Target-Aligned Reinforcement Learning

Leonard S. Pleiss
Technical University of Munich
leonard.pleiss@tum.de

James Harrison
Google DeepMind
jamesharrison@google.com

Maximilian Schiffer
HEC Paris
schiffer@hec.fr

Abstract

Many value-based deep reinforcement learning algorithms rely on target networks—lagged copies of the online network—to stabilize training. While effective, this mechanism introduces a fundamental stability-recency trade-off: slower target updates improve stability but reduce the recency of learning signals, hindering convergence speed. We propose Target-Aligned Reinforcement Learning (TARL), a simple drop-in refinement for existing algorithms that emphasizes transitions for which the target and online network estimates are highly aligned. By focusing updates on well-aligned targets, TARL mitigates the adverse effects of stale target estimates while retaining the stabilizing benefits of target networks. We empirically demonstrate consistent improvements within discrete and continuous control algorithms across various benchmark environments without any hyperparameter tuning, including a 38.18% median peak score gain on Atari-10, while incurring less than a 4% increase in wall-clock time.

1 Introduction

In Reinforcement Learning (RL), agents optimize their policy through repeated interaction with an environment, aiming to maximize expected cumulative return. A central mechanism enabling this process is Temporal Difference (TD) learning [Sutton, 1988], which relies on bootstrapped target estimates—so-called target values. Current state estimates are iteratively updated toward the estimates of future states, thereby allowing the agent to anticipate long-term reward.

However, as shown by Mnih et al. [2015], naïvely using the online network to produce target values can destabilize learning. To address this issue, they introduced a designated target network. A target network is a lagged copy of the online network used exclusively to generate target values. By decoupling the online parameter updates from target value computation, the bootstrapped targets are generated by a comparatively stationary function approximator. This reduces temporal correlations between updates and targets, suppresses self-reinforcing error feedback, and substantially improves training stability and convergence behavior. This innovation contributed to landmark achievements such as human-level performance on the Atari benchmark [Mnih et al., 2015], and remains widely used [Fujimoto et al., 2018, Hessel et al., 2018, Haarnoja et al., 2018, Badia et al., 2020, Schwarzer et al., 2023].

Despite their widespread success, target networks introduce a critical limitation: by construction, their estimates are lagged relative to the online network. The lag can cause target estimates to become stale, reflecting outdated value predictions that no longer align with the current policy or representation [Vincent et al., 2026]. In essence, it creates a dynamic analogous to aiming at a moving target using delayed coordinates: if the value function changes direction, the lagged target steers the update astray.

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

Consequently, improved target stability necessarily comes at the cost of reduced recency, while increasing recency risks reintroducing instability. This phenomenon constitutes the fundamental *stability-recency trade-off*: Ideally, target values should be as up-to-date as possible while preserving the level of stability required for reliable bootstrapped learning.

In this work, we argue that sacrificing recency is not necessary to ensure stability. We propose to mitigate this trade-off by using the unstable but recent online network to validate the stable but partially stale target network. Our approach—Target-Aligned Reinforcement Learning (TARL)—prioritizes updates where online and target estimates agree, enabling the agent to exploit recent information without compromising stability.

Background. Even though individual papers have recently claimed to bypass the need for a target network by proposing alternate update rules [Kim et al., 2019], functional regularization [Piché et al., 2023], decision transformers [Chen et al., 2021] or streaming algorithms [Elsayed et al., 2024], target networks remain the de facto standard in value-based RL. They are a core component in most popular RL algorithms, including Deep Q-Network (DQN), Rainbow, Soft Actor-Critic (SAC) or Deep Deterministic Policy Gradient [Lillicrap et al., 2016, Hessel et al., 2018, Mnih et al., 2015, Haarnoja et al., 2018]. Target networks further remain a cornerstone of state-of-the-art RL algorithms to this day [Badia et al., 2020, Hessel et al., 2018, Haarnoja et al., 2018, Schwarzer et al., 2023].

As such, the stability-recency trade-off remains a fundamental consideration in RL, particularly during hyperparameter tuning. Most algorithms navigate the trade-off through design choices that govern the target network's update mechanism and cadence. There are two major update mechanisms, *hard* and *soft* target network updates [Li, 2023].

Methods like DQN [Mnih et al., 2015] traditionally perform *hard* updates. At some fixed interval—denoted as the target update interval K—the target network is overwritten by the online network. This interval is then tuned to balance stability and recency.

Other algorithms like SAC or Deep Deterministic Policy Gradient usually perform *soft* updates [Lillicrap et al., 2016, Kobayashi and Ilboudo, 2021, Haarnoja et al., 2018, Zhang et al., 2021]. A soft update is a mechanism to incrementally adjust the target network's parameters toward those of the online network. Rather than copying parameters at fixed intervals, the target network is gradually updated at every step. These gradual updates are usually performed via exponential moving or Polyak averages [Polyak and Juditsky, 1992], typically of the form

$$\theta_{\text{target}} \leftarrow \tau\theta_{\text{online}} + (1-\tau)\theta_{\text{target}}, \tag{1}$$

where $\tau \in (0, 1]$ controls the update magnitude. Soft updates reduce the variance and non-stationarity of bootstrapped targets by smoothing parameter changes over time, improving training stability while still allowing target estimates to track the evolving online network.

Regardless of whether frequent soft updates or infrequent hard updates are employed, the challenge remains the same: to identify an optimal balance where stability is maintained while recency is maximized. Current methods therefore essentially treat this trade-off as a zero-sum game, merely adjusting the slider between stability (low τ or high K) and recency (high τ or low K).

In contrast, we approach this challenge from a novel angle. We posit that we do not need to make the target network fresher (and more unstable); we only need to defer updates where the target network's staleness currently leads to outdated update directions. Specifically, we focus on the alignment between the stable estimate—the target network's target value—and the most recent estimate—the online network's target value. We propose to prioritize updates where both target estimates are well-aligned, indicating that our learning target is both stable *and* recent. In doing so, TARL is the first method to address the stability-recency trade-off inherent to target networks on a per-sample basis, shifting from network-level approaches (e.g., tuning update frequency or Polyak coefficients) to individual transitions. Operating at this finer granularity bypasses the conventional trade-off, preserving stability without sacrificing recency.

Contribution. Our contribution is threefold: First, we introduce a novel offline-online target alignment metric that quantifies agreement between the online and target network value estimates. Second, we present TARL, an algorithmic framework that integrates target alignment through oversampling into any standard RL algorithm employing target networks. Finally, we support our claims with empirical studies across various discrete and continuous control benchmarks and multiple algorithms, showing that TARL consistently improves performance.

2 Problem statement

We consider a standard Markov Decision Process (MDP) as usually studied in an RL setting [Sutton and Barto, 2018]. We characterize this MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, p)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a stochastic kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\gamma \in (0, 1)$ is a discount factor, and $p \in \Delta(\mathcal{S})$ denotes a probability mass function describing the distribution of the initial state, $S_1 \sim p$.

We denote by S_t and A_t the random variables representing the state and action at time t, and by $s \in \mathcal{S}$ and $a \in \mathcal{A}$ their respective realizations. At time step t, the system is in state $S_t \in \mathcal{S}$. If an agent takes action $A_t \in \mathcal{A}$, it receives a corresponding reward $r(s, a)$, and the system transitions to the next state $S_{t+1} \sim P(\cdot|s, a)$. We define the random reward at time t as $R_t = r(S_t, A_t)$. The agent selects actions based on a policy $\pi : \mathcal{S} \to \mathcal{A}$ via $A_t = \pi(S_t)$. Let $\mathbb{P}_p^\pi(\cdot) = \text{Prob}(\cdot \mid \pi, S_1 \sim p)$ denote the probability of an event when following a policy π, starting from an initial state $S_1 \sim p$, and let $\mathbb{E}_p^\pi[\cdot]$ denote the corresponding expectation operator. Let n denote the number of transitions within the episode. Let G_t denote the discounted return at time t, with $G_t = \sum_{i=t}^{n} \gamma^{i-t} R_i$. We define the Q-function (or action-value function) for a policy π as

$$Q^\pi(s, a) = \mathbb{E}_p^\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_p^\pi \left[\sum_{i=t}^{n} \gamma^{i-t} R_i \mid S_t = s, A_t = a \right], \tag{2}$$

where action a is taken at time t and π is followed thereafter. The ultimate goal of value-based RL is to learn a policy that maximizes the Q-function, leading to $Q^\star(s, a) = \max_\pi Q^\pi(s, a)$.

The policy is gradually improved by repeatedly interacting with the environment and learning from previously experienced transitions. A transition C_t is a 5-tuple, $C_t = (S_t, A_t, R_t, S_{t+1}, d_t)$, where d_t is a binary episode termination indicator, $d_t = \mathbb{1}_{t=n}$. One popular approach to learn $Q^\star$ is via Watkins' Q-learning [Watkins, 1989, Watkins and Dayan, 1992], where Q-values are gradually updated via

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \eta \cdot \delta_t, \tag{3}$$

with $\eta \in (0, 1]$ being the learning rate and δ_t being the TD error. In the classical tabular formulation, this TD error is defined using the *online target*,

$$Q_{\text{target}}(S_t) = R_t + (1 - d_t) \cdot \gamma \cdot \max_a Q(S_{t+1}, a, \theta), \tag{4}$$

yielding the online TD error $\delta_t = Q_{\text{target}}(S_t) - Q(S_t, A_t, \theta)$.

In deep RL, and specifically in the DQN algorithm [Mnih et al., 2015], stability is improved by introducing a separate *target network* with parameters $\bar{\theta}$. Crucially, $\bar{\theta}$ is a time-lagged copy of the online parameters θ, such that, under hard updates, $\bar{\theta} \approx \theta_{t-k}$ for some delay k. This target network improves training stability: When a single function approximator is used to both estimate action values and define the bootstrap targets, parameter updates induce correlated changes in predictions and targets, which can lead to divergence. The target network mitigates this issue by decoupling target computation from the online updates, yielding a temporally more stationary learning objective.

The *offline target*—i.e., the target value from the target network, as used in standard DQN—is defined as

$$\bar{Q}_{\text{target}}(S_t) = R_t + (1 - d_t) \cdot \gamma \cdot \max_a Q(S_{t+1}, a, \bar{\theta}), \tag{5}$$

which leads to the offline or target TD error $\bar{\delta}_t = \bar{Q}_{\text{target}}(S_t) - Q(S_t, A_t, \theta)$.

3 Target-Aligned Reinforcement Learning

In this section, we present TARL, a framework designed to mitigate the stability-recency trade-off in deep RL. We first build the intuition for target alignment by analyzing the interplay between offline and online value estimates. We then formalize this intuition into a computable alignment score. Finally, we introduce an alignment-based oversampling mechanism that integrates this metric into standard off-policy algorithms to prioritize stable *and* recent updates.

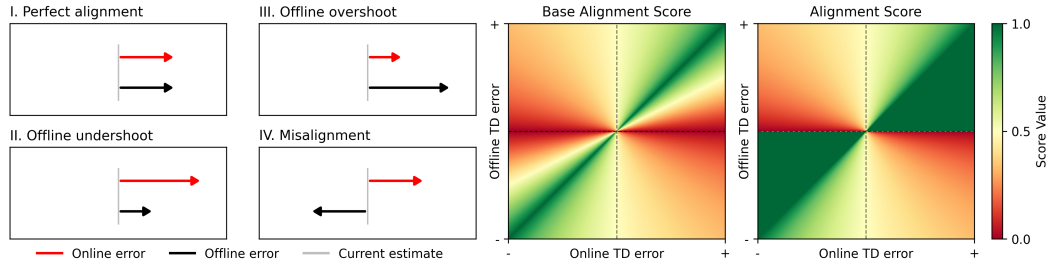


Figure 1: *Left:* Stylized visualization of alignment scenarios. We distinguish between updates fully supported by the online network (I and II) and updates that are only partially or not at all supported by the online network (III and IV). *Right:* Visualization of base alignment and alignment as a function of online and offline temporal difference error.

3.1 Target alignment

In contemporary deep RL, value estimates are updated using target values provided by a lagged offline network. While this lag increases stability, it naturally sacrifices recency. Because the target network lacks the most recent updates, it is prone to staleness bias. TARL targets this staleness bias by quantifying the agreement between the stable (offline) target and the most recent (online) estimate. Specifically, it measures the extent to which the update direction and magnitude proposed by the offline target are *supported* by the online network. As displayed in Figure 1, we identify four distinct alignment scenarios.

I) Perfect Alignment. The offline and online updates are identical. The offline target is perfectly aligned with the most recent estimate from the online network.

II) Offline Undershoot. Both networks agree on direction, but the online network proposes a more extreme update. The conservative offline update is therefore *fully supported* by the online view. We treat this as a safe update: the stable target points in the right direction but is simply more cautious than the volatile online network.

III) Offline Overshoot. Both networks agree on direction, but the offline target proposes a more extreme update than the online network. The offline update is therefore only *partially supported*, indicating some level of risk that the frozen offline target overshoots the ideal update magnitude, introducing the potential need for partial update reversion in the future. In essence, the stale target may push the agent too hard, ignoring recent evidence that the value is actually closer than the offline target suggests.

IV) Misalignment. The networks disagree on update direction. The offline target is not at all supported by the current online estimate.

We hypothesize that, all else equal, updates with higher online support are more beneficial for learning. Specifically, we posit that learning is most effective in scenarios I and II, where the recent online target fully supports the stable offline target. Conversely, scenarios III and IV introduce varying degrees of risk. In the following, we first motivate this hypothesis empirically and then define a metric to formalize the established preference for updates with aligned targets.

3.2 Empirical motivation

We subsequently elaborate why the online network's view matters when the update itself is computed from the offline target. Figure 2 tracks the online network between two synchronizations of DQN on four MINATAR games. In nearly all snapshots (97.8%, SD=2.1%), the online network is closer to the upcoming target network than the current target network is, and on average its distance is about a third smaller. This finding suggests that the online target is usually a better indication of the next offline target than the current offline target itself.

This observation gives the four scenarios above a concrete interpretation in terms of which updates are likely to last. An update in Scenarios I or II moves the estimate toward the online target, and therefore likely also toward the next offline target, so it tends to persist. An update in Scenario IV moves the estimate in the opposite direction and is likely to be undone once the target network is synchronized. Scenario III lies in between, as only the part of the step up to the online target is likely to persist.

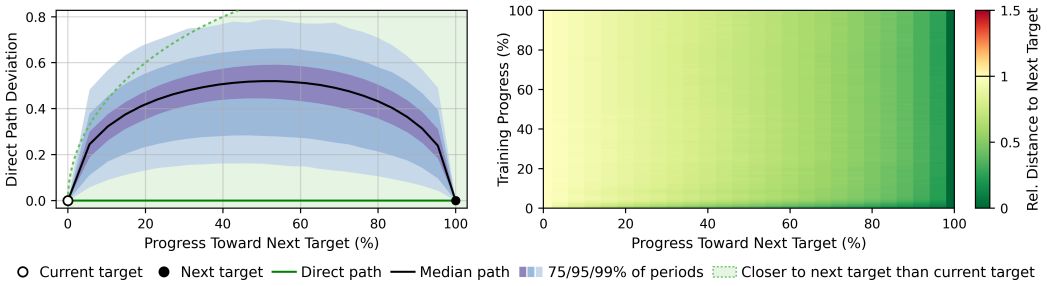


Figure 2: How the online network moves between target-network updates (DQN over four MinAtar games, in parameter-space). *Left:* Median path of the online network between target updates. Shaded bands contain the central 75%, 95% and 99% of deviations. In the green region, the online network is closer to the next target than the current target is. *Right:* the online network's distance to the next target, relative to the current target's distance. The online net is a better predictor of the next target than the current target in 97.8% (SD=2.1%; excluding endpoints) of snapshots, and improves upon the current target by 32.4% (SD=5.2%) on average. Further details are presented in Appendix F.

Note that this reasoning is a heuristic rather than a theoretical guarantee. The online estimate is an imperfect preview, and our motivation considers each transition in isolation, whereas in deep RL an update on one transition also changes the estimates of others.

3.3 Target alignment score

In the following, we formalize the intuition derived in Sections 3.1 and 3.2 into a computable *Target Alignment Score* $\mathcal{A}_t$. This score quantifies, on a per-transition basis, the degree to which the update step proposed by the stable target network is supported by the most recent online estimate. As such, it provides a principled signal for identifying updates that are both stable and up-to-date.

We first define the residual online error δ_t^{next}, which approximates the online error that would remain after applying the update proposed by the offline target under unit step-size, $\delta_t^{\text{next}} = \delta_t - \bar{\delta}_t$. Using this residual, we define the base alignment score $\mathcal{A}_t^{\text{base}} \in [0, 1]$, which measures the extent to which the offline update reduces the magnitude of the current online error,

$$\mathcal{A}_t^{\text{base}} = \frac{|\delta_t|}{|\delta_t| + |\delta_t^{\text{next}}| + \varepsilon}. \tag{6}$$

The denominator $|\delta_t| + |\delta_t^{\text{next}}|$ represents the total variation in error space, and $\varepsilon = 10^{-8}$ is a small constant added for numerical stability. We choose this specific fractional formulation over simpler alternatives—such as unbounded dot products or purely directional cosine similarities—because it yields a bounded $[0, 1]$ score that rigorously penalizes magnitude disagreements independent of the absolute scale of the TD errors. If the residual error $|\delta_t^{\text{next}}|$ is small relative to the initial error $|\delta_t|$, base alignment is high. In the ideal scenario where the offline update fully resolves the online error (i.e., $\delta_t^{\text{next}} \to 0$), the score is maximized (i.e., $\mathcal{A}^{\text{base}} \to 1$). Conversely, in cases of Misalignment (Scenario IV) or severe Offline Overshoot (Scenario III), the proposed update exacerbates the residual error, causing the denominator to grow relative to the numerator and diminishing the score towards zero.

However, strict adherence to this base metric would penalize offline over- and undershoots equally. Yet, as established in Section 3.1, in the case of an Offline Undershoot (Scenario II), the designated target is fully supported by the most recent estimate, while it is only partially supported in the Offline Overshoot case (Scenario III). As such, offline undershoots are inherently safer, whereas offline overshoots pose a risk of inducing parameter oscillation. To account for this asymmetry, we define the final alignment metric $\mathcal{A}_t$,

$$\mathcal{A}_t = \begin{cases} 1 & \text{if } (\delta_t \cdot \bar{\delta}_t > 0) \text{ and } (|\delta_t| \geq |\bar{\delta}_t|) \\ \mathcal{A}_t^{\text{base}} & \text{else.} \end{cases} \tag{7}$$

This metric formally captures our previous visual intuition: The conditional check isolates updates where the direction is identical and the offline magnitude is conservative. Consequently, the score equals one for Perfect Alignment (Scenario I) and Offline Undershoots (Scenario II), while falling back to the strictly penalizing $\mathcal{A}_t^{\text{base}}$ for Offline Overshoots (Scenario III) and Misalignments (Scenario IV).

Note on online target stability. The proposed methodology relies on the alignment between online and offline targets. In the broader literature, online targets are often considered too unstable to yield valuable information. However, this instability arises primarily when the online network is used to provide bootstrapped learning targets, as this creates immediate feedback loops. In TARL, the online network is used only for directional validation. As long as the online network is updated based on stable targets provided by the offline network, its estimates remain sufficiently stable to serve as a directional reference for alignment.

3.4 Alignment-based oversampling

Building on the intuition that well-aligned targets yield more reliable learning signals, we introduce a simple extension to standard RL pipelines that explicitly incorporates target alignment to improve sample efficiency. We propose *alignment-based oversampling*, which serves as the backbone for TARL: Given a target batch size m and an oversampling margin $b \in \mathbb{N}$, we sample $m + b$ transitions from the replay buffer, compute their target alignment $\mathcal{A}_t$, and select the m transitions with the highest alignment for the network update. This mechanism effectively acts as a dynamic noise filter. By temporarily omitting poorly aligned transitions, it defers their updates until subsequent network parameter changes naturally clarify the alignment signal.

Crucially, this building block is a drop-in refinement that effortlessly converts any off-policy RL algorithm employing target networks into its target-aligned version. Algorithm 1 presents an instantiation for Target-Aligned DQN (TA-DQN). Note the boxed lines, which highlight that TARL requires only a localized, minimal modification to the standard training loop. Alternative approaches to integrating alignment, such as prioritized sampling or loss weighting, are discussed in Appendix A.

Implicit stratification. Unlike traditional prioritization schemes, alignment-based selection limits the persistence of its own sampling bias through regular resets: Under a hard update regime, synchronizing the target network instantly resets the target alignment to 1 across the entire replay buffer. This periodic global reset frequently reintroduces temporarily neglected transitions into the active pool with maximum priority, preventing localized biases from accumulating across target periods.

Algorithm 1 Target-Aligned Deep Q-Network (TA-DQN)

Input: Replay buffer $\mathcal{D}$, discount factor γ, batch size m, oversampling margin b, target update interval K, budget T
Initialize online parameters θ, target parameters $\bar{\theta} \leftarrow \theta$, and replay buffer $\mathcal{D} \leftarrow \emptyset$
Sample initial state $S_1 \sim p$
for $t = 1, \ldots, T$ **do**
  Select action A_t using ϵ-greedy policy w.r.t. $Q(S_t, \cdot, \theta)$
  Execute A_t, observe reward R_t and next state S_{t+1}
  Store transition $C_t = (S_t, A_t, R_t, S_{t+1}, d_t)$ in $\mathcal{D}$
  Sample random oversized minibatch $\{M_j\}_{j=1}^{m+b}$ from $\mathcal{D}$
  Compute offline target $\bar{Q}_{\text{target}}(S_j)$, online target $Q_{\text{target}}(S_j)$ and alignment metric $\mathcal{A}_j$ for M_j
  Obtain final minibatch $\mathcal{C} \leftarrow \text{TopK}_m(M, \mathcal{A})$
  Update θ by minimizing loss, $L(\theta) = \frac{1}{m} \sum_{j \in \mathcal{C}} \left(\bar{Q}_{\text{target}}(S_j) - Q(S_j, A_j, \theta) \right)^2$
  if $t \pmod K = 0$ **then**
    Update target network: $\bar{\theta} \leftarrow \theta$
  end if
end for

3.5 Algorithmic intuition

Our hypothesis that—all else equal—updates with higher online support are more beneficial for learning can be understood through three complementary observations, each pointing to the same underlying principle: disagreement between online and offline estimates signals temporally inconsistent targets.

Avoidance of outdated targets. Misalignment or offline overshoots suggest that the proposed offline update is no longer fully supported by the online estimate. By performing updates on those targets, we

effectively chase targets which are no longer aligned with the most recent information. Avoiding such updates prevents the agent from committing to update steps that have been identified as directionally incorrect or excessively large by the most recent value estimates.

Temporal ensembling. The online and offline networks provide complementary estimates of the same value function. While they are correlated, they function as a temporal ensemble. Lower agreement between them may indicate lower temporal consistency. Prioritizing alignment then acts as an inconsistency reduction mechanism.

Reversion of misaligned changes. Since the online network anticipates the next target network (Section 3.2), updates that conflict with the online view are likely to be reverted after the next target network update. As such, Q-value updates based on misaligned targets induce oscillatory behavior and are potentially wasteful.

A bias-variance perspective. Our previous observations allow a unified interpretation through the bias-variance lens. TARL's empirical success admits a compact reading: *a high per-transition alignment score suggests that the staleness bias of the offline target has not yet manifested on a given transition*. Existing target-network schemes navigate this axis *globally* via K or τ, whereas TARL navigates it *per transition*.

We treat each bootstrapped TD residual as a noisy estimate of the expected residual $\delta_t^\star = \mathbb{E}[\delta_t \mid S_t, A_t]$, where the expectation is taken over transition noise and training stochasticity; variance is taken over the same training stochasticity, and bias is measured relative to $\delta_t^\star$, not to $Q^\star$. The offline residual $\bar{\delta}_t$ then has *low variance*—$\bar{\theta}$ is held constant or smoothed over many updates—but carries *bias* growing with its staleness. The online residual δ_t is the converse: continually refreshed, hence low-bias but high-variance. Traditional algorithms must select a static point along this Pareto frontier.

TARL leaves the pointwise estimator unchanged: every selected transition is regressed onto the offline target $\bar{Q}_{\text{target}}$ of the base algorithm, so the target, together with its bias and variance, is identical to that of the base algorithm. What TARL changes is which transitions enter the update. Because the selection event—agreement between δ_t and $\bar{\delta}_t$—depends on the sampled transition itself, the resulting minibatch update is outcome-dependent: its expectation is taken under an alignment-tilted rather than the uniform replay distribution, making it a biased estimate of the base algorithm's expected update. TARL deliberately accepts this selection bias in exchange for reduced staleness bias, by conditioning updates on an event correlated with low staleness bias. Decomposing the disagreement,

$$\delta_t - \bar{\delta}_t \;=\; \underbrace{(\beta_{\delta_t} - \beta_{\bar{\delta}_t})}_{\text{systematic gap}} + \underbrace{(\nu_{\delta_t} - \nu_{\bar{\delta}_t})}_{\text{stochastic gap}}, \tag{8}$$

with β, ν denoting bias and noise components, the small variance of $\bar{\delta}_t$ implies that small disagreement is, up to online noise, evidence that offline bias has not yet diverged on that transition. The same decomposition recovers the asymmetry of $\mathcal{A}_t$ in Eq. (7): Scenario II is the low-bias regime where we accept variance reduction by retaining the conservative offline magnitude; Scenario III flags an offline magnitude likely biased upward by staleness; Scenario IV indicates that the directional component of the bias dominates the stochastic gap entirely.

A caveat is in order. Since $\bar{\theta}$ is a delayed copy of θ, both estimators share a common ancestor and could, in principle, agree by failing in the same direction. However, agreement nonetheless remains a highly informative filter. Since the last target synchronization, the online network has absorbed gradient information from numerous transitions that the offline copy has not seen. While co-failure remains possible in deep non-convex optimization, empirical performance suggests that filtering out *known* disagreements successfully prevents the agent from committing to actively conflicting, oscillatory updates. In practice, target alignment serves as a necessary, if not strictly sufficient, condition for reliable learning steps.

4 Numerical results

We evaluate TARL across both continuous and discrete control domains to assess its generality. Our experimental design is chosen to test the hypothesis that target alignment is a fundamental principle, agnostic to the specific algorithm, target update mechanism, or experience replay mechanism. We test this hypothesis in discrete control environments using DQN with uniform sampling, Prioritized Experience Replay (PER) and Reliability-adjusted Prioritized Experience Replay (ReaPER), as well

as in continuous control environments using SAC. This selection allows us to test performance across substantially different learning dynamics, action modalities, target value formulations, experience replay strategies and target network update regimes. Specifically, we evaluate TARL in three distinct experimental settings:

Discrete control: We evaluate the impact of target alignment in a Double Deep Q-Network (DDQN) using hard target updates on the ATARI-10 benchmark, which approximates the median score across the seminal ATARI-57 benchmark with $R^2 \approx 0.99$ [Aitchison et al., 2023].

Continuous control: We further evaluate the impact of target alignment in SAC using soft target updates on six continuous control environments [Todorov et al., 2012].

Ablations: To better understand the inner workings of TARL, its sensitivity to the oversampling margin and the target update interval, its compatibility with prioritized replay methods as well as the relevance of the transition selection criterion, we conduct additional experiments within four MINATAR environments [Young and Tian, 2019].

Crucially, we performed no hyperparameter tuning for TARL. We applied TARL as a direct drop-in refinement without re-tuning the base hyperparameters of the original algorithms. While we added an ablation on the TARL-specific oversampling hyperparameter b, we did not tune b for the main experiments, but fixed $b = m$. This design choice aims to show the flexibility and robustness of TARL, and highlights its ease of integration into existing pipelines.

Evaluation metric. To evaluate learning efficiency and asymptotic performance simultaneously, our primary evaluation metric is the normalized Area Under the Curve (nAUC). This metric yields a unitless score in $[0, 1]$, where 0 indicates performance no better than a random agent throughout training, and 1 represents achieving peak performance immediately, rewarding both sample-efficient convergence and high asymptotic returns. A formal definition is provided in Appendix E.

For additional details on our experimental setting, we refer to Appendix C. For more in-depth results and nAUC tables for all experiments, we refer to Appendices G and H.

Discrete control. We compared DDQN performance with and without TARL along the ATARI-10 benchmark. As displayed in Figure 3, Target-Aligned DDQN (TA-DDQN) achieves higher peak scores than its vanilla counterpart in eight out of ten games. TA-DDQN achieved a median peak score gain of 38.18% (95% CI across games = [3.64, 98.69]; Mean=51.17% (SD=63.03)). Further, TA-DDQN scores a higher nAUC on all games, raising the mean nAUC by 85.1% $(0.174 \rightarrow 0.323)$. Notably, TA-DDQN achieved this outperformance while incurring negligible increases in wall-clock time: TA-DDQN trained for 4.78 hours (SD=0.15 hours) on average, vanilla DDQN trained for 4.63 hours (SD=0.17 hours). Thus, TARL incurred an overhead in wall-clock time of merely 3.29%.

To control for the confounding effect of additional network evaluations, we evaluated a vanilla DDQN using a doubled batch size. This baseline performs the same number of forward passes as TA-DDQN, but applies backward passes to twice as many transitions. This variant performs similarly to DDQN.

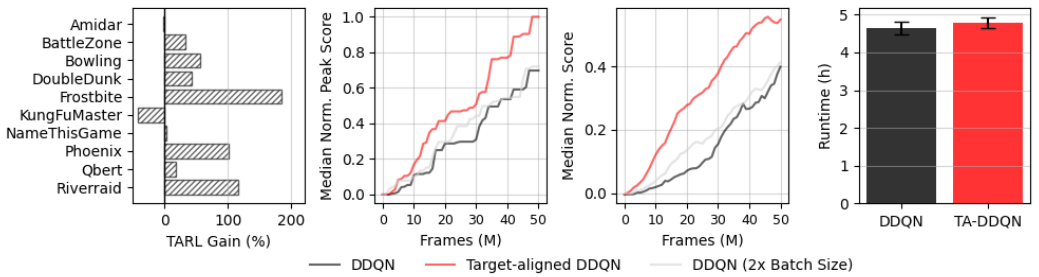


Figure 3: Comparison of DDQN, Target-aligned DDQN (TA-DDQN), and DDQN with doubled batch size across the Atari-10 benchmark. (Left) Per-game peak score improvement of TA-DDQN over DDQN (%). (Center-left) Median normalized cumulative maximum score aggregated over all games as a function of environment frames. (Center-right) Median normalized score over training, reflecting instantaneous rather than best-seen performance. (Right) Mean wall-clock runtime for DDQN and TA-DDQN. Score normalization per game is performed using the range between the random baseline and the best observed performance across all conditions. Score curves are smoothed over 10 evaluations.

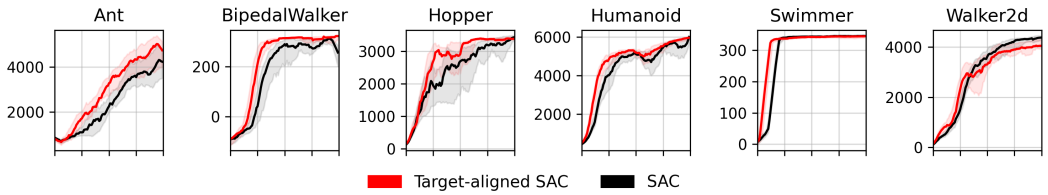


Figure 4: Performance comparison between SAC and target-aligned SAC on six continuous control environments over five seeds. Curves represent median returns, smoothed over 10 evaluations, and shaded areas indicate interquartile ranges.

Continuous control. We compared SAC performance with and without TARL along six different continuous control environments, namely ANT, BIPEDALWALKER, HOPPER, HUMANOID, SWIMMER, and WALKER2D. As in Haarnoja et al. [2018], we report performances across five seeds. As displayed in Figure 4, Target-Aligned SAC (TA-SAC) achieved faster convergence than SAC in five out of six environments, slightly underperforming in WALKER2D (-0.7% nAUC). Nonetheless, across all environments, TA-SAC raises the mean nAUC by 10.0% $(0.640 \rightarrow 0.705)$. Unlike DDQN, where additional online forward passes can be effectively amortized via parallelization, vanilla SAC does not perform online forward passes for the next states. Consequently, the naïve implementation of TA-SAC introduces a 17.82% $(SD = 3.44)$ increase in wall-clock training time. Yet, the performance gains in this regime highlight that even within modern algorithms under soft updates, explicit alignment filtering improves training dynamics.

Ablations. All ablations were conducted across five seeds. Results are displayed in Figure 5. nAUC results are reported in Table 6.

Ablation 1: Experience replay. We compared DQN performance with and without TARL for three different experience replay strategies, namely uniform sampling, PER [Schaul et al., 2016] and ReaPER [Pleiss et al., 2026]. Target-aligned methods consistently outperform their non-aligned counterpart in every game, indicating that TARL acts as a standalone mechanism that complements, rather than substitutes, the replay strategy. Notably, PER and ReaPER alone fall below uniform DQN (-9.4% and -5.4%), but yield substantial gains over uniform DQN when combined with TARL ($+82.7\%$ and $+60.9\%$). Per-game results are displayed in Figure 9.

Ablation 2: Oversampling margin. We further conducted an ablation to evaluate the impact of oversampling margin. We report the normalized scores across the environments, as well as the minimum and mean batch alignment score. Our findings suggest that performance generally increases with higher alignment: mean nAUC rises from 0.343 $(b = 0.1m)$ to 0.399 $(b = 0.5m)$, 0.420 $(b = m)$ and 0.515 $(b = 3m)$, corresponding to gains over DQN of $+17.6$, $+36.8$, $+44.0$ and $+76.4\,\%$, respectively. Naturally, minimum and mean batch alignment increase as well, and are close to saturation at around 0.9 when $b = 3m$. Per-game results are displayed in Figure 10.

Ablation 3: Target update interval. We evaluated the impact of different target update intervals (see Figure 5). We find that TA-DQN outperforms its non-aligned counterpart across all intervals, and

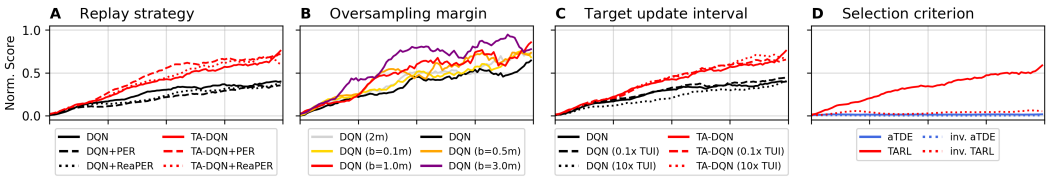


Figure 5: Ablation studies of TA-DQN on the MINATAR benchmark. **A:** Experience replay. DQN with and without TARL under three replay strategies: uniform sampling, PER, and ReaPER. **B:** Standard DQN and DQN (2m) compared against TA-DQN with oversampling margins $b \in \{0.1m, 0.5m, 1.0m, 3.0m\}$, denoted as DQN ($b$=·) in the legend. Shown are the normalized score and the mean and minimum batch alignment after discarding. **C:** DQN and TA-DQN evaluated across different TUIs (default is 1,000). **D:** Selection by alignment score ($\mathcal{A}$) compared with inverted alignment ($-\mathcal{A}$), largest absolute TD error ($|\bar{\delta}_t|$), and smallest absolute TD error ($-|\bar{\delta}_t|$; $b = 1m$). Results are aggregated across four MINATAR environments, with scores min-max normalized per environment before aggregation. Curves show the median over five seeds, smoothed over 10 evaluations (m: batch size; b: oversampling margin; DQN (2m): DQN with doubled batch size; TUI: target update interval; aTDE: $|\bar{\delta}_t|$).

that TARL helps to mitigate the adverse effects of very large update intervals: A tenfold longer target update interval lowered DQN's mean nAUC from 0.292 to 0.226 (-22.6% relative to DQN), while TA-DQN was largely unaffected (0.420 $\rightarrow$ 0.411), suggesting that target alignment increases robustness to infrequent target-network updates. Per-game results are displayed in Figure 11.

Ablation 4: Transition selection criterion. We evaluated the impact of different transition selection criteria. We consider selection by $\mathcal{A}$ (TARL), $-\mathcal{A}$ (inverted TARL), $|\bar{\delta}_t|$ (largest absolute TD error), and $-|\bar{\delta}_t|$ (smallest absolute TD error). We find that selection by $\mathcal{A}$ is the sole criterion that facilitates stable learning ($+44.0\%$ over DQN), whereas all other criteria fall well below the DQN baseline ($-\mathcal{A}$: -71.9%; $|\bar{\delta}_t|$: -73.7%; $-|\bar{\delta}_t|$: -97.0%). Per-game results are displayed in Figure 12.

5 Discussion

Across a diverse set of environments, for multiple algorithms, and under both hard and soft target network update regimes, algorithms employing TARL consistently outperform their counterparts without alignment correction. Importantly, these improvements are achieved without any additional hyperparameter tuning, highlighting the robustness of the proposed approach. Moreover, TARL can be seamlessly integrated into existing RL pipelines, making it a practical drop-in enhancement that consistently improves performance across multiple domains.

While TARL improves performance, it naturally introduces computational overhead by increasing the number of forward passes required during training. However, because forward passes are relatively inexpensive and highly parallelizable in modern RL frameworks, the impact on wall-clock time remains modest given sufficient device memory, particularly in algorithms natively employing online batch forward passes like DDQN (incurring a mere 3.29% increase for $b = m$ in Atari). Thus, to optimally balance computational costs with performance benefits, the oversampling margin b should simply be scaled to device limits: Figure 10 indicates that larger values of b generally yield stronger gains. Consequently, we recommend setting b such that the total query size $m + b$ matches the maximum single-pass batch size supported by the available hardware.

Limitations. While TARL offers considerable empirical benefits, it is inherently restricted to RL algorithms that utilize target networks and does not apply to purely on-policy architectures. Further, while TARL shares structural similarities with ensemble methods, it does not yield independent estimates, nor does it constitute a measure of uncertainty. Rather, it is specifically designed to mitigate the staleness bias caused by lagged target networks (see Appendix B). Additionally, relying on oversampling requires a wider forward pass, inducing some computational overhead. Finally, like any other method that alters the sampling distribution, TARL lacks formal global guarantees against harmful i.i.d. violation. Providing such theoretical guarantees in deep RL currently remains a fundamental, open challenge across the entire field. However, under hard updates, the regular synchronization of the target network routinely clears the priority landscape, counteracting the buildup of systematic bias. As with other established non-uniform sampling methods like PER [Schaul et al., 2016], we rely on our extensive empirical validation, which demonstrates robust performance across diverse environments without observing any instances of degenerate state-space concentration.

Conclusion. We introduced TARL, a method that prioritizes updates on transitions with high target alignment between offline and online targets. Central to our approach is a novel alignment metric that quantifies the consistency between update targets across target network updates. By leveraging this metric as a transition selection criterion, TARL promotes target recency *and* stability, thereby mitigating the stability-recency trade-off induced by target networks. Empirical evaluation across a variety of benchmarks and algorithms validates TARL's effectiveness.

TARL can be seamlessly integrated into existing RL pipelines without requiring additional hyperparameter tuning. Beyond its immediate practical benefits, our work highlights a fundamental insight: the online network—despite being too unstable to provide reliable bootstrap targets—holds highly valuable information for directional update validation. We hope TARL inspires further research into overcoming the structural bottlenecks of value-based RL. Promising future directions include the design of fully differentiable alignment-aware loss functions and the exploration of alignment signals for dynamic representation learning.

Acknowledgments and Disclosure of Funding

L.S.P. was supported by a Roland Berger doctoral fellowship. Roland Berger had no role in the design, execution, analysis, or reporting of this research. The authors declare no competing interests.

References

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, 2023.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the Atari human benchmark. In *International Conference on Machine Learning*, 2020.

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.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: reinforcement learning via sequence modeling. In *International Conference on Neural Information Processing Systems*, 2021.

Mohamed Elsayed, Gautham Vasan, and A. Rupam Mahmood. Deep reinforcement learning without experience replay, target networks, or batch updates. In *NeurIPS Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*, 2024.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: Removing the need for a target network in Deep Q-learning. In *International Joint Conference on Artificial Intelligence*, 2019.

Taisuke Kobayashi and Wendyam Eric Lionel Ilboudo. t-soft update of target network for deep reinforcement learning. *Neural Networks*, 136:63–71, 2021.

Shengbo Eben Li. *Deep Reinforcement Learning*, pages 365–402. Springer Nature Singapore, Singapore, 2023.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Alexandre Piché, Valentin Thomas, Joseph Marino, Rafael Pardinas, Gian Maria Marconi, Christopher Pal, and Mohammad Emtiyaz Khan. Bridging the gap between target networks and functional regularization. *Transactions on Machine Learning Research*, 2023.

Leonard S. Pleiss, Tobias Sutter, and Maximilian Schiffer. Reliability-adjusted prioritized experience replay. In *International Conference on Learning Representations*, 2026.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Antonin Raffin. RL Baselines3 Zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In *International Conference on Machine Learning*, 2023.

Richard Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3: 9–44, 1988.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Jordan Terry, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. In *International Conference on Neural Information Processing Systems*, 2025.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *AAAI Conference on Artificial Intelligence*, 2016.

Théo Vincent, Yogesh Tripathi, Tim Faust, Abdullah Akgül, Yaniv Oren, Melih Kandemir, Jan Peters, and Carlo D'Eramo. Bridging the performance-gap between target-free and target-based reinforcement learning. In *International Conference on Learning Representations*, 2026.

Christopher Watkins. *Learning From Delayed Rewards*. PhD thesis, King's College, University of Cambridge, 1989.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

Kenny Young and Tian Tian. MinAtar: An Atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv:1903.03176*, 2019.

Hongming Zhang, Fengshuo Bai, Chenjun Xiao, Chao Gao, Bo Xu, and Martin Müller. Beta-DQN: Improving Deep Q-learning by evolving the behavior. In *International Conference on Autonomous Agents and Multiagent Systems*, 2025.

Shangtong Zhang, Hengshuai Yao, and Shimon Whiteson. Breaking the deadly triad with a target network. In *International Conference on Machine Learning*, 2021.

A Alternative integration strategies

Several alternatives to oversampling are worth considering to integrate alignment into the learning process:

Threshold-based filtering. A natural alternative to our top-k oversampling scheme is to discard all transitions whose alignment score falls below a fixed threshold and to update on whatever remains. We deliberately opted against this design. A threshold would reintroduce exactly the kind of hyperparameter that TARL is designed to avoid. Further, it would introduce variable batch sizes, which would confound our comparisons: by keeping the number of backward passes identical to the baselines, we ensure that observed gains are attributable to *which* transitions are selected rather than to *how many* updates are performed. Variable batch size would further cause a practical challenge, as it would lead to inconsistent accelerator utilization and memory allocation.

Prioritized sampling. Replay transitions could be sampled based on alignment, favoring well-aligned updates. However, this would require computing alignment for the full buffer, which is computationally prohibitive. Approximate solutions, such as maintaining stored alignment values updated on sampling (analogous to Prioritized Experience Replay [Schaul et al., 2016]), are likely not applicable, as the alignment metric relies on frequently changing online estimates.

Loss weighting. Alignment can be used as a weight within the loss function, upweighting well-aligned samples and downweighting poorly aligned ones. While flexible, this approach requires hyperparameter tuning and may interfere with other loss-weighting schemes (e.g., importance sampling in PER), thereby limiting the method's general applicability.

Loss scaling. Alternatively, alignment may be used to directly scale the loss, effectively modulating the learning rate on a per-sample basis. This is conceptually clean but risks destabilization unless normalization or re-scaling strategies are applied, requiring careful hyperparameter tuning.

We consider oversampling the most robust and conceptually sound approach, as it does not require hyperparameter tuning, is easily integrated into existing pipelines and ensures target alignment while causing limited computational overhead. We therefore deliberately chose to limit this paper's scope to oversampling. Yet, we plan to pursue other directions in future work.

B Distinctions from ensemble methods

TARL shares superficial structural similarities with ensemble methods, as both evaluate multiple network estimates. However, the underlying mechanisms, computational costs, and theoretical drivers of TARL are fundamentally distinct. This section clarifies why TARL's performance gains may not be explained by generic ensemble-based variance reduction and are instead explained by the mitigation of the stability-recency trade-off.

Standard ensemble methods rely on independently initialized and/or trained networks to achieve variance reduction. TARL diverges from this paradigm in several critical ways:

Lack of independent estimators: The target network in TARL is not an independent estimator; it is strictly a lagged clone of the online network. They are identically initialized, and the sole mathematical and informational difference between them is the parameter updates derived since the last synchronization—a delta we define as the *recency gap*.

Asymmetric metric design: TARL does not strictly prioritize for consensus, but instead, TARL employs an asymmetric alignment score to specifically target directional temporal consistency.

Computational efficiency: Ensembles demand considerable overhead, as they require maintaining multiple networks and additional backward passes. TARL exploits existing architectures, requiring only wider forward passes. This results in a significantly reduced computational footprint (less than a 4% increase in wall-clock time on Atari).

Further, because the recency gap is the exclusive structural difference between the online network and the target network, it is the only variable that allows the online network to meaningfully alter the target network's predictions. The gains observed in TARL are therefore explained by recency, not by variance reduction.

The functional distinction between TARL and generic ensembling is further supported by two empirical observations. First, SAC natively employs a twin-critic ensemble to mitigate overestimation bias. If the benefit of TARL stemmed merely from combining multiple value estimates, it would be largely redundant in this setup. Instead, integrating TARL into twin-critic SAC raises the mean nAUC by 10.0% and improves performance in five of six environments, indicating that mitigating the staleness bias operates orthogonally to traditional ensembling. Second, the target update interval ablation (Table 6) directly manipulates the recency gap. A tenfold longer interval lowers the mean nAUC of DQN by 22.6%, whereas TA-DQN loses only 2.1%. Consequently, the advantage of TARL over DQN grows with the interval, from $+40.1\%$ ($0.1\times$) to $+44.0\%$ ($1\times$) and $+82.2\%$ ($10\times$). The gains of TARL are thus largest where staleness is most severe, and TARL removes nearly all of the degradation that staleness inflicts on DQN, a pattern expected from a mechanism that targets the recency gap.

C Experimental design

Hyperparameters for SAC were set to the standards defined in the RL Baselines3 Zoo [Raffin, 2020]. In ANT, HOPPER, SWIMMER, and WALKER2D, we trained for 1 million timesteps per run. In BIPEDALWALKER, we trained for 500,000 steps. In HUMANOID, we trained for 2 million timesteps per run. DQN hyperparameters for MINATAR were set to recommendations published within Zhang et al. [2025]. We trained for 2 million timesteps per run. DDQN hyperparameters for ATARI were set in accordance with seminal work [Mnih et al., 2015, Schaul et al., 2016, van Hasselt et al., 2016]. We trained for 50 million frames (12.5 million timesteps; one seed per game). PER and ReaPER were configured as in the original work [Schaul et al., 2016, Pleiss et al., 2026]. Importance sampling weights are batch-normalized by division with the maximum weight within the (filtered) batch. When using TARL with PER or ReaPER, sampling priorities are updated for the enlarged batch. Oversampling margins are, when required, converted to integers via flooring (e.g., when $m = 32$ and $b = 0.1m$, $b \leftarrow 3$). Atari experiments employ the standard modification steps extensively described in seminal work [Mnih et al., 2015, van Hasselt et al., 2016, Schaul et al., 2016], including frame skips, grayscaling and resizing, reward clipping, no-op starts and terminal-on-life-loss. For DQN and DDQN, we employ the standard network architecture described in Mnih et al. [2015]. SAC employs ReLU activations and the STABLEBASELINES3 [Raffin et al., 2021] standard architecture: two hidden layers of 256 units each (except BipedalWalker, which employs hidden layers of 400 and 300 units).

Table 1: Hyperparameters for SAC on continuous control env.[2]

PARAMETER	VALUE
LEARNING RATE	$3e-4$
BUFFER SIZE	$1e6$
BATCH SIZE	256
LEARNING STARTS	$1e4$
DISCOUNT FACTOR	0.99
SOFT UPDATE COEFF.	0.005
UPDATE PER STEPS	1
# GRADIENT STEPS	1
OPTIMIZER	ADAM

Table 2: Hyperparameters for DQN on MinAtar.

PARAMETER	VALUE
LEARNING RATE	$2.5e-4$
BUFFER SIZE	$1e5$
BATCH SIZE	32
LEARNING STARTS	5,000
DISCOUNT FACTOR	0.99
TARGET UPDATE INT.	1,000
STARTING EXPL.	1.0
EXPL. FRACTION	0.05
FINAL EXPL.	0.01
OPTIMIZER	RMSPROP

Table 3: Hyperparameters for DDQN on Atari.

PARAMETER	VALUE
LEARNING RATE	$2.5e-4$
TOTAL FRAMES	50,000,000
BUFFER SIZE	$1e6$
BATCH SIZE	32
LEARNING STARTS	50,000
DISCOUNT FACTOR	0.99
TARGET UPDATE INT.	30,000
STARTING EXPL.	1.0
EXPL. FRACTION	0.08
FINAL EXPL.	0.01
EVAL EXPL. FRACTION	0.001
TRAIN FREQUENCY	4
GRADIENT STEPS	1
FRAME STACK	4
OPTIMIZER	RMSPROP

Evaluation. For all environments, 100 evaluations were evenly spaced throughout the training procedure. Each agent evaluation consisted of five full trajectories in the environment, going from initial to terminal state. The evaluation score of a single agent evaluation is the average total score across those five evaluation trajectories. SAC employs deterministic action selection during evaluation.

[2] In Swimmer-v4, a discount factor of 0.999 was used. For BipedalWalker-v3, we employ generalized state-dependent exploration, a learning rate of 7.3e-4, a buffer size of 3e5, a discount factor of 0.98, a soft update coefficient of 0.02, and perform 64 gradient steps every 64 environment steps.

D Implementation details

TA-DDQN The implementation of TA-DDQN is identical to TA-DQN, except that it uses the established DDQN offline target, $\bar{Q}_{\text{target}}(S_t) = R_t + (1-d_t)\cdot\gamma\cdot Q\left(S_{t+1}, \arg\max_{a'} Q(S_{t+1}, a'; \theta); \bar{\theta}\right)$ instead of the DQN offline target for alignment calculation and parameter updates.

TA-SAC To adapt the alignment metric for the SAC algorithm, we make specific adjustments to account for both the twin-critic architecture and the maximum entropy objective. For the twin-critic setup, the TD errors and subsequent alignment scores are first computed independently for each of the two Q-networks. To ensure a conservative evaluation of a transition's utility and prevent overestimation, the final alignment score assigned to each transition is the minimum of the alignment scores across the twin critics. The stochastic policy's entropy term is integrated into the metric by incorporating it directly into the TD targets. Specifically, the scaled log-probability of the next action, $\alpha \log \pi(a'|s')$, is subtracted from the next-state Q-value estimates for both the online and offline target calculations. This ensures that the alignment calculation evaluates the critics based on the standard soft Bellman equations, requiring no fundamental structural changes to the alignment formulation itself.

E Normalized area under the curve

To evaluate learning efficiency and asymptotic performance simultaneously, we report the nAUC. For a given environment, let R_t denote the mean evaluation return at step t, and let R_{min} and R_{max} represent the environment-specific random baseline and the maximum return achieved across all compared algorithms, respectively. To ensure the metric focuses on learning progress relative to the baseline and remains robust to "cold start" noise, we calculate the nAUC by clipping normalized scores at zero and averaging over the total number of evaluation checkpoints T,

$$\text{nAUC} = \frac{1}{T}\sum_{t=1}^{T} \max\left(0, \frac{R_t - R_{\text{min}}}{R_{\text{max}} - R_{\text{min}}}\right) \tag{9}$$

This metric provides a within-study relative, unitless score in the range $[0, 1]$, where a score of 0 indicates performance no better than a random agent throughout training, and a score of 1 represents achieving the peak performance observed across all conditions at the very first evaluation checkpoint, and maintaining it consistently ever after. We report nAUC across all experimental conditions as a comparable metric of sample efficiency.

F Experiment on target-to-target trajectories

Figure 2 displays how DQN's online network moves during a target period, the interval between two consecutive target-network updates. We train DQN on the four MINATAR games ASTERIX, BREAKOUT, FREEWAY, and SPACE INVADERS, according to the hyperparameter settings described in Appendix C, with 5 seeds per game.

Within each target update interval, we record the online network every 40 environment steps. This gives 26 snapshots per period, wherein the first is the current target, the last is the next target. Each snapshot stores the full parameter vector of the online network (all weights and biases, flattened). We record $1{,}037{,}400$ individual measurements ($957{,}600$ excluding endpoints) across $39{,}900$ target periods, stratified over training time. For the heatmap, we average the relative distance over all periods within 100 equal bins of training time and over seeds, then over games with equal weight.

Each snapshot is described relative to its period's direct path, using three quantities: *Progress* is the projection of the snapshot's displacement from the current target onto the direct path, divided by the length of the direct path. It is 0 at the current target and 1 at the next target. *Deviation* is the snapshot's distance from the straight line through both targets, divided by the same length. *Relative distance to the next target* describes the snapshot's distance to the next target, divided by the distance between the two targets. Values below 1 mean the online network is closer to the next target than the current target is. Every quantity is relative to the distance between consecutive targets. Periods from different games, stages of training and seeds therefore share one scale and can be pooled.

G Normalized area under the curve tables

In Tables 4–6, gains denote the relative change in nAUC over the respective baseline (DDQN, SAC, and DQN). Per-environment gains compare both methods within a single environment, whereas aggregate gains compare their mean nAUC across environments, i.e., $\left(\overline{\text{nAUC}}_{\text{method}}/\overline{\text{nAUC}}_{\text{baseline}} - 1\right) \cdot 100$. Averaging per-environment relative gains instead yields considerably larger values (e.g., $+152\%$ rather than $+85\%$ on ATARI-10, and $+74\%$ rather than $+44\%$ for TA-DQN on MINATAR). These values are dominated by environments with near-zero baseline nAUC; we therefore report the relative change between mean nAUC values as the more conservative estimate.

Table 4: Normalized area under the curve for Atari-10.

ENVIRONMENT	DDQN	TA-DDQN	GAIN (%)
AMIDAR	0.325	0.414	+27.4
BATTLEZONE	0.088	0.270	+205.8
BOWLING	0.035	0.193	+446.5
DOUBLEDUNK	0.108	0.296	+175.0
FROSTBITE	0.077	0.338	+337.0
KUNGFUMASTER	0.239	0.298	+24.7
NAMETHISGAME	0.341	0.439	+28.7
PHOENIX	0.113	0.255	+126.5
QBERT	0.201	0.361	+79.3
RIVERRAID	0.216	0.363	+67.9
OVERALL AVERAGE	**0.174**	**0.323**	**+85.1**

Table 5: Normalized area under the curve for continuous control environments.

ENVIRONMENT	SAC	TA-SAC	GAIN (%)
ANT-V4	0.452	0.534	+18.2
BIPEDALWALKER-V3	0.657	0.784	+19.2
HOPPER-V4	0.630	0.715	+13.5
HUMANOID-V4	0.660	0.733	+11.2
SWIMMER-V4	0.857	0.879	+2.5
WALKER2D-V4	0.587	0.583	-0.7
OVERALL AVERAGE	**0.640**	**0.705**	**+10.0**

Table 6: Normalized area under the learning curve for all ablations in Figure 5. Evaluation returns are min-max normalized per game across all runs in this table to enable comparability within each column.

Condition	Asterix	Breakout	Freeway	SpaceInv.	Mean	Gain (%)
DQN	0.090	0.368	0.482	0.227	0.292	–
DQN ($2m$)	0.096	0.443	0.530	0.290	0.340	$+16.5$
TA-DQN	0.272	0.474	0.631	0.304	0.420	$+44.0$
(A) Replay strategy						
DQN+PER	0.066	0.401	0.345	0.245	0.264	-9.4
DQN+ReaPER	0.075	0.376	0.411	0.242	0.276	-5.4
TA-DQN+PER	0.506	0.522	0.647	0.457	0.533	$+82.7$
TA-DQN+ReaPER	0.394	0.480	0.630	0.374	0.470	$+60.9$
(B) Oversampling margin						
TA-DQN (b=$0.1m$)	0.108	0.439	0.565	0.260	0.343	$+17.6$
TA-DQN (b=$0.5m$)	0.195	0.451	0.616	0.335	0.399	$+36.8$
TA-DQN (b=$3.0m$)	0.480	0.470	0.727	0.382	0.515	$+76.4$
(C) Target update interval						
DQN (0.1× TUI)	0.089	0.394	0.500	0.238	0.305	$+4.6$
DQN (10× TUI)	0.050	0.281	0.346	0.226	0.226	-22.6
TA-DQN (0.1× TUI)	0.312	0.486	0.603	0.310	0.428	$+46.6$
TA-DQN (10× TUI)	0.303	0.452	0.603	0.287	0.411	$+41.0$
(D) Selection criterion						
aTDE	0.007	0.005	0.276	0.019	0.077	-73.7
inv. aTDE	0.007	0.022	0.002	0.004	0.009	-97.0
inv. TARL	0.013	0.062	0.198	0.055	0.082	-71.9

H Additional figures

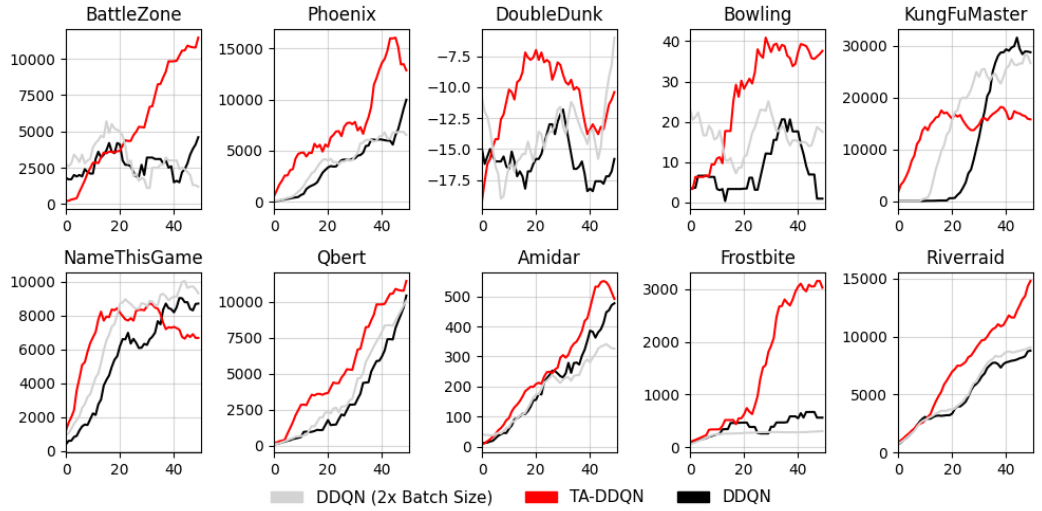


Figure 6: Per-game performance comparison between DDQN, DDQN with doubled batch size and TA-DDQN on the Atari-10 benchmark. Curves represent returns, smoothed over 10 evaluations.

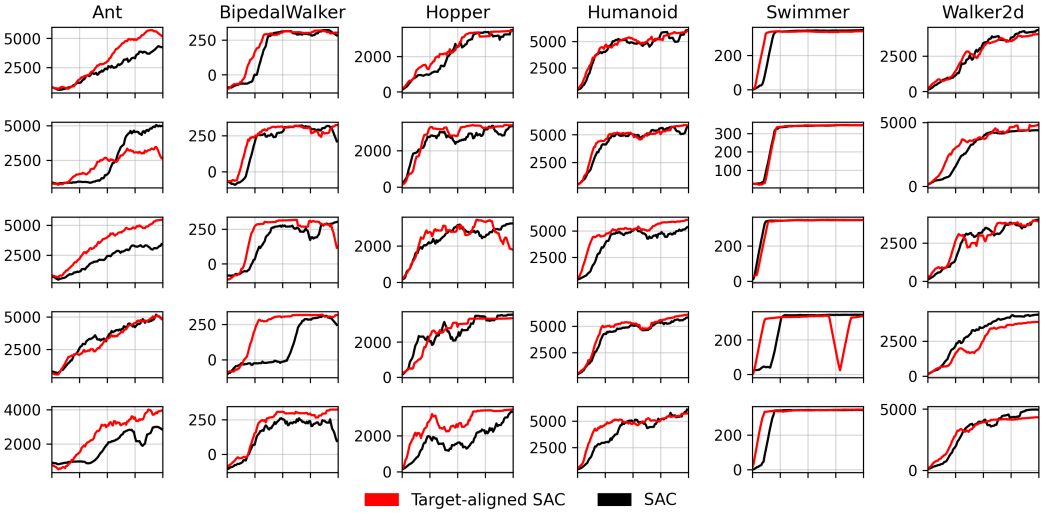


Figure 7: Per-seed performance comparison between SAC and TA-SAC on six continuous control environments. Curves represent returns, smoothed over 10 evaluations.

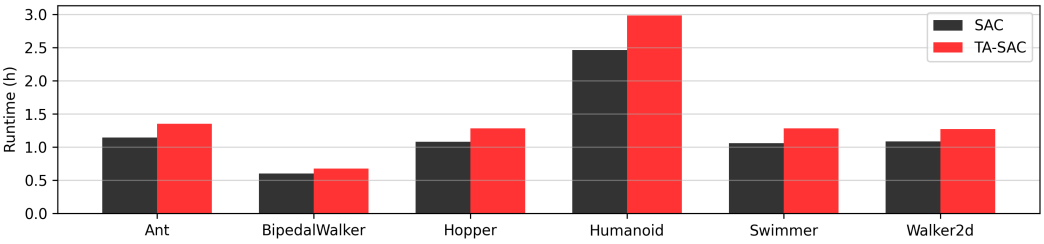


Figure 8: Wall-clock training time for SAC and TA-SAC on six continuous control environments.

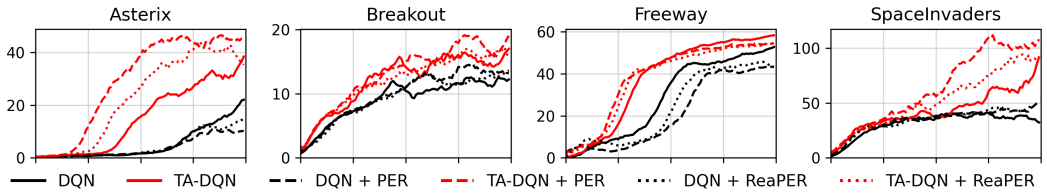


Figure 9: Performance comparison between DQN and target-aligned DQN under different experience replay strategies on four games of the MINATAR benchmark over five seeds. Curves represent median returns, smoothed over 10 evaluations.

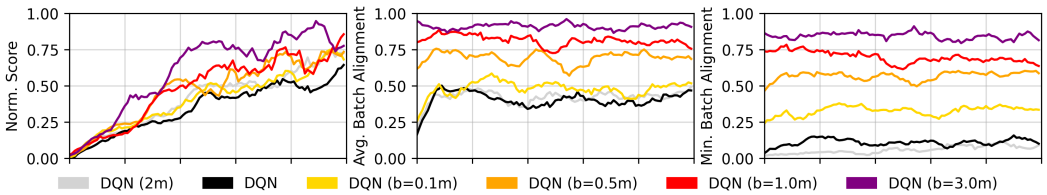


Figure 10: Impact of the oversampling margin on performance and batch alignment. Results are aggregated across four MinAtar environments. The subplots compare standard DQN and a DQN with doubled batch size—denoted as DQN (2m)—against TA-DQN, evaluated at oversampling margins $b \in \{0.1m, 0.5m, 1.0m, 3.0m\}$, denoted as DQN ($b$=$\cdot$) in the legend. All curves are smoothed over 10 evaluations. (Left) Median normalized evaluation scores. Scores were min-max normalized per environment prior to aggregation. (Center) Average batch alignment post-discard. (Right) Minimum batch alignment post-discard.

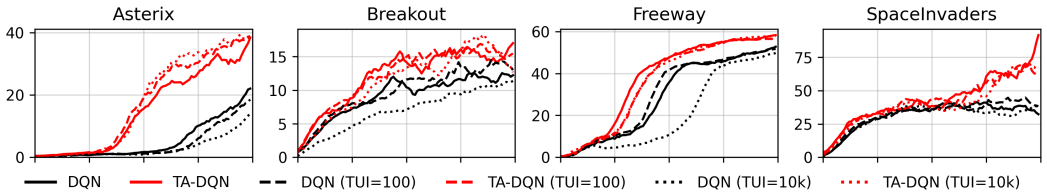


Figure 11: Impact of the target update interval. All curves are smoothed over 10 evaluations (TUI = target update interval).

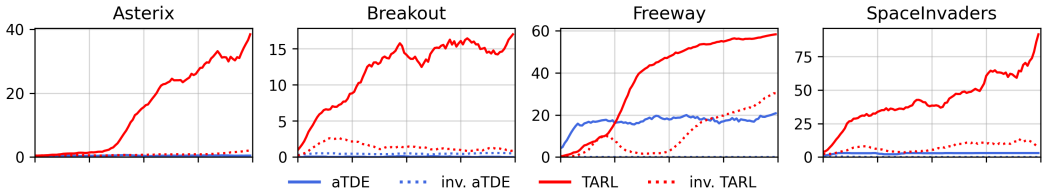


Figure 12: Impact of the chosen metric. All curves are smoothed over 10 evaluations. Batches in each condition were generated by sampling 2m candidates and selecting the top m candidates. Selection metrics were aTDE ($|\bar{\delta}_t|$), inv. aTDE ($-|\bar{\delta}_t|$), TARL ($\mathcal{A}$) and inv. TARL ($-\mathcal{A}$).

I Target alignment and upstream error propagation

We illustrate the effect of TARL on error propagation in a minimal tabular MDP in which all quantities can be computed in closed form. The example is not a general guarantee; it shows how successor-value drift enters the selection criterion and how this changes the propagation of stale target values when the target network is synchronized infrequently.

I.1 The mechanics of the alignment metric

For the online and lagged targets

$$y = r + \gamma \max_{a'} Q(s', a', \theta), \qquad \bar{y} = r + \gamma \max_{a'} Q(s', a', \bar{\theta}), \tag{10}$$

with residuals $\delta = y - Q(s, a)$ and $\bar{\delta} = \bar{y} - Q(s, a)$, the difference between the two evaluates to

$$\delta - \bar{\delta} = \gamma \Big[\max_{a'} Q(s', a', \theta) - \max_{a'} Q(s', a', \bar{\theta}) \Big]. \tag{11}$$

Because the reward and the current-state estimate cancel, the only information that TARL adds to the standard residual $\bar{\delta}$ is the drift of the successor value between the online and target critics; it does not directly measure target bias or uncertainty. The top-m mechanism modulates update frequencies based on this drift without altering the TD targets used for the selected transitions. For terminal transitions, $y = \bar{y} = r$ and hence $\delta = \bar{\delta}$.

In the analysis below, we only use the following property of the alignment score $\mathcal{A}(\delta, \bar{\delta})$: for $\bar{\delta} \neq 0$, the score attains its maximum $\mathcal{A}(\bar{\delta}, \bar{\delta}) = 1$, and $\mathcal{A}(\delta, \bar{\delta}) < 1$ whenever $\delta\bar{\delta} \leq 0$ or $|\delta| < |\bar{\delta}|$. That is, a transition is less aligned when the online target contradicts the step prescribed by the lagged target or supports only part of it.

I.2 Uniform sampling and the stale successor problem

We analyze a minimal two-step chain with a single action:

$$s_0 \xrightarrow{0} s_1 \xrightarrow{R} \perp. \tag{12}$$

We write $u = Q(s_0)$ and $v = Q(s_1)$, whose true values are γR and R, and define the value errors as

$$e_u = u - \gamma R, \qquad e_v = v - R. \tag{13}$$

Let $\bar{u}, \bar{v}$ denote the target-network values, equal to u, v at the start of each interval. Each update is a tabular step with learning rate $\eta \in (0, 1)$ towards the lagged target, and the target network is synchronized every K updates. We write $e_{u,j}$ and $e_{v,j}$ for the errors at the start of the j-th target interval. During a frozen-target interval, if u and v are updated n_u and n_v times, respectively, their updated values become

$$u' = (1 - \eta)^{n_u} u + \big[1 - (1 - \eta)^{n_u}\big] \gamma \bar{v}, \tag{14}$$

$$v' = (1 - \eta)^{n_v} v + \big[1 - (1 - \eta)^{n_v}\big] R. \tag{15}$$

Under uniform sampling, each update selects either transition with probability $1/2$, so that $n_u, n_v \sim \mathrm{Binomial}(K, 1/2)$. Since $\mathbb{E}\big[(1 - \eta)^n\big] = (1 - \eta/2)^K$ for $n \sim \mathrm{Binomial}(K, 1/2)$, the expected errors evolve according to the linear system

$$\mathbb{E} \begin{bmatrix} e_{u,j+1} \\ e_{v,j+1} \end{bmatrix} = \begin{bmatrix} \lambda & \gamma(1 - \lambda) \\ 0 & \lambda \end{bmatrix} \mathbb{E} \begin{bmatrix} e_{u,j} \\ e_{v,j} \end{bmatrix}, \qquad \lambda = (1 - \eta/2)^K. \tag{16}$$

Consequently, the expected errors after j target intervals are

$$\mathbb{E}\, e_{v,j} = \lambda^j e_{v,0}, \tag{17}$$

$$\mathbb{E}\, e_{u,j} = \lambda^j e_{u,0} + j\gamma(1 - \lambda)\lambda^{j-1} e_{v,0}. \tag{18}$$

The second term in (18) shows that, under uniform sampling, a stale successor error is propagated upstream and must be unlearned again after subsequent target synchronizations. In particular, for $e_{u,0} = 0$ and $e_{v,0} = c$, the first interval yields

$$\mathbb{E}\, e_{u,1}^{\text{uniform}} = \gamma c \big[1 - (1 - \eta/2)^K\big], \tag{19}$$

which approaches the full discounted successor error γc as $K \to \infty$.

I.3 Mitigation by target alignment

TARL can mitigate this upstream propagation. Consider again a correct upstream value and an overestimated successor, $e_{u,0} = 0$ and $e_{v,0} = c > 0$. Assume a batch size $m = 1$, a candidate pool that contains both transitions at every update (i.e., $b = m$ with candidates drawn without replacement), and random tie-breaking.

Because s_1 is terminal, $\delta_v = \bar{\delta}_v$, so $\mathcal{A}_v = 1$ as long as its TD error is nonzero. Before the first update of v, we have $v = \bar{v}$ and hence $\delta_u = \bar{\delta}_u$, so both transitions have $\mathcal{A} = 1$ and each update selects either with probability $1/2$. Once v has been updated $n_v \geq 1$ times, its online value has moved towards R while $\bar{v}$ remains frozen, and by (11)

$$\delta_u = \bar{\delta}_u - \gamma c\big[1 - (1-\eta)^{n_v}\big], \qquad \bar{\delta}_u = \gamma c\,(1-\eta)^{n_u} > 0, \tag{20}$$

where n_u is the number of preceding updates of u. The online residual is therefore either smaller than the lagged residual or of opposite sign, so $\mathcal{A}_u < 1 = \mathcal{A}_v$, and v is selected for the remainder of the interval. Letting N denote the number of updates of u in the interval, which all occur before the first update of v, its probability distribution is given by

$$\Pr(N = n) = \begin{cases} 2^{-(n+1)}, & n = 0, \dots, K-1, \\ 2^{-K}, & n = K. \end{cases} \tag{21}$$

At the conclusion of the interval, the errors are

$$e_{v,1} = (1-\eta)^{K-N} c, \qquad e_{u,1} = \gamma c\big[1 - (1-\eta)^{N}\big]. \tag{22}$$

Taking the expectation over N yields

$$\mathbb{E}\, e_{u,1}^{\text{TARL}} = \gamma c\, \frac{\eta}{1+\eta}\Big[1 - \Big(\frac{1-\eta}{2}\Big)^{K}\Big] \;\leq\; \gamma c\, \frac{\eta}{1+\eta}. \tag{23}$$

Compared with (19), the two coincide for $K = 1$, and for $K \geq 2$ TARL propagates strictly less of the stale successor error. As $K \to \infty$, the propagated error approaches γc under uniform sampling but remains bounded by $\gamma c\,\eta/(1+\eta)$ under TARL. The benefit thus grows with the target update interval, which is consistent with the corresponding ablation in Table 6: a tenfold longer interval lowers the mean nAUC of DQN from 0.292 to 0.226, whereas that of TA-DQN changes from 0.420 to 0.411.

Two caveats apply. First, the bound relies on the misaligned transition being excluded deterministically. If candidates are drawn with replacement, u is still updated with probability $1/4$ per step after the first update of v, and the propagated error again approaches γc as $K \to \infty$, although more slowly than under uniform sampling. Hence, the $\eta/(1+\eta)$ bound describes an idealized limit. Second, the same mechanism may also defer useful updates: If u carries a genuine error of its own, a drift of its successor can likewise lower $\mathcal{A}_u$, so that its correction is largely deferred until the next target synchronization, which resets the alignment of all transitions.

J Hardware specification

The ATARI experiments were conducted on a workstation equipped with an AMD Ryzen 9 7950X CPU (16 cores at 4.5 GHz), 128 GB of RAM, and an NVIDIA RTX 4090 GPU with 24 GB of memory (CUDA Toolkit version 12.3). All other numerical experiments were performed on a 2024 MacBook Air with an Apple M3 processor.

K Assets

We use Gymnasium [Towers et al., 2025] (MIT) with MuJoCo [Todorov et al., 2012] (Apache 2.0) and Box2D (zlib) for continuous control, the Arcade Learning Environment [Bellemare et al., 2013] (GPL-2.0) with Atari 2600 ROMs obtained via AutoROM for Atari-10, MinAtar [Young and Tian, 2019] (GPL-3.0) for the ablations, and Stable-Baselines3 [Raffin et al., 2021] and RL Baselines3 Zoo [Raffin, 2020] (both MIT) for the SAC implementation and its hyperparameters.

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: We claim to propose a novel algorithm that incurs consistent empirical improvements across DQN, DDQN and SAC, directly mapping to Algorithm 1, and Figure 3, 4 and 5 in the main body.

   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: The Limitations paragraph in Section 5 discusses the restriction to algorithms with target networks, the distinction from uncertainty estimation, the computational overhead, and the lack of formal guarantees induced by non-uniform sampling.

   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: [N/A]

Justification: The paper states no theorems or formal guarantees. Sections 3.2 and 3.5 provide heuristic motivation and are explicitly framed as such. Appendix I works through a closed-form example on a two-state tabular chain to illustrate how alignment-based selection limits upstream error propagation; it is not presented as a general result, and the assumptions it relies on are stated.

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: The paper exhaustively lists all utilized algorithms, environments and hyperparameters.

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 release our implementations of TA-DQN, TA-DDQN, and TA-SAC as drop-in models compatible with Stable-Baselines3 at `https://github.com/leonardpleiss/TARL`. Together with Stable-Baselines3 and the hyperparameters in Appendix C, they allow reproducing our main experiments.

   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: Experimental details, including all utilized hyperparameters, are described in Section 4 and Appendices C and D.

   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 follow reporting standards from seminal papers in our field. Continuous control (SAC, five seeds): Figure 4 shows median returns with interquartile ranges across seeds, and Figure 7 shows all individual seeds. Atari-10 (DDQN): we ran a single seed per game (Appendix C), so no seed-level error bars are available; instead, we report variability across games, namely a 95% confidence interval, mean, and standard deviation of the per-game peak-score gain (Section 4), and show each game's learning curve (Figure 6). MinAtar ablations (five seeds): we report median curves (Figures 5 and 9) and mean nAUC (Table 6). Wall-clock times are reported as means with standard deviations.

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 information on computer resources in Appendix J, and provide wall-clock time statistics for the main experiments.

   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 paper proposes a simple drop-in refinement for off-policy reinforcement learning algorithms, and therefore does not conflict with the NeurIPS Code of Ethics in any way.

   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: [N/A]

    Justification: This paper proposes a simple methodological refinement for off-policy reinforcement learning algorithms to accelerate convergence. The authors do not anticipate any potential for malicious or unintended use through improvements in convergence speed.

    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: Our work does not include the release of datasets or models.

    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 assets are cited, and their licenses are stated in Appendix K; we use them in accordance with their terms.

    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: Our implementations are released at `https://github.com/leonardpleiss/TARL` under the MIT license, with a README documenting usage and the TARL-specific arguments.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    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.