SPACE-SAMPLED VALUE DECAY: FORGETTING MECHANISMS FOR NON-STATIONARY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning agents deployed on physical systems must adapt continually, since degradation and shifting environment conditions change the dynamics (they *drift*) over time. In the hardest version of this problem, the agent interacts with a single system that might drift at every timestep, leaving no opportunity to revisit past conditions – a setting we call Single Environment, One-Shot Non-Stationary Reinforcement Learning (SEOS-NSRL). We argue that this setting calls for selective forgetting rather than re-learning, and introduce Space-sampled Value Decay (SsVD), which pulls value estimates of randomly chosen elements of the state space to a baseline value, so that outdated information in non visited regions is discarded. SsVD does not require resetting or change-point detection and plugs into modern off-policy algorithms; we integrate it into Soft Actor Critic and Deep Q-Networks. Across 6 non-stationary environments, SsVD improves upon its direct base algorithms and attains the best mean rank across all. The SsVD mechanism can also induce optimism which we show on hard-exploration tasks, although we investigate the connection here only briefly.

1 INTRODUCTION

Humans are inherently capable of adapting to changing environmental conditions. Replicating this behavior in intelligent agents is however challenging even though an ever increasing focus is placed upon the need for such life-long learning capabilities of agents based on experience (Silver & Sutton, 2025). As of today, most modern Reinforcement Learning architectures lack mechanisms for adapting to changes (we refer to such changes as *drift*) in the environment – see Figure 1. We coarsely refer to this problem setting as *Non-Stationary Reinforcement Learning* (NSRL). In particular, we are interested in environmental drift (e.g. of the dynamics) that necessitates a policy adaptation.

Whilst the setting of non-stationary Reinforcement Learning has been acknowledged for a while (Sutton, 1991; Sutton et al., 2007), the limited view of clearly distinct, finite set of *tasks* is very prevalent (see Khetarpal et al., 2022, for an overview). Recently, arguments for a more general view of NSRL with continuous or ever-going changes have been voiced (Silver & Sutton, 2025; Behdin et al., 2026). We take and further refine this setting as *single environment, one-shot* NSRL assuming that the agent can only interact with a single environment which in the worst case changes at every timestep – thus being *one-shot*. This corresponds to the setting when RL agents are a) deployed (Behdin et al., 2026) or b) interact with real-world systems (Dulac-Arnold et al., 2021).

The concept of forgetting mechanisms plays a vital role in the design of machine learning algorithms (such as LSTMs (Hochreiter & Schmidhuber, 1997)) but is also a key aspect of neurology research (Richards & Frankland, 2017; Ryan & Frankland, 2022):

> “In other words, degrading stored information contained in a memory allows organisms to behave more flexibly and promotes better memory-guided decision-making.”
>
> — (Ryan & Frankland, 2022, p. 179)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

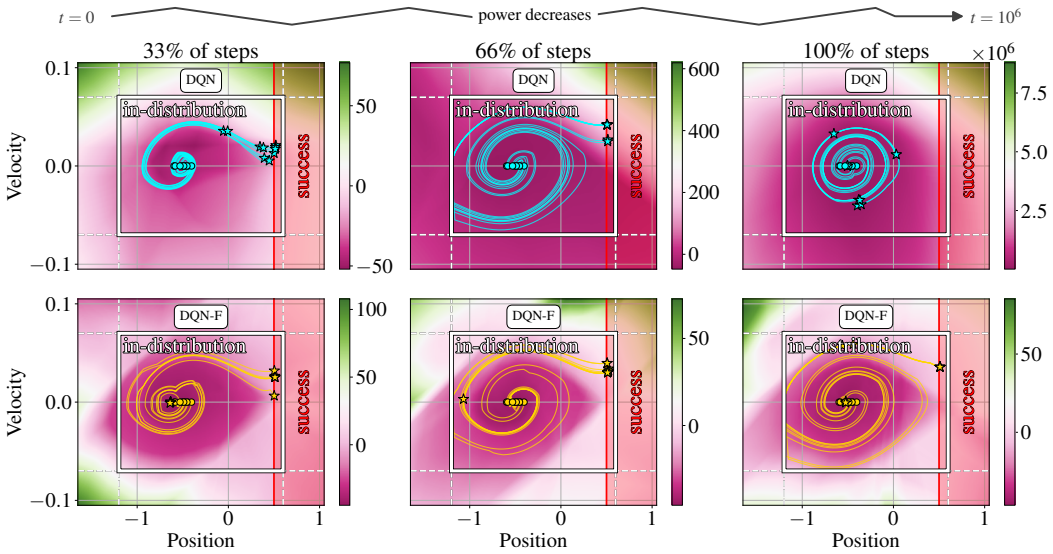


Figure 1: Heatmap snapshots of the learned action-value function ($\max_a Q(s, a)$ over the state-space) by DQN (top, weight decay enabled) and DQN-F (DQN + SsVD, bottom, weight decay disabled, baseline $b = 0$), on a non-stationary MountainCar environment where the engine power decreases across time. Sample trajectories of different rollouts at the same intermediate steps (colored lines), start position (circle marker) and end position (star marker). Total of 1M training steps. A rollout is successful if a position with $x \geq 0.5$ is reached (red shaded region, right side).

This work takes inspiration from "Q-Learning with forgetting" (Ito & Doya, 2009; Beron et al., 2022) designed to model the behaviour of rodents in non-stationary conditions and extend the key concepts to modern Reinforcement Learning architectures. Our contributions are the following:

- We introduce *Space-sampled Value Decay* (SsVD), a bio-inspired forgetting mechanism that samples states from the state space pulling their values toward a baseline b. It adds one auxiliary loss, needs no resets or drift detection, and plugs into any off-policy RL agent.
- Across 6 non-stationary environments, SsVD improves upon its direct base algorithms and attains the best mean rank across all tested methods. We provide qualitative insights about SsVD by performing a visual analysis on `MountainCar`. An optimistic baseline b turns the same mechanism into an exploration bonus which we demonstrate on 3 environments.

2 Problem Setting

Reinforcement Learning Reinforcement Learning assumes an underlying Markov Decision Process (MDP) which is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s, a)$ is the transition dynamics, $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is a discount factor. At each timestep t, an agent observes state $s_t \in \mathcal{S}$, selects action $a_t \in \mathcal{A}$ according to a policy $\pi(a|s)$, and receives reward $r_t = R(s_t, a_t)$. The goal is to find a policy π^* that maximizes the expected discounted return $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

Q-Learning (Watkins & Dayan, 1992) Default Q-Learning assumes a discrete state and action space with the goal of learning an action-value function that provides the expected discounted return of taking action a_t in state s_t from timestep t onwards. The idea is to update the current action-value estimate based on the immediately received reward r_t and a discounted expected future return provided by bootstrapping the action-value function (denoted by Q) as follows:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha[r_t + \gamma \max_{\tilde{a}} Q(s_{t+1}, \tilde{a})]. \tag{1}$$

The policy is then found by taking the actions that maximize the action-value function, usually in combination with a ε-greedy policy for exploration.

Deep Reinforcement Learning A key element of modern architectures is the use of neural network based function approximation to extract relevant features for value estimation. In this work, we focus on Deep Q-Networks (DQN) (Mnih et al., 2015) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as action-value based off-policy agents. Being off-policy allows to reuse collected experience for sample efficiency but causes problems if the agent should adapt to drifting conditions as the environment might behave differently than the stored experience suggests. In contrast to the basic ε-greedy used by DQN, SAC additionally uses entropy guided exploration which includes automatic entropy adjustment and thus should be more robust to the influence of drift.

2.1 Non-stationary Reinforcement Learning and Uncertainty

In the broadest sense, non-stationary Reinforcement Learning is defined as an MDP where the transition dynamics as well as the reward function can change with time t, i.e. $(\mathcal{S}, \mathcal{A}, P_t, R_t, \gamma)$. Note however, that there is large body of related work discussing different possible definitions; a non-exhaustive list includes the case of bandits (Liu et al., 2023), NSRL as a special case of partially observable MDPs (POMDPs) (Khetarpal et al., 2022) or an agent centric perspective (Abel et al., 2023). From a formal perspective the broad definition is sufficient for this work and we in contrast want to focus on practical constraints that we discuss in the following.

In stationary RL settings it is common to scale simulations by using multiple copies of an environment, in some cases this is also transferred to real-world systems (Levine et al., 2017; Kalashnikov et al., 2018). However, most deployed industrial systems cannot be scaled in this way: usually only a single copy of a system is available (Dulac-Arnold et al., 2021) rendering data collection expensive and scaling simulated copies suffers from the sim-to-real gap (Zhao et al., 2020). Further, changes in industrial systems occur throughout their lifetime and can often be attributed to influences such as temperature or wear-and-tear: hence it is highly unrealistic to assume the availability of multiple environments affected by the same form of non-stationarity. Thus, a deployed RL usually interacts with a *single system* at a *single point in time*, hence this work addresses what we term SEOS-NSRL.

Assumption 1 (Single Environment, One-Shot Non-Stationary RL (SEOS-NSRL))**.** *An agent is only able to interact with a* ***single environment****. This environment might drift* ***at every timestep*** *a) without the knowledge of the agent, b) across episodes and c) rendering the current policy suboptimal –* ***requiring*** *policy adaptation.*

Drifting environments are common throughout industry. One example is the use of linear transport systems: such systems are ubiquitous in industrial plants and range from simpler conveyor-like systems to more complex independent cart systems. The latter usually rely on bearings, rollers or similar technology which are slowly affected by wear-and-tear over long periods of time (Cavalaglio Camargo Molano et al., 2020). Further, the wear manifests locally due to the use of multiple independent carts, each again subject to significantly different extent of usage (Jabbar et al., 2025). Therefore, an RL agent interacting with such a system has to expect such variations at all times and separately for each affected component.

In this setting, an RL agent has to face several challenges; two of those are particularly relevant to this work and address the uncertainty the agent has to handle. More precisely, uncertainty can occur in (at least) two different ways:

a) The agent relies **on old** information: deprecated transitions are still present in the replay buffer or indirectly encoded in for example the weights of a neural network.

b) The agent has **no** information: neural networks are not designed for extrapolation, i.e. states that are not visited might take arbitrary values (over- or underestimation).

The former can only occur in NSRL whereas the latter is more general: yet as shown in Figure 1 overestimation seems to be particularly relevant in NSRL settings. The goal of our novel forgetting mechanism is to enforce a baseline value b in states where we do not have any or only old information which is done by sampling possible states independent of the used policy and collected experience – we provide a biological motivation for such an approach next.

2.2 Biological Motivation for Forgetting Mechanisms

In behavioral biology, Q-Learning with forgetting mechanisms emerged (Ito & Doya, 2009; Beron et al., 2022) for simple, usually single state, environments and a set of discrete actions. Whilst slight variations of these mechanisms exist, their idea is to extend eq. (1) by a forgetting mechanism as follows:

$$Q(s_t, a) \leftarrow (1-\eta)Q(s_t, a) + \eta b, \quad \forall a \neq a_t. \tag{2}$$

That is, whilst the taken action is updated, the action values for all other actions decay towards a baseline b (often $b = 0$) with a forgetting rate η as it is assumed that the information is less reliable.

3 Method: Space-sampled Value Decay

Again considering the setting of tabular Q-Learning with discrete actions, we now need to define a distribution $\mathcal{D}_\mathcal{S}$ on the state space $\mathcal{S}$ to allow sampling. Terming this novel approach *Space-sampled Value Decay* (SsVD) we extend eq. (1) by another update which can be written as:

$$Q(s', a) \leftarrow (1-\eta)Q(s', a) + \eta b, \quad s' \sim \mathcal{D}_\mathcal{S}, \quad \forall a \in \mathcal{A}. \tag{3}$$

Thus, we now sample the state randomly from the state-space and decay the values of all possible actions (it is possible to also sample the action which we make use of later for SAC). In many real world applications the observation space is known a priori, for example the workspace configuration of a robot is usually fixed (e.g. acceptable joint angles). Such an approach is not directly transferable to settings where no such assumption can be made, e.g. in many video games there is no direct way to sample states – we discuss this and possible remedies in Section B.4.

3.1 Theoretical Intuition

Whilst the idea of SsVD is straight-forward, this section is dedicated to theoretically analyzing if this novel update behaves as expected from a theoretical viewpoint. Assume the current timestep is t and that a state-action pair (s, a) has not been updated by the classical Q-learning update for τ timesteps. Further, we assume that the change of the optimal action-value function between two consecutive timesteps is bounded by $\Delta \geq 0$:

$$|Q^*_{t+1}(s, a) - Q^*_t(s, a)| \leq \Delta, \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A},\ \forall t \geq 0. \tag{4}$$

Theorem 1. *Let $\eta \in [0, 1]$, fix a state-action pair (s, a) and let t_0 denote the last time step at which $Q(s, a)$ was updated by the Q-learning update. For $t \geq t_0$, let $\tau := t - t_0$ denote the number of time steps since this update, and assume that the SsVD update is applied to (s, a) at every time step $k \in \{t_0 + 1, \dots, t\}$. Then*

$$Q_t(s, a) - Q^*_t(s, a) = (1-\eta)^\tau \big(Q^*_{t_0}(s, a) - Q^*_t(s, a) + \epsilon_{t_0}\big) + \big(1 - (1-\eta)^\tau\big)\big(b - Q^*_t(s, a)\big),$$

where $\epsilon_{t_0} := Q_{t_0}(s, a) - Q^*_{t_0}(s, a)$ is the estimation error at time t_0.

Proof. All proofs are found in Appendix A. □

Theorem 1 now allows us to analyze how we can bound the difference between estimated and optimal action-value function. Taking the norm and applying some simple transformations yields:

Corollary 1. *Let η, (s, a), t_0 and τ be as in Theorem 1. Then*

$$\big|Q_t(s, a) - Q^*_t(s, a)\big| \leq \big(1 - (1-\eta)^\tau\big)\,\big|Q^*_t(s, a) - b\big| + (1-\eta)^\tau\big(|\epsilon_{t_0}| + \Delta\tau\big).$$

Note that we aim for this bound to be as small as possible. Hence, this now becomes an optimization problem with straightforward optima where we search for an optimal value of the forgetting rate η.

Corollary 2. *Let (s, a), t_0 and $\tau \geq 1$ be as in Theorem 1, and let*

$$D := \big|Q^*_t(s, a) - b\big| \quad \textit{and} \quad S := |\epsilon_{t_0}| + \Delta\tau.$$

Then the bound in Corollary 1, viewed as a function of $\eta \in [0, 1]$, is minimized for $\eta = 1$ if $S > D$ and for $\eta = 0$ if $S < D$.

The interpretation is the following: it is optimal to either completely forget ($\eta = 1$) or not forget at all ($\eta = 0$), in practice however, we usually do not know whether $S > D$ or $S < D$ holds (e.g. is the baseline more informative than the estimation error), hence we assume $\eta \in (0, 1)$. In this case, when a state is not visited but regularly sampled using SsVD (i.e. $\lim_{\tau\to\infty}$) we as intended have:

Corollary 3. *Let $\eta \in (0, 1)$ and let (s, a) be a state-action pair that was last updated by the classical Q-learning update at time t_0, i.e. it receives no Q-learning update for all $t > t_0$, while the SsVD update is applied in every timestep. Then, with $\tau = t - t_0$,*

$$\lim_{t\to\infty} Q_t(s, a) = b.$$

i.e. a state not updated by an agent (for example it is not visited) assumes the baseline value. If this baseline value is chosen roughly neutral, e.g. $b \approx 0$[1], this imposes a neutrality bias on states the agent is uncertain. A different mode of operation can be achieved by setting an optimistic prior by choosing b optimistically, e.g. $b \geq V_{\max}$, which drives the agent to explore.

3.2 Extension to Function Approximation

Note that SsVD *does not* rely on the associated rewards of the actions, i.e. it is sufficient to sample states for DQN and actions *and* states for continuous actor-critic models. Adapting the forgetting mechanisms for modern RL algorithms is done by adding an additional auxiliary loss to optimize.

DQN Deep Q-networks rely on fixing the target networks used in the bootstrapping step, we denote the frozen network as Q^- (Mnih et al., 2015). If the default DQN loss is given by $\mathcal{L}_{DQN}$ based on a mini-batch of size m we additionally sample $p \leq m$ states $\hat{s}_1, ..., \hat{s}_p$ and compute

$$\mathcal{L}_F = \frac{1}{p}\sum_{i=1}^{p} \|Q(\hat{s}_i, \cdot) - (1 - \eta)Q^-(\hat{s}_i, \cdot) - \eta\mathbf{b}\|^2 \tag{5}$$

with baseline $\mathbf{b} \in \mathbb{R}^{|\mathcal{A}|}$ written as a vector. Note that we write $Q(\hat{s}_i, \cdot)$ to denote that the action-value in DQN is implemented as $Q : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$ where $Q(s, a)$ is computed by picking the corresponding entry in the resulting function output which is possible as $\mathcal{A}$ is discrete and finite. Thus, the final loss is given by $\mathcal{L} = \mathcal{L}_F + \mathcal{L}_{DQN}$. In practice, we choose a fraction ξ of the original mini-batch size as $p = \max(\lceil \xi m \rceil, 1)$, i.e. the "max" always guarantees that at least a single sample is used.

SAC A key difference to DQN is that the critic network is implemented as $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which together with a separate actor/policy network allows the extension to continuous action spaces (Haarnoja et al., 2018). In contrast to DQN we now sample $p \leq m$ states *and* actions $\hat{s}_1, ..., \hat{s}_p, \hat{a}_1, ..., \hat{a}_p$ to compute the loss as:

$$\mathcal{L}_F = \frac{1}{p}\sum_{i=1}^{p} \left[Q_j(\hat{s}_i, \hat{a}_i) - (1 - \eta)Q_j^-(\hat{s}_i, \hat{a}_i) - \eta b\right]^2 \tag{6}$$

for $j = 1, ..., k$ with Q_j representing one of the k (usually $k = 2$) networks as SAC uses multiple critics (Fujimoto et al., 2018) to avoid overestimation, and b being a scalar baseline value. The number of samples p is also chosen according to $p = \max(\lceil \xi m \rceil, 1)$.

Sampling Sampling complex state spaces (such as images) is generally difficult: in most cases this will require problem specific adaptations. Environments used in this work allow to generate arbitrary environment configurations which we can generate on the fly for sampling – for details see Section B.4. A similar limitation applies to higher dimensional spaces which are difficult to sample due to the curse of dimensionality and if it is unclear which states actually represent realistic environment configurations (e.g. some dimensions might allow theoretically unbounded values).

4 Empirical Experiments

The code used for all experiments and to reproduce the shown results exactly is publicly available on GitHub[2]. Further details can be found in Appendix B.

[1] What counts as *neutral* is environment and scale-dependent.

[2] `https://anonymous.4open.science/r/SpaceSampledValueDecay-3F20`

Table 1: Overview of used environments grouped by type. Stationary environments to evaluate SsVD for exploration are marked in purple and by a † symbol.

	Environment	Non-st. Parameter	Type of Drift	Scheduler	Actions
Classic	CartPole-v1	Force	Decr. $(-)$	Probab.	Discrete
	MountainCar-v0	Force	Decr. $(-)$	Probab.	Discrete
	MountainCarContinuous-v0†	*None*	*None*	*None*	Continuous
MJ	InvertedPendulum-v5	Cart mass	Decr. $(-)$	Probab.	Continuous
Misc.	DeepSea†	*None*	*None*	*None*	Discrete
	RiverSwim†	*None*	*None*	*None*	Discrete
	MultiPadLunarLander (cont./discr.)	Reward	Flip (la. pad)	Probab.	Discrete/Continuous
	ChooseBox	Reward	Flip $(-1\cdot)$	Fixed	Discrete

4.1 Non-stationary Environments

Algorithms We run our proposed SsVD forgetting mechanism on top of the base algorithms DQN and SAC (we denote these extensions by -F, i.e. DQN-F for Deep Q-Networks with SsVD). Further, we include limited (i.e. non-tracking) versions of the base algorithms to illustrate what happens if no further adaptation is done. As an on-policy tracking baseline (Sutton et al., 2007), we include PPO (Schulman et al., 2017) restricted to a single environment instance to strictly adhere to the SEOS-NSRL setting (Assumption 1), alongside off-policy tracking baselines (DQN and SAC). Further, we compare against two general purpose baselines; Nikishin et al. (2022) use recurring resets of the network parameters while maintaining the replay buffer and one with L2 regularization towards the initial parameters of the network (Kumar et al., 2025). All implementations rely on Stable Baselines 3 (Raffin et al., 2021) using the *exact same* (non method-specific) parameters adapted to the SEOS-NSRL setting (i.e. we use a single environment throughout). The used parameters correspond to the pre-tuned values found in RL zoo (Raffin, 2020): further details can be found in Section B.5.

Environments and Drifts We use the classic control environments `CartPole` and `MountainCar` as well as `InvertedPendulum` as a MuJoCo based environments. For these environments, we utilize Keplinger et al. (2025) to define continuous and incremental drifts that modify a parameter of the dynamics (e.g. the power of the car in `MountainCar` as indicated in Figure 1). We additionally introduce two novel non-stationary environments with flipping rewards as drifting dynamics: `MultiPadLunarLander`, an extension of the classic `LunarLander` environment with a second landing pad and `ChooseBox`, a Minigrid (Chevalier-Boisvert et al., 2023) based environment with the goal of picking up one of two boxes, for details see Section B.1. Only `ChooseBox` uses a number of fixed timesteps where the drift occurs – all other drifts are stochastic as the drift is applied with a given probability (these thus vary between seeds) – details on this are found in Section B.2. Table 1 provides a full list of environments used herein.

Evaluation Episode rewards are mean aggregated across timesteps using fixed-width bins of size $B = 5T_{\max}$ where T_{max} is the maximum episode length. For each random seed (we run 10 iterations on each environment), rewards within a bin are averaged, and the mean and standard deviation are then computed across seeds centered at each bin. Number of runs (i.e. seeds) per environment are listed in Table 4. We further apply Gaussian smoothing with $\sigma_{\text{smooth}} = 10$ to both.

Results Figure 2 shows the online return during training; the mean online return is listed in Table 3 (Appendix C). Of the base algorithms, DQN fails to adapt likely because its ε-schedule is not designed for drifting environments. Adding SsVD clearly improves over the respective base algorithm on all six environments (Figure 3, right). The -F variants also attain a higher mean online return than Resetting on all six environments and than L2ToInit on four; L2ToInit is ahead on `MountainCar` and on both `MultiPadLunarLander` variants the differences between all three methods are small. PPO is a strong tracking baseline since, being on-policy, it is not affected by outdated transitions: it attains the highest mean online return on `CartPole`, `MountainCar` and `MultiPadLunarLander (discr.)` and is on par with SAC-F on `MultiPadLunarLander (cont.)`. However, PPO fails to learn `ChooseBox` and its return on `InvertedPendulum` drops sharply late in training. Thus, neither approach dominates. Aggregated over all environments, the -F variants attain the best mean rank (closely followed by PPO, Figure 3, left), although this is not statistically significant (Herbold, 2020).

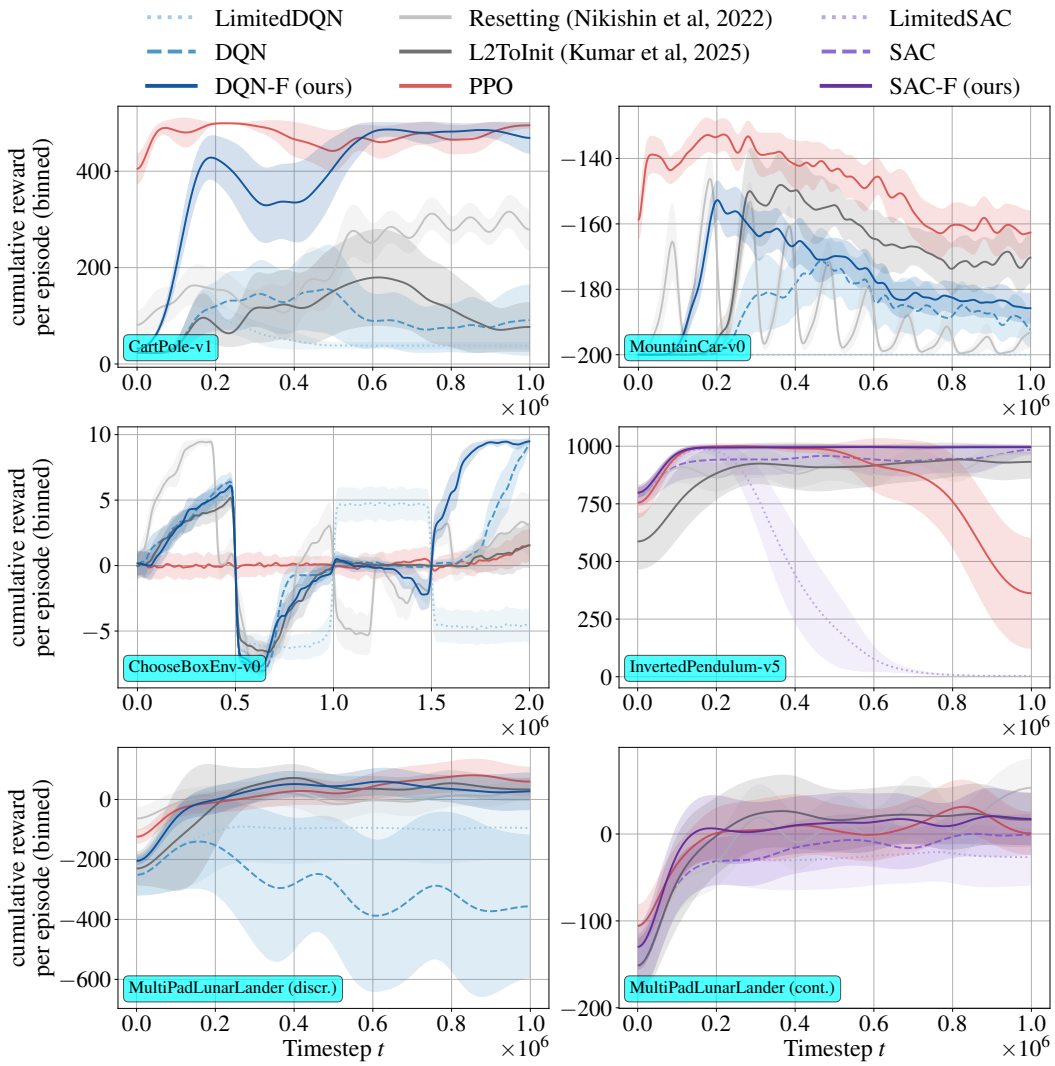


Figure 2: Empirical evaluation on 6 non-stationary environments. Smoothed cumulative returns of terminating episodes mean-aggregated across bins (y-axis, solid lines) with 0.5σ empirical standard deviation (shaded). Timesteps indicates the number of environment interactions (x-axis).

Stationary Ablation We rerun part of the same experiment with stationary environments – a setting where our method also improves on the base algorithms, see Appendix C.

4.2 Qualitative Analysis on MountainCar

For a qualitative, visual analysis we choose the *discrete* non-stationary `MountainCar` environment as its state space is two-dimensional; we visualize the learned value surface ($\max_a Q(s,a)$) by taking snapshots at intermediate timesteps and sampling rollout trajectories. Note that despite the problem's age and apparent simplicity, it has been shown by Huber et al. (2026) that modern algorithms only achieve suboptimal regret on the stationary (albeit continuous) variant of this task.

From Figure 1 it is evident that vanilla DQN starts to drastically overestimate the values associated around the goal state in the top right. Weight decay is enabled here and does not prevent this. As classic Q-learning is known to be overestimating (van Hasselt, 2010) we compare with DoubleDQN (van Hasselt et al., 2016) which is aimed at mitigating overestimation and also an ablation using a bounded activation function instead of the ReLU default. Figure 10 (Appendix C) clearly indicates that using DoubleDQN already reduces but not removes the issue of overestimation – however

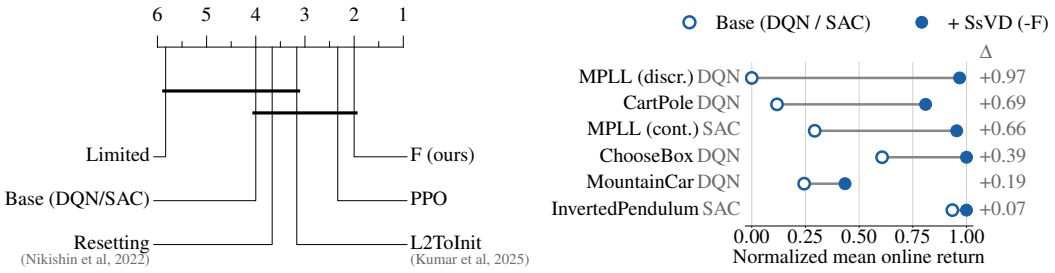


Figure 3: **Left:** Mean ranking (lower is better) of the algorithms within-environment ranks using Autorank (Herbold, 2020). The within-environment rank is computed based on the mean online return. The black-bar marks models where the ranking cannot be guaranteed with statistical significance. **Right:** Comparison of the normalized (across all methods) mean online return (higher is better) between the base variant and its combination with SsVD across all environments.

when using a bounded activation function this issue does not occur. Hence, we believe that the overestimation is caused by the continued bootstrapping over longer-than-usual time horizons.

In DQN-F, the region usually visited (centered diamond-like shape) is well recognizable. Values that are still in-distribution in principle but outside this typical region are pulled towards the $b = 0$ baseline which thus form a white colored *ring* around it – note that $b = 0$ is an *optimistic* baseline for this environment and that weight decay is *not* used. This is Corollary 3 realized empirically as the ring is a mild exploration bonus surrounding the current behavior: as the power decreases and the previously successful swing pattern stops reaching $x > 0.5$, these surrounding states become comparatively attractive and re-exploration follows. This is also evident from sampled policy rollouts: the trajectories lengthen as more swings are required but in nearly all cases reach the goal state.

4.3 Hard Exploration

Setup A different interpretation of SsVD is possible by choosing optimistic values for the baseline b to aid in exploration. To evaluate this additional property we choose a set of three standard benchmark environments for hard exploration tasks (see Table 1, marked) whilst the environments are kept stationary and the baseline value is initialized optimistically. As commonly done with exploration heuristics, we start with a large bonus ($b \propto V_{\max}$) which we linearly decay (either b, ξ or η) over roughly the first half of training, enabling asymptotic policy convergence (see Section B.5).

For `MountainCarContinuous` and `RiverSwim` we perform the same evaluation protocol as before but for `DeepSea` (details on the latter two environments can be found in Section B.3) we scale the problem complexity (i.e. grid size), following Osband et al. (2019) we include BootstrappedDQN with random priors as a baseline.

Discussion Figure 4 clearly shows that SsVD with positive baseline values is beneficial for exploration. Note that the use of generalized State Dependent Exploration (gSDE) (Raffin et al., 2022) is disabled on `MountainCarContinuous` to isolate the effect of using SsVD. On `DeepSea` we can see that our proposed method reliably solves the environment up to $N = 50$ and occasionally up to $N = 70$; outperforming the BootstrappedDQN baseline for $N \geq 30$.

5 Related Work

Planning and Trajectory Sampling Classical approaches rely on sampling states for model-based planning (e.g. trajectory sampling) or exploring starts (Sutton & Barto, 2018) and reward baselines are widely used to reduce variance (Williams, 1992; Baird, 1993; Weaver & Tao, 2001); SsVD however works *independent* of a) visiting a state (though choosing visited states would be possible) and b) of the actual dynamics of the environment as SsVD does not aim to estimate their values.

Non-Stationary / Continual RL A large body of literature focuses on a setting of discrete tasks or sets of change points (Kirkpatrick et al., 2017; Al-Shedivat et al., 2018; Rolnick et al., 2019; Kauvar et al., 2023; Caccia et al., 2023; Ahn et al., 2025), some also assuming intra episode change that

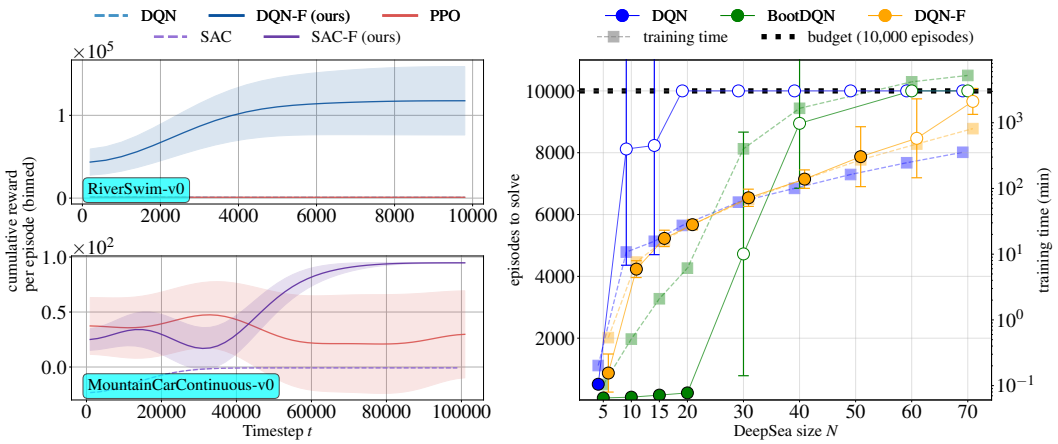


Figure 4: SsVD evaluated on hard exploration tasks. **Left:** SsVD with an optimistic baseline b on 2 stationary exploration environments. Otherwise same evaluation setup as in Figure 2 for 10 runs with $\sigma_{\text{smooth}} = 5$. **Right:** Exploration on `DeepSea` for different grid sizes (x-axis) and the mean number of episodes required to solve (y-axis, left) as well as wall-clock time required (y-axis, right) across 5 seeds (if a seed is unsolved the marker is hollow and counts as the maximum budget).

resets (Keplinger et al., 2025). Many (though not all e.g. Caccia et al., 2023) also assume additional information such as context or task IDs. Such a setting is also referred to as Life-long RL (Abel et al., 2018) – usually an additional goal is to mitigate forgetting of previous knowledge. These assumptions are however unrealistic for real-world applications (Behdin et al., 2026). In contrast, we here take the more realistic viewpoint of SEOS-NSRL (see Assumption 1), requiring that old information has to be forgotten to maintain optimality. To the best of the authors' knowledge, only a few works explicitly address this view: Xie et al. (2021); Liu et al. (2025) both rely on learning latent-variables to infer the current or next state the environment is in; (Liu et al., 2025) also uses change-points detection. Others rely on more general strategies that work independent of the setting such as periodic resets (Nikishin et al., 2022) or modified L2 regularization (Kumar et al., 2025).

Exploration and Optimism When parametrized with an optimistic baseline (see also Corollary 3), SsVD yields a variant of optimistic initialization (Machado et al., 2015; Sutton & Barto, 2018) for function approximation. While visitation-count methods (Bellemare et al., 2016; Sutton & Barto, 2018) and curiosity mechanisms like RND (Burda et al., 2019) compute intrinsic rewards along visited or simulated trajectories, SsVD operates directly on unvisited state representations, driving exploration toward unexplored regions without requiring environment transitions.

6 Conclusion, Limitations, and Future Work

Biologically inspired, we introduced *Space-sampled Value Decay* (SsVD) as a simple, effective forgetting mechanism for deep off-policy RL agents (DQN and SAC). SsVD stabilizes learning in diverse settings: from non-stationary adaptation to mitigating overestimation and driving exploration.

Limitations Non-stationarity in physical systems manifests heterogeneously, and no single adaptation strategy is optimal across all drift profiles without prior knowledge. A primary practical limitation of SsVD is its reliance on sampling valid environment states, which becomes non-trivial in high-dimensional or image-based observation spaces without world models or generative priors. Additionally, while the baseline b can be chosen based on task return scales, SsVD introduces auxiliary hyperparameters (η and ξ) that may require tuning depending on rate and magnitude of drift.

Future Work A natural next step is to explore more adaptive sampling strategies that focus forgetting on states where action-value uncertainty is highest, bypassing uniform sampling in complex state spaces. This encompasses both domain-specific geometric schemes (e.g., Yershova & LaValle, 2004) but also task-agnostic representations. Finally, developing adaptive mechanisms that dynamically tune the forgetting rate η could further be beneficial for autonomously deploying SsVD.

AI USE STATEMENT

We used generative AI tools for finding related and relevant work (deep research) and also as a coding agent to support in the development of software; all generated code and found related work were manually checked for correctness. Further, an extensive manual survey of the literature was performed. The elements of the theory section (i.e. theorems and proofs) were manually drafted and then generative AI was applied to clean and improve their presentation. Generative AI was *not* used for writing or contributing to the main text of the paper except for grammar checking, minor rephrasing and providing high-level feedback on the general structure of the paper. Generative AI helped in writing parts of the text found in the Appendix.

REPRODUCIBILITY STATEMENT

Full code is made available on GitHub (see Section 4). The code not only includes all implementations of benchmarked algorithms including the environments, but also all files used to run the empirical evaluation and to exactly generate the herein shown plots. Further, results underlying the quantitative analysis are provided as ".json" files. A requirements file is provided for installing the used software libraries and to document their used versions.

Used parameters are not only found in the provided code but also listed in Table 4. The two newly introduced environments are described in more detail in Section B.1 and the introduced forms of non-stationarities are also shown visually and discussed in detail in Section B.2.

REFERENCES

David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. Policy and value transfer in lifelong reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 20–29. PMLR, July 2018.

David Abel, Andre Barreto, Benjamin Van Roy, Doina Precup, Hado van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 50377–50407. Curran Associates, Inc., 2023.

Hongjoon Ahn, Jinu Hyeon, Youngmin Oh, Bosun Hwang, and Taesup Moon. Prevalence of negative transfer in continual reinforcement learning: Analyses and a simple baseline. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 75036–75060, 2025.

Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments. In *International Conference on Learning Representations*, February 2018.

Leemon C. Baird, III. Advantage updating. Technical Report WL-TR-93-1146, Wright Laboratory, Wright-Patterson Air Force Base, Dayton, OH, 1993.

Parnian Behdin, Kevin Roice, and Golnaz Mesbahi. Position: Deployed Reinforcement Learning should be Continual. In *Forty-Third International Conference on Machine Learning*, 2026.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Celia C. Beron, Shay Q. Neufeld, Scott W. Linderman, and Bernardo L. Sabatini. Mice exhibit stochastic and efficient action switching during probabilistic decision making. *Proceedings of the National Academy of Sciences*, 119(15):e2113961119, April 2022.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Massimo Caccia, Jonas Mueller, Taesup Kim, Laurent Charlin, and Rasool Fakoor. Task-agnostic continual reinforcement learning: Gaining insights and overcoming challenges. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup (eds.), *Proceedings of the 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pp. 89–119. PMLR, August 2023.

Jacopo Cavalaglio Camargo Molano, Luca Capelli, Riccardo Rubini, Davide Borghi, and Marco Cocconcelli. A bearing fault model for Independent Cart Conveyor System and its validation. *Applied Acoustics*, 159:107069, 2020. ISSN 0003-682X.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Nips '23, New Orleans, LA, USA, 2023. Curran Associates Inc.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, September 2021. ISSN 1573-0565.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, July 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 Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, July 2018.

Steffen Herbold. Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173, 2020.

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

Stefan Huber, Hannes Unger, Georg Schäfer, and Jakob Rehrl. Chebyshev Policies and the Mountain Car Problem: Reinforcement Learning for Low-Dimensional Control Tasks. In *Forty-Third International Conference on Machine Learning*, June 2026.

Makoto Ito and Kenji Doya. Validation of Decision-Making Models and Analysis of Decision Variables in the Rat Basal Ganglia. *The Journal of Neuroscience*, 29(31):9861–9874, August 2009. ISSN 0270-6474, 1529-2401.

Abdul Jabbar, Marco Cocconcelli, Gianluca D'Elia, Davide Borghi, Luca Capelli, Jacopo Cavalaglio Camargo Molano, Matteo Strozzi, and Riccardo Rubini. MOIRA-UNIMORE bearing data set for independent cart systems. *Applied Sciences*, 15(3691), 2025. ISSN 2076-3417.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (eds.), *Proceedings of the 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 651–673. PMLR, October 2018.

Isaac Kauvar, Chris Doyle, Linqi Zhou, and Nick Haber. Curious replay for model-based adaptation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 16018–16048. PMLR, July 2023.

Nathaniel S. Keplinger, Baiting Luo, Yunuo Zhang, Kyle H Wray, Aron Laszka, Abhishek Dubey, and Ayan Mukhopadhyay. NS-gym: A comprehensive and open-source simulation framework for non-stationary markov decision processes. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards Continual Reinforcement Learning: A Review and Perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, December 2022. ISSN 1076-9757.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. In Vincenzo Lomonaco, Stefano Melacci, Tinne Tuytelaars, Sarath Chandar, and Razvan Pascanu (eds.), *Proceedings of the 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 410–430. PMLR, July 2025.

Sergey Levine, Peter Pastor Sampedro, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. In *International Symposium on Experimental Robotics*, 2017.

Yueyang Liu, Xu Kuang, and Benjamin Van Roy. A Definition of Non-Stationary Bandits. In *arXiv:2302.12202*. arXiv, July 2023.

Zihe Liu, Jie Lu, Junyu Xuan, and Guangquan Zhang. Learning latent and changing dynamics in real non-stationary environments. *IEEE Transactions on Knowledge and Data Engineering*, 37 (4):1930–1942, 2025.

Marlos C. Machado, Sriram Srinivasan, and Michael Bowling. Domain-Independent Optimistic Initialization for Reinforcement Learning. In *Learning for General Competency in Video Games: Papers from the 2015 AAAI Workshop*, 2015.

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, February 2015. ISSN 1476-4687.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847. PMLR, July 2022.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020.

Antonin Raffin. 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*, 22(268):1–8, 2021.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1634–1644. PMLR, November 2022.

Blake A. Richards and Paul W. Frankland. The Persistence and Transience of Memory. *Neuron*, 94 (6):1071–1084, June 2017. ISSN 1097-4199.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Tomás J. Ryan and Paul W. Frankland. Forgetting as a form of adaptive engram cell plasticity. *Nature Reviews Neuroscience*, 23(3):173–186, March 2022. ISSN 1471-0048.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

David Silver and Richard S Sutton. Welcome to the Era of Experience. 2025.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008. ISSN 0022-0000.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991. ISSN 0163-5719.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition, 2018.

Richard S. Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th International Conference on Machine Learning*, Icml '07, pp. 871–878, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, J Terry, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025.

Hado van Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), March 2016. ISSN 2374-3468.

Christopher J.C.H. Watkins and Peter Dayan. Technical Note: Q-Learning. *Machine Learning*, 8 (3):279–292, May 1992. ISSN 1573-0565.

Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 538–545, Seattle, Washington, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11393–11403. PMLR, July 2021.

A. Yershova and S.M. LaValle. Deterministic sampling methods for spheres and SO(3). In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pp. 3974–3980 Vol.4, 2004.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744, 2020.

A PROOFS

A.1 PROOF FOR THEOREM 1

Proof. Consider that only the SsVD update Equation (3) is applied for τ consecutive steps starting from the value $Q_{t-\tau}$, where we from now on omit the state-action pair (s, a) for readability. Note that $t_0 = t - \tau$ holds. Unrolling the recursion in Equation (3), the value of Q_t is given by:

$$Q_t = (1-\eta)^\tau Q_{t-\tau} + \eta b \sum_{k=0}^{\tau-1} (1-\eta)^k \tag{7}$$

$$= (1-\eta)^\tau Q_{t-\tau} + \eta b \frac{1-(1-\eta)^\tau}{1-(1-\eta)} \tag{8}$$

$$= (1-\eta)^\tau Q_{t-\tau} + (1-(1-\eta)^\tau) b \tag{9}$$

where we used that $\eta \in (0, 1]$ and the geometric sum formula. For $\eta = 0$, Equation (9) holds trivially since then $Q_t = Q_{t-\tau}$. Further, note that we can denote Q_t^* as a convex combination:

$$Q_t^* = (1-\eta)^\tau Q_t^* + (1-(1-\eta)^\tau) Q_t^*. \tag{10}$$

The goal is now to get an expression for the difference of the action-value function to the optimal action-value function at timestep t if it was not updated for τ steps by the classic Q-learning rule. Denote the estimation error at time $t - \tau$ as $\epsilon_{t-\tau} = Q_{t-\tau} - Q_{t-\tau}^*$. Then the difference is given by:

$$\begin{aligned} Q_t - Q_t^* &= (1-\eta)^\tau Q_{t-\tau} + (1-(1-\eta)^\tau) b - Q_t^* \quad \text{(using Equation (9))} \\ &= (1-\eta)^\tau (Q_{t-\tau}^* + \epsilon_{t-\tau}) + (1-(1-\eta)^\tau) b - Q_t^* \\ &= (1-\eta)^\tau (Q_{t-\tau}^* + \epsilon_{t-\tau}) + (1-(1-\eta)^\tau) b \\ &\quad - (1-\eta)^\tau Q_t^* - (1-(1-\eta)^\tau) Q_t^* \quad \text{(using Equation (10))} \\ &= (1-\eta)^\tau (Q_{t-\tau}^* - Q_t^* + \epsilon_{t-\tau}) + (1-(1-\eta)^\tau)(b - Q_t^*) \end{aligned} \tag{11}$$

which was to be shown. □

A.2 PROOF FOR COROLLARY 1

Proof. As before, we omit the state-action pair (s, a) for readability. Taking absolute values in Equation (11) and applying the triangle inequality, using that $(1-\eta)^\tau \geq 0$ and $1-(1-\eta)^\tau \geq 0$, yields

$$|Q_t - Q_t^*| \leq (1-\eta)^\tau \big(|Q_{t-\tau}^* - Q_t^*| + |\epsilon_{t-\tau}|\big) + (1-(1-\eta)^\tau)\,|b - Q_t^*|.$$

By Equation (4) applied to each term using again the triangle inequality this yields

$$|Q_t - Q_t^*| \leq (1-(1-\eta)^\tau)\,|Q_t^* - b| + (1-\eta)^\tau \big(|\epsilon_{t-\tau}| + \Delta\tau\big),$$

which was to be shown. □

A.3 PROOF FOR COROLLARY 2

Proof. The bound in Corollary 1 can be written as

$$(1-(1-\eta)^\tau) D + (1-\eta)^\tau S = D + (1-\eta)^\tau (S - D).$$

For $\tau \geq 1$, $(1-\eta)^\tau$ is strictly decreasing in η on $[0, 1]$, with value 1 at $\eta = 0$ and 0 at $\eta = 1$. Hence, if $S > D$, the bound is minimized for $\eta = 1$, and if $S < D$, it is minimized for $\eta = 0$. If $S = D$, the bound is independent of η. □

A.4 PROOF FOR COROLLARY 3

Proof. As before, we omit the state-action pair (s, a) for readability. Rearranging Equation (9) gives

$$Q_t - b = (1-\eta)^\tau \big(Q_{t-\tau} - b\big) = (1-\eta)^\tau \big(Q_{t_0} - b\big).$$

Since no Q-learning update occurs after t_0, we have $\tau = t - t_0 \to \infty$ as $t \to \infty$. As $0 < 1 - \eta < 1$ and Q_{t_0} is finite, the right-hand side converges to 0, which was to be shown.

The same holds if the SsVD update is only applied to (s, a) independently with some probability $p > 0$ in each timestep, in which case the limit holds almost surely. □

B Additional Experimental Details

B.1 Custom Non-Stationary Environments

We introduce two novel non-stationary environments, both are shown in Figure 5 together with an example of how these environment drift.

MultiPadLunarLander extends the established `LunarLander` environment (Towers et al., 2025) with a larger area and a second landing pad. The target pad switches with a given probability. The agent receives a positive bonus for landing safely on the target pad, a penalty for crashing, and neither if it hovers until the episode terminates. Since we modify only the position of the target pad, the default shaping rewards of the original environment are retained on top of these terminal rewards. Similar to the original, the observation space remains 8 dimensional and the action space can either be discrete (4 actions) or continuous (2 dimensions for the booster strengths of the side and main engines) – we benchmark both variants.

ChooseBox is a custom MiniGrid environment (Chevalier-Boisvert et al., 2023) in which the agent must pick up one of two boxes. At predefined timesteps, the target box switches, so the agent has to learn new behavior. Picking up a box yields a reward of $a \cdot \left(10 - 9\,\frac{t}{T}\right)$, where t is the current step, T is the maximum episode length, and $a = 1$ for the correct box and $a = -1$ for the incorrect one. The reward therefore decays linearly from 10 to 1 over the course of an episode, which favors fast solutions. Truncated episodes yield a reward of zero.

Neither environment signals the change to the agent. In `ChooseBox`, the boxes swap roles without changing color, so the image-like observations (see Section B.4) look exactly the same before and after a switch. In `MultiPadLunarLander`, the target pad does change color, but color is not part of the observation. However, because the shaping rewards depend on the target position, the agent is informed of a switch fairly quickly through the reward signal.

B.2 Forms of Non-stationarity

For the newly introduced custom environments see Section B.1. For the other environments, the implementation of changing parameters is done with NS-Gym (Keplinger et al., 2025) which provides tools to *schedule* drifts (when and how often these occur) and to define the performed *update* when a scheduler fires. In all cases we use induced changes that retain across episodes. We utilize a probabilistic scheduler and incremental updates which were customized with upper/lower bounds. In other words, at every timestep with a given probability an incremental modification of the dynamics (see Table 1 for which exact parameters are modified) is applied where due to the probabilistic nature we include bounds on the maximum allowed change (e.g. decreasing gravity should not become zero). A handful of different drifting trajectories per environment are shown in Figure 6 with the full details on the parametrizations of these listed in Table 2.

For `MultiPadLunarLander` the chosen drift probability yields an expected number of 8 flips across the duration of 10^6 timesteps: note that this exact number is different for each run, with the

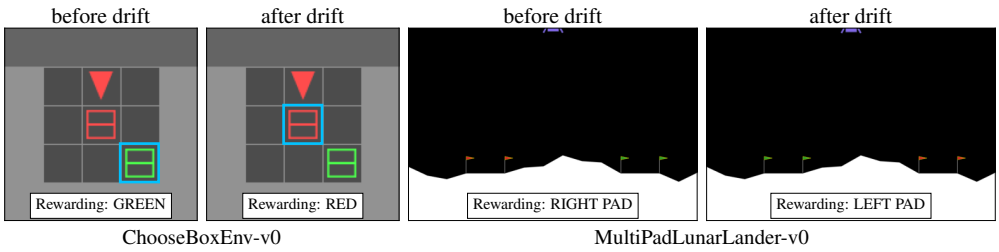


Figure 5: Visualization of two custom non-stationary environments. `ChooseBox` a minigrid environment with switching boxes (left) and `MultiPadLunarLander` which extends the classic `LunarLander` by an additional landing pad (right).

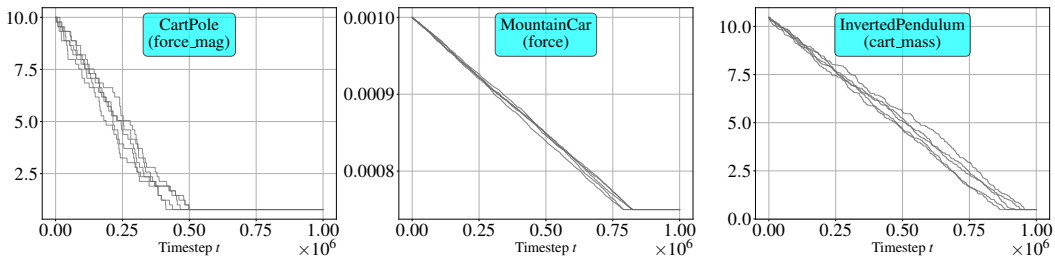


Figure 6: Trajectories with different seeds of the induced non-stationarity on each environment.

Environment	Scheduler	Update Function	Parameter	Initial Value
CartPole	Random($p = 0.00009$)	BoundedDecrement($k = 0.225$, lo $= 0.9$)	`force_mag`	10.0
MountainCar	Random($p = 0.0025$)	BoundedDecrement($k = 0.000000125$, lo $= 0.00075$)	`force`	0.001
InvertedPendulum	Random($p = 0.0004$)	BoundedDecrement($k = 0.0275$, lo $= 0.5$)	`cart_mass`	≈ 10.47
MultiPadLunarLander (discr./cont.)	Random($p = 0.000008$)	(shaping) reward flip	`reward`	–
ChooseBox	Fixed $[5, 10, 15] \cdot 10^5$	reward flip	`reward`	–

Table 2: Scheduler configuration and update functions. "lo" denotes lower bound. The affected environment parameter and its initial parameter values (at $t = 0$) are listed.

code on GitHub (see Section 4) we additionally provide the original result files as ".json" format which records the exact timesteps.

B.3 Special Exploration Environments

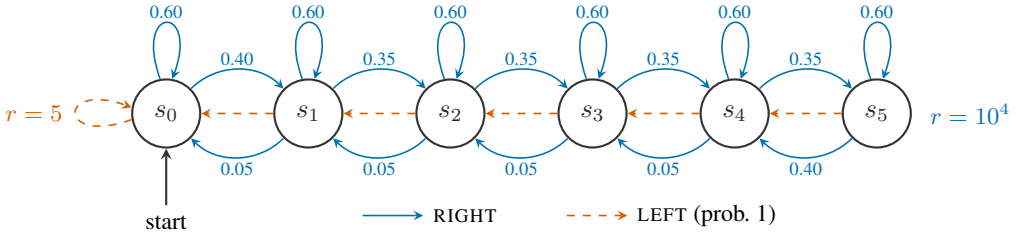


Figure 7: RiverSwim with $S = 6$ states. Solid blue arrows show transitions under RIGHT with their probabilities; dashed orange arrows show LEFT, which is deterministic. Choosing LEFT in s_0 pays 5 and choosing RIGHT in s_5 pays 10^4 (regardless of the resulting transition); zero otherwise.

RiverSwim (Strehl & Littman, 2008) is a chain of $S = 6$ states $s_0, \ldots, s_5$ in a river that flows to the left (Figure 7). Swimming LEFT, with the current, always succeeds, whereas swimming RIGHT, against the current, advances the swimmer only with probability 0.35 (0.40 in s_0) and otherwise leaves it in place or pushes it back. As in the original formulation, the agent receives a reward of 5 for choosing LEFT in s_0 and 10^4 for choosing RIGHT in s_5; all other rewards are zero. Episodes start in s_0 and are truncated after $H = 200$ steps, the agent observes a one-hot encoding of its state.

`RiverSwim` tests deep exploration under stochastic transitions. The small reward in s_0 is available immediately and yields a return of 1000 per episode, whereas the large reward requires swimming against the current for many consecutive steps without any intermediate feedback. An agent that dithers around the locally optimal "always LEFT" policy is carried back by the current and almost never reaches s_5.

DeepSea (Osband et al., 2019) is an $N \times N$ grid with different values for N; we follow the `bsuite` implementation (Osband et al., 2020) (Figure 8). The agent starts in the top-left cell and in every step descends one row while moving one column to the right or to the left, so each episode terminates after exactly N steps. Which of the two actions moves right is drawn at random for every cell from a fixed seed and stays the same across episodes. Every move to the right costs $0.01/N$, and moving right in the bottom-right cell additionally yields a treasure of $+1$. The agent observes a one-hot encoding of its cell (N^2 dimensions).

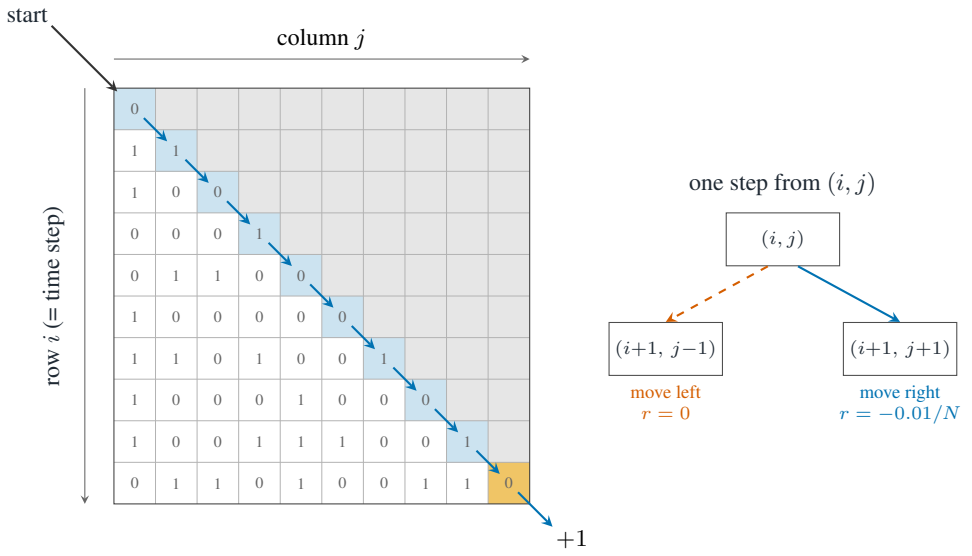


Figure 8: DeepSea with $N = 10$. The agent starts in the top-left cell and descends one row per step while moving one column left or right. Grey cells are unreachable. The only rewarding trajectory is the diagonal (blue), which moves right in every step and collects the treasure $(+1)$ by moving right in the bottom-right cell (gold). Digits show which action index moves right in each cell.

`DeepSea` isolates the need for deep exploration in a deterministic environment. Because the horizontal position can increase by at most one per step, the treasure is obtained by exactly one of the 2^N action sequences, moving right in every step, which yields the optimal return of 0.99. Every other sequence returns $-0.01\,k/N \leq 0$, where k is the number of moves to the right, so until the treasure has been found the move cost makes moving left look optimal everywhere. A uniformly random policy finds the treasure with probability 2^{-N} per episode, so dithering strategies need on the order of 2^N episodes, and the per-cell randomization of actions rules out shortcuts such as always choosing the same action.

B.4 Sampling

ChooseBox The observations in this environment (see Section B.1) are encoded with the Minigrid default as a symbolic 7×7 grid where each cell contains a triple of integers (object_idx, color_idx, state) which encode the type, color and state of the observed cell. Clearly, naively sampling states is not possible: instead we observe that the amount of possible game states is finite (4 agent orientations, 9 positions for the agent, 8 for the first box and thus 7 possible choices for the second box, yielding $9 \cdot 8 \cdot 7 \cdot 4 = 2016$ different states) and we can use the Minigrid interface to generate one of these possible game states and use these as samples.

Others The observation and action spaces of the other environments are constrained and thus we rely on the built-in sampling methods of the defined observation spaces (Towers et al., 2025).

Table 3: Mean online return per environment, averaged over seeds, with the within-environment rank in parentheses (best in bold). Returns are only comparable within a row.

Environment	Base (DQN/SAC)	F (ours)	L2ToInit	Limited	PPO	Resetting
CartPole-v1	98.9 (5)	394.3 (2)	108.3 (4)	48.7 (6)	**475.5 (1)**	213.8 (3)
ChooseBoxEnv-v0	0.7 (2)	**1.6 (1)**	-0.1 (5)	-0.6 (6)	0.1 (4)	0.4 (3)
InvertedPendulum-v5	945.0 (2)	**984.1 (1)**	901.1 (4)	390.2 (6)	860.1 (5)	911.0 (3)
MountainCar-v0	-187.3 (4)	-177.3 (3)	-171.9 (2)	-200.0 (6)	**-147.8 (1)**	-187.3 (5)
MultiPadLunarLander (cont.)	-25.7 (5)	-0.3 (3)	**1.5 (1)**	-37.0 (6)	-0.2 (2)	-4.1 (4)
MultiPadLunarLander (discr.)	-280.6 (6)	14.3 (2)	6.2 (3)	-110.9 (5)	**24.1 (1)**	1.8 (4)

B.5 HYPERPARAMETERS

Non-stationary (Section 4.1) All used hyper-parameters are found in Table 4 – the base parameters were chosen according to the tuned parameter values found in RL zoo (Raffin, 2020). The reset interval for Nikishin et al. (2022)'s approach is chosen heuristically; additionally we use an ε warmup strategy for DQN: this overrides the default chosen exploration strategy by applying the original schedule after each reset (including an initial reset at $t = 0$) so that the final `exploration_final_eps` is reached again after 10^4 steps. The regularization strength for L2ToInit (Kumar et al., 2025) is determined by running the algorithm for $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ on each environment and picking the best value based on the result. The parameters for our proposed forgetting mechanism SsVD were also chosen heuristically testing a few variations in magnitude for the parameter η and selected values for b. Note that the baseline $b = -50$ for `MultiPadLunarLander` can be seen as either optimistic or pessimistic (see Section B.1).

Exploration (Section 4.3) We recall the hyperparameters of SsVD (see Section 3.2 for details): the forgetting ratio ξ, the forgetting learning rate η, and the baseline value b. Some of these follow a linear schedule, denoted $\mathrm{Lin}(a; b; c)$: the value starts at a and decays linearly towards b until a fraction c of the total training time has passed, after which it remains at b.

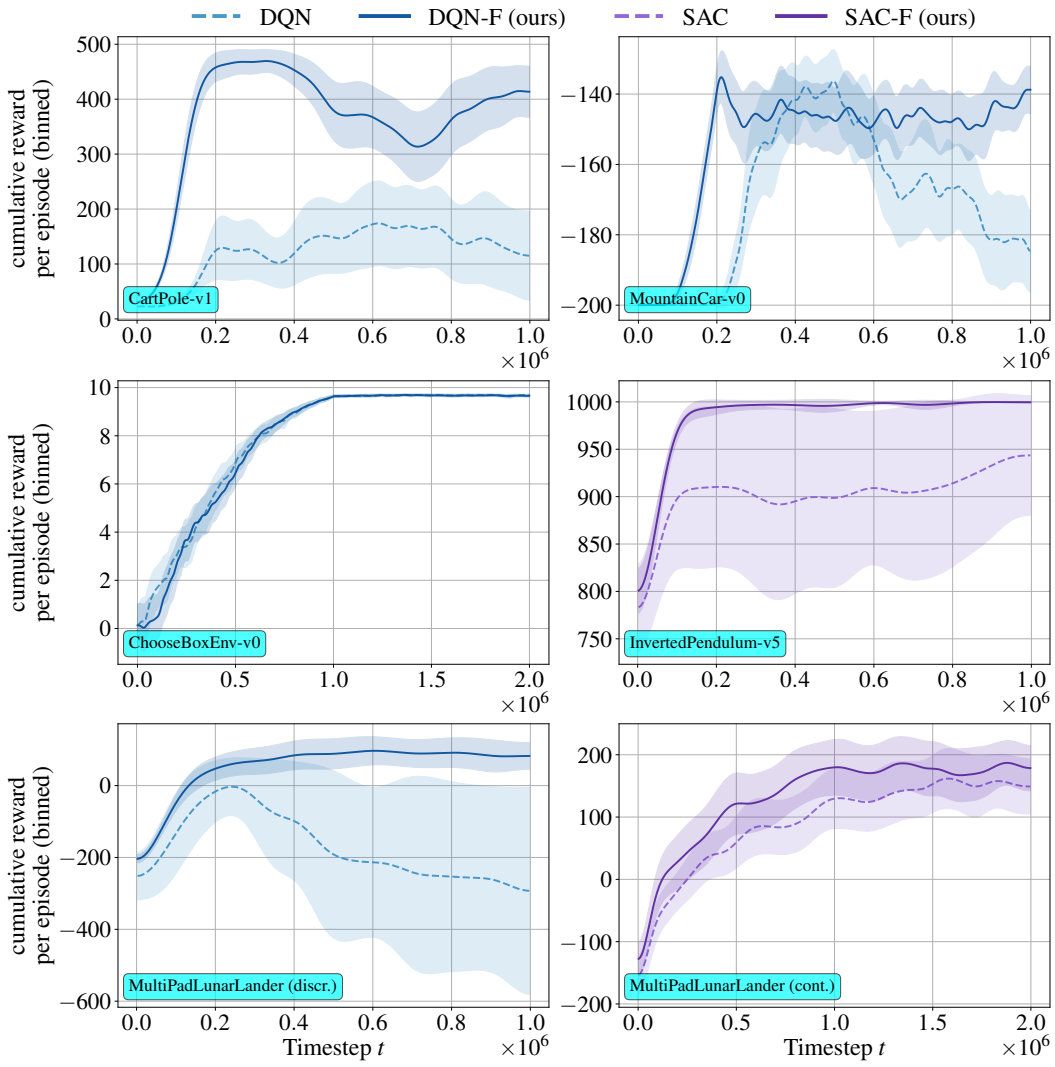


Figure 9: Same experimental setup as Figure 2 but with stationary environments. Only the base algorithm (SAC or DQN) and their counterpart using our proposed SsVD mechanisms are shown.

On `MountainCarContinuous`, we use $\eta = \text{Lin}(0.15; 0; 0.5)$ and $\xi = \text{Lin}(0.25; 0; 0.5)$, so SsVD becomes inactive after 50% of training. The baseline is fixed and chosen optimistically as $b = 100$. On `RiverSwim`, we use fixed values $b = 5000$ and $\eta = 0.25$, together with $\xi = \text{Lin}(1; 0; 0.25)$. On `DeepSea`, we use a fixed $\xi = 0.25$, together with $b = \text{Lin}(100; 1; 0.33)$ and $\eta = \text{Lin}(1; 0.005; 0.66)$.

C Additional Visualizations and Results

Within-environment Ranking The mean ranking in Figure 3 is based on per-environment rankings of the tested algorithms. We rank within each environment because reward scales differ substantially across environments, which would dominate any direct comparison of raw returns. Within each environment, we rank the algorithms by their mean online return, i.e., the average reward attained over the entire course of training, and then average these ranks across environments. Since the base algorithm differs between environments (DQN or SAC), we treat all limited variants (e.g., Limited DQN and LimitedSAC) as a single "Limited" entry, and likewise group the base algorithms into a single entry, enabling a consistent comparison across environments. The per-environment rankings, along with the raw scores for all environments and algorithms, are reported in Table 3.

Ablation: Stationary Environments In Figure 9 we illustrate the same empirical evaluation as done in the main text but now the environments are kept stationary. We can see that our proposed mechanism *does not harm performance* even if the environments are kept stationary – in particular our approach even *improves* learning on all environments except on `ChooseBox` where both perform equally. How strongly an environment is affected by the induced change – or at least how this exhibits in the reward function – varies across the tested environments. On `InvertedPendulum`, for instance, removing the drift yields little improvement, since our approach already performs near-

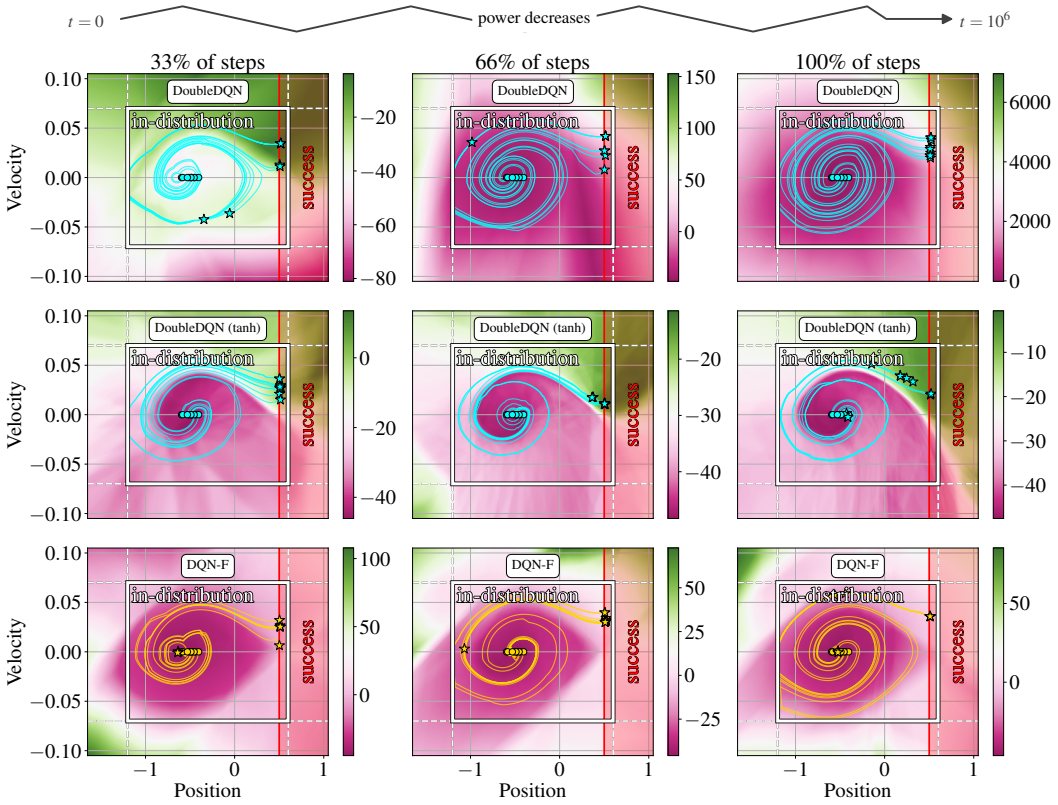


Figure 10: Same experimental setup as Figure 1 but with DoubleDQN instead of DQN and an ablation where the default activation function (ReLU) is replaced by tanh. DoubleDQN is otherwise using the exact same parameters as in Figure 1.

optimally when it is affected by drift. The environment is nonetheless clearly affected by the drift, as evidenced by the performance drop of the "Limited" variant in Figure 2. In other environments, such as `ChooseBox` and `MountainCar`, the stationary setting yields clear improvements. This illustrates the heterogeneity of challenges encountered in non-stationary RL.

Extended Qualitative Analysis Figure 10 extends Figure 1 by DoubleDQN (van Hasselt et al., 2016), and DoubleDQN with $\tanh$ instead of ReLU activations; all other parameters are unchanged. Note that each panel has its own color scale. As every step yields $r_t = -1$, all true values lie below -1 *independent of the engine power*, so any positive estimate is an overestimate of the true value.

DoubleDQN reduces the final magnitude up to $\approx 6 \cdot 10^3$, as well as intermediately after 66% of steps. As online and target network are strongly correlated, the double estimator removes noise-induced maximization bias but not such strong divergence. With $\tanh$, the values stay within the admissible range throughout, so the divergence appears to require unbounded hidden features. As discussed in the main text DQN-F keeps the ReLU architecture without weight decay, yet its values stay at a few tens and nearly all rollouts reach the goal.

Table 4: Hyperparameters used for DQN and SAC. Hyperparameters are used as reported in RL Zoo (Raffin, 2020). "—" indicates that the parameter does not apply to the method (e.g. exploration_fraction in SAC). Parameters for the forgetting mechanisms (ξ, η, b) are found at the bottom and only apply to the -F (SsVD) variants which otherwise use the same parameters as the base algorithm. Number of runs and random seeds used for evaluation is shown as "num_iterations". "max_training_steps" indicates maximum for "Limited" versions. Lin(a; b; c) means that the value is initialized with a and linearly decays towards b until a fraction of the total time c has passed. "gradient_steps" $= -1$ means as many gradient steps as environment steps collected per rollout.

		CartPole-v1	MountainCar-v0	InvertedPendulum-v5	MPLL (discr.)	MPLL (cont.)	ChooseBox
	Algorithm / Hyperparameter	DQN	DQN	SAC	DQN	SAC	DQN
Base	n_timesteps	10^6	10^6	10^6	10^6	10^6	$2 \cdot 10^6$
	policy	MlpPolicy	MlpPolicy	MlpPolicy	MlpPolicy	MlpPolicy	MlpPolicy
	learning_rate	$2.3 \cdot 10^{-3}$	$4 \cdot 10^{-3}$	$3 \cdot 10^{-4}$	$6.3 \cdot 10^{-4}$	Lin($7.3 \cdot 10^{-4}$; 0; 1)	10^{-4}
	batch_size	64	128	256	128	256	64
	buffer_size	100_000	10_000	10_000	50_000	1_000_000	1_000_000
	gamma (γ)	0.995	0.98	0.99	0.99	0.99	0.95
	tau	1	1	0.005	1	0.01	1
	train_freq	256	16	1	4	1	1
	gradient_steps	128	8	1	-1	1	1
	target_update_interval	10 (hard)	600 (hard)	1 (polyak)	250 (hard)	1 (polyak)	20 (hard)
	exploration_fraction	0.16	0.2	—	0.12	—	0.5
	exploration_final_eps	0.05	0.07	—	0.1	—	0.05
	ent_coef	—	—	auto	—	auto	—
	use_sde	—	—	False	—	False	—
	net_arch	[256, 256]	[256, 256]	[256, 256]	[256, 256]	[400, 300]	[64, 64]
SsVD	ξ (for _F only)	0.05	0.05	0.25	0.05	0.05	0.25
	η (for _F only)	0.015	0.0015	0.0015	0.015	0.015	0.015
	b (for _F only)	0	0	0	0	-50	5
Res.	Interval J	10^5	10^5	$2 \cdot 10^5$	10^5	$2 \cdot 10^5$	$4 \cdot 10^5$
L2	λ	10^{-5}	10^{-4}	10^{-5}	10^{-5}	10^{-4}	10^{-5}
Lim.	max_training_steps	$2 \cdot 10^5$	$2 \cdot 10^5$	10^5	$2 \cdot 10^5$	$2.5 \cdot 10^5$	$3.5 \cdot 10^5$
	num_iterations	10	10	10	10	10	10