. The policy, value, and dynamics networks each have two 256-unit layers; the encoder hidden state has dimension 128. The proxy reward network has three 256-unit layers and takes only the current state and action as input.

Trajectory and transition replay hold 256 episodes and 100000 transitions, with batch sizes 8 and 256, respectively. We use 1000 random-exploration steps and begin updates at step 1000. The learning rate is 3×10^{-4}, the discount is 0.99, and the target-network update coefficient is 0.005. Return scaling is 1.0; SAC tunes the entropy temperature automatically.

The encoder order and credit order are independent parameters, both initialized to 0.6. The subtrajectory loss weight is 0.1. The prefix cut point is the fixed midpoint, and $\mathcal{L}_{\text{sub}}$ is omitted for one-step trajectories. After each episode, the credit module and proxy reward network each receive eight updates. SAC receives one update per environment step on average. The default freezing point in the supplementary controls is 300k environment steps. Return decomposition baselines use official implementations or checked reimplementations, with the same feedback, interaction budget, and network scale.

All methods use the same environment versions, termination rules, and evaluation seeds. Evaluation steps are excluded from the training budget. At each checkpoint, we first average true episodic returns over the five evaluation episodes within each training seed. Final returns are taken from the last checkpoint, and their mean and sample standard deviation are computed across the five training seeds.

Algorithm notation. In Algorithm 1, $\mathcal{D}_{\text{tr}}$ is transition replay and $|\tau|$ counts decision steps. $\text{FCA}_\Theta(\mathcal{B})$ encodes the trajectories in $\mathcal{B}$ and returns their stepwise credits. The losses are minibatch estimates; $\mathcal{L}_{\text{dis}}(\mathcal{B}, \mathbf{y})$ pairs the state–action samples with labels $\mathbf{y}$. OptStep performs one optimizer step to minimize the corresponding loss. Rollout uses random actions during the first $N_0 = 1000$ environment steps and the current policy thereafter. SACStep updates only Ω, including critics, actor, entropy temperature, and target networks, with r_θ fixed.

B.2 EXTENDED CONTROLS AND SENSITIVITY

We examine sensitivity to the main hyperparameters of FCA on Hopper using five seeds. We change one factor at a time and keep all other settings at their default values. The factors include order initialization, fixed credit orders, the subtrajectory loss weight, trajectory batch size, learning rates, and updates per episode. We also vary exploration duration, return scaling, the freezing point, network width and architecture, and prefix cut-point strategy. Extended controls examine terminal-return noise and trajectory length. The noise standard deviation is 10% of the return. The length control doubles the episode length while retaining the task generation rules.

B.3 Diagnosing Bias from Reward Distillation and the Optimization Objective

Reward distillation compresses credits that depend on complete trajectories into a proxy reward that depends only on the current state and action. As discussed in Section 4.4, distillation, discounting, and entropy regularization may change the optimization objective. We quantify this bias through comparisons on Hopper and Ant. The conditions include full FCA and direct policy training with trajectory-level credit labels, without distillation. We also consider distillation after freezing the credit model, policy optimization after freezing the proxy reward, and removing stop-gradient from distillation labels. A simple control uniformly distributes the episodic return before distillation. Further controls use $\gamma = 1$ for a finite-horizon objective, with bootstrapping stopped at episode boundaries, and different entropy regularization settings. These last two controls are also applied to SAC (terminal).

On complete trajectories excluded from training, we record the distillation mean squared error and the discrepancy between summed proxy rewards and episodic returns. We also measure ranking agreement for a fixed set of policies under three criteria. These are the true undiscounted return, the undiscounted proxy return, and the discounted proxy objective. Ranking agreement describes the empirical alignment between the proxy and original objectives. It does not prove that optimal policies remain unchanged.

B.4 Metric Definitions

Final return uses the last evaluation checkpoint; peak return is the maximum over the evaluation grid. Degradation is peak return minus final return. Each seed is a statistical unit; across-seed dispersion uses the sample standard deviation. Method differences are preferentially paired by seed. Reward drift is the mean absolute prediction change between consecutive checkpoints on a fixed diagnostic set. Return discrepancy is the sum of proxy rewards along a trajectory minus its observed return. We report its absolute value and its value normalized by a fixed scale.