Reward Inflation: A Healthy Stimulus for Reinforcement Learning

Ganghun Lee[1] **Minji Kim**[2] **Minsu Lee**[3]* **Byoung-Tak Zhang**[1,2]*
[1]AIIS, Seoul National University
[2]Department of Computer Science, Seoul National University
[3]School of AI Convergence, Sungshin Women's University
{zxc8594, cyqkcy01}@snu.ac.kr, mslee@sungshin.ac.kr, btzhang@snu.ac.kr

Abstract

Reward serves as the primary learning signal in reinforcement learning (RL). However, while reward magnitudes are typically held fixed throughout training, their temporal modulation remains underexplored. In this paper, we propose *reward inflation*, a gradual scaling of rewards over the course of training, and show that it can act as a healthy stimulus for RL. Theoretically, reward inflation induces an implicit recency weighting that upweights recent transitions during policy updates, enabling faster adaptation. We further show that reward inflation suppresses the emergence of dormant neurons by sustaining gradient signals as the policy saturates, suggesting a possible contribution to plasticity preservation. Empirical results on ALE games and MuJoCo tasks corroborate these findings, showing that an appropriate level of reward inflation benefits a broad range of tasks. Finally, we introduce *Fed*, an adaptive variant that adjusts the inflation level on the fly, and find that it often improves upon fixed inflation. Source code is available at https://github.com/northadventure/RewardInflation.

1 Introduction

Reinforcement Learning (RL) has shown promising results in diverse domains, such as robotics [1], games [2], art [3], autonomous driving [4], recommendation systems [5], finance [6], and gaining traction to finetune and improve large language models (LLMs) [7]. Since RL learns from rewards, optimizing the reward function is central to RL research. Although reward design is largely task-specific, some previous works have explored various techniques to enhance or augment fully or partially task-agnostic reward design [8], such as reward normalization [9], potential-based reward shaping [10], adaptive reward shaping [11], intrinsic motivation [12], curriculum rewards [13], and, more recently, human- or LLM-augmented rewards [14]. However, the temporal dynamics of the reward scale throughout training remain underexplored.

In this paper, we introduce *reward inflation*, a novel approach that scales rewards slowly yet exponentially over the course of RL. The idea is simple and widely applicable to implement: maintaining a positive variable called the inflation level, which is periodically updated by an inflation rate and multiplied with the reward. The agent then learns from the inflated rewards during training, while previously stored rewards in the replay buffer remain unchanged. Additionally, we introduce an optional component, *Fed*, which adaptively adjusts the inflation rate during training by monitoring whether the moving average of the policy gradient norm becomes excessive or insufficient.

Despite its simplicity, reward inflation has rich theoretical implications. Based on our theoretical analysis under a fixed inflation rate, reward inflation induces an implicit recency weighting in the TD

*Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

update—a non-uniform weighting across the replay buffer that assigns greater effective weight to recent transitions. In addition, reward inflation suppresses the emergence of dormant neurons: by gradually increasing the overall gradient norm, it may help preserve plasticity, potentially contributing to improved learning.

Our experimental results on ALE [15] games and MuJoCo [16] tasks show that, under a moderate inflation rate, reward inflation benefits a broad range of tasks, supporting the claims from our theoretical analysis. Specifically, on ALE, reward inflation achieved an 11.53% improvement over the baseline, with Fed further increasing the gain to 12.59%. On MuJoCo, based on the normalized score, reward inflation achieved an 18.51% improvement over the non-inflation baseline, while the Fed variant further increased the improvement to 35.21%. These results indicate more consistent improvements than related approaches, including reward normalization, prioritized experience replay, learning rate inflation, and recency-weighted sampling. Additional experiments with TD3 and tabular Q-learning further examine the scope and recency-weighting effect of reward inflation, including its potential costs under deceptive rewards. Comparisons with plasticity methods further examine the relationship between dormant-neuron suppression and learning performance.

Our contributions are summarized as follows:

- We propose reward inflation, a simple and nearly cost-free modification to RL training. We further introduce Fed, an adaptive variant that adjusts the inflation rate on the fly.
- We analyze how reward inflation induces recency weighting and sustains gradient signals, and investigate dormant-neuron suppression as a possible contributing mechanism for improved learning.
- We present empirical results demonstrating that reward inflation and Fed benefit a broad range of tasks across ALE and MuJoCo.

2 Related works

Reward shaping and normalization. These techniques have long been recognized as important factors in stabilizing and accelerating RL training. A first line of work modifies the reward signal to ease optimization, including potential-based shaping [10], intrinsic reward shaping [17, 18], meta-learned shaping [19, 20], and curriculum-based reward design [21, 22, 23, 24]. A closely related family normalizes reward or value magnitudes for stable training, including reward normalization [25, 26], return-based normalization [9], and value rescaling [27, 28], with extensions to temporally heterogeneous rewards [29]. Reward inflation also modifies the reward signal, but differs in two respects: it preserves reward semantics and operates along a temporal axis, gradually scaling reward magnitudes over training rather than reshaping them per-state.

Implicit weighting of replayed experience. Such weighting schemes have also been explored to improve sample efficiency and learning stability in off-policy RL. Prioritized experience replay (PER) weights transitions by TD error [30, 31, 32], with extensions based on state distribution [33], freshness [34], and learnability [35]. Other approaches emphasize recent experience (ERE) [36], learn sampling policies [31], decouple actor-critic sampling [37], or prioritize via reward prediction errors [38]. Reward inflation similarly increases the influence of recent transitions on TD updates, but does so by progressively amplifying reward magnitudes rather than altering replay or sampling mechanisms. Daley et al. [39] analyze the benefits of recency-based temporal credit assignment in TD return estimators, providing a conceptual connection to our approach, which instead weights replayed transitions according to their collection time.

Plasticity in deep RL. Recent work has identified a range of pathologies in deep RL, including dormant neurons [40], capacity loss [41], and primacy bias [42]. These collectively reflect a loss of plasticity and pose a growing obstacle to sustained learning. Proposed remedies include periodic resets of network parameters [42, 43], regularization toward initialization [44], and architectural interventions. Reward inflation offers an indirect alternative: by gradually amplifying gradient magnitudes, it sustains the effective update signal, which can equivalently be interpreted as maintaining the effective learning rate [45], thereby mitigating the emergence of dormant neurons without explicit resetting or architectural change.

3 Method

In this section, we present the notation and formulation of reward inflation, followed by the introduction of Fed, an adaptive inflation control mechanism.

3.1 Preliminaries

We formulate the RL problem as a Markov decision process (MDP), defined by $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{P}$ the transition function, $\mathcal{R}$ the reward function, and γ the discount factor. A policy $\pi \in \Pi$ maps states to action distributions. At each timestep t, the agent samples an action $a_t \sim \pi(\cdot \mid s_t)$, transitions to $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$, and receives reward r_t. The return is defined as $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$, where T is the episode horizon. The goal of RL is to find the optimal policy $\pi^* = \arg\max_\pi \mathbb{E}_\pi[G_0]$.

3.2 Reward Inflation

We consider off-policy RL with experience replay. Reward inflation is applied before rewards are inserted into the replay buffer, and stored rewards remain unchanged thereafter. Under reward inflation, the agent learns from the nominal reward r'_t rather than the real reward r_t.

We denote *global step* $g \in \mathbb{N}_0$ as a unique index that counts the total number of environment interactions throughout training. The inflation is represented as *inflation rate* $\rho_Y(g\,;\,Y)$, where Y denotes the number of global steps to incrementally apply the inflation rate. From the definition, *per-step inflation rate* $\tilde{\rho}(g)$ can be derived as

$$\tilde{\rho}(g) \coloneqq (1 + \rho_Y(g\,;\,Y))^{\frac{1}{Y}} - 1. \tag{1}$$

For example, a constant inflation rate of 2% over $Y = 10^5$ steps corresponds to a per-step rate of $\tilde{\rho} = 1.02^{1/10^5} - 1 \approx 1.98 \times 10^{-7}$. Compounding this per-step rate over 10^5 steps increases the reward scale by 2%. This implies that, if the inflation rate remains constant, the reward scale will increase by 2% after 10^5 steps. Then we define *inflation level* $I(g) > 0$ at global step g as

$$I(g) \coloneqq \textstyle\prod_{k=1}^{g} (1 + \tilde{\rho}(k))\,, \qquad r'_t(g) \coloneqq r_t \cdot I(g), \tag{2}$$

where $I(0) = 1$ by convention. At any step g, the inflation level $I(g)$ represents the cumulative reward scaling relative to the initial value $I(0)$, under which the nominal and real rewards are identical.

As training progresses, the inflation level increases exponentially under a constant positive per-step inflation rate $\tilde{\rho} > 0$, a condition referred to as *positive inflation*. If $\tilde{\rho} = 0$, the inflation level remains constant, reducing the setting to standard RL when maintained throughout training. For $-1 < \tilde{\rho} < 0$, the reward scale gradually contracts over time. This case, termed *negative inflation* (or deflation), produces effects opposite to those of positive inflation. We exclude the degenerate case of $\tilde{\rho} \leq -1$, in which rewards vanish or invert, thereby distorting the original learning problem.

3.3 Fed: Adaptive Inflation Control

Building on reward inflation, we further propose *Fed*, an adaptive heuristic that controls the inflation rate during training. Inspired by the role of the Federal Reserve in stabilizing the economy by lowering interest rates during downturns and raising them during overheating, Fed adjusts the inflation rate in response to the recent dynamics of the gradient norm.

Fed monitors the gradient norm $\mathcal{G}(\phi) = \|\nabla_\phi J(\phi)\|_2$, where $J(\phi)$ is the actor loss for actor-critic methods or the Q-function loss for value-based methods. We use an exponential moving average (EMA) over a target sequence $\{x_k\}$:

$$m_k(x) = \beta \cdot m_{k-1}(x) + (1 - \beta) \cdot x_k. \tag{3}$$

Fed maintains short- and long-term EMAs m_k^s and m_k^l of $\mathcal{G}(\phi)$, with $\beta_s < \beta_l$ so that m_k^s tracks recent fluctuations while m_k^l captures longer-term trends. Their ratio

$$m_k^{s/l} = m_k^s / m_k^l \tag{4}$$

serves as a state indicator: $m_k^{s/l} > 1$ signals overheating (gradients growing faster than the long-term trend), while $m_k^{s/l} < 1$ signals a downturn.

To control inflation, Fed adjusts ρ_Y as

$$\rho_Y^{\text{new}} = \rho_Y^* - \lambda_1(m_k^{s/l} - 1) - \lambda_2(m_k^{s/l} - m_{k-1}^{s/l}), \tag{5}$$

where ρ_Y^* is the target inflation rate. The first correction term, scaled by λ_1, is a proportional response to the deviation $m_k^{s/l} - 1$, reducing ρ_Y to slow the amplification when $m_k^{s/l} > 1$, and vice versa. The second term, scaled by λ_2, is a derivative response to the rate of change $m_k^{s/l} - m_{k-1}^{s/l}$, providing anticipatory damping against rapid swings. This update rule is inspired by the Taylor Rule [46] from monetary policy and extends it with a derivative term, yielding a PD-style controller that directly modulates the inflation rate.

To prevent abrupt changes, we apply a soft update for the next ρ_Y:

$$\rho_Y = \tau_\rho \cdot \rho_Y^{\text{old}} + (1 - \tau_\rho) \cdot \rho_Y^{\text{new}}, \tag{6}$$

where ρ_Y^{old} is the previous rate and $\tau_\rho \in [0, 1)$ is the soft update coefficient. Every Y global training steps, Fed performs K updates of $m_k^{s/l}$ and ρ_Y.

4 Theoretical Analysis of Reward Inflation

In this section, we analyze the several effects of reward inflation on RL under fixed inflation.

Optimality under uniform reward scaling. A natural concern is whether scaling rewards alters the problem being solved. If all rewards are multiplied by the same positive factor $I(g)$, the resulting reward function $r'(s, a) = I(g)r(s, a)$ induces the same optimal policy under the standard return objective, since positive scaling leaves the arg max over policies unchanged. During learning, however, the replay buffer contains rewards scaled at different collection times, and its overall reward scale evolves with training progress. We now examine how these dynamics reshape TD learning.

Recency-weighting on TD update. To investigate how the inflated reward $r' = I(g) \cdot r$ behaves in Q-learning, we derive the TD-update equation at the gradient level. Throughout this analysis, we treat θ as the parameters at the current training step (held fixed). Each transition in the replay buffer $\mathcal{D}$ takes the form (s, a, r', s') and is associated with the global step g at which it was collected, so that $r' = I(g) \cdot r$ for the underlying raw reward r.

We further assume:

- (A1) The replay buffer $\mathcal{D}$ induces an approximately stationary joint distribution over transitions and collect times, so that the marginal sample distribution and the inflation factor $I(g)$ are approximately decoupled. In particular, the conditional distribution of (s, a, r, s') given collect time g is approximately equal to its marginal under $\mathcal{D}$.
- (A2) The bootstrapped term $\max_{a'} Q(s', a')$ is treated as a stop-gradient target (e.g., a target network), so ∇_θ does not pass through it.

The TD-update for Q_θ is described by the loss

$$L(\theta) = \mathbb{E}_{(s,a,r',s')\sim\mathcal{D}}\left[\tfrac{1}{2}\big(y - Q_\theta(s, a)\big)^2\right], \tag{7}$$

where $y = r' + \gamma \max_{a'} Q(s', a')$ denotes the TD-target. Its gradient is

$$\nabla_\theta L(\theta) = -\mathbb{E}_{(s,a,r',s')\sim\mathcal{D}}\big[\big(y - Q_\theta(s, a)\big)\nabla_\theta Q_\theta(s, a)\big]. \tag{8}$$

The gradient update rule for θ is $\theta \leftarrow \theta - \eta\nabla_\theta L(\theta)$, where η denotes the learning rate. To isolate the effect of collect-time inflation, we examine the expected update contribution from transitions collected at a specific global step g. Letting $\mathcal{D} \mid g$ denote the conditional distribution over $\mathcal{D}$ given collect time g, we define

$$\Delta\theta(g) = \eta\,\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}|g}\Big[\big(I(g) \cdot r + \gamma \max_{a'} Q(s', a') - Q_\theta(s, a)\big)\nabla_\theta Q_\theta(s, a)\Big], \tag{9}$$

and let $\|\Delta\theta(g)\|$ denote the corresponding update strength.

Under (A1), the conditional distribution $\mathcal{D} \mid g$ shares its marginal over (s, a, r, s') with $\mathcal{D}$, so $\Delta\theta(g)$ decomposes as

$$\Delta\theta(g) = \underbrace{\eta\, I(g)\, \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[r\, \nabla_\theta Q_\theta(s,a)]}_{\text{reward-driven component}} + \underbrace{\eta\, \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\big(\gamma \max_{a'} Q(s',a') - Q_\theta(s,a)\big)\, \nabla_\theta Q_\theta(s,a)\Big]}_{\text{bootstrapping component}}. \tag{10}$$

With Q_θ held fixed, the reward-driven component explicitly depends on the collection-time inflation factor $I(g)$, while the expected bootstrapping component has no explicit dependence on collection time g under (A1). Consequently, when comparing two collection times $g_1 < g_2$, the bootstrapping components cancel and only the reward-driven components contribute to the difference:

$$\Delta\theta(g_2) - \Delta\theta(g_1) = \big(I(g_2) - I(g_1)\big) \cdot \eta\, \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[r\, \nabla_\theta Q_\theta(s,a)]\,. \tag{11}$$

Taking norms,

$$\big\|\Delta\theta(g_2) - \Delta\theta(g_1)\big\| = \big|I(g_2) - I(g_1)\big| \cdot \eta\, \big\|\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[r\, \nabla_\theta Q_\theta(s,a)]\,\big\|. \tag{12}$$

By (12), the update discrepancy scales linearly with $|I(g_2) - I(g_1)|$, so the transition with the larger inflation factor contributes the larger reward-driven component. Under *positive inflation*, $I(g)$ increases monotonically with g, so recently collected transitions (g_2) tend to dominate older ones (g_1). Conversely, under *negative inflation* (deflation), older transitions tend to dominate.

Time-invariance of inflation weighting. For a fixed collection-time difference $g_2 - g_1$, the absolute inflation gap $|I(g_2) - I(g_1)|$ grows under constant positive inflation but shrinks under constant deflation as training progresses. The *shape* of recency-weighting in the reward-driven update, however, is invariant to absolute training time: isolating the reward-driven component $\Delta\theta_{\text{rew}}(g) := \eta\, I(g)\, \mathbb{E}_{\mathcal{D}}[r\, \nabla_\theta Q_\theta(s,a)]$ from the decomposition of $\Delta\theta(g)$, its ratio between any two transitions reduces to

$$\frac{\Delta\theta_{\text{rew}}(g_2)}{\Delta\theta_{\text{rew}}(g_1)} = \frac{I(g_2)}{I(g_1)} = (1+\tilde{\rho})^{\,g_2 - g_1}. \tag{13}$$

Under exponential inflation $I(g) = (1+\tilde{\rho})^g$ with $\tilde{\rho} > -1$, this ratio depends only on the relative time difference $g_2 - g_1$, not on the absolute training step—a consequence of the multiplicative closure $I(g+\Delta) = (1+\tilde{\rho})^\Delta\, I(g)$. The shape of recency-weighting in the reward-driven update therefore remains stationary across training: two transitions Δ steps apart always carry the same relative weight $(1+\tilde{\rho})^\Delta$, with positive inflation ($\tilde{\rho} > 0$) weighting recent transitions more heavily and deflation ($-1 < \tilde{\rho} < 0$) doing the reverse. In contrast, linear or polynomial inflation profiles induce a recency-weighting whose shape flattens over time, since $I(g_2)/I(g_1) \to 1$ as $g_1, g_2 \to \infty$ for fixed $g_2 - g_1$.

Cumulative effect via bootstrapping. While the above analysis isolates the instantaneous effect of inflation on a single update step, the bootstrapping component absorbs inflation indirectly over the course of training. Since Q_θ itself is learned from inflated rewards stored in the buffer, the term $\gamma \max_{a'} Q_\theta(s', a')$ progressively reflects the average inflation level

$$\bar{I} := \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[I(g)] \tag{14}$$

of the buffer. Under (A1), in a quasi-stationary regime where Q_θ approximately tracks the fixed point of the inflated Bellman operator, unrolling the Bellman recursion yields the approximate scaling

$$Q_\theta(s,a) \;\approx\; \bar{I} \cdot Q^*(s,a), \tag{15}$$

where Q^* denotes the optimal Q-function under the un-inflated reward r. With a finite (e.g., FIFO) replay buffer, $\bar{I}$ tracks the recent inflation level rather than the full training history. Consequently, $\bar{I}$ drifts as training progresses—upward under positive inflation and downward under deflation—inducing a non-stationary shift in the magnitude of Q_θ in the corresponding direction. In this sense, the reward-driven component injects inflation locally at each transition, while the bootstrapping component cumulatively scales the entire Q-function by the buffer-averaged inflation $\bar{I}$.

Gradient norm scaling under inflation. We now examine how the gradient norm $\|\nabla_\theta L(\theta)\|$ scales with the buffer-averaged inflation $\bar{I}$. Under the quasi-stationary scaling $Q_\theta \approx \bar{I} \cdot Q^*$ established earlier, the parametric gradient inherits the same factor, $\nabla_\theta Q_\theta \approx \bar{I} \cdot \nabla_\theta Q^*$. Since the TD-error $y - Q_\theta(s,a)$ is itself of order $\bar{I}$ (with $I(g) \approx \bar{I}$ on average within $\mathcal{D}$), the loss gradient scales as

$$\nabla_\theta L(\theta) = -\mathbb{E}_{\mathcal{D}}\big[\big(y - Q_\theta(s,a)\big)\,\nabla_\theta Q_\theta(s,a)\big] \implies \|\nabla_\theta L(\theta)\| \propto \bar{I}^2. \tag{16}$$

A similar analysis for the policy gradient yields a different scaling. For the advantage-based estimator $\nabla_\phi J(\phi) = \mathbb{E}[\nabla_\phi \log \pi_\phi(a|s) \cdot A^\pi(s,a)]$, the score function $\nabla_\phi \log \pi_\phi$ is independent of the reward scale, while the advantage $A^\pi = Q^\pi - V^\pi$ inherits the $\bar{I}$ scaling of the value functions. Hence

$$\|\nabla_\phi J(\phi)\| \propto \bar{I}. \tag{17}$$

In both cases, gradient norms grow with $\bar{I}$ under *positive inflation*, and shrink under *negative inflation* (deflation).

Convergence considerations. The approximate scaling in Eq. (15) describes tracking a moving value scale, rather than convergence to a fixed Q-function. Under persistent positive inflation, the buffer-averaged reward scale can grow without bound, so convergence to a finite Q-function is generally not guaranteed. If the reward scale instead approaches a finite positive limit and the replay buffer refreshes, the learning problem approaches a stationary, uniformly scaled MDP. Further details are provided in Appendix C.

Connection to plasticity loss. The cumulative gradient-norm scalings $\|\nabla_\theta L\| \propto \bar{I}^2$ and $\|\nabla_\phi J(\phi)\| \propto \bar{I}$ derived above suggest a mechanistic connection to plasticity loss in deep RL, where the fraction of dormant neurons grows over the course of training [41, 40]. By gradually amplifying gradient magnitudes, reward inflation sustains the effective learning rate [45] that would otherwise diminish as the policy saturates. This is consistent with prior findings that weak gradients are a primary cause of dormant neurons [47, 48].

5 Experiment

We organize our experiments around the following questions: (1) Do reward inflation and Fed improve learning performance? (2) How does reward inflation compare with related approaches? (3) What evidence supports the proposed mechanisms, and what trade-offs accompany reward inflation?

5.1 Experimental Setup

We conduct experiments on 30 games from the Arcade Learning Environment (ALE) 2600 [15] using DQN [49] to assess the overall performance impact of reward inflation. We further evaluate on eight MuJoCo tasks [16] (six locomotion and two manipulation) using Soft Actor-Critic (SAC) [50] to corroborate our theoretical analysis and conduct ablation studies, and using TD3 [51] as an alternative RL algorithm. Additionally, we use tabular Q-learning [52] with a replay buffer to examine the recency-weighting effect of reward inflation in a more controlled setting without neural function approximation.

Results are averaged over multiple random seeds, with counts specified alongside the corresponding results. All agents are evaluated using the original environment rewards without inflation. The $\pm$ values in the tables denote standard deviations across seeds. The main ALE and MuJoCo experiments run for 10M and 3M environment steps, respectively. For the inflation rate $\rho_Y(g\,;\,Y)$, we set $Y = 10^5$ steps. We set Fed's inflation target at 2%. Detailed RL hyperparameters, Fed hyperparameters, and comparison-baseline implementations are provided in Appendices A.1, A.2, and A.3, respectively.

5.2 Performance Evaluation

On both ALE and MuJoCo, we first compare baseline agents against two variants: one trained with a fixed 2% inflation, and one trained with Fed at a 2% target inflation rate. We then broaden the comparison on MuJoCo to additional inflation rates and to related approaches.

Table 1: Performance summary on 30 ALE tasks. Tied methods each receive a win.

Metric	**0%**	**2%**	**Fed (2%)**
# Winning Tasks	2	**16**	13
Mean HNS	219.5	244.8	**247.2**

Table 2: Performance comparison under varying inflation rates (upper rows) and other methods (lower rows) across MuJoCo tasks. Bold and underlined means indicate the best and second-best results across all methods, respectively. The dagger (†) marks the best value among fixed inflation rates. The last column reports the average within-table min-max normalized score across tasks. Returns are evaluated without reward inflation.

Method	**HalfCheetah**	**Walker2d**	**Humanoid**	**Swimmer**	**Hopper**	**Ant**	**Pusher**	**Reacher**	**NS**
-2%	14020±1082	4342±461	5300±176	92±43	2421±460	3929±1090	-24.25±4.10	-3.46±0.23	36.8
0%	15241±734	4399±527	5826±313	116±36	2869±299	4763±1642	-23.28±2.89	**-3.41**±0.23†	68.6
1%	**15707**±715†	5007±560	5859±363	121±25	2821±406	5245±1312†	-22.71±2.87	<u>-3.42</u>±0.23	79.8
2%	15607±372	<u>5102</u>±520†	5861±199	<u>136</u>±11†	3040±499	4346±1267	<u>-22.07</u>±1.58†	<u>-3.42</u>±0.24	<u>81.3</u>†
3%	15091±927	4690±624	<u>5873</u>±306†	126±11	3035±445	4392±1298	-22.58±1.58	-3.44±0.24	71.5
5%	14484±996	4615±518	5545±492	131±12	<u>3161</u>±346	4795±1089	-22.20±0.89	-3.45±0.23	71.5
10%	14186±1200	4167±650	5209±1241	129±19	**3166**±288†	4248±1219	-24.60±1.47	-3.54±0.33	46.6
Fed (2%)	<u>15621</u>±991	**5395**±452	**6048**±436	**143**±13	3105±293	5180±1352	**-21.99**±1.23	-3.43±0.24	**92.8**
2% (lr)	14566±1006	4922±515	5417±298	77±41	2899±359	4458±1888	-23.95±2.88	<u>-3.42</u>±0.24	59.2
PER	14734±1934	4797±592	4542±2290	57±23	2683±529	**5762**±979	-26.41±2.98	-3.43±0.24	48.7
RN	15022±436	5036±466	5578±285	118±13	3149±272	5017±1666	-29.17±13.37	-3.46±0.24	64.1
RTN	11877±369	4454±571	4451±633	87±21	2742±463	4046±1603	-23.41±2.05	-3.45±0.13	32.1
RWS	15482±687	5000±546	5730±362	104±40	2923±469	<u>5466</u>±1318	-23.32±3.55	-3.43±0.20	76.8

5.2.1 Arcade Learning Environment

To investigate how reward inflation and Fed generally affect training performance, we evaluated 30 ALE Atari 2600 games using standard DQN. Table 1 compares the DQN baseline, 2% reward inflation, and Fed with a target inflation rate of 2%, summarizing the average performance over the final 1M training steps, with the last row reporting the average Human-Normalized Score (HNS) [53]. Results are averaged over 8 random seeds per condition. Surprisingly, despite the small modification, reward inflation significantly improved performance in most games, yielding an average HNS improvement of 11.53% over the baseline (mean baseline-normalized improvement: 17.28%, 95% CI $[5.16\%, 29.41\%]$; one-sample t-test against zero, $t(29) = 2.92$, $p = 0.0068$). Fed further improved performance, achieving a 12.59% HNS gain over the baseline (mean baseline-normalized improvement: 19.60%, 95% CI $[2.35\%, 36.85\%]$; $t(29) = 2.32$, $p = 0.027$). These results highlight the beneficial impact of reward inflation. The full table, learning curves, and bootstrap CIs are provided in Table 10, Figure 5, and Figure 6, respectively.

5.2.2 MuJoCo

We further evaluated on eight MuJoCo tasks. This time, we evaluated various inflation rates to examine how each rate differently affects training, using rates of -2%, 0%, 1%, 2%, 3%, 5%, and 10%. For comparison with related approaches, we additionally evaluated 2% learning rate (LR) inflation, PER, reward normalization (RN), return normalization (RTN), and recency-weighted sampling (RWS). The 2% (lr) baseline increases learning rates rather than scaling rewards. RWS, inspired by ERE, is designed to match the recency-weighting effect of 2% reward inflation (see Appendix A.3).

Table 2 reports the episode return averaged over the final 300K training steps. Results are averaged over 10 random seeds per condition. The last column reports the average within-table min-max normalized score (NS) across tasks, where normalization is performed independently for each task using the minimum and maximum scores among all compared methods. On MuJoCo, compared to the baseline (0%), 2% inflation improved the NS by 18.51%, while Fed with a target inflation rate of 2% further increased the improvement to 35.21%. This result again highlights the general benefits of reward inflation.

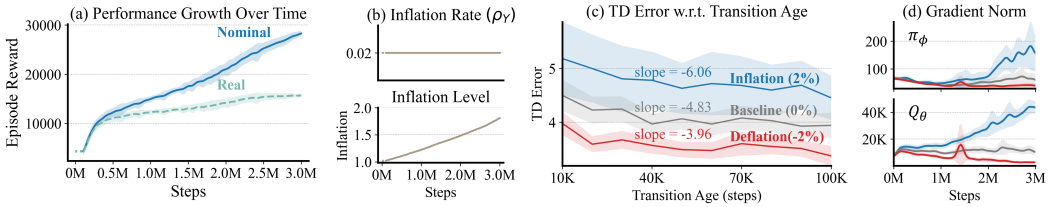


Figure 1: (a) The nominal and real performance growth over training time under 2% inflation. (b) The inflation rate and inflation level $I(g)$ over the training time. (c) The TD error w.r.t. the transition age in replay buffer, across the inflation rates. (d) The changes of gradient norms during training.

Performance under variety of inflation rates. Overall, based on NS, while 1–5% inflation showed improvements over the baseline, the two extremes, -2% deflation and 10% inflation, resulted in severe degradation (-46.43% and -32.16%, respectively). Since -2% deflation induces reversed recency weighting, emphasizing older samples and shrinking gradient norms, performance was generally degraded. Conversely, the degradation under 10% inflation suggests that while moderate inflation can act as a healthy stimulus, excessive inflation introduces side effects that hurt training. However, since 10% inflation achieved the best performance on `Hopper`, where scores increased alongside the inflation rate, the most beneficial rate appears to be task-dependent. Statistically, the set of per-task best rates yielded a mean baseline-normalized improvement of 5.40% (95% CI $[0.30\%, 10.50\%]$; one-sample t-test against zero, $t(7) = 2.50$, $p = 0.041$), while Fed yielded 7.31% ($[0.83\%, 13.79\%]$; $t(7) = 2.67$, $p = 0.032$). Bootstrap CIs are also provided in Figure 7.

Comparison with related approaches. We compare reward inflation against several methods that share a related mechanism but intervene at different stages or through different signals. While reward inflation induces an implicit recency weighting that assigns a different effective learning rate to each sample depending on its recency, LR inflation raises the update magnitude uniformly across all samples, degrading performance well below the baseline. PER likewise reshapes the effective gradient signal, but does so by selectively up-weighting transitions with large TD errors; while this yields the strongest result on `Ant`, the gains are inconsistent across tasks. RN and RTN also rescale rewards, but use running statistics rather than an explicit inflation schedule. Their performance varies across tasks, with neither method exceeding the baseline's aggregate NS. While RWS yields only marginal gains over the baseline, moderate reward inflation and Fed deliver larger and more consistent improvements across tasks.

Results with TD3. To examine whether the benefits of reward inflation extend beyond SAC, we additionally evaluated TD3 with and without 2% inflation on the same MuJoCo tasks over 8 random seeds per condition (Table 11). Inflation yielded modest AUC gains on seven of the eight tasks, while improvements in final scores were smaller and less consistent. These results suggest that the benefit can also occur without entropy-temperature tuning, although the larger gains observed with SAC may partly reflect its interaction with reward scaling.

5.3 Empirical Analysis of Reward Inflation

Nominal and real episode reward. Maximizing nominal return is one distinctive property of reward inflation. Figure 1(a,b) illustrates the gap between nominal and real episode rewards, alongside the change in the inflation level $I(g)$ during training. This example uses `HalfCheetah-v4` with a 2% inflation rate. The nominal episode reward increased consistently faster than the real episode reward, in proportion to $I(g)$. Note that the real episode reward is reported as the performance metric.

Recency weighting and gradient norms. To investigate the impact of reward inflation on recency weighting and gradient norms, we evaluated agents with 2% (inflation), 0% (baseline), and -2% (deflation) inflation rates on `HalfCheetah-v4`. Recency weighting was quantified by analyzing the dependence of TD error on the age of transitions. As shown in Figure 1(c), older transitions, which have been sampled more frequently, tended to exhibit lower TD errors. However, inflation amplified the TD error gap between older and more recent transitions, whereas deflation reduced it. Figure 1(d)

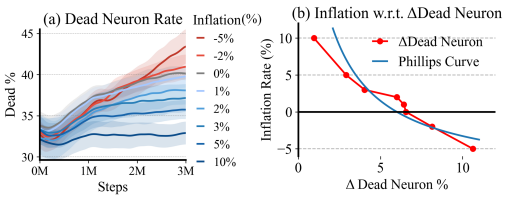


Figure 2: (a) The changes in dormant (dead) neuron rate during training from deflation (-5%) to inflation (10%). (b) The total increased rate of dead neurons under each inflation rate.

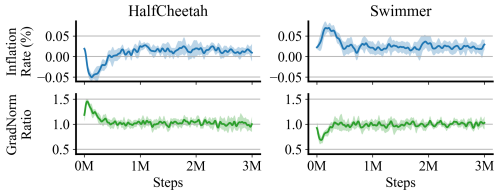


Figure 3: Inflation rates (top) adjusted by Fed in response to the observed short-long gradient norm ratios $m^{s/l}$ (bottom), on `HalfCheetah-v4` and `Swimmer-v4`.

further illustrates how the progression of gradient norms over time differs across the three inflation rates. These observations support the findings from the theoretical analysis.

Impacts on dormant neurons. A dormant (or dead) neuron [54] is defined as a neuron that persistently outputs zero, contributing little or nothing to further learning. To investigate how reward inflation affects dormant neurons, we tracked the proportion of dormant neurons in both the Q_θ and π^ϕ networks throughout training.

Figure 2(a) shows how the proportion of dormant neurons evolves under inflation rates ranging from -5% to 10% on `HalfCheetah-v4`. Surprisingly, a clear negative correlation emerges: the final proportion of dormant neurons decreases as the inflation rate increases, with deflation yielding a higher proportion than the baseline. Although dormant neurons are commonly associated with ReLU, similar behavior has been reported under other activations [55]; the same pattern accordingly holds across other tasks and activation functions (Figures 8, 9, and 10). Jointly observing Figure 2(a) and Figure 1(d) shows that under the baseline (0%), the gradient norm generally decreases while the proportion of dormant neurons concurrently increases. The decrease in gradient norm is expected as the agent's predictions align more closely with its learning targets. Positive inflation instead increases the gradient norm over training, suppressing the emergence of dormant neurons, whereas deflation accelerates gradient decay and produces a more rapid rise in dormant neurons.

Intuitively, the decrease in dormant neuron rates with increasing inflation in Figure 2(b) resembles the Phillips curve [56], which captures the inflation-unemployment trade-off and echoes Keynesian ideas of activating idle resources [57]. Analogously, reducing dormant neurons may improve network utilization. However, just as hyperinflation is undesirable, the lowest dormant neuron rate does not imply the best performance: 10% inflation yields the fewest dormant neurons but the worst overall result. This may reflect a trade-off between network utilization [40] and the benefits of sparsity [58], although the present results do not isolate these factors.

We additionally compare reward inflation with related plasticity baselines ReDo [40], Reset [42], and Shrink & Perturb [59] (Tables 12 and 13; implementation details in Appendix A.3). These methods generally reduce dormant neurons more strongly, but their performance gains are not consistent across tasks. This suggests that dormant-neuron reduction alone is insufficient to explain the performance benefits of reward inflation. A HalfCheetah task-switch experiment also suggests a potential benefit for adaptation to changing objectives (Appendix B.1).

Fed dynamics. Fed adjusts the inflation rate in response to changes in the short-long gradient ratio $m^{s/l}$. Figure 3 illustrates how Fed controls inflation in two contrasting cases. In `HalfCheetah-v4`, $m^{s/l}$ surged rapidly at the beginning of training, indicating overheating caused by strong learning signals. As this task is known for rapid early performance gains, Fed responded by sharply lowering the inflation rate, even temporarily entering deflation. Once $m^{s/l}$ stabilized, the inflation rate recovered to the target level of 2% and remained relatively stable thereafter. In contrast, `Swimmer-v4` exhibited the opposite trend. Due to its slow and reward-scarce early phase, $m^{s/l}$ initially declined, prompting Fed to increase the inflation rate to stimulate learning. As training progressed and $m^{s/l}$ stabilized, the inflation rate gradually converged to the target level.

Table 3: TD-error recency and learning performance in tabular Q-learning on slippery 8×8 FrozenLake. The TD-error slope is fitted to mean $\log |\delta|$ as a function of transition age; more negative values indicate stronger recency weighting. Success rate and learning-curve AUC are reported as mean $\pm$ standard deviation over 32 seeds.

Inflation	TD-error slope	Success rate	AUC
0%	-2.039×10^{-4}	0.478 ± 0.132	0.407 ± 0.014
1%	-7.645×10^{-4}	0.483 ± 0.166	0.408 ± 0.015
5%	-3.220×10^{-3}	0.520 ± 0.109	0.422 ± 0.015
10%	-6.464×10^{-3}	0.507 ± 0.109	0.431 ± 0.018
50%	-2.600×10^{-2}	0.584 ± 0.047	0.475 ± 0.015
100%	-3.753×10^{-2}	0.585 ± 0.030	0.482 ± 0.015
500%	-8.895×10^{-2}	0.586 ± 0.033	$\mathbf{0.484} \pm 0.014$
1000%	-1.231×10^{-1}	$\mathbf{0.594} \pm 0.031$	$\mathbf{0.484} \pm 0.018$
5000%	-1.930×10^{-1}	0.574 ± 0.035	0.474 ± 0.019

5.4 Tabular Q-Learning: Recency Weighting and Its Trade-Offs

Recency weighting and learning performance. To examine reward inflation without function approximation, we trained tabular Q-learning agents on slippery 8×8 FrozenLake for 2M steps over 32 seeds, using inflation rates from 0% to $5{,}000\%$. Algorithmic and environment details are provided in Appendix A.4. We measure recency weighting through the slope of mean log absolute TD error against transition age for reward-bearing transitions; a more negative slope indicates larger TD errors for recent transitions.

In Table 3, the slope becomes more negative with inflation despite uniform replay sampling. Success rate and AUC generally improve, although gains diminish at the highest rates. These results support stronger recency weighting and faster learning in this tabular setting.

Trap-entry evaluation. However, stronger recency weighting may increase susceptibility to deceptive rewards. We therefore introduce an easily reached low-reward goal alongside a harder-to-reach high-reward goal and compare two exploration schedules. As shown in Table 14, reward inflation increases trapping at high rates, while slower ϵ decay reduces it. This suggests that inflation may require a larger exploration budget to avoid deceptive rewards.

5.5 Limitations

Several limitations remain. First, our experiments cover finite training horizons. Prolonged inflation can destabilize optimization through reward-scale growth, motivating stabilization or saturation mechanisms (see Appendix C). Second, our current formulation relies on off-policy learning with replay, where transitions collected at different reward scales coexist in the buffer. Extending this mechanism to on-policy learning remains future work. Third, the optimal inflation behavior appears to depend on task characteristics and training dynamics, suggesting that a universally optimal inflation schedule may not exist. Finally, Fed currently relies on multiple hyperparameters and a heuristic control rule, motivating future work on more principled and adaptive inflation-control strategies.

6 Conclusion

In this paper, we introduced reward inflation, a simple temporal reward-scaling scheme that can act as a healthy stimulus for off-policy RL with experience replay. Experiments on ALE and MuJoCo demonstrate performance benefits across a broad range of tasks under moderate inflation, with Fed often improving upon fixed inflation. Our analysis characterizes implicit recency weighting and gradient scaling, while experiments identify dormant-neuron suppression as a possible contributing mechanism. Tabular Q-learning experiments further support the recency effect while revealing increased susceptibility to deceptive rewards at high inflation rates. These findings motivate further investigation into temporal reward dynamics as a lightweight approach to improving RL.

Acknowledgment

This work was supported in part by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) grant funded by the Defense Acquisition Program Administration (DAPA) (No. KRIT-CT-23-003/10%, Development of AI Researchers Based on Deep Reinforcement Learning and Establishment of Virtual Combat Experiment Environment), and in part by the IITP (RS-2021-II211343-GSAI/10%, RS-2022-II220951-LBA/10%, RS-2022-II220953-PICA/10%, RS-2026-25553157-MIACC/10%), the NRF (RS-2024-00353991-SPARC/10%, RS-2023-00274280-HEI/10%, RS-2024-00358416-AutoRL/10%), the KEIT (RS-2025-25453780/10%), and the KIAT (RS-2025-25460896/10%), funded by the Korean government.

References

[1] Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28694–28698, 2025.

[2] Konstantinos Souchleris, George K Sidiropoulos, and George A Papakostas. Reinforcement learning in game industry—review, prospects and challenges. *Applied Sciences*, 13(4):2443, 2023.

[3] Ganghun Lee, Minji Kim, Yunsu Lee, Minsu Lee, and Byoung-Tak Zhang. Neural collage transfer: Artistic reconstruction via material manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2394–2405, 2023.

[4] Rui Zhao, Yun Li, Yuze Fan, Fei Gao, Manabu Tsukada, and Zhenhai Gao. A survey on recent advancements in autonomous driving using deep reinforcement learning: Applications, challenges, and solutions. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[5] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

[6] Yahui Bai, Yuhe Gao, Runzhe Wan, Sheng Zhang, and Rui Song. A review of reinforcement learning in financial applications. *Annual Review of Statistics and Its Application*, 12(1):209–232, 2025.

[7] Keliang Liu, Dingkang Yang, Ziyun Qian, Weijie Yin, Yuchi Wang, Hongsheng Li, Jun Liu, Peng Zhai, Yang Liu, and Lihua Zhang. Reinforcement learning meets large language models: A survey of advancements and applications across the llm lifecycle. *arXiv preprint arXiv:2509.16679*, 2025.

[8] Jonas Eschmann. Reward function design in reinforcement learning. *Reinforcement learning algorithms: Analysis and Applications*, pages 25–33, 2021.

[9] Tom Schaul, Georg Ostrovski, Iurii Kemaev, and Diana Borsa. Return-based scaling: Yet another normalisation trick for deep rl. *arXiv preprint arXiv:2105.05347*, 2021.

[10] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer, 1999.

[11] Haozhe Ma, Zhengding Luo, Thanh Vinh Vo, Kuankuan Sima, and Tze-Yun Leong. Highly efficient self-adaptive reward shaping for reinforcement learning. *arXiv preprint arXiv:2408.03029*, 2024.

[12] Arthur Aubret, Laetitia Matignon, and Salima Hassas. An information-theoretic perspective on intrinsic motivation in reinforcement learning: A survey. *Entropy*, 25(2):327, 2023.

[13] Kashish Gupta, Debasmita Mukherjee, and Homayoun Najjaran. Extending the capabilities of reinforcement learning through curriculum: A review of methods and applications. *SN Computer Science*, 3(1):28, 2022.

[14] Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: A survey, 2025.

[15] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[16] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[17] Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Automatic intrinsic reward shaping for exploration in deep reinforcement learning. In *International Conference on Machine Learning*, pages 40531–40554. PMLR, 2023.

[18] Arthur Aubret, Laetitia Matignon, and Salima Hassas. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

[19] Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. Reward shaping via meta-learning. *arXiv preprint arXiv:1901.09330*, 2019.

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

[21] Junseok Park, Hyeonseo Yang, Min Whoo Lee, Won-Seok Choi, Minsu Lee, and Byoung-Tak Zhang. From sparse to dense: Toddler-inspired reward transition in goal-oriented reinforcement learning. *arXiv preprint arXiv:2501.17842*, 2025.

[22] Kilian Freitag, Kristian Ceder, Rita Laezza, Knut Åkesson, and Morteza Haghir Chehreghani. Curriculum reinforcement learning for complex reward functions. *arXiv preprint arXiv:2410.16790*, 2024.

[23] Mihai Anca, Jonathan D Thomas, Dabal Pedamonti, Mark Hansen, and Matthew Studley. Achieving goals using reward shaping and curriculum learning. In *Proceedings of the Future Technologies Conference*, pages 316–331. Springer, 2023.

[24] Yash Shukla, Abhishek Kulkarni, Robert Wright, Alvaro Velasquez, and Jivko Sinapov. Automaton-guided curriculum generation for reinforcement learning agents. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pages 605–613, 2023.

[25] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[27] Hado P Van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. *Advances in neural information processing systems*, 29, 2016.

[28] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado Van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

[29] Michael Dann and John Thangarajah. Adapting to reward progressivity via spectral reinforcement learning. *arXiv preprint arXiv:2104.14138*, 2021.

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

[31] Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. Experience replay optimization. *arXiv preprint arXiv:1906.08387*, 2019.

[32] Hongjie Zhang, Cheng Qu, Jindou Zhang, and Jing Li. Self-adaptive priority correction for prioritized experience replay. *Applied sciences*, 10(19):6925, 2020.

[33] Peiquan Sun, Wengang Zhou, and Houqiang Li. Attentive experience replay. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5900–5907, 2020.

[34] Jue Ma, Dejun Ning, Chengyi Zhang, and Shipeng Liu. Fresher experience plays a more important role in prioritized experience replay. *Applied sciences*, 12(23):12489, 2022.

[35] Shivakanth Sujit, Somjit Nath, Pedro Braga, and Samira Ebrahimi Kahou. Prioritizing samples in reinforcement learning with reducible loss. *Advances in Neural Information Processing Systems*, 36:23237–23258, 2023.

[36] Che Wang and Keith Ross. Boosting soft actor-critic: Emphasizing recent experience without forgetting the past. *arXiv preprint arXiv:1906.04009*, 2019.

[37] Baturay Saglam, Furkan B Mutlu, Dogan C Cicek, and Suleyman S Kozat. Actor prioritized experience replay. *Journal of Artificial Intelligence Research*, 78:639–672, 2023.

[38] Hoda Yamani, Yuning Xing, Lee Violet C Ong, Bruce A MacDonald, and Henry Williams. Reward prediction error prioritisation in experience replay: The rpe-per method. *arXiv preprint arXiv:2501.18093*, 2025.

[39] Brett Daley, Marlos C Machado, and Martha White. Demystifying the recency heuristic in temporal-difference learning. *arXiv preprint arXiv:2406.12284*, 2024.

[40] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pages 32145–32168. PMLR, 2023.

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

[42] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR, 2022.

[43] Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.

[44] Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado Van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv:2402.18762*, 2024.

[45] Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. *Advances in Neural Information Processing Systems*, 37:106440–106473, 2024.

[46] John B Taylor. Discretion versus policy rules in practice. In *Carnegie-Rochester conference series on public policy*, volume 39, pages 195–214. Elsevier, 1993.

[47] Isac Arnekvist, J Frederico Carvalho, Danica Kragic, and Johannes A Stork. The effect of target normalization and momentum on dying relu. *arXiv preprint arXiv:2005.06195*, 2020.

[48] Coşku Can Horuz, Geoffrey Kasenbacher, Saya Higuchi, Sebastian Kairat, Jendrik Stoltz, Moritz Pesl, Bernhard A Moser, Christoph Linse, Thomas Martinetz, and Sebastian Otte. The resurrection of the relu. *arXiv preprint arXiv:2505.22074*, 2025.

[49] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[50] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[51] Stephen Dankwa and Wenfeng Zheng. Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In *Proceedings of the 3rd international conference on vision, image and signal processing*, pages 1–5, 2019.

[52] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[53] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR, 2020.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[55] Murilo Gustineli. A survey on recently proposed activation functions for deep learning. *arXiv preprint arXiv:2204.02921*, 2022.

[56] Alban W Phillips. The relation between unemployment and the rate of change of money wage rates in the united kingdom, 1861-1957. *economica*, 25(100):283–299, 1958.

[57] Paul A Samuelson and Robert M Solow. Analytical aspects of anti-inflation policy. *The American economic review*, 50(2):177–194, 1960.

[58] Simon Dufort-Labbé, Pierluca D'Oro, Evgenii Nikishin, Razvan Pascanu, Pierre-Luc Bacon, and Aristide Baratin. Maxwell's demon at work: Efficient pruning by leveraging saturation of neurons. *arXiv preprint arXiv:2403.07688*, 2024.

[59] Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

[60] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

A Experimental Details

A.1 RL Hyperparameters

SAC. Unless otherwise specified, SAC experiments use the hyperparameters listed in Table 4. These settings largely follow the CleanRL implementation [60]. We train for 3M environment steps to examine the effects of reward inflation over an extended training horizon. We increase the warm-up period from 5K to 100K steps because the shorter warm-up occasionally produced poor performance for both the baseline and inflated agents. For *Hopper-v4* and *Ant-v4*, we adjust the discount factor to reduce performance volatility. The SAC baseline agents with these adjusted discount factors also outperform those using the default $\gamma = 0.99$. The task-specific discount factors in Table 5 are used for both SAC and TD3.

Table 4: SAC hyperparameter configuration.

Parameter	**Value**
Total environment steps	3M
Replay buffer size	1M
Target update coefficient (τ)	0.005
Batch size	256
Warm-up steps	100K
Learning rate (Q_θ)	0.001
Learning rate (π_ϕ)	0.0003
Update interval (Q_θ)	1
Update interval (π_ϕ)	2
Update interval (target Q_θ)	1
Initial entropy coefficient (α)	1.0
Automatic α tuning	True
Activation	ReLU
Optimizer	Adam

Table 5: Task-specific discount factors for SAC and TD3.

Task	**Discount factor (γ)**
`Hopper-v4`	0.995
`Ant-v4`	0.975
Others	0.99

TD3. Our TD3 implementation also follows CleanRL [60], with hyperparameters summarized in Table 6 and discount factors specified in Table 5. Actions are sampled uniformly during the initial 25K steps, followed by actor actions with Gaussian exploration noise. Exploration and target policy noise are scaled by half the action-range width, with target noise clipped before scaling. The resulting actions are clipped to the valid action bounds. The actor and target networks are updated every two training steps, while both critics are updated every step. We evaluate the deterministic policy without exploration noise every 10K steps, using one episode per checkpoint and reporting the original, uninflated return.

DQN. Our DQN implementation follows CleanRL [60], with hyperparameters summarized in Table 7. ALE preprocessing includes up to 30 random no-op actions at reset, action repeat of four with max pooling, episodic-life handling, and fire-reset handling where applicable. Sticky actions are disabled. Observations are resized to 84×84 grayscale images and stacked over four frames. Training rewards are sign-clipped before reward inflation is applied and stored in replay; reported episodic returns use the original, unclipped and uninflated environment rewards. Step counts refer to interactions with the wrapped environment rather than individual emulator frames.

Table 6: TD3 hyperparameter configuration. Noise parameters are expressed relative to half the action-range width.

Parameter	**Value**
Total environment steps	3M
Replay buffer size	1M
Batch size	256
Warm-up steps	25K
Actor learning rate	0.0003
Critic learning rate	0.0003
Discount factor (γ)	See Table 5
Target update coefficient (τ)	0.005
Critic update interval	1
Actor update interval	2
Target network update interval	2
Exploration noise standard deviation	0.2
Target policy noise standard deviation	0.2
Target policy noise clipping	$[-0.5, 0.5]$
Hidden layers (actor and critics)	$256, 256$
Hidden activation	ReLU
Optimizer	Adam
Evaluation interval	10K steps
Evaluation episodes per checkpoint	1

Table 7: DQN hyperparameter configuration for ALE. Convolutional specifications list channels, kernel size, and stride.

Parameter	**Value**
Total environment steps	10M
Replay buffer size	1M
Batch size	32
Warm-up steps	80K
Learning rate	10^{-4}
Discount factor (γ)	0.99
Gradient update interval	4
Target network update interval	1K
Target update coefficient (τ)	1.0 (hard update)
Initial exploration rate (ϵ)	1.0
Final exploration rate (ϵ)	0.01
Linear exploration decay duration	1M steps
Input shape	$4 \times 84 \times 84$
Input normalization	Divide by 255
Convolutional layer 1	32 channels, kernel 8, stride 4
Convolutional layer 2	64 channels, kernel 4, stride 2
Convolutional layer 3	64 channels, kernel 3, stride 1
Fully connected hidden layer	512 units
Output dimension	Number of actions
Hidden activation	ReLU
Optimizer	Adam

A.2 Fed Hyperparameters

We selected the Fed hyperparameters through a simple search. Table 8 summarizes the configuration.

Table 8: Fed hyperparameter configuration.

Parameter	Value
Short-term EMA coefficient (β_s)	0 (instantaneous update)
Long-term EMA coefficient (β_l)	0.83
Inflation interval (Y)	10^5 environment steps
Controller updates per interval (K)	12
Target inflation rate (ρ_Y^*)	0.02
Controller coefficient (λ_1)	0.2
Controller coefficient (λ_2)	0.2
Inflation-rate smoothing coefficient (τ_ρ)	0.83

A.3 Implementation of Comparison Baselines

Reward normalization (RN). We normalize rewards using the running standard deviation of observed rewards:

$$r_t^{\mathrm{RN}} = \frac{r_t}{\sigma_r}.$$

We do not subtract the running mean. This avoids introducing an additive reward shift, which can alter incentives for episode termination.

Return normalization (RTN). Following the normalization form used by VecNormalize, we divide rewards by the running standard deviation of discounted returns:

$$r_t^{\mathrm{RTN}} = \frac{r_t}{\sigma_R}.$$

Both RN and RTN use running statistics rather than fixed statistics. Automatic entropy-temperature tuning remains enabled in all SAC comparisons.

Learning-rate inflation. The "2% (lr)" baseline applies the same multiplicative schedule as 2% reward inflation to the learning rates instead of rewards. Thus, the learning rates increase throughout training, rather than remaining at a fixed larger value.

Prioritized experience replay (PER). We use proportional prioritized experience replay [30], sampling transition i with probability $P(i) = p_i^{0.6} / \sum_j p_j^{0.6}$. Priorities are updated as $p_i = |\delta_i| + 10^{-6}$, where δ_i is the first critic's TD error computed using the SAC target. New transitions receive the maximum priority currently in the buffer. Both critic losses use importance-sampling weights $w_i = (NP(i))^{-\beta}$, normalized by the maximum weight within the sampled minibatch, where N is the current buffer size. We initialize $\beta = 0.4$ and increase it by 10^{-6} after each sampling call, capped at 1. Actor and entropy-temperature losses use the same sampled transitions without importance weighting.

Recency-weighted sampling (RWS). We designed RWS using a recency-biased replay buffer, where samples are drawn according to the probability mass function:

$$p(a) = \frac{(1-r)r^{a-1}}{1-r^N},$$

where $a = 1, 2, \ldots, N$ denotes the sample age, N is the replay buffer size, and $0 < r < 1$ is the decay rate. This assigns higher probability to more recent transitions (smaller a), with the ratio $\frac{p(a_2)}{p(a_1)} = r^{a_2 - a_1}$ for $a_2 > a_1$. Thus, the sampling scheme imposes a geometric decay over sample age, analogous to the geometric scaling of rewards across global steps induced by reward inflation. We set $r = 0.98^{1/100000}$ to align with the notion of 2% reward inflation.

RWS biases the sampling process, causing the networks (Q_θ or π_ϕ) to encounter recent transitions more frequently. In contrast, reward inflation preserves uniform sampling but assigns relatively larger rewards to recently acquired transitions.

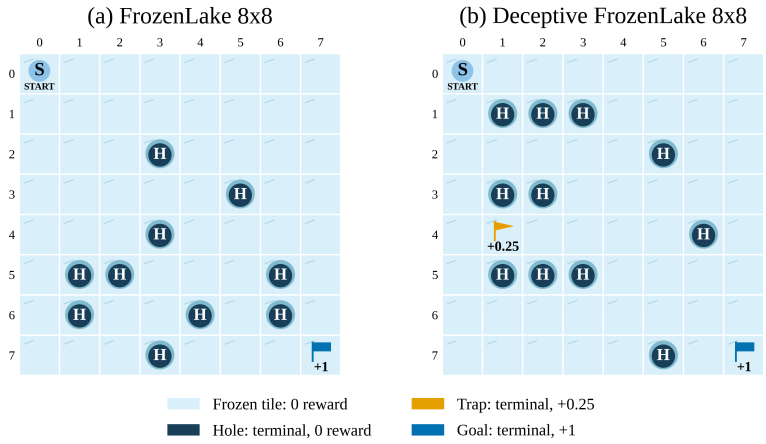


Figure 4: FrozenLake maps: (a) the standard map and (b) the deceptive-reward map, with terminal goal rewards of 0.25 and 1.0. Both environments use slippery transitions.

Plasticity baselines. For ReDo, we use a recycling period of 1K steps and a dormant-neuron threshold of 0.025. Reset is applied every 200K steps. Shrink & Perturb is applied every 1K steps with a shrink factor of 0.99.

A.4 Tabular Q-Learning Settings

Environments. Figure 4 shows the two 8×8 FrozenLake maps. The standard setting uses the slippery `FrozenLake-v1` environment. The deceptive-reward setting also uses slippery transitions, with a start at $(0, 0)$, a low-reward goal at $(4, 1)$ yielding 0.25, and a high-reward goal at $(7, 7)$ yielding 1.0, using zero-based row and column indices. All other transitions yield zero reward. Reaching either goal or falling into a hole terminates the episode; episodes are truncated after 200 steps. The deceptive-reward experiment shares the replay Q-learning settings in Table 9, with ϵ reaching 0.01 at 1M steps under the lower-exploration schedule and at 2M steps under the higher-exploration schedule. We classify a run as trapped if its final greedy policy reaches the low-reward goal more frequently than the high-reward goal over 1,000 evaluation episodes under slippery dynamics. The reported percentage is the fraction of the 32 runs classified as trapped.

Algorithm and hyperparameters. We use tabular Q-learning with a zero-initialized Q-table and uniform experience replay. After collecting 100 transitions, we perform one update per environment step using a single transition sampled uniformly from the replay buffer. Rewards are multiplied by the collection-time inflation level before insertion and remain unchanged thereafter. Bootstrapping is disabled for transitions ending in either termination or truncation. Table 9 summarizes the settings. The ϵ-greedy exploration rate decreases linearly from 1.0 to 0.01 over the first 1M steps and remains at 0.01 thereafter. Only the higher-exploration condition in the deceptive-reward experiment extends this decay to the full 2M training steps. For the standard-map experiment, greedy policies are evaluated every 10K steps over 200 episodes, with a separate final evaluation over 1,000 episodes. Success rate is the fraction of evaluation episodes reaching the goal. Learning-curve AUC is computed by trapezoidal integration and divided by the training horizon. For the recency analysis, we record $\log(|\delta| + 10^{-12})$ before each replay update, aggregate it by transition age for reward-bearing transitions, and fit weighted least squares using the number of samples at each age as the weight.

A.5 Resources

All experiments were conducted on a system with multiple Linux servers including GPUs as NVIDIA RTX 4090, RTX A6000, RTX A5000, and TITAN RTX, paired with CPUs as AMD EPYC 9354 32-Core, AMD EPYC 9554 64-Core, AMD Ryzen Threadripper PRO 5975WX 32-Cores and AMD

Table 9: Tabular Q-learning hyperparameters. The longer exploration decay applies only to the higher-exploration condition in the deceptive-reward experiment.

Parameter	**Value**
Total environment steps	2M
Learning rate	0.1
Discount factor (γ)	0.99
Initial Q-values	0
Replay buffer capacity	10K
Warm-up transitions	100
Replay updates per environment step	1
Transitions per replay update	1
Initial exploration rate (ϵ)	1.0
Final exploration rate (ϵ)	0.01
Linear exploration decay duration	1M steps
Higher-exploration decay duration	2M steps
Inflation interval (Y)	10^5 steps

Ryzen Threadripper 3970X 32-Cores, respectively. On average, a single experimental run for both ALE (10M steps) and MuJoCo (3M steps) takes approximately 8 hours on the reported hardware.

B Additional Results

Table 10: Performance comparison of DQN agents in ALE games under reward inflation (0%, 2%, and Fed (2%)).

Task	**0%**	**2%**	**Fed (2%)**	**Task**	**0%**	**2%**	**Fed (2%)**
Alien	1250±352	**1513**±124	1482±96	Frostbite	2732±364	**3377**±543	3172±579
Amidar	296±33	**468**±64	456±83	Gopher	12595±3870	**16160**±4948	15954±5352
Assault	3654±252	**3814**±449	3401±396	Hero	**2758**±1779	2731±624	2549±215
Asterix	22178±7214	**31424**±5245	29492±3913	Jamesbond	486±63	**601**±154	567±120
BankHeist	650±251	776±189	**797**±109	Kangaroo	7821±1762	10885±2742	**11983**±2016
BattleZone	22659±1074	22195±4406	**24501**±945	Krull	7584±2605	7969±2602	**8799**±419
BeamRider	6906±662	**8682**±904	7908±759	KungFu	3853±4542	8257±8977	**12075**±11337
Bowling	22.46±0.63	**23.55**±0.90	23.30±0.50	MsPacman	2218±141	**2365**±102	2181±135
Boxing	91.37±2.06	94.93±1.02	**95.44**±1.00	Pong	20.40±0.1	**20.58**±0.09	20.30±0.40
Breakout	367±19	**375**±14	**375**±21	PrivateEye	**621**±1229	63±117	81±143
ChopperCmd	1178±508	1492±658	**1553**±1301	Qbert	11754±1849	12317±1376	**12831**±1756
CrazyClimber	93919±8113	103214±10735	**105703**±10671	Riverraid	8228±641	**8699**±1122	8350±840
DemonAttack	32572±9621	50007±6358	**50466**±6846	RoadRunner	34017±1172	35502±6092	**40112**±3054
Enduro	1275±108	1542±165	**1553**±150	Seaquest	4074±632	**5138**±1024	4635±1048
Freeway	23.44±3.58	**24.81**±4.80	23.53±3.73	SpaceInvade	1349±106	**1565**±148	1483±222

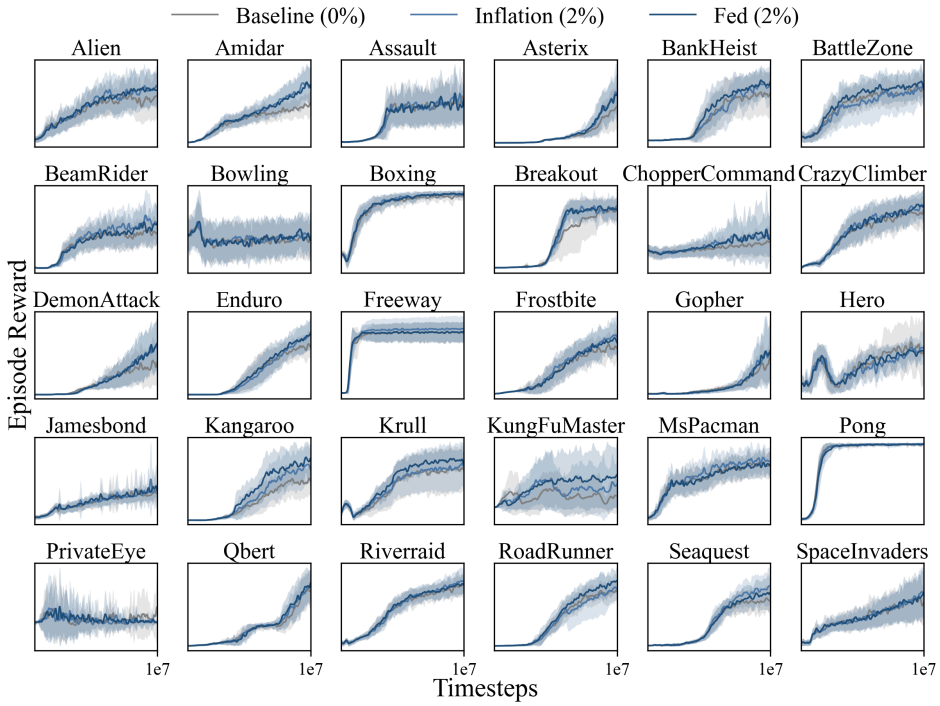


Figure 5: Learning curves of DQN agents across all ALE games under different reward inflation settings (0%, 2%, and Fed with 2% target).

Table 11: TD3 performance on eight MuJoCo tasks with and without 2% reward inflation. Results are mean ± standard deviation across eight runs trained for 3M timesteps. Final scores are averaged over the final 10% of training, and AUC is computed over the full learning curve. All evaluations use the original, uninflated rewards. Bold indicates the higher mean within each metric and task.

	Final score		**AUC**	
Task	**0%**	**2%**	**0%**	**2%**
HalfCheetah	13662 ± 651	$\mathbf{13730} \pm 618$	11213 ± 592	$\mathbf{11537} \pm 220$
Walker2d	4862 ± 471	$\mathbf{4881} \pm 557$	3858 ± 443	$\mathbf{3889} \pm 773$
Humanoid	5788 ± 338	$\mathbf{6023} \pm 503$	4854 ± 222	$\mathbf{5010} \pm 279$
Swimmer	$\mathbf{122} \pm 25$	120 ± 37	94 ± 23	$\mathbf{97} \pm 27$
Hopper	3075 ± 397	$\mathbf{3154} \pm 383$	2650 ± 330	$\mathbf{2796} \pm 148$
Ant	$\mathbf{6540} \pm 224$	6207 ± 255	5192 ± 284	$\mathbf{5212} \pm 202$
Pusher	-23.69 ± 3.66	$\mathbf{-22.45} \pm 1.73$	-25.45 ± 1.54	$\mathbf{-25.33} \pm 1.28$
Reacher	-3.67 ± 0.21	$\mathbf{-3.66} \pm 0.23$	$\mathbf{-3.73} \pm 0.08$	-3.74 ± 0.10

Table 12: Performance comparison with plasticity baselines on MuJoCo. Results are mean ± standard deviation of returns under the original, uninflated rewards, averaged over the final 300K training steps unless otherwise indicated. Reset (100K) and Reset (10K) use shorter evaluation windows, not different reset intervals. Bold indicates the highest mean in each task.

Method	**HalfCheetah**	**Walker2d**	**Humanoid**	**Swimmer**	**Hopper**	**Ant**	**Pusher**	**Reacher**
0%	15241 ± 734	4399 ± 527	5826 ± 313	116 ± 36	2869 ± 299	4763 ± 1642	-23.28 ± 2.89	$\mathbf{-3.41} \pm 0.23$
2%	15607 ± 372	5102 ± 520	5861 ± 199	136 ± 11	3040 ± 499	4346 ± 1267	-22.07 ± 1.58	-3.42 ± 0.24
Fed (2%)	$\mathbf{15621} \pm 991$	$\mathbf{5395} \pm 452$	$\mathbf{6048} \pm 436$	$\mathbf{143} \pm 13$	3105 ± 293	$\mathbf{5180} \pm 1352$	-21.99 ± 1.23	-3.43 ± 0.24
ReDo	13692 ± 1933	5223 ± 652	5494 ± 247	108 ± 35	$\mathbf{3380} \pm 122$	4986 ± 1231	$\mathbf{-21.58} \pm 0.63$	-3.46 ± 0.24
Reset	8823 ± 257	3379 ± 373	3486 ± 301	71 ± 33	2555 ± 170	1919 ± 1093	-26.61 ± 1.81	-4.19 ± 0.54
(100K)	10915 ± 649	3862 ± 505	4531 ± 462	77 ± 37	2892 ± 238	2397 ± 1553	-24.38 ± 3.76	-3.87 ± 0.58
(10K)	11661 ± 517	3936 ± 1813	5351 ± 694	78 ± 38	3199 ± 748	2746 ± 1618	-23.11 ± 3.79	-3.85 ± 1.13
S&P	10936 ± 431	3958 ± 219	4660 ± 592	99 ± 31	2739 ± 306	2550 ± 1162	-24.43 ± 0.85	-3.52 ± 0.24

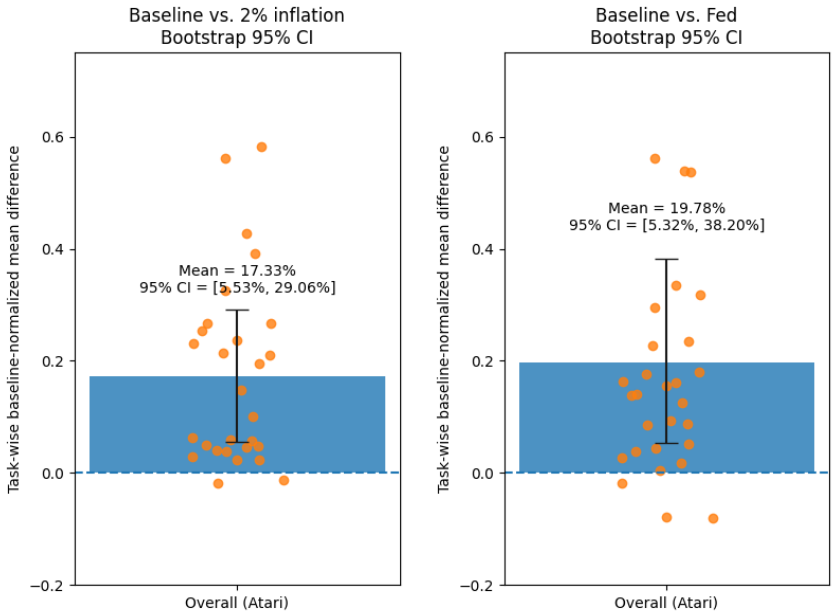


Figure 6: Task-wise bootstrap 95% CI over Atari tasks. (Left) Baseline vs. 2% inflation rate. (Right) Baseline vs. Fed.

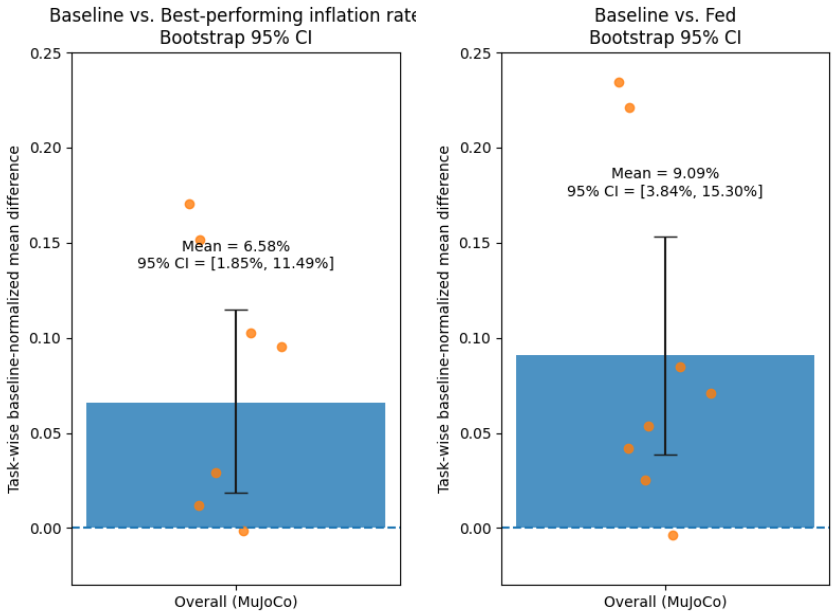


Figure 7: Task-wise bootstrap 95% CI over MuJoCo tasks. (Left) Baseline vs. the best-performing inflation rate identified in Table 1 of the main paper. (Right) Baseline vs. Fed.

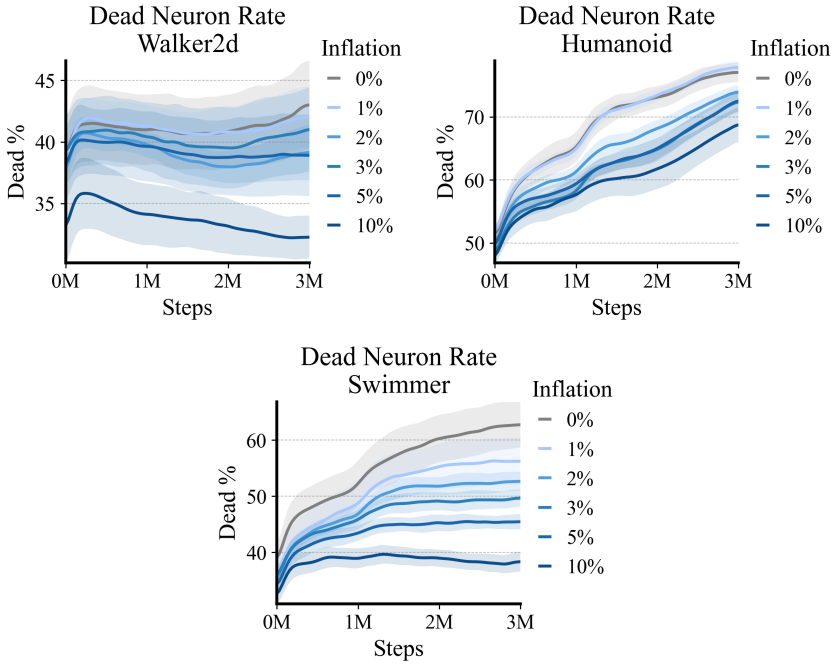


Figure 8: Additional results of dormant (dead) neuron ratios across different tasks under varying reward inflation rates (Fig. 2(a) in the main paper corresponds to HalfCheetah-v4).

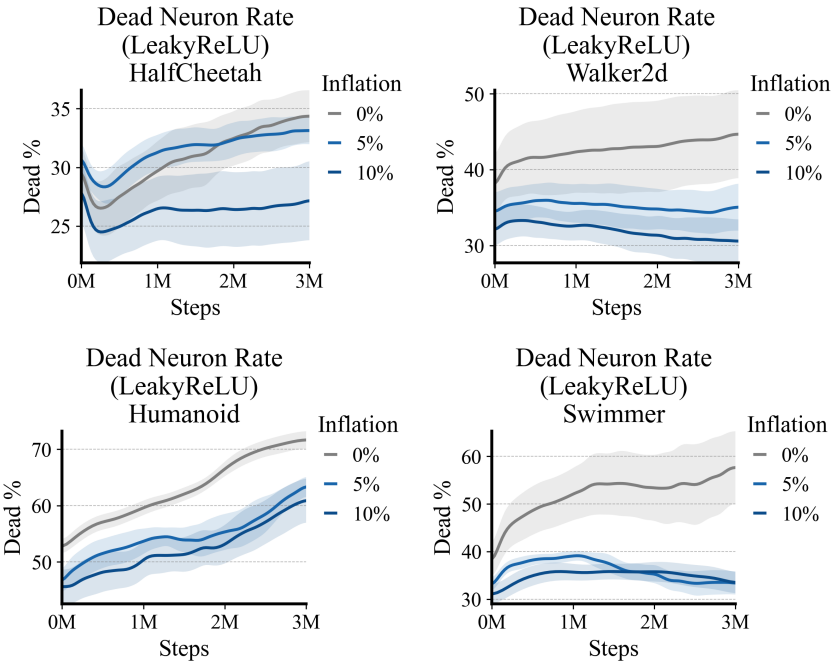


Figure 9: Dormant (dead) neuron ratios across different tasks under varying reward inflation rates, using `LeakyReLU` instead of `ReLU`. Neurons whose negative activation ratio exceeded 99% across inputs were counted as dormant.

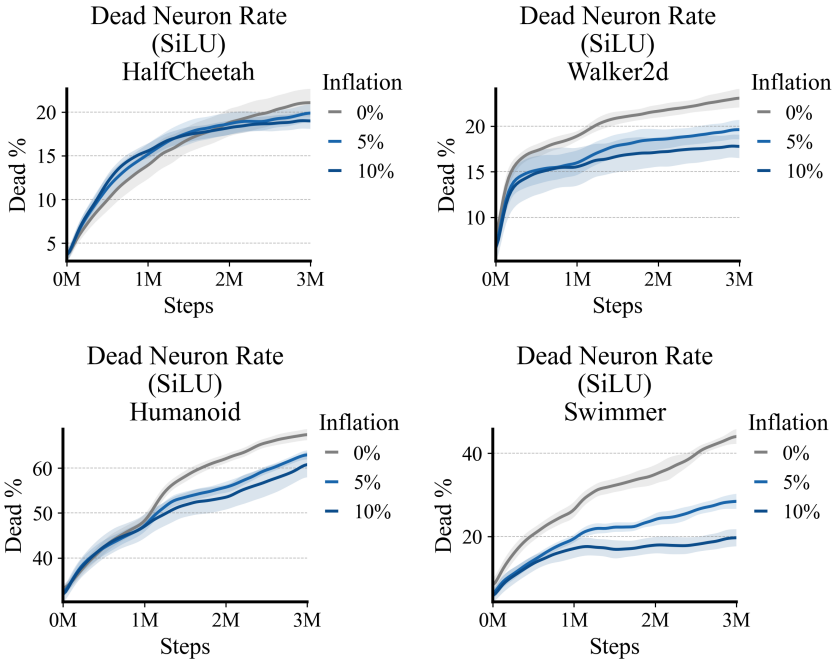


Figure 10: Dormant (dead) neuron ratios across different tasks under varying reward inflation rates, using `SiLU` instead of `ReLU`. Neurons whose activation variance across inputs was below 0.0001 were counted as dormant.

Table 13: Final dormant-neuron ratios (%) on MuJoCo. Plasticity baselines generally suppress dormant neurons more strongly than reward inflation, but this does not consistently translate into better performance (Table 12).

Method	HalfCheetah	Walker2d	Humanoid	Swimmer	Hopper	Ant	Pusher	Reacher
0%	40.12	43.06	77.06	62.83	44.60	53.59	73.17	76.26
2%	38.09	39.20	74.01	52.70	49.83	38.46	72.42	76.17
10%	32.88	32.32	69.13	38.30	43.15	36.89	68.42	78.02
ReDo	12.43	8.81	34.55	27.20	11.77	12.63	31.51	48.21
Reset	18.24	19.60	71.35	51.35	28.45	24.48	58.18	67.52
S&P	17.96	17.68	58.18	51.81	23.62	26.74	54.35	67.99

Table 14: Percentage of runs whose final greedy policy reaches the low-reward goal more often than the high-reward goal during evaluation. Higher exploration uses slower epsilon decay.

Inflation	Low exploration	Higher exploration
0%	18.8%	0.0%
1%	15.6%	0.0%
5%	31.2%	0.0%
10%	28.1%	0.0%
50%	93.8%	59.4%
100%	100.0%	96.9%

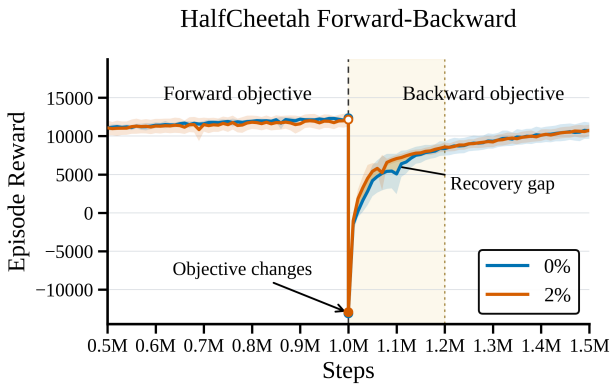


Figure 11: Evaluation episodic returns for SAC with 0% and 2% reward inflation (8 seeds per method). At 1M steps, the objective switches from forward to backward locomotion. Both the final forward-task and initial backward-task evaluations are shown at the switch. The highlighted region marks the early recovery period from 1.0M to 1.2M steps, during which 2% inflation achieves higher mean returns.

B.1 Adaptation after a Task Switch

If the dormant-neuron suppression induced by reward inflation helps preserve plasticity, we would also expect benefits when learning successive tasks. To examine this possibility, we evaluate HalfCheetah agents by switching from forward to backward locomotion after 1M training steps, comparing 0% and 2% inflation across eight runs per condition. As shown in Figure 11, reward inflation leads to faster recovery following the objective change, while the performance of the two conditions becomes similar later in training. The 2% condition achieved a higher post-switch AUC than the 0% baseline (Welch's t-test, $p = 0.044$). These results suggest a potential benefit for adaptation to changing objectives, although they do not establish dormant-neuron suppression as the cause of improved adaptation.

B.2 Fed Hyperparameter Study

We conducted a hyperparameter study to assess the sensitivity of Fed to its hyperparameter choices. Since the global horizon Y mainly aids interpretation, while the effective inflation mechanism depends on the stepwise rate $\tilde{\rho}$, we fix $Y = 10^5$ and vary only the target inflation rate ρ^*_Y. The remaining parameters control the volatility of inflation updates: larger K, λ_1, λ_2, and smaller τ_ρ increase fluctuations, while the relative scale of β_s and β_l affects the variability of gradient-norm measurements driving these updates. Figure 12 summarizes how these hyperparameters influence inflation-rate stability, while Figure 13 illustrates the sensitivity of normalized performance across four MuJoCo tasks when varying five representative hyperparameters and keeping the others fixed at their default settings. Although some settings (e.g., $K = 6, 12$ or $\rho^* = 0.2$) slightly improve performance, the overall variation remains relatively small. This suggests that Fed is reasonably robust to moderate changes in hyperparameter settings.

B.3 Environmental Uncertainty

Since reward inflation continuously changes the reward scale during training, we further investigated whether its benefits persist under additional environmental noise. To examine this, we injected three types of Gaussian noise: (1) observation noise (0.01σ), (2) reward noise (1.0σ), and (3) action noise (0.1σ) into four MuJoCo tasks and compared SAC agents under 0% and 2% inflation. As shown in Figure 14, both agents experienced performance degradation under noisy conditions; however, the 2% inflation agent maintained comparable or slightly better performance than the baseline. These results suggest that the benefits of reward inflation can persist, at least partially, even when environmental noise is compounded with the inherent temporal inconsistency induced by inflation.

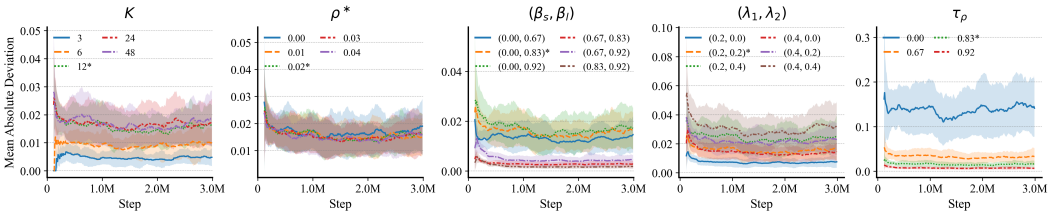


Figure 12: Mean absolute deviation curves of inflation rate under various Fed hyperparameter settings. Specifically, Larger K, λ_1, and λ_2, and smaller τ_ρ lead to more aggressive fluctuations in the inflation rate. For β_s and β_l, smaller values and larger gaps between them amplify variations in gradient-norm measurements, which in turn induce greater inflation-rate volatility. Note that target inflation ρ^* does not impact volatility.

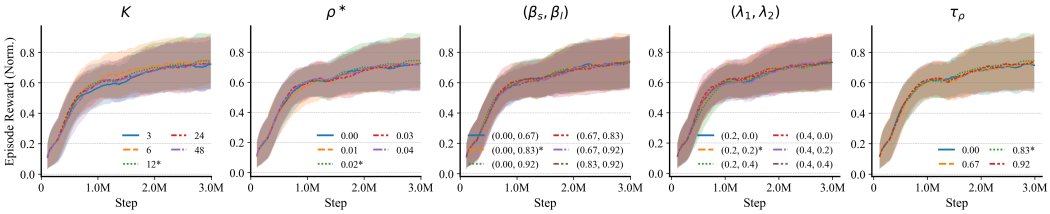


Figure 13: Learning curves of Fed varying 5 sets of hyperparameters. Default values are marked with $*$. Episode rewards are normalized across 4 tasks.

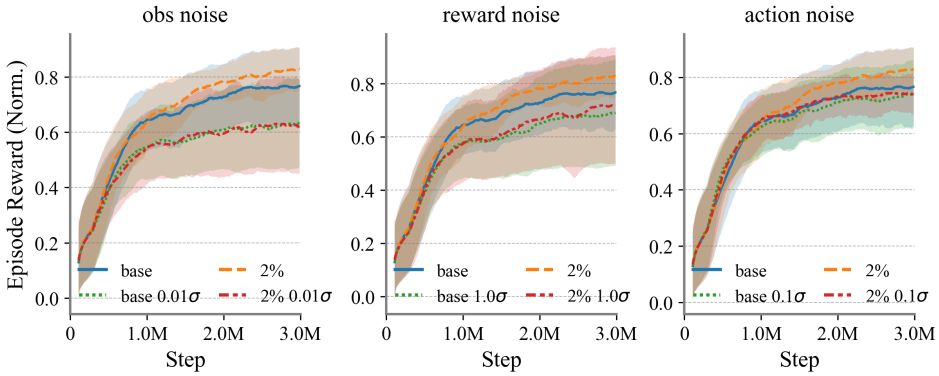


Figure 14: Non-inflation (0%) vs. inflation (2%) on more uncertain conditions.

C Convergence Discussion

Reward inflation introduces two related effects relevant to convergence: the overall reward scale changes during training, and the replay buffer contains transitions collected at different reward scales. We discuss how these effects behave under persistent positive inflation and when inflation stops. We consider discounted tabular Q-learning with bounded original rewards, assuming the standard convergence conditions on state-action coverage, diminishing learning rates, and replay sampling.

Changing reward scales. Let g denote the current training step and $h \leq g$ the collection step of a transition in the replay buffer $\mathcal{D}_g$. Its stored reward is $I(h)r$. Under assumption (A1), the buffer-averaged multiplier

$$\bar{I}_g := \mathbb{E}_{\mathcal{D}_g}[I(h)]$$

defines the approximate reference value function $\bar{I}_g Q^*$ in Eq. (15). Thus, changes in $\bar{I}_g$ move the reference that the learner tracks.

Individual transitions also differ from this common reference scale. For the same transition and a fixed Q-function, the targets using $I(h)r$ and $\bar{I}_g r$ differ by

$$|y_{g,h} - \bar{y}_g| = |I(h) - \bar{I}_g|\,|r| \leq R_{\max}|I(h) - \bar{I}_g|,$$

where $|r| \leq R_{\max}$. This captures the additional target variation arising from heterogeneous reward scales in replay.

Persistent positive inflation. Under a constant positive inflation rate, $I(g)$ grows exponentially. With a finite FIFO replay buffer, $\bar{I}_g$ follows this growth, so the reference $\bar{I}_g Q^*$ generally does not approach a finite limit. Transitions of different ages also retain different reward multipliers, with persistent relative differences and growing absolute differences. Consequently, convergence to a finite Q-function is generally not guaranteed. This does not imply that the greedy policy must diverge, since a common positive scaling preserves action rankings.

Stopping or decaying inflation. If inflation stops at a finite step G, the multiplier remains fixed at $I(G)$. Once earlier transitions leave the replay buffer, all stored rewards use this same multiplier. Both effects described above disappear, and learning reduces to ordinary Q-learning with uniformly scaled rewards. Under the stated convergence conditions, the Q-function converges to $I(G)Q^*$, which induces the same greedy optimal policies as Q^*.

The same limiting reward structure arises if inflation decays sufficiently rapidly that $I(g) \to I_\infty \in (0, \infty)$ and the buffer refreshes. In this case, convergence additionally requires the remaining scale perturbations to satisfy the relevant stochastic-approximation conditions. A summable sequence of nonnegative per-step inflation rates ensures a finite limiting multiplier; merely having the rate approach zero does not. Our fixed-inflation experiments examine finite-horizon learning benefits rather than such a convergence-oriented schedule.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims regarding theoretical derivations and empirical validations are accurately reflected and supported in the main paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss potential risks such as training instability under excessive inflation and the heuristic dependencies of the Fed controller in Section 5.5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide the assumptions, derivations, and supporting analysis for the theoretical results, with additional details in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all essential details for reproduction, including the adaptive control algorithm (Fed) in Section 3, environmental setups in Section 5.1, and a comprehensive hyperparameter table in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the core implementation of our proposed method, specifically the reward inflation mechanism, the Fed controller, and baseline implementations, as well as code for other experiments. Our implementation is based on the CleanRL library, and we provide sufficient instructions to reproduce the core experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all training details, including the Adam optimizer configuration, batch size, and network architectures in Appendix **??**. We also describe the rationale for choosing the target inflation rates and provide a sensitivity analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report results across 8 seeds for ALE, 10 for the main SAC comparisons, 8 for TD3 and the task-switch experiment, and 32 for tabular Q-learning. Tables report mean and standard deviation where applicable; confidence intervals and statistical tests are provided for the main performance comparisons.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide full hardware specifications and the typical execution time for each task in Appendix A.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research conforms to the NeurIPS Code of Ethics. We have ensured that our methodology and experiments do not involve any unethical practices, including data manipulation or harmful applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: As this paper focuses on foundational reinforcement learning methodology, it offers broad potential for positive impact across diverse domains such as robotics, autonomous driving, and industrial optimization by enhancing learning efficiency. Our research aims to improve general algorithmic performance and does not introduce any specific features or components that facilitate misuse beyond the standard advancement of the field.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Not applicable, as this study focuses on foundational reinforcement learning algorithms and does not involve high-risk models (e.g., generative models) or sensitive datasets that require specific safeguards against misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets, including MuJoCo, the ALE, and the CleanRL library, are properly cited in the Section 5.1 and Appendix A, and were used in accordance with their licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The core implementation of reward inflation, the Fed controller, and other baselines, as well as code for other experiments, is provided as a new asset in the supplemental material.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This research does not involve any human subjects, crowdsourcing, or manual data collection involving human labor.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve any human subjects, crowdsourcing, or experiments requiring IRB approval. All experiments were conducted using standard reinforcement learning benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Not applicable, as the core methodology developed in this research (Reward Inflation and the Fed controller) does not involve the use of Large Language Models (LLMs) as an original or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.