Reweighted Bellman Targets for Continual Reinforcement Learning

Ke Sun[1], **Jun Jin**[1], **Xi Chen**[2], **Wulong Liu**[2], **Linglong Kong**[1*]
[1]University of Alberta, Edmonton, Canada
[2]Huawei Noah's Ark Lab
{ksun6,jjin5,lkong}@ualberta.ca
{xi.chen4,liuwulong}@huawei.com

Abstract

One roadblock to building general AI agents is the inability to continually learn and adapt to environmental changes without dramatically forgetting previous knowledge. This deficiency is highly linked to most reinforcement learning (RL) methods being designed based on the critical assumption of a fixed environment transition dynamics and reward function. To address these limitations, in this paper, we first dive deeper into the less studied foundations of continual RL, focusing on defining the MDP distance and catastrophic forgetting based on the difference of optimal value functions. In particular, we analyze the learning behaviors of continual RL algorithms with the sole stability or plasticity ability. A theoretically principled continual RL algorithm is further proposed by reweighting the historical and current Bellman targets, explicitly balancing the stability and plasticity in continual RL. We conduct rigorous experiments in the tabular setting to corroborate our analytical results, suggesting the potential of our proposed algorithm in real continual RL scenarios.

1 Introduction

It is notoriously challenging to develop general artificial intelligence agents that can continually learn new tasks while maintaining the knowledge they previously acquired in the historical tasks. This research field also referred to *continual reinforcement learning (RL)* [21], which has gained increasing attention in recent years with solutions growing substantially [4, 20, 19, 6, 18, 15, 42, 39, 3, 37]. Developing practical continual RL algorithms is crucial yet challenging as most successful RL algorithms are designed for a single Markov decision process (MDP) [33], where they assume the underlying MDP is stationary with a fixed reward function and state transition dynamics. Nonetheless, this assumption is often violated in practical problems [7], limiting the generality of RL algorithms to continually learn and adapt to environmental changes toward building human-level agents.

Challenges in Continual RL. Unlike the human brain, deep neural networks are prone to *catastrophic forgetting* issue [14, 27], as deep nets or deep RL agents often quickly worsen their performance on the previous tasks when sequentially trained on a sequence of new tasks [38], such as new datasets or environments. Despite the promising progress of continual RL to ameliorate catastrophic forgetting, we still have a limited understanding of how to define and solve this continual learning problem. Recently, a conceptual basis of continual RL was provided in [1] to formalize the notion that "agents can never stop learning". Nevertheless, an explicit gap exists between their formalism and a practical algorithm design. How to explicitly quantify foundations in continual RL, including catastrophic forgetting, plasticity, and stability, has yet to be explored.

*Corresponding author

Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2024).

Our Contributions. In this paper, we build out the foundations of continual RL. Specifically, we leverage the difference of optimal value functions in the respective MDPs to define the MDP distance, based on which the catastrophic forgetting is explicitly quantified. We further characterize the learning behaviors of a continual RL algorithm with the sole stability or plasticity capability. The sole plasticity degrades continual RL to a Finetune algorithm, suffering from complete catastrophic forgetting. On the other hand, continual RL that merely reduces catastrophic forgetting is infeasible in computation. To circumvent their drawbacks, we introduce a practical continual RL algorithm by reweighting previous and current Bellman targets. This leads to a favorable trade-off between mitigating catastrophic forgetting and reducing computation burden. Our algorithm is thus theoretically principled and is accomplished by maintaining a fixed copy of value networks trained on each previous MDP. Finally, we conduct extensive experiments on the tabular setting to demonstrate our theoretical results and evaluate the performance of considered continual RL algorithms. Our empirical results substantiate the applicability of our proposed framework and the potential of our proposed algorithm in real applications. Our contributions are summarized as follows:

(1) We analyze the foundations for continual RL, defining the MDP distance, catastrophic forgetting, and adaptivity regarding the optimal Q functions.

(2) We develop a theoretically principled continual RL algorithm by reweighing the previous and current Bellman targets, which explicitly balances the plasticity and stability.

(3) We conduct extensive experiments in the tabular setting to verify the effectiveness of our proposed continual RL algorithms.

2 Related Work

Continual RL. Continual learning (CL) [34, 9] has been one of the most critical milestones on the path to building artificial general intelligence. Existing methods can be mainly classified into three groups, including rehearsal methods [25, 8], regularization-based methods [29, 22, 2] as well as parameter isolation approaches [41, 26]. Within continual learning, there has also been growing interest in the problem of training agents on the sequence of tasks, also referred to as continual RL [21]. Existing continual RL algorithms are designed from a variety of perspectives, including the synaptic model [18], behavioral cloning that queries all previous policies [39], sparse prompting [42], policy consolidation [19] and policy subspace building [15]. [3] introduces permanent and transient value functions by performing an interplay between fast and slow learning. By contrast, the reweighed Bellman targets from the previous and current value functions in our method also integrate learning at various speeds, but we maintain the Bellman targets and employ the reweighing strategy. In summary, the design of continual RL algorithms seeks a trade-off between the performance and model size [15]. However, most of the existing continual RL approaches tend to be heuristic, and there is still a fundamental lack of a theoretical analysis framework. Experimentally, only a few benchmarks have been recently proposed [40, 16, 30, 35], and they still need to be verified widely in the future.

Ensemble and Reweighted Methods in RL. Reweighted and ensemble methods have suggested substantial success in a wide range of RL problems, including the weighted Q learning [10] that can reduce the bias in target estimates, Anderson Acceleration [36] that reweights previous target estimates in the fixed-point iteration to speed up the convergence [32, 24], and ensemble RL [23] that helps reduce the variance in target estimates. Reweighting past data was typically utilized to search for a policy that maximizes future performance in one single non-stationary MDP [7], while our work is the first to explore the efficacy of reweighting target estimates in the continual RL setting.

3 Foundations of Continual RL

3.1 Continual RL Setting and MDP Distance

Consider we have a sequence of T tasks denoted by $t = 1, ..., T$, where each task t is modeled by a Markov Decision Process (MDP) $\mathcal{M}_t = \langle \mathcal{S}_t, \mathcal{A}_t, P_t, R_t, \gamma \rangle$, with a set of states $\mathcal{S}_t$ and actions $\mathcal{A}_t$, the environment transition dynamics $P_t : \mathcal{S}_t \times \mathcal{A}_t \to \mathcal{P}(\mathcal{S}_t)$ and the reward function $R_t : \mathcal{S}_t \times \mathcal{A}_t \to \mathbb{R}$. In our work, we assume the same state and action space across T tasks and the known task boundaries for the RL agent. In continual RL, we aim to seek an optimal policy π_{CL} after sequential training that can be generalized favorably across all tasks. When training on each

task t, we typically require a *training budget*, including a moderate model size and an allowable computation cost. For practical continual RL algorithms, having full access to interact with prior MDPs or an arbitrarily large model size is typically infeasible. We define the action-value function $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$ given the state s, the action a, and a policy π.

MDP Distance. A desirable definition of the MDP distance should fully consider the variation of both reward functions and state transition dynamics between two MDPs. To this end, we use the normalized distance between two MDP-determined optimal Q functions $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ to define MDP distance in Definition 1.

Definition 1. ***(MDP Distance)*** *For two finite MDPs $\mathcal{M}_1 = (\mathcal{S}, \mathcal{A}, R_1, P_1, \gamma)$ and $\mathcal{M}_2 = (\mathcal{S}, \mathcal{A}, R_2, P_2, \gamma)$, we denote their optimal Q functions as Q_1^* and Q_2^*. The p-norm (normalized) MDP distance $d_p(\mathcal{M}_1, \mathcal{M}_2)$ is defined as*

$$d_p(\mathcal{M}_1, \mathcal{M}_2) \stackrel{def}{=} \left(\sum_{s,a}\left|\frac{Q_1^*(s,a)}{\|Q_1^*\|} - \frac{Q_2^*(s,a)}{\|Q_2^*\|}\right|^p\right)^{1/p}. \tag{1}$$

Unless otherwise stated, we mainly consider d_2 ($p = 2$) in our paper and $\|\cdot\|$ represents ℓ_2-norm. Remarkably, we recommend employing normalization for each $Q^*(s,a)$ to maintain *scale-invariance* and *comparability* properties. Any MDP with scaled rewards retains the same policy and should remain the MDP distance as the original. The normalization ensures the scale invariance, enabling the capture of the MDP difference more accurately, regardless of the influence of reward scaling. In Appendix A, we show that the normalized ℓ_2 distance in Eq. 1 is equivalent to Cosine distance. In addition, the normalization standardizes the comparison between different MDPs, which is instrumental in quantifying catastrophic forgetting in Section 3.2. In general, we equally compare the forgetting among past environments, necessitating the same magnitude in measuring MDP distance.

Note that employing the optimal Q function to measure MDP distance may not uniquely identify the difference of a state transition and reward function pair between two MDPs. However, it remains a preferable measure as the difference between two optimal Q functions effectively captures the variations in both state transitions and reward functions in a unified and concise manner. A follow-up definition of a weighted MDP distance is provided in Appendix B.1), which establishes an underlying connection with the concept of catastrophic forgetting discussed in Section 3.2.

3.2 Stability: Catastrophic Forgetting

Our definition of catastrophic forgetting in continual RL is inspired by *distribution drift* and *catastrophic forgetting* quantified in deep learning scenario [11], which we have a brief recap in Appendix B.2. Before introducing the general catastrophic forgetting across a sequence of MDPs, we first consider a simple continual learning from a source MDP $\mathcal{M}_{\text{sou}}$ and a target MDP $\mathcal{M}_{\text{tar}}$. For any learning algorithm, we denote $\widehat{Q}_{\text{sou}}$ and $\widehat{Q}_{\text{tar}}$ as estimated Q functions after training from $\mathcal{M}_{\text{sou}}$ to $\mathcal{M}_{\text{tar}}$. We define the resulting target policy π_{tar} based on $\widehat{Q}_{\text{tar}}$, following the greedy rule, i.e., $\pi_{\text{tar}}(\cdot|s) = \arg\max_a \widehat{Q}_{\text{tar}}(s,a)$. Grounded in the definition of distribution drift between two MDPs (see Appendix B.3), we introduce catastrophic forgetting between two MDPs in Definition 2.

Definition 2. ***(Catastrophic Forgetting between two MDPs)*** *For any given algorithm, if we apply the target policy π_{tar} to interact within the source MDP $\mathcal{M}_{sou}$, we denote the resulting state distribution as $\mu_{sou}^{\pi_{tar}}$. The catastrophic forgetting $\Delta_{sou,tar}^{\pi_{tar}}(\mathcal{M}_{sou})$ is formulated as*

$$\Delta_{sou,tar}^{\pi_{tar}}(\mathcal{M}_{sou}) = \sum_{s,a}\mu_{sou}^{\pi_{tar}}(s)\pi_{tar}(a|s)\left(\frac{\widehat{Q}_{sou}(s,a)}{\|\widehat{Q}_{sou}\|} - \frac{\widehat{Q}_{tar}(s,a)}{\|\widehat{Q}_{tar}\|}\right)^2, \tag{2}$$

where the catastrophic forgetting $\Delta_{\text{sou,tar}}^{\pi_{\text{tar}}}(\mathcal{M}_{\text{sou}})$ in Eq. 2 can be seen as a variant of weighted MDP distance by letting $\widehat{Q}_{\text{sou}} = Q_{\text{sou}}^*$, $\widehat{Q}_{\text{tar}} = Q_{\text{tar}}^*$, and the weight as $\mu_{\text{sou}}^{\pi_{\text{tar}}}(s)\pi_{\text{tar}}(a|s)$. Notably, the state distribution is simultaneously determined by the MDP transition and the behavior policy. We next define the general catastrophic forgetting across a sequence of MDPs $\{\mathcal{M}_t\}_{t=1}^T$. The quantity of interest is one single optimal Q function estimator Q_T associated with the policy π_{CL}, which is expected to perform favorably across all considered MDPs. Importantly, we are capable of storing the typically lightweight Q functions $\{\widehat{Q}_t\}_{t\le T}$ and querying them to develop Q_T. This strategy also implies that the model size of our continual RL algorithm is linearly increasing in terms of the number of tasks.

Definition 3. ***(Catastrophic Forgetting in Continual RL)*** *Consider continual RL setting across a series of MDP $\{\mathcal{M}_t\}_{t=1}^T$. The catastrophic forgetting CF(Q_T) regarding Q_T is defined as*

$$CF(Q_T) = \sum_{t=1}^{T} \Delta_{t,T}^{\pi_{CL}}(\mathcal{M}_t) = \sum_{t=1}^{T} \sum_{s,a} \mu_t^{\pi_{CL}}(s)\pi_{CL}(a|s) \left(\frac{\widehat{Q}_t(s,a)}{\|\widehat{Q}_t\|} - Q_T(s,a) \right)^2, \tag{3}$$

where we use $\Delta_{i,j}^{\pi}(\mathcal{M}_i)$ in general to represent the catastrophic forgetting from the i-th to j-th MDP by leveraging the policy π. By solving $Q_T^{\text{opt}} = \arg\min \text{CF}(Q_T)$, we obtain a global policy π_{CL} following the greedy rule $\pi_{\text{CL}}(\cdot|s) = \arg\max_a Q_T^{\text{opt}}(s,a)$. Intuitively, when fixing $\{\widehat{Q}_t\}_{t\leq T}$ we can find the optimal Q_T^{opt} in the subspace spanned by the basic functions $\{\widehat{Q}_t\}_{t\leq T}$, a representation for Q_T^{opt} via $\{\widehat{Q}_t\}_{t\leq T}$. However, it is almost infeasible to directly attain Q_T^{opt} by minimizing Eq. 3, as it will lead to solving an *implicit equation* in Proposition 1. The proof is provided in Appendix D.

Proposition 1. *(Implicit Optimality Equation) Denote the weight $w_t^{\pi_{CL}}(s,a) = \mu_t^{\pi_{CL}}(s)\pi_{CL}(a|s)$. When fixing $w_t^{\pi_{CL}}$, the optimal estimator Q_T^{opt} by minimizing CF(Q_T) in Eq. 3 satisfies the equation:*

$$Q_T^{opt}(s,a) = \sum_{t=1}^{T} w_t^{\pi_{CL}}(s,a) \frac{\widehat{Q}_t(s,a)}{\|\widehat{Q}_t\|} \Big/ \sum_{t=1}^{T} w_t^{\pi_{CL}}(s,a), \tag{4}$$

where π_{CL} is the function of Q_T^{opt}, following the greedy rule, i.e., $\pi_{CL}(a^*|s) = 1$ if $a^* = \arg\max_{a'} Q_T^{opt}(s,a')$, otherwise, $\pi_{CL}(\cdot|s) = 0$.

Implicit Optimality Equation, Online Alternating Algorithm, and Infeasible Computation. For conciseness, we use w_t^{π} to denote $w_t^{\pi_{\text{CL}}}$. For practical algorithms, the weight $w_t^{\pi}(s,a)$ can be approximated by Monte Carlo, i.e., $\widehat{w}_t^{\pi}(s_i,a_i) = \frac{1}{N_t}\sum_{i=1}^{N_t} \mathbf{1}_{\{s_t=s_i,a_t=a_i\}}$, where the state-action pairs (s_i,a_i) are collected by applying the behavior policy π_{CL} in $\mathcal{M}_t$ and N_t is the number of collected sample pairs. Despite the feasibility of applying the Monte Carlo method, the minimizer Q_T^{opt} is coupled with π_{CL} in w_t^{π} within this *implicit optimality equation*. For a straightforward solution, we introduce an online alternating algorithm (see Algorithm 1 in Appendix E), which alternately updates the optimal Q function Q_T^{opt} and the weight w_t^{π}. Nonetheless, the algorithm requires as many interactions within all T MDPs as possible to accurately approximate the true weight $w_t^{\pi}(s,a)$ for each MDP, resulting in typically intractable computation, especially for MDPs with large state and action spaces. This computational bottleneck motivates us to devise an efficient algorithm while minimizing the catastrophic forgetting CF(Q_T), on which we elaborate in Section 4.

3.3 Plasticity: Adaptivity and Convergence

In contrast to the stability and catastrophic forgetting, *plasticity* refers to the capability of an RL agent that can quickly learn new experiences and adapt to changes in the environment. In fact, a continual RL algorithm that merely minimizes catastrophic forgetting may not quickly adjust to the new environment and can even diverge in the optimization process. Specifically, the one that fully captures the plasticity and ignores the stability is the commonly used *Finetune algorithm*. The Finetune algorithm follows MDP-wise training by deploying a specific RL algorithm without imposing any continual learning strategy. Consequently, the obtained Q function after training the previous MDP serves as the initialization of Q functions on the current MDP. The global policy π_{CL} is directly obtained based on $\widehat{Q}_T$ after training on the T-th MDP immediately, i.e., $Q_T^{\text{opt}} = \widehat{Q}_T$. We contend that understanding the continual learning behaviors of the Finetune algorithm is indispensable for analyzing the plasticity of any continual RL algorithms. For the Finetune continual RL algorithm, we make the following conclusions from the perspectives of convergence and convergence rate:

- **Convergence.** The estimator $\widehat{Q}_t$ we obtain in each MDP will converge to the MDP-dependent optimal Q function Q_t^* regardless of MDP distance.
- **Convergence rate.** The convergence rate on the current t-th MDP is determined by MDP distance only with its preceding MDP, i.e., $d_p(\mathcal{M}_{t-1}, \mathcal{M}_t)$.

We present our analytical results in the general framework of Fitted Q Iteration (FQI) [31, 12] that provides a favorable interpretation of many practical deep RL algorithms, such as DQN [28]. The FQI captures two key features, including the leverage of the target network and the experience replay.

The objective function in the k-th phrase of FQI is $\widehat{Q}_\theta^{k+1} = \arg\min_{Q_\theta} \frac{1}{n}\sum_{i=1}^n \left[y_i - Q_\theta\left(s_i, a_i\right)\right]^2$, where the target $y_i = r(s_i, a_i) + \gamma \max_{a\in\mathcal{A}} Q_{\theta^*}^k\left(s_i', a\right)$ is fixed within every T_{target} steps to update target network Q_{θ^*} by letting $\theta^* = \theta$. We define Bellman optimality operator $\mathcal{T}^{\text{opt}}$ such that $\mathcal{T}^{\text{opt}}Q(s,a) = \mathbb{E}\left[R(s,a)\right] + \gamma \max_{a'} \mathbb{E}_{s'}\left[Q(s',a')\right]$. Theorem 1 demonstrates the convergence behaviors of Finetune continual RL algorithm, denoted as *Finetune FQI*:

Theorem 1. *(Convergence of Finetune FQI.) Denote Q_t^* as the optimal Q function for the t-th MDP, $\widehat{Q}_t^k$ as the Q function estimate after the k-th phase of FQI in the t-th MDP. If the regression error $e_t^k(n) = \|\widehat{Q}_t^{k+1} - \mathcal{T}^{opt}Q_{\theta^*}^k\|_\infty$ decreases to 0 as the sample size $n \to +\infty$, we have:*

(1) $\|\widehat{Q}_t^k - Q_t^*\|_\infty \le \gamma^k d_\infty(\mathcal{M}_{t-1}, \mathcal{M}_t)$ as $n \to +\infty$; $\|\widehat{Q}_t^k - Q_t^*\|_\infty \to 0$ if we further let $k \to +\infty$.

(2) The iteration complexity is $\mathcal{O}(\log \frac{d_\infty(\mathcal{M}_{t-1},\mathcal{M}_t)}{\epsilon})$ in the t-th MDP given the tolerance error ϵ.

Please refer to Appendix C for the detailed proof. Theorem 1 demonstrates the estimated Q function has an *Markov-like property* in Finetune FQI, whose convergence rate is determined by the MDP distance only between the preceding and current MDPs, regardless of other MDPs. More importantly, the Q function estimator in the Finetune FQI algorithm in each MDP can asymptotically converge to the MDP-dependent optimal Q function Q_t^*, suffering from complete catastrophic forgetting about the knowledge of previous environments. While the results in Theorem 1 are direct follow-up conclusions of FQI and may not be surprising, they serve as foundations to understand the plasticity ability of continual RL algorithms. Overall, without incorporating any strategy to minimize catastrophic forgetting, the resulting learning algorithm possesses full plasticity ability under mild conditions. The principal caveat is the convergence rate, which hinges on two adjacent MDPs.

Remark: Non-decreasing Regression Error $e_t^k(n)$. In deep RL, the lack of access to sufficient samples often leads to violations of the decreasing regression error condition. This results in $\widehat{Q}_t$ being additionally affected by the neural network parameters from previous or even older environments. This issue is, in fact, related to the transfer challenges in continual RL, as discussed in [39], and goes beyond the conventional stability and plasticity concerns. We leave this direction as future work.

4 Our Approach: Continual RL with Reweighted Bellman Targets

Motivation: Trade-off between Stability and Plasticity. The analysis in Section 3 motivates us to find a trade-off between plasticity and catastrophic forgetting. A key insight is the optimal Q function Q_T^{opt} in Proposition 1 by minimizing the catastrophic forgetting has a weighted average form of the estimated Q functions among all MDPs. It is, therefore, theoretically principled to design a continual RL algorithm by reweighting Bellman targets, e.g., the estimated Q functions from MDPs that the agent has interacted with. Our algorithm explicitly considers the two crucial characteristics of continual learning, i.e., stability and plasticity. **(1) Stability.** The past Q functions encompass the knowledge from previous MDPs. Incorporating them into the target to guide the learning in the current MDP contributes to stability. **(2) Plasticity.** As the weighted target also includes the current Q function estimate, the agent can adapt to the new environment, which enhances plasticity.

4.1 Algorithm Framework with Reweighted Bellman Targets

Recap the updated rule $Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \eta_k\left[r(s,a) + \gamma \max_{a'} Q_k(s',a') - Q_k(s,a)\right]$ in the vanilla Q learning, where η_k is the step size in the k-th step. Similarly, the updating rule with reweighted targets in continual Q learning for the t-th MDP is

$$Q_k^t(s,a) \leftarrow Q_k^t(s,a) + \eta_k\left[r^t(s,a) + \gamma \sum_{i=1}^t \alpha_i \max_{a'} \widehat{Q}_i(s',a') - Q_k^t(s,a)\right],$$

where $\sum_{i=1}^t \alpha_i \max_{a'} \widehat{Q}_i(s',a')$ with the weight α_i plus the reward $r^t(s,a)$ collected in the current t-th MDP serves as the target, and $\alpha = [\alpha_1, ..., \alpha_t]^\top$. In the FQI, we further have

$$\widehat{Q}_t^{k+1} = \underset{Q_t^k}{\text{argmin}} \frac{1}{n}\sum_{i=1}^n \left[\bar{y}_i^t(\alpha) - Q_t^k\left(s_i, a_i\right)\right]^2, \quad \text{s.t. } \alpha \succeq 0, \alpha^\top \mathbf{1} = 1, \tag{5}$$

where the reweighted target is $\bar{y}_i^t(\alpha) = r^t(s_i,a_i) + \gamma \sum_{j=1}^t \alpha_j \max_{a'} \widehat{Q}_j^k(s_i',a')$. $\widehat{Q}_t^k = Q_{\theta_t^*}^k$ is the target network in the t-th MDP, acting as the Bellman target for the learning in the current MDP.

Remarkably, the proposed algorithm in Eq. 5 is a variant of FQI as the reweighted targets are fixed within each updating phase, akin to the vanilla FQI.

Principle of Selecting α: Balancing Stability and Plasticity. The optimal α is selected by considering both stability and plasticity: 1) to minimize catastrophic forgetting CF(Q_T) while incorporating the knowledge from previous MDPs, and 2) to possess the convergence guarantee in the current MDP.

4.2 Stability: Minimizing Catastrophic Forgetting

With a pre-specified weighted form of Q_T^{opt} as shown in Proposition 2, we consider to plug this parametric from of Q_t in catastrophic forgetting CF(Q_T) defined in Eq. 3. This leads to a bi-level optimization with the stability constraint to minimize catastrophic forgetting:

$$\begin{aligned}
&\widehat{Q}_t^{k+1}(\alpha^*) = \underset{Q_t^k}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left[\bar{y}_i^t(\alpha^*) - Q_t^k\left(s_i, a_i\right) \right]^2, \\
&\text{s.t. } \alpha^*(\widehat{Q}_t^k) = \arg\min_{\alpha} \sum_{i=1}^{t} \sum_{s,a} \mu_i^\pi(s)\pi(a|s) \left(\frac{\widehat{Q}_i(s,a)}{\|\widehat{Q}_i\|} - \sum_{j=1}^{t} \alpha_j \widehat{Q}_j^k(s,a) \right)^2, \alpha \succeq 0, \alpha^\top \mathbf{1} = 1,
\end{aligned} \tag{6}$$

where the constraint objective function CF(Q_T) is reformulated by replacing Q_T with $\sum_{j=1}^{t} \alpha_j \widehat{Q}_j^k$. In updating the target network, we have $\widehat{Q}_t^k = Q_{\theta^*}^k$. In the constraint, $\alpha^*(\widehat{Q}_t^k)$ is first obtained by minimizing CF(Q_T) regarding a fixed $\widehat{Q}_t^k$, which further consists of $\bar{y}_i^t(\alpha^*)$ in the objective function. By solving the regression problem, we attain $\widehat{Q}_t^{k+1}$, which is then utilized to optimize $\alpha^*(\widehat{Q}_t^{k+1})$ in the $(k+1)$-th phrase of FQI. From the perspective of computational cost, however, the resulting lower-level optimization is also expensive as the exact evaluation on $\mu_i^\pi(s)\pi(a|s)$ requires the agent to interact with previous MDPs additionally. To ease this burden, we choose to simplify CF(Q_T) by approximating $\mu_i^\pi(s)\pi(a|s)$ with a uniform distribution, i.e., $\mu_i^\pi(s)\pi(a|s) \approx \frac{1}{|S||A|}$:

$$\sum_{i=1}^{t} \sum_{s,a} \mu_i^\pi(s)\pi(a|s) \left(\frac{\widehat{Q}_i(s,a)}{\|\widehat{Q}_i\|} - \sum_{j=1}^{t} \alpha_j \widehat{Q}_j^k(s,a) \right)^2 \approx \sum_{i=1}^{t} \|\delta_i^k \alpha\|_2^2 = \|\Gamma^k \alpha\|_2^2, \tag{7}$$

where $\delta_i^k = [\widehat{Q}_i^k/\|\widehat{Q}_i^k\| - \widehat{Q}_1^k, \widehat{Q}_i^k/\|\widehat{Q}_i^k\| - \widehat{Q}_2^k ..., \widehat{Q}_i^k/\|\widehat{Q}_i^k\| - \widehat{Q}_t^k] \in \mathbb{R}^{|S\times A|\times t}$ and $\Gamma^k = \left[\delta_1^T, \delta_2^T, \ldots, \delta_t^T\right]^T \in \mathbb{R}^{t|S\times A|\times t}$ are the constant matrixes within each lower-level optimization of FQI. Such a simplification leads to a quadratic objective function w.r.t. α, enjoying a unique solution and facilitating commonly-used optimization tools, e.g., the CVXOPT toolbox in Python. We can tolerate a certain amount of interactions with the environments within the training budget or collect offline datasets to approximate the $\mu_i^\pi(s)\pi(a|s)$ more accurately than the uniform strategy we use here. We leave this direct improvement as future work and focus on the simplest form of stability in our methodology. Moreover, the optimization form in Eq. 7 has an interesting connection with a specific type of acceleration algorithm, called Anderson Acceleration [36], for which we provide a discussion in Appendix F for interested readers.

Tractable Computation and Space Complexity. Within the stochastic optimization paradigm for practical algorithms, the state space $|\mathcal{S}|$, which consists of the dimension of Γ^K, will be reduced to the batch size for the batch-level training. The Q function is typically lightweight instead of parameterized by very deep nets to avoid the optimization instability issue [5].

4.3 Plasticity: Convergence Guarantee

The plasticity ability of a continual RL algorithm enables rapid adaptation and convergence within a new MDP. To examine the convergence properties of our continual RL algorithm by introducing a *continual learning Bellman Optimality Operator* $\mathcal{T}_{\text{CL}}^{\text{opt}}$ within the k-phase of FQI framework:

$$\begin{aligned}
\mathcal{T}_{\text{CL}}^{\text{opt}} Q_t^k(s,a) &= \mathbb{E}\left[R_t(s,a)\right] + \gamma \sum_{s'} \mathcal{P}_{s,s'}^a \max_{a'} Q_t^k(s',a'), \\
Q_t^k(s,a) &= \mathbb{E}\left[R_t(s,a)\right] + \gamma \sum_{s'} \mathcal{P}_{s,s'}^a \sum_{i=1}^{t} \alpha_i^k \max_{a'}, Q_i^{k-1}(s',a')
\end{aligned} \tag{8}$$

where $\mathcal{P}^a_{s,s'} = P(s'|s,a)$ is the state transition dynamics. We define $\sum_{s'} \mathcal{P}^a_{s,s'} \sum_{a'} \pi^*(a'|s') = \sum_{s'} \mathcal{P}^{\pi^*}_{s,s'}$, omitting a in $\mathcal{P}^{\pi^*}_{s,s'}$ for brevity, with $\pi^*(\cdot|s') = \arg\max Q_i(s',\cdot)$. To guarantee the convergence of Q^k_t in the t-th MDP under $\mathcal{T}^{\text{opt}}_{\text{CL}}$, we have the following condition in Proposition 2.

Proposition 2. *$\mathcal{T}^{opt}_{CL}$ has a γ-linear convergence rate $\|\mathcal{T}^{opt}_{CL} Q^k_t - Q^k_t\| \leq \gamma \|\mathcal{T}^{opt}_{CL} Q^{k-1}_t - Q^{k-1}_t\|$, if*

$$\alpha^k = \arg\min_{\alpha} \| \sum_{i=1}^{t} \alpha_i (\mathcal{T}^{opt}_{CL} Q^{k-1}_i - Q^{k-1}_i)\|, \quad \textit{s.t. } \alpha \succeq 0, \alpha^\top \mathbf{1} = 1 \tag{9}$$

where $Q^{k-1} = [Q^{k-1}_1, ..., Q^{k-1}_t] \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times t}$ and $\mathcal{T}^{opt}_{CL} Q^{k-1} = [\mathcal{T}^{opt}_{CL} Q^{k-1}_1, ..., \mathcal{T}^{opt}_{CL} Q^{k-1}_t]$.

The proof of Proposition 2 is given in Appendix G. Proposition 2 demonstrates that α_k should be the obtained by solving Eq. 9 for γ-linear convergence of $\mathcal{T}^{\text{opt}}_{\text{CL}}$, the plasticity ability in the current MDP. In practice, we leverage $\widehat{Q}_i = Q^k_i$ for all k in each $i = 1, ..., t-1$.

4.4 Putting All Together: Continual RL algorithm with Reweighted Bellman Targets

To combine the two constraints on α^k regarding stability and plasticity, we introduce the coefficient λ. The complete bi-level optimization of our continual RL algorithm with reweighted Bellman targets is:

$$\begin{aligned} \widehat{Q}^{k+1}_t(\alpha^k) &= \underset{Q^k_t}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left[\bar{y}^t_i(\alpha^k) - Q^k_t(s_i, a_i) \right]^2, \\ \text{s.t.} \quad \alpha^k(\widehat{Q}^k_t) &= \arg\min_{\alpha} \| \sum_{i=1}^{t} \alpha_i (\mathcal{T}^{\text{opt}}_{\text{CL}} \widehat{Q}^{k-1}_i - \widehat{Q}^{k-1}_i)\|^2 + \lambda \|\Gamma^k \alpha\|^2, \alpha \succeq 0, \alpha^\top \mathbf{1} = 1. \end{aligned} \tag{10}$$

where λ controls the strength of catastrophic forgetting over the convergence, explicitly trading off the stability-plasticity dilemma in continual RL. It is worth noting that the upper-level optimization is an iterative regression problem in terms of Q^k_t given α^k, while the lower-level one is a quadratic convex optimization problem regarding α^k. We can initialize $\alpha^0 = [1/t, .., 1/t]^\top$.

Interpolation between Sole Plasticity and Stability. When $\lambda = 0$, our algorithm only emphasizes plasticity and will degrade to Finetune FQI if we further let $\alpha^k = [0, ..., 1]$. Conversely, as $\lambda \to +\infty$ and without the uniform approximation in Γ^k, the solution to Eq. 10 in our algorithm also satisfies the optimal condition in Proposition 1, solely focusing on reducing catastrophic forgetting.

5 Experiments

In this section, we conduct experiments to verify the analytical results in Sections 3.2 and 3.3 and the effectiveness of our proposed continual RL algorithm with reweighted Bellman targets. Concretely, we sequentially deploy different continual RL algorithms on an array of MDPs with distinct reward functions and state transition dynamics. As our analysis and the proposed algorithm are mainly value-based in the tabular setting, we demonstrate our results on a simple MDP and the Grid World environment.

Baselines. (1) **Minimizing Catastrophic Forgetting (MCF)**. As discussed in Section 3.2, a continual RL algorithm that solely minimizes catastrophic forgetting is typically infeasible in computation. However, we can deploy an online alternating algorithm (Algorithm 1), which alternately solves the implicit optimality equation in Proposition 1, to approximately minimize catastrophic forgetting. We call this baseline algorithm Minimizing Catastrophic Forgetting (MCF), which normally serves as the upper bound of performance if the approximation is accurate. (2) **Finetune**. As analyzed in Section 3.3, the Finetune continual algorithm has a rapid adaptation capability to learn in the new environment but suffers from complete catastrophic forgetting. (3) $\boldsymbol{V}^*$. We also report the estimated Q function by deploying separate Q learning on each MDP. We denote this baseline algorithm as V^* because it can achieve almost optimal value function, although it is not a continual learning algorithm in nature. (4) **Optimal**. Since searching-based algorithms can solve many simple MDPs, we thus evaluate the optimal value function by leveraging the well-known Floyd's algorithm [13] to find the shortest path, equivalent to finding the optimal policy to achieve the largest cumulative rewards. We denote this search-based algorithm as "Optimal". Note that "Optimal" evaluated by the deterministic Floyd's algorithm only relies on the reward function, ignoring the transition randomness. We only employ this baseline algorithm in the first simple MDP environment.

5.1 Continual Q Learning on Simple MDPs

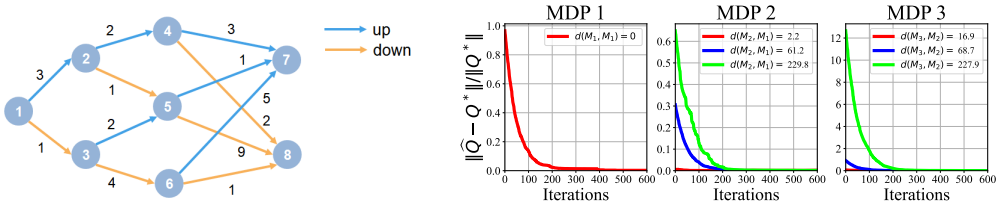

Figure 1: (**Left**) The simple MDP with different reward functions and environment dynamics. **(Right)** Learning curves of the Finetune algorithm on three MDPs, where Q^* is evaluated as the separate Q learning on each MDP. Different colors represent learning curves under MDP 2 with different reward ranges, thus having distinct MDP distances between adjacent MDPs.

Experimental Setup. The 8-state MDP structure is illustrated in Figure 1 (Left), where the action space includes "Up" and "Down". When the agent takes the action "Up" ("Down"), it will reach the upper (lower) state with a certain probability p, typically close to 1. Otherwise, it will reach the converse state in the reversed direction with the probability $1 - p$. We construct three MDPs with distinct reward functions and state transition dynamics. In particular, we randomly select rewards in each edge, ranging from (0, 20), (10, 20), and (0, 30) for these three MDPs, respectively. We set the transition probabilities p in the three MDPs as 0.9, 0.8, and 0.9. To verify the convergence behaviors of Finetune algorithm in Theorem 1 regarding MDP distance, we specify three MDP 2 with different reward ranges, i.e., (10, 20), (30, 45), (80, 100), in the green, blue and red colors in Figure 1 (**Right**).

Results: Adaptivity under different MDP distances. To investigate the adaptivity ability of continual RL, we deployed the Finetune algorithm across three MDPs and plotted the learning curves in Figure 1. As shown in Figure 1 (Right), the Finetune algorithm converges faster in the 2nd and 3rd MDPs when the MDP distance $d_\infty(\mathcal{M}_t, \mathcal{M}_{t-1})$ is smaller, as indicated by the transition from green to blue and red curves. However, with sufficient training, the normalized Q function difference $\|\widehat{Q} - Q^*\|/\|Q^*\|$ tends to 0, indicating that the algorithm asymptotically converges to the MDP-dependent optimal Q function. These results corroborate Theorem 1 and support our intuition that initializing the Q function with an MDP closer to the target MDP, speeds up adaptation.

Average Return	MDP 1	MDP 2	MDP 3	Average Performance
Optimal	49.8 ($\pm$3.7)	63.6 ($\pm$3.2)	161.2 ($\pm$15.5)	91.5 ($\pm$5.7)
V^*	48.5 ($\pm$3.7)	60.9 ($\pm$2.7)	159.4 ($\pm$15.4)	89.6 ($\pm$5.6)
Finetune	38.6 ($\pm$4.6)	55.2 ($\pm$3.4)	159.7 ($\pm$15.3)	84.5 ($\pm$5.5)
MCF	47.8 ($\pm$5.5)	60.2 ($\pm$5.8)	154.0 ($\pm$15.0)	**87.4** ($\pm$5.9)
Ours ($\lambda = 0$)	42.6 ($\pm$5.8)	57.0 ($\pm$4.2)	156.8 ($\pm$15.3)	85.4 ($\pm$5.5)
Ours ($\lambda = 3$)	43.0 ($\pm$6.2)	58.9 ($\pm$3.9)	155.6 ($\pm$16.0)	85.8 ($\pm$5.3)
Ours ($\lambda = 6$)	42.6 ($\pm$5.2)	57.5 ($\pm$5.5)	156.8 ($\pm$15.9)	85.6 ($\pm$5.6)
Ours ($\lambda = 10$)	40.6 ($\pm$4.5)	58.2 ($\pm$4.7)	156.9 ($\pm$15.9)	85.2 ($\pm$5.9)

Table 1: Achieved average return with standard deviations of different continual RL algorithms over 20 runs. Our method with $\lambda = 3$ performs best in this simple MDP.

Results: Algorithm Performance. We run each algorithm 20 times and report the average return for each and among all three MDPs. Table 1 shows that the performance of our algorithm interpolates between the Finetune and MCF, with $\lambda = 6$ yielding the best result. While the Finetune algorithm performs best in the last MDP compared to Ours and MCF, it suffers a significant performance drop when evaluated in the 1st and 2nd MDP. In contrast, MCF effectively reduces catastrophic forgetting and retains consistent performance across three MDPs at the cost of intensively evaluating the true weight function in Eq. 4. We also demonstrate that MCF is more likely to outperform the Finetune algorithm when the MDPs differ dramatically. Precisely, we specify the reward in MDP 3 as $C - \frac{1}{2}R_1 - \frac{1}{2}R_2$, where R_1 and R_2 are the rewards in the 1st and 2nd MDPs and C is a constant generated in the range of (60, 80). Detailed empirical results are provided in Appendix H.

5.2 Continual Q Learning on the Grid World Environment

Experimental Setup. The Grid World environment, as used to evaluate continual RL algorithms in [18], features a stochastic MDP with two-dimensional states, where the same action could lead to

different outcomes and transition to different states. In particular, the environment moves the agent in the intended direction with a certain probability p; with probability $1-p$, the agent moves in a random other direction. We set the width and height as ten and select four MDPs with different transition probabilities p, namely 0.75, 0.8, 0.85, and 0.9. As illustrated in Figure 2 (Left), the agent receives a +100 reward upon encountering the gold and a -100 reward upon encountering a bomb. In the four MDPs, we gradually shift the locations of gold and bomb from the right upper part to the left lower part, thereby altering both the reward functions and the environment transitions.

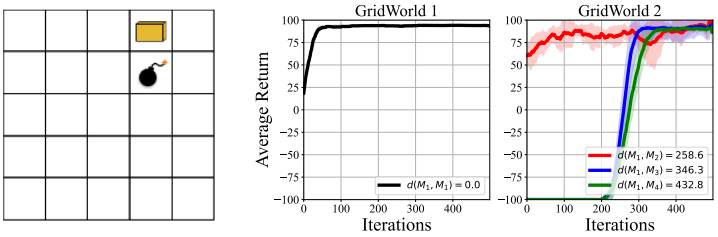

Figure 2: (**Left**) The Grid World Environment. **(Right)** Learning curves of the Finetune algorithm on Grid World under the scenarios with different MDP distances. Results are averaged over ten runs.

Results: Adaptivity under different MDP distances. We begin by demonstrating the algorithm's adaptivity ability presented in Theorem 1 of the Finetune algorithm. As illustrated in Figure 2 (Right), the convergence rate of the Finetune algorithm declines as we increase the MDP distance between the current and previous MDPs. This trend is evident from the red, blue to the green lines, indicating that a greater MDP distance between two MDPs impedes the convergence speed of adaptivity.

Results: Algorithm Performance. The results in Table 2 suggest that our proposed algorithm performs favorably compared with the other baselines, with the best performance observed for $\lambda = 18.0$. Although MCF involves significant computation by interacting with previous MDPs, it performs poorly in this environment. This underperformance is due to the online alternating algorithm's possible divergence issue when deployed in this relatively complex environment, resulting in an inaccurate approximation of the true weight function described in Proposition 2.

Average Return	**MDP 1**	**MDP 2**	**MDP 3**	**MDP 4**	**Average Performance**
V^*	90.7 ($\pm$1.3)	93.2 ($\pm$4.5)	92.3 ($\pm$6.1)	90.7 ($\pm$3.9)	90.7 ($\pm$1.9)
Finetune	-100.0 ($\pm$0.0)	-100.0 ($\pm$0.0)	-92.3 ($\pm$8.9)	26.2 ($\pm$87.0)	-66.5 ($\pm$23.2)
MCF	-93.0 ($\pm$5.6)	-98.0 ($\pm$1.9)	-61.2 ($\pm$17.3)	18.1 ($\pm$67.8)	-58.5 ($\pm$20.7)
Ours ($\lambda = 0.0$)	-99.0 ($\pm$0.6)	37.4 ($\pm$53.7)	-102.8 ($\pm$3.9)	-100.0 ($\pm$0.0)	-66.1 ($\pm$12.9)
Ours ($\lambda = 12.0$)	-70.2 ($\pm$42.3)	42.6 ($\pm$44.0)	-100.0 ($\pm$0.2)	-100.0 ($\pm$0.0)	-56.9 ($\pm$18.4)
Ours ($\lambda = 18.0$)	-48.8 ($\pm$67.3)	84.4 ($\pm$8.6)	-98.0 ($\pm$2.2)	-99.0 ($\pm$0.2)	**-40.6** ($\pm$16.2)

Table 2: Average return of the considered continual RL algorithms over five runs on the Grid World.

6 Conclusion, Limitations and Future Work

In this paper, we study the foundations of continual RL and propose a theoretically principled algorithm by reweighting Bellman targets. We begin by defining the MDP distance and then characterizing catastrophic forgetting in continual RL. Next, we propose a practical continual RL algorithm by reweighing Bellman targets, and extensive experiments in the tabular setting have demonstrated promising results for our algorithm.

Limitations and Future Work. There are still some limitations to our work. First, we focus on continual RL algorithms in the tabular setting by balancing plasticity and stability/catastrophic forgetting. However, we have not explicitly considered the knowledge-transferring effect from the representation of deep RL algorithms across multiple MDPs. Moreover, our continual RL setting requires access to the task boundary and assumes the same state and action spaces across different MDPs. Instead, the research community could put more effort into the task-agnostic case with arbitrarily different state and action spaces across various MDPs, which is more challenging but closer to real-world scenarios. We leave the exploration in these directions as future work.

Acknowledgements

Linglong Kong was partially supported by grants from the Canada CIFAR AI Chairs program, the Alberta Machine Intelligence Institute (AMII), and Natural Sciences and Engineering Council of Canada (NSERC), and the Canada Research Chair program from NSERC. We also thank all the constructive suggestions and comments from the reviewers.

References

[1] David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *Advances in neural information processing systems*, 2023.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[3] Nishanth Anand and Doina Precup. Prediction and control in continual reinforcement learning. *Advances in neural information processing systems*, 2023.

[4] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020.

[5] Johan Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning. *Advances in neural information processing systems (NeurIPS)*, 2021.

[6] Massimo Caccia, Jonas Mueller, Taesup Kim, Laurent Charlin, and Rasool Fakoor. Task-agnostic continual reinforcement learning: In praise of a simple baseline. *arXiv preprint arXiv:2205.14495*, 2022.

[7] Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip Thomas. Optimizing for the future in non-stationary mdps. In *International Conference on Machine Learning*, pages 1414–1425. PMLR, 2020.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *International Conference on Learning Representations*, 2019.

[9] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[10] Andrea Cini, Carlo D'Eramo, Jan Peters, and Cesare Alippi. Deep reinforcement learning with weighted q-learning. *arXiv preprint arXiv:2003.09280*, 2020.

[11] Thang Doan, Mehdi Abbana Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In *International Conference on Artificial Intelligence and Statistics*, pages 1072–1080. PMLR, 2021.

[12] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pages 486–489. PMLR, 2020.

[13] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345–345, 1962.

[14] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[15] Jean-Baptiste Gaya, Thang Doan, Lucas Caccia, Laure Soulier, Ludovic Denoyer, and Roberta Raileanu. Building a subspace of policies for scalable continual learning. 2023.

[16] Peter Henderson, Wei-Di Chang, Florian Shkurti, Johanna Hansen, David Meger, and Gregory Dudek. Benchmark environments for multitask learning in continuous domains. *International Conference on Machine Learning*, 2017.

[17] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[18] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In *International Conference on Machine Learning*, pages 2497–2506. PMLR, 2018.

[19] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *International Conference on Machine Learning*, 2019.

[20] Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Same state, different task: Continual reinforcement learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7143–7151, 2022.

[21] 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, 2022.

[22] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[23] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

[24] Yujun Li. Accelerated value iteration via anderson mixing. *Science China Information Sciences*, 64:1–15, 2021.

[25] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[26] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[27] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

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

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

[30] Emmanouil Antonios Platanios, Abulhair Saparov, and Tom Mitchell. Jelly bean world: A testbed for never-ending learning. *International Conference on Learning Representations (ICLR)*, 2020.

[31] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[32] Ke Sun, Yafei Wang, Yi Liu, Bo Pan, Shangling Jui, Bei Jiang, Linglong Kong, et al. Damped anderson mixing for deep reinforcement learning: Acceleration, convergence, and stabilization. *Advances in Neural Information Processing Systems*, 34:3732–3743, 2021.

[33] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An Introduction*. MIT press, 2018.

[34] Sebastian Thrun. A lifelong learning perspective for mobile robot control. In *Intelligent robots and systems*, pages 201–214. Elsevier, 1995.

[35] Tristan Tomilin, Meng Fang, Yudi Zhang, and Mykola Pechenizkiy. Coom: A game benchmark for continual reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[36] Homer F Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[37] Yi Wan, Ali Rahimi-Kalahroudi, Janarthanan Rajendran, Ida Momennejad, Sarath Chandar, and Harm H Van Seijen. Towards evaluating adaptivity of model-based reinforcement learning methods. In *International Conference on Machine Learning*, pages 22536–22561. PMLR, 2022.

[38] John T Wixted. The psychology and neuroscience of forgetting. *Annu. Rev. Psychol.*, 55:235–269, 2004.

[39] Maciej Wolczyk, Michal Zajac, Razvan Pascanu, Lukasz Kucinski, and Piotr Milos. Disentangling transfer in continual reinforcement learning. *Advances in Neural Information Processing Systems*, 35:6304–6317, 2022.

[40] Maciej Wolczyk, Michał Zajkac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

[41] Ju Xu and Zhanxing Zhu. Reinforced continual learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[42] Yijun Yang, Tianyi Zhou, Jing Jiang, Guodong Long, and Yuhui Shi. Continual task allocation in meta-policy network via sparse prompting. 2023.

Appendix

Table of Contents

A Equivalence to Cosine Distance

For two vectors $\mathbf{x}$ and $\mathbf{y}$, the cosine distance $d_{\cos}$ is defined as

$$d_{\cos}(\mathbf{x}, \mathbf{y}) = 1 - \cos(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}. \tag{11}$$

Although cosine distance is scale-invariant ($d_{\cos}(a\mathbf{x}, b\mathbf{y}) = d_{\cos}(\mathbf{x}, \mathbf{y})$) and can be used as a distance in general, it does not satisfy triangle inequality properties and thus is not a metric in a mathematical sense. In our study, we prefer to use a normalized ℓ_2 distance, which is more commonly used. In addition, the normalized ℓ_2 distance can be shown to be proportional to the cosine distance, therefore satisfying the scale-invariant. For two ℓ_2 normalized vectors $\mathbf{x}$ and $\mathbf{y}$ with $\|\mathbf{x}\|_2 = 1$ and $\|\mathbf{y}\|_2 = 1$, we have

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = (\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) = \mathbf{x}^\top \mathbf{x} - 2\mathbf{x}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} = 2 - 2\cos(\mathbf{x}, \mathbf{y}) = 2d_{\cos}(\mathbf{x}, \mathbf{y}). \tag{12}$$

B More Definitions with Detailed Explanation

B.1 Weighted MDP Distance

In the basic version of MDP distance in Definition 1, each state-action pair possesses the same weight, while we further define a weighted version of d_2 associated with a state-action weight function $w(s, a)$ in Definition 4. We will show later that the catastrophic forgetting between two MDPs is a particular weighted MDP distance.

Definition 4. ***(Weighted MDP Distance)*** *The weighted MDP distance $\bar{d}_w$ is defined as:*

$$\bar{d}_w(\mathcal{M}_1, \mathcal{M}_2) = \|Q_1^* - Q_2^*\|_w = \left(\sum_{s,a} w(s, a) \left(\frac{Q_1^*(s, a)}{\|Q_1^*\|_2} - \frac{Q_2^*(s, a)}{\|Q_2^*\|_2} \right)^2 \right)^{\frac{1}{2}} \tag{13}$$

where we have a state-action weight function $w(s, a)$, with $\sum_{s,a} w(s, a) = 1$ and $w(s, a) \geq 0$ for each s, a.

B.2 Definition of Distribution Drift and Catastrophic Forgetting in Deep Learning

We first introduce the concept of *drift* in the process of learning a parameterized function f from the source data distribution τ_S with the dataset $\mathcal{D}_{\tau_S}$ to the target data distribution τ_T with the dataset $\mathcal{D}_{\tau_T}$. After learning f on the source dataset $\mathcal{D}_{\tau_S}$, we obtain the estimated function $\widehat{f}_{\tau_S}$. Then we apply the same model architecture f on the target dataset $\mathcal{D}_{\tau_T}$ with any learning algorithms, and finally we evaluate the drift of the attained $\widehat{f}_{\tau_T}$ via $\delta^{\tau_S \to \tau_T}$ defined as [11]:

$$\delta^{\tau_S \to \tau_T}\left(X^{\tau_S}\right) = \left(\widehat{f}_{\tau_T}(x) - \widehat{f}_{\tau_S}(x) \right)_{(x,y) \in \mathcal{D}_{\tau_S}} \tag{14}$$

Based on the definition of *drift*, we define the *vanilla catastrophic forgetting* $\Delta^{\tau_S \to \tau_T}$ as

$$\Delta^{\tau_S \to \tau_T}\left(X^{\tau_S}\right) = \|\delta^{\tau_S \to \tau_T}\left(X^{\tau_S}\right)\|_2^2 = \sum_{(x,y) \in \mathcal{D}_{\tau_S}} \left(\widehat{f}_{\tau_T}(x) - \widehat{f}_{\tau_S}(x) \right)^2, \tag{15}$$

where the catastrophic forgetting can be further simplified as $\Delta^{\tau_S \to \tau_T} = \left\| \phi\left(X^{\tau_S}\right) \left(\omega_{\tau_T}^* - \omega_{\tau_S}^*\right) \right\|_2^2$ in the Neural Tangent Kernel (NTK) regime [11, 17], allowing the proposal of new continual learning approaches. In deep learning, minimizing the catastrophic forgetting $\Delta^{\tau_S \to \tau_T}$ is equivalent to minimizing a weighted drift in terms of the prediction function $\widehat{f}$ with the weights determined by the dataset.

B.3 Definitions of Distribution Drift between two MDPs

Before defining the catastrophic forgetting, we first measure the distribution drift between two MDPs for a given RL algorithm in Definition 5 by leveraging the estimated Q functions:

Definition 5. ***(Drift between two MDPs)*** *Consider the continual RL setting across the source and target MDP $\mathcal{M}_{sou}$ and $\mathcal{M}_{tar}$. The drift, denoted as $\delta^{\pi_{tar}}_{sou,tar}(\mathcal{M}_{sou})$, with respect to π_{tar} between two MDPs is defined as:*

$$\delta^{\pi_{tar}}_{sou,tar}(\mathcal{M}_{sou}) = \left(\sum_a \pi_{tar}(a|s) \left(\frac{\widehat{Q}_{sou}(s,a)}{\|\widehat{Q}_{sou}\|_2} - \frac{\widehat{Q}_{tar}(s,a)}{\|\widehat{Q}_{tar}\|_2} \right)^2 \right)_{s\in|S|}. \tag{16}$$

Notably, the drift term $\delta^{\pi_{\text{tar}}}_{\text{sou,tar}}(\mathcal{M}_{\text{sou}})$ in Definition 5 can be interpreted as the state-level difference between estimated Q functions with each element weighted by the target policy π_{tar} that determines the proportion of the action a. Furthermore, suppose we apply a behavior policy to interact within a specified MDP. In that case, the resulting state distribution will be simultaneously determined by the behavior policy and the MDP-specific state transition dynamics. This allows us to define catastrophic forgetting between two MDPs by additionally incorporating the state distribution.

C Proof of Theorem 1

Proof. **Convergence.** Recap the objective function in the k-th phrase of FQI is

$$\widehat{Q}^{k+1}_\theta = \arg\min_{Q_\theta} \frac{1}{n} \sum_{i=1}^{n} \left[y_i - Q_\theta(s_i, a_i) \right]^2, \tag{17}$$

where the target $y_i = r(s_i, a_i) + \gamma \max_{a\in\mathcal{A}} Q^k_{\theta^*}(s'_i, a)$ is fixed within every T_{target} steps to update the target network Q_{θ^*} by letting $\theta^* = \theta$. The periodical updating of the target network Q_{θ^*} in continual RL indicates that

$$Q^k_{\theta^*}(s,a) = \widehat{Q}^k_t(s,a). \tag{18}$$

Since we require $\widehat{Q}^k_t$ has a decreasing regression error, the regression error denoted by $e^k_t(n)$ between it and the expectation target $\mathcal{T}^{\text{opt}} Q^k_{\theta^*}$ converges to 0 as $n \to +\infty$, i.e.,

$$e^k_t(n) = \|\widehat{Q}^{k+1}_t - \mathcal{T}^{\text{opt}} Q^k_{\theta^*}\|_\infty = \|\widehat{Q}^{k+1}_t - Q^{k+1}_t\|_\infty \to 0, \quad \text{as} \quad n \to +\infty, \tag{19}$$

where we denote the expectation target as Q^{k+1}_t. Thus, we have $Q^{k+1}_t = \mathcal{T}^{\text{opt}} Q^k_{\theta^*} = \mathcal{T}^{\text{opt}} \widehat{Q}^k_t(s,a)$. Putting all notations together, we have the following result:

$$\begin{aligned}
\sup_{s,a} \left| \widehat{Q}^k_t(s,a) - Q^*_t(s,a) \right| &\le \sup_{s,a} \left| \widehat{Q}^k_t(s,a) - Q^k_t(s,a) \right| + \sup_{s,a} \left| Q^*_t(s,a) - Q^k_t(s,a) \right| \\
&= e^k_t(n) + \sup_{s,a} \left| \mathcal{T}^{\text{opt}} Q^*_t(s,a) - \mathcal{T}^{\text{opt}} \widehat{Q}^{k-1}_t(s,a) \right| \\
&\le e^k_t(n) + \gamma \sup_{s,a} \left| Q^*_t(s,a) - \widehat{Q}^{k-1}_t(s,a) \right| \\
&= e^k_t(n) + \gamma e^{k-1}_t(n) + \gamma^2 \sup_{s,a} \left| \widehat{Q}^{k-2}_t(s,a) - Q^*_t(s,a) \right| \\
&= \sum_{i=0}^{k-1} \gamma^i e^{k-i}_t(n) + \gamma^k \sup_{s,a} \left| \widehat{Q}^0_t(s,a) - Q^*_t(s,a) \right| \\
&\overset{(a)}{=} \sum_{i=0}^{k-1} \gamma^i e^{k-i}_t(n) + \gamma^k \sup_{s,a} \left| \widehat{Q}_{t-1}(s,a) - Q^*_t(s,a) \right| \\
&\overset{(b)}{\to} \gamma^k \sup_{s,a} \left| \widehat{Q}_{t-1}(s,a) - Q^*_t(s,a) \right| \qquad (n \to +\infty) \\
&\overset{(c)}{\to} 0 \qquad (k \to +\infty)
\end{aligned} \tag{20}$$

where (a) relies on the fact that $Q^1_t = \mathcal{T}^{\text{opt}} \widehat{Q}^0_t$ in the first phase for the t-th MDP, and $\widehat{Q}^0_t = \widehat{Q}_{t-1}$ which is initialized as the final Q estimation $\widehat{Q}_{t-1}$ after traning the $t-1$-th MDP. The limit (b)

leverages the decreasing property of regression error $e_t^k(n) \to 0$ as $n \to +\infty$. The last arrow (c) of proof holds when $k \to +\infty$ as long as $\|\widehat{Q}_{t-1} - Q_t^*\|_\infty$ is bounded. This assumption is also mild, as most environments have bounded Q functions or impose a bounded reward range.

Putting all together, the proof above indicates that $\|\widehat{Q}_t^k - Q_t^*\|_\infty \to 0$ as $n, k \to +\infty$ without the optimization error in each FQI **regardless of the initialization** $\widehat{Q}_{t-1}$ **or** $\widehat{Q}_t^0$. As $\widehat{Q}_1^k \to Q_1^*$ under the above conditions, we can easily prove recursively, showing that $\widehat{Q}_t^k \to Q_t^*$ for each $t = 1, ..., T$ in this Finetune FQI algorithm, i.e., $\widehat{Q}_t = Q_t^*$. As such, we further plug this condition into the last two arrows in the proof above. Therefore, given the k-th phase in the t-th MDP, as $n \to +\infty$, we have the limiting upper bound as follows:

$$\begin{aligned} \sup_{s,a} |\widehat{Q}_t^k(s,a) - Q_t^*(s,a)| &\leq \gamma^k \sup_{s,a} \left| Q_{t-1}^*(s,a) - Q_t^*(s,a) \right| \\ &= \gamma^k d_\infty(\mathcal{M}_{t-1}, \mathcal{M}_t), \end{aligned} \tag{21}$$

where the RHS in terms of the Q function is exactly the MDP distance defined in Definition 1.

Iteration complexity. Let RHS in Eq. 21 be less than ϵ, after taking log transformation, we have:

$$k \log \gamma + \log d_\infty(\mathcal{M}_{t-1}, \mathcal{M}_t) \leq \log \epsilon$$

Finally, we have:

$$k \geq C \log \frac{d_\infty(\mathcal{M}_{t-1}, \mathcal{M}_t)}{\epsilon}, \tag{22}$$

where $C = -\frac{1}{\log \gamma}$. This indicates that the iteration complexity is $\mathcal{O}(\log d_\infty(\mathcal{M}_{t-1}, \mathcal{M}_t)/\epsilon)$ given an ϵ iteration error. In other words, a larger MDP distance between the current and preceding ones would require a larger number of iterations for $\widehat{Q}_t^k$ in order to converge to the MDP-dependent optimal Q_t^* in the t-th MDP.

□

D Proof of Proposition 1

Proof. The catastrophic forgetting CF(Q_T) regarding Q_T is defined as

$$\begin{aligned} \text{CF}(Q_T) &= \sum_{t=1}^{T} \sum_{s,a} \mu_t^{\pi_{\text{CL}}}(s) \pi_{\text{CL}}(a|s) \left(\frac{\widehat{Q}_t(s,a)}{\|\widehat{Q}_t\|} - Q_T(s,a) \right)^2 \\ &= \sum_{t=1}^{T} \sum_{s,a} w_t^{\pi_{\text{CL}}}(s,a) \left(\frac{\widehat{Q}_t(s,a)}{\|\widehat{Q}_t\|} - Q_T(s,a) \right)^2 \end{aligned} \tag{23}$$

where we denote the weight as $w_t^{\pi_{\text{CL}}}(s,a) = \mu_t^{\pi_{\text{CL}}}(s)\pi_{\text{CL}}(a|s)$. Since we aim to find the optimal Q estimator for the whole objective function, the Q function mapping from s, a to the Q is just the inner mapping. We only need to consider the optimality equation for a specific s, a. Although $w_t^{\pi_{\text{CL}}}(s,a)$ is coupled with $Q_T(s,a)$ following the greedy rule in terms of the policy π, when we fix $w_t^{\pi_{\text{CL}}}$ it leads to a bi-level optimization. In particular, thanks to a fixed $w_t^{\pi_{\text{CL}}}$, the objective function above is equivalent to a quadratic function regarding $Q_T(s,a)$. By taking the derivative of CF(Q_T) regarding $Q_T(s,a)$ and then let the gradient equal to zero for each s, a, it arrives at

$$\sum_{t=1}^{T} w_t^{\pi_{\text{CL}}}(s,a) \left(\frac{\widehat{Q}_t(s,a)}{\|\widehat{Q}_t\|} - Q_T(s,a) \right) = 0 \tag{24}$$

Consequently, we have the following equation:

$$Q_T^{\text{opt}}(s,a) = \sum_{t=1}^{T} w_t^{\pi_{\text{CL}}}(s,a) \frac{\widehat{Q}_t(s,a)}{\|\widehat{Q}_t\|} / \sum_{t=1}^{T} w_t^{\pi_{\text{CL}}}(s,a) \quad \forall s, a, \tag{25}$$

where the constraint is the weight $w_t^{\pi_{\text{CL}}}(s,a) = \mu_t^{\pi_{\text{CL}}}(s)\pi_{\text{CL}}(a|s)$ with $\pi(a^*|s) = 1$ if $a^* = \arg\max_{a'} Q_T^{\text{opt}}(s,a')$, otherwise $\pi_{\text{CL}}(a|s) = 0$.

□

E Online Alternating Algorithm to Minimize Catastrophic Forgetting

The algorithm procedure includes two iterative steps. The first is the weight evaluation while fixing the estimated Q function, in which case, the policy is also fixed. The weight evaluation $\widehat{w}_t^{\pi}(s,a)$ typically proceeds via the Monte Carlo method, and a more significant number of simulations leads to a more accurate evaluation at the cost of the more computational cost. The second step is to update the Q function with the evaluated weights $\widehat{w}_t^{\pi}(s,a)$ in step one. This procedure requires online interaction with all MDPs, and hence, it is called an online alternating algorithm. This alternating optimization in Algorithm 1 is similar to the policy/value iteration algorithm that interacts between the Q function and the policy. In our implementation, we select a sufficiently large L to guarantee a favorable convergence of our online alternating algorithm, although an overly large L will increase the computational cost significantly.

In summary, to leverage all optimal Q functions and seek an optimal Q function estimator in continual RL, we must interact with all MDPs. This additional computation is likely helpful in continual learning provided that the weights $\widehat{w}_t^{\pi}(s_i,a_i)$ can be approximated favorably via the online alternating algorithm within the computational budget.

Algorithm 1 Online Alternating Algorithm to Minimize Catastrophic Forgetting

Given the $\{\widehat{Q}_t\}$ for $t=1,..,T$, and initialize $Q^{(0)}$. Set the total training steps K, evaluation step L, and the number of samples N_t for each $t=1,...,T$. Initialize $l=1$.
while $l \leq L$ **do**
  */ * Step 1: Weight Evaluation via π determined by $Q^{(l-1)}$ * /*
  for $t=1$ to T **do**
    Observe the initial state s_0 in the t-th MDP;
    for $i=1$ to N_t **do**
      Select $a_i = \arg\max_a Q^{(l-1)}(s_i,a)$ via the greedy rule.
      Perform the action a_i on t-th MDP, obtain r_i and s_{i+1}.
      Store the transition (s_i,a_i) in the t-th buffer.
    end for
    Estimate w_t^{π} based on samples in the t-th buffer:

$$\widehat{w}_t^{\pi}(s,a) \leftarrow \frac{1}{N_t}\sum_{i=1}^{N_t} \mathbf{1}_{\{s_t=s_i,a_t=a_i\}}$$

  end for
  */ * Step 2: Q Function Updating * /*
  Sample the batch of transitions (s_i,a_i) from all T buffers.
  Update Q Function for each (s_i,a_i) via

$$Q^{(l)}(s_i,a_i) \leftarrow \sum_{t=1}^{T} \widehat{w}_t^{\pi}(s_i,a_i)\widehat{Q}_t(s_i,a_i) / \sum_{t=1}^{T} \widehat{w}_t^{\pi}(s_i,a_i)$$

  $l \leftarrow l+1$
end while

F Discussion with Anderson Mixing

Comparison with Anderson Acceleration. The optimization form in Eq. 7 is similar to the widely used *Anderson Acceleration* [36] technique that can speed up RL algorithms [32, 24]. In Anderson Acceleration, within the FQI framework, weights can be solved by

$$\alpha^*(\widehat{Q}_t^k) = \underset{\alpha}{\operatorname{argmin}} \left\| \sum_{i=1}^{m} \alpha_i \left(\mathcal{T}\widehat{Q}_t^{k+1-i}(s,a) - \widehat{Q}_t^{k+1-i}(s,a) \right) \right\|_2 \tag{26}$$

where anderson acceleration use the m historical values $\left\{\widehat{Q}_t^{k+1-i}\right\}_{i=1}^{m}$ and $\left\{\mathcal{T}\widehat{Q}_t^{k+1-i}\right\}_{i=1}^{m}$, aiming at accelerating the training on the current MDP. It is worth noting that the weighted targets in Anderson acceleration are from the iteration *in the current MDP*. Alternatively, our reweighted method in continual RL uses Q functions *in the previous tasks from different MDPs*. While both Anderson acceleration and our reweighted algorithm have a similar optimization form to solve the weight, they are fundamentally different and in distinct contexts.

G Proof of Proposition 2

Directly considering two $Q_t(s,a)$ to derive the contraction mapping would be difficult as α depends on the $Q_t(s,a)$ and decoupling α and $Q_t(s,a)$ is hard. Instead, we consider the proof of the convergence rate below by following [24]:

$$
\begin{aligned}
&\mathcal{T}_{\text{CL}}^{\text{opt}} Q_t^k(s,a) - Q_t^k(s,a) \\
&= \mathbb{E}\left[R_t(s,a)\right] + \gamma \sum_{s'} \mathcal{P}_{s,s'}^{\pi^k} Q_t^k(s',a') - Q_t^k(s,a) \\
&= \mathbb{E}\left[R_t(s,a)\right] + \gamma \sum_{s'} \mathcal{P}_{s,s'}^{\pi^k} Q_t^k(s',a') - \left(\mathbb{E}\left[R_t(s,a)\right] + \sum_{i=1}^{t} \gamma \sum_{s'} \mathcal{P}_{s,s'}^{\pi^{k-1}} \alpha_i^k Q_i^{k-1}(s',a') \right) \\
&\overset{(a)}{\leq} \gamma \sum_{s'} \mathcal{P}_{s,s'}^{\pi^*} \left(Q_t^k(s',a') - \sum_{i=1}^{t} \alpha_i^k Q_i^{k-1}(s',a') \right) \\
&= \gamma \sum_{s'} \mathcal{P}_{s,s'}^{\pi^*} \left(\mathbb{E}\left[R_t(s',a')\right] + \sum_{i=1}^{t} \gamma \sum_{s''} \mathcal{P}_{s',s''}^{\pi^{k-1}} \alpha_i^k Q_i^{k-1}(s'',a'') - \sum_{i=1}^{t} \alpha_i^k Q_i^{k-1}(s',a') \right) \\
&= \gamma \sum_{s'} \mathcal{P}_{s,s'}^{\pi^*} \left(\sum_{i=1}^{t} \alpha_i^k \left(\mathbb{E}\left[R_t(s',a')\right] + \gamma \sum_{s''} \mathcal{P}_{s',s''}^{\pi^{k-1}} Q_i^{k-1}(s'',a'') - Q_i^{k-1}(s',a') \right) \right) \\
&= \gamma \sum_{s'} \mathcal{P}_{s,s'}^{\pi^*} \left(\sum_{i=1}^{t} \alpha_i^k \left(\mathcal{T}_{\text{CL}}^{\text{opt}} Q_i^{k-1}(s',a') - Q_i^{k-1}(s',a') \right) \right)
\end{aligned}
\tag{27}
$$

where $\mathcal{T}_{\text{CL}}^{\text{opt}} Q^k$ indicates that we apply the operator $\mathcal{T}_{\text{CL}}^{\text{opt}}$ on the vector $Q^k = \{Q_i^k\}_{i=1}^t$ in an column-wise way. (a) relies on the inequality $\max_x f(x) - \max_y g(y) = f(x^*) - g(y^*) \leq f(x^*) - g(x^*) \leq \max_x f(x) - g(x)$ if we assume the difference is positive without loss of generality, and $\sum_i b_i f_i - \sum_j b_j g_j = \sum_i b_i (f_i - g_i)$. $Q^k = [Q_1^k, ..., Q_t^k]$. Consequently, we have

$$
\begin{aligned}
\|\mathcal{T}_{\text{CL}}^{\text{opt}} Q_t^k - Q_t^k\| &\leq \gamma \|\mathcal{P}_{s,s'}^{\pi^*}\| \| \sum_{i=1}^{t} \alpha_i^k \left(\mathcal{T}_{\text{CL}}^{\text{opt}} Q_i^{k-1} - Q_i^{k-1} \right) \| \\
&\leq \gamma \| \sum_{i=1}^{t} \alpha_i^k \left(\mathcal{T}_{\text{CL}}^{\text{opt}} Q_i^{k-1} - Q_i^{k-1} \right) \| \\
&\overset{(b)}{\leq} \gamma \|\mathcal{T}_{\text{CL}}^{\text{opt}} Q_t^{k-1} - Q_t^{k-1}\|,
\end{aligned}
\tag{28}
$$

where (b) relies on the optimization regarding α in Proposition 2:

$$
\alpha^k = \arg\min_{\alpha} \| \sum_{i=1}^{t} \alpha_i \left(\mathcal{T}_{\text{CL}}^{\text{opt}} Q_i^{k-1} - Q_i^{k-1} \right) \|. \tag{29}
$$

The inequality (b) holds because we can use a specific $\alpha^k = [0, ..., 1]^\top$.

H Experiments on MDP

Further, we investigate when MCF is significantly superior to the Finetune algorithm. Instead of randomly assigning rewards in a specific range in the previous setting, we construct a "reverse" reward function setting in the third MDP in contrast to the reward functions in the first and second MDPs, respectively. In particular, given the reward function in the first and second MDP, we set rewards in the third MDP as $C - r_1$, $C - r_2$ and $C - r_1 - r_2$, where C is a pre-specified constant. It is expected that if the reward function in the three MDPs is $C - r_1$, for example, the Finetune algorithm will "overfit" to the last MDP, which is reversed to the reward distribution in the first MDP. Therefore, its performance on the first MDP would be undesirable and worse than that of MCF, which simultaneously considers all MDPs. To make a detailed comparison, we evaluate the difference

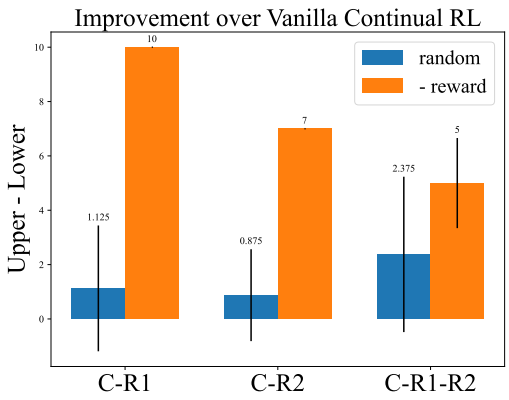

Figure 3: The first two orange bars for $C - r_1$, $C - r_2$ are calculated on MDP 1 and MDP 2, respectively. The third bar for $C - r_1 - r_2$ is averaged over both MDP 1 and MDP 2. The 'Upper - Lower' represents the performance difference between MCF and the Finetune algorithm.

between the Finetune and MCF algorithms on random (rewards are sampled randomly) or reverse reward, e.g., $C - r_1$.

As suggested in Figure 3, when the reward function in the third MDP is the reverse one of the previous MDP, MCF performs much better than the Finetune algorithm as the latter tends to overfit the last MDP, which is dramatically different from previous environments.