Improving Deep Reinforcement Learning by Reducing the Chain Effect of Value and Policy Churn

Hongyao Tang
Mila - Québec AI Institute
Université de Montréal
tang.hongyao@mila.quebec

Glen Berseth
Mila - Québec AI Institute
Université de Montréal
glen.berseth@mila.quebec

Abstract

Deep neural networks provide Reinforcement Learning (RL) powerful function approximators to address large-scale decision-making problems. However, these approximators introduce challenges due to the non-stationary nature of RL training. One source of the challenges in RL is that output predictions can *churn*, leading to uncontrolled changes after each batch update for states not included in the batch. Although such a churn phenomenon exists in each step of network training, how churn occurs and impacts RL remains under-explored. In this work, we start by characterizing churn in a view of Generalized Policy Iteration with function approximation, and we discover a *chain effect of churn* that leads to a cycle where the churns in value estimation and policy improvement compound and bias the learning dynamics throughout the iteration. Further, we concretize the study and focus on the learning issues caused by the chain effect in different settings, including greedy action deviation in value-based methods, trust region violation in proximal policy optimization, and dual bias of policy value in actor-critic methods. We then propose a method to reduce the chain effect across different settings, called Churn Approximated ReductIoN (**CHAIN**), which can be easily plugged into most existing DRL algorithms. Our experiments demonstrate the effectiveness of our method in both reducing churn and improving learning performance across online and offline, value-based and policy-based RL settings, as well as a scaling setting.

1 Introduction

One fundamental recipe for the success of Deep Reinforcement Learning (DRL) is powerful approximation and generalization provided by deep neural networks, which augments the ability of RL with tabular or linear approximation to large state spaces. However, on the other side of this benefit is less control over the function dynamics. Network outputs can change indirectly to unexpected values after any random batch update for input data not included in the batch, called *churn* in this paper. This change is particularly problematic for an RL agent due to its non-stationary nature, which can exacerbate instability, suboptimality, and even collapse. Therefore, it is important to understand and control these undesired dynamics to address learning issues and improve performance.

Consistent efforts have been devoted by the RL community to gain a better understanding of the learning dynamics from different perspectives [Achiam et al., 2019, Kumar et al., 2022, Liu et al., 2023, Lyle et al., 2022b]. Recently, Schaul et al. [2022] studied a novel churn phenomenon in the learning process of typical value-based RL algorithms like DoubleDQN [van Hasselt et al., 2016]. The phenomenon reveals that the greedy actions of about 10% of states in the replay buffer change after a single regular batch update. Such a dramatic churn can persist throughout the learning process of DoubleDQN, causing instabilities.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0490

In this paper, we aim to take a step further to understand how churn occurs and influences learning in different DRL settings beyond value-based RL, as well as to propose a method to control churn. We start by formally characterizing churn in view of Generalized Policy Iteration (GPI) with function approximation to best cover most DRL settings. The impact of churn is two-fold in this view: the churn in policy improvement (called the *policy churn*) changes policy outputs on states that are not directly updated, while the churn in value estimation (called the *value churn*) also changes the action-value landscape, thus altering greedy action and action gradient. We then discover a *chain effect of churn* that exhibits a cycle where the two types of churn compound and bias the learning dynamics throughout the iteration. Further, we move on from the general analysis to concrete DRL settings. We focus on the learning issues caused by the chain effect including greedy action deviation in value-based methods, trust region violation in proximal policy optimization [Schulman et al., 2017] and dual bias of policy value in actor-critic methods. The connection between the chain effect of churn and the issues necessitates an explicit control of the churn.

To this end, we propose a method called Churn Approximated ReductIoN (**CHAIN**) to reduce the chain effect of churn across different DRL settings. The main idea of CHAIN is to reduce the undesirable changes to the outputs of the policy and value networks for states (and actions) outside of the current batch of data for regular DRL training. This reduction is achieved by minimizing the change in target values for a separate batch of data when optimizing the original policy or value learning objective. CHAIN is easy to implement and plug in most existing DRL algorithms with only a few lines of code[1]. In our experiments, we evaluate the efficacy of CHAIN in a range of environments, including MinAtar [Young and Tian, 2019], OpenAI MuJoCo [Brockman et al., 2016], DeepMind Control Suite [Tassa et al., 2018] and D4RL [Fu et al., 2020]. The results show that our method can effectively reduce churn and mitigate the learning issues, thus improving sample efficiency or final performance across online and offline, value-based and policy-based DRL settings. Moreover, our results also show that our method helps to scale DRL agents up and achieves significantly better learning performance when using wider or deeper networks.

The main contributions of this work are summarized as follows: (1) We study how churn occurs and influences learning from the perspective of GPI with function approximation and present the chain effect of churn. (2) We show how churn results in three learning issues in typical DRL settings. (3) We propose a simple and general method and demonstrate its effectiveness in reducing churn and improving learning performance across various DRL settings and environments.

2 Prior Work

In the past decade, a significant effort has been made to understand the learning issues of DRL agents and propose improvements that make DRL more stable and effective. The early stage of this effort studied bias control for value approximation with deep neural networks introducing many improvements after DQN [Mnih et al., 2015] and DDPG [Lillicrap et al., 2015] to address overestimation or underestimation for value-based methods [van Hasselt et al., 2016, Bellemare et al., 2017, Hessel et al., 2018] and deep AC methods [Fujimoto et al., 2018, Haarnoja et al., 2018, Lan et al., 2020, Kuznetsov et al., 2020, Chen et al., 2021] respectively. Additional works dig deeper to diagnose the learning issues regarding instability and generalization, related to the Deadly Triad in DRL [van Hasselt et al., 2018, Achiam et al., 2019], stabilizing effect of target network [Zhang et al., 2021b, Chen et al., 2022, Piché et al., 2022], difficulty of experience replay [Schaul et al., 2016, Kumar et al., 2020, Ostrovski et al., 2021], over-generalization [Ghiassian et al., 2020, Pan et al., 2021, Yang et al., 2022], representations in DRL [Zhang et al., 2021a, Li et al., 2022, Tang et al., 2022], delusional bias [Lu et al., 2018, Su et al., 2020], off-policy correction [Nachum et al., 2019, Zhang et al., 2020, Lee et al., 2021], interference [Cobbe et al., 2021, Raileanu and Fergus, 2021, Bengio et al., 2020] and architecture choices [Ota et al., 2020].

One notable thread is to understand the learning dynamics of DRL agents with a focus on the non-stationary nature of RL. A prominent phenomenon of representation ability loss is studied in [Dabney et al., 2021, Igl et al., 2021, Kumar et al., 2021, 2022, Ma et al., 2023], which reveals how representations become less useful in later stages of learning, leading to myopic convergence. Further, empirical studies in [Nikishin et al., 2022, D'Oro et al., 2023, Sokar et al., 2023, Nauman et al., 2024] demonstrate that the loss of approximation ability becomes severe and leads to collapse when high

[1] `https://github.com/bluecontra/CHAIN`

replay-ratios are adopted for better sample efficiency, while network resets and normalization methods can be simple and effective remedies. This is further identified as plasticity loss in DRL [Lyle et al., 2022a, Abbas et al., 2023, Dohare et al., 2023, Lyle et al., 2024, Xu et al., 2024].

Recently, it has been found that there is a dramatic change in the policy distribution where a large portion of the greedy actions change after each batch update, called *policy churn* Schaul et al. [2022]. Although intuitively related to generalization [Bengio et al., 2020] and interference [Liu et al., 2023], it presents a lack of understanding of churn's effect on the learning behaviors of DRL agents regarding stability, convergence, exploration, etc. Kapturowski et al. [2023] takes the inspiration and proposes a method to robustify the agent's behavior by adding an additional policy head to the value network that fits the ϵ-greedy policy via policy distillation. In this work, we further the study of churn in a more general formal framework, where churn occurs in both value and policy learning. In particular, we focus on the dynamics of churn during the learning process and how it incurs issues in different DRL algorithms and propose a practical method to reduce churn and improve learning performance.

3 Preliminaries

Reinforcement Learning (RL) is formulated within the framework of a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho_0, T \rangle$, defined with the state set $\mathcal{S}$, the action set $\mathcal{A}$, the transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the discounted factor $\gamma \in [0, 1)$, the initial state distribution ρ_0 and the horizon T. The agent interacts with the MDP by performing actions from its policy $a_t \sim \pi(s_t)$ that defines the mapping from states to actions or action distributions. The objective of an RL agent is to optimize its policy to maximize the expected discounted cumulative reward $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$, where $s_0 \sim \rho_0(s_0)$, $s_{t+1} \sim \mathcal{P}(s_{t+1} \mid s_t, a_t)$ and $r_t = \mathcal{R}(s_t, a_t)$. The state-action value function q^π defines the expected cumulative discounted reward for all $s, a \in \mathcal{S} \times \mathcal{A}$ and the policy π, i.e., $q^\pi(s, a) = \mathbb{E}_\pi\big[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, a_0 = a\big]$.

Policy and value functions are approximated with deep neural networks to cope with large and continuous state-action space. Conventionally, q^π can be approximated by Q_θ with parameters θ typically through minimizing Temporal Difference (TD) loss [Sutton and Barto, 1988], i.e., $L(\theta) = \mathbb{E}_{s,a\sim D}\, \delta_\theta(s, a)^2$ where D is a replay buffer and $\delta_\theta(s, a)$ is a type of TD error. A parameterized policy π_ϕ with parameters ϕ can be updated by taking the gradient of the objective, i.e., $\phi' \leftarrow \phi + \alpha \nabla_\phi J(\pi_\phi)$ with a step size α. Value-based methods like Deep Q-Network (DQN) [Mnih et al., 2015] trains a Q-network Q_θ by minimizing $L(\theta)$ where $\delta(s, a) = Q_\theta(s, a) - (r + \gamma \max_{a'} Q_{\theta^-}(s', a'))$ and θ^- denotes the target network. For policy-based methods, TD3 [Fujimoto et al., 2018] is often used to update a deterministic policy with Deterministic Policy Gradient (DPG) theorem [Silver et al., 2014]: $\nabla_\phi J(\pi_\phi) = \mathbb{E}_{s\sim D}\left[\nabla_\phi \pi_\phi(s) \nabla_a Q_\theta(s, a)|_{a=\pi_\phi(s)}\right]$; Soft Actor-Critic (SAC) [Haarnoja et al., 2018] learns a stochastic policy with the gradient: $\nabla_\phi \hat{J}(\pi_\phi) = \mathbb{E}_{s\sim D}\big[\nabla_\phi \log \pi_\phi(a|s) + (\nabla_a \log \pi_\phi(a|s)) - \nabla_a Q_\theta(s, a))\nabla_\phi f_\phi(\epsilon; s))|_{a=f_\phi(\epsilon;s)}\big]$, with noise ϵ and implicit function f_ϕ for re-parameterization.

4 A Chain Effect of Value and Policy Churn

In this section, we present a formal study on value and policy churn and their impact on learning. We first introduce an intuitive overview of how churn is involved in DRL (Section 4.1). Then, we propose the definitions of the value and policy churn (Section 4.2), followed by a chain effect that reveals how the churns interplay and bias parameter update (Section 4.3).

4.1 Generalized Policy Iteration under Churn

Generalized Policy Iteration (GPI) [Sutton and Barto, 1988] is widely used to refer to the general principle of learning in an Evaluation-Improvement iteration manner, which applies to almost all RL methods. In the context of DRL, i.e., with network representation and mini-batch training, the value and policy networks' outputs can have unexpected changes, i.e., the *churn*, after each mini-batch training for the states not included in the batch. Such churns are neglected in most DRL methods, let alone their influence on the practical learning process. In Figure 1, we extend the classic GPI diagram by considering churn to show how it is involved in the learning process intuitively.

In the evaluation process, the parameterized Q-network Q_θ approximates the value of the current policy via repeated mini-batch training. Under the impact of churn, the Q-network is not likely to

 https://doi.org/10.52202/079017-0490

have output predictions the same as what was updated with explicit mini-batch training. For example, let's imagine we have a *virtual* network $\bar{Q}_\theta$ that only accepts the changes for the states updated by mini-batch training directly and remains unchanged for the others.

Thus, the *value churn*, denoted by $\bar{Q}_\theta \rightsquigarrow Q_\theta$, is an implicit process that alters the virtual network $\bar{Q}_\theta$ to the approximation Q_θ we obtained in practice. Similarly, the *policy churn* $\bar{\pi}_\phi \rightsquigarrow \pi_\phi$ occurs in the improvement process. As illustrated in Figure 1, the value churn and the policy churn are interwoven in the Evaluation-Improvement process. Usually, we can assume that the churns make non-negligible changes, i.e., $Q_\theta \neq \bar{Q}_\theta$ and $\pi_\phi \neq \bar{\pi}_\phi$. Therefore, we delve into the cause of churn, its impact on learning, and possible remedies to mitigate its negative impact in the following sections.

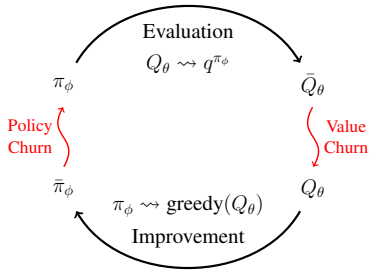


Figure 1: Generalized Policy Iteration (GPI) under the value and policy churn.

4.2 Definition of Value and Policy Churn

A deep neural network can have the form $f_\theta : X \to Y$ with parameters θ. The network is optimized for a set of input data $B_{\text{train}} = \{x_i\}$ with a loss function, leading to a parameter update of $\theta \to \theta'$. Given a reference set of input data $B_{\text{ref}} = \{\bar{x}_i\}$ (where $B_{\text{ref}} \cap B_{\text{train}} = \emptyset$) and a metric d for the output space, the churn is formally defined as:

$$\mathcal{C}_f(\theta, \theta', B_{\text{ref}}) = \frac{1}{|B_{\text{ref}}|} \sum_{\bar{x} \in B_{\text{ref}}} d(f_{\theta'}(\bar{x}), f_\theta(\bar{x})).$$

Arguably, churn is an innate property of neural networks, and it is closely related to problems like interference [Liu et al., 2020, 2023] and catastrophic forgetting [Lan et al., 2023] in different contexts.

In this paper, we focus on the churn in Q-value network Q_θ and policy network π_ϕ. We then obtain the definitions of the Q-value churn ($\mathcal{C}_Q$, w.r.t. $\theta \to \theta'$) and the policy churn ($\mathcal{C}_\pi$, w.r.t. $\phi \to \phi'$, using a deterministic policy for demonstration) for an arbitrary state-action pair $\bar{s}, \bar{a} \in B_{\text{ref}}$ as follows:

$$c_Q(\theta, \theta', \bar{s}, \bar{a}) = Q_{\theta'}(\bar{s}, \bar{a}) - Q_\theta(\bar{s}, \bar{a}),\ c_\pi(\phi, \phi', \bar{s}) = \pi_{\phi'}(\bar{s}) - \pi_\phi(\bar{s}). \tag{1}$$

Then, the definitions regarding B_{ref} can be generalized to the batch setting by aggregating data in B_{ref}: $\mathcal{C}_Q(\theta, \theta', B_{\text{ref}}) = \frac{1}{|B_{\text{ref}}|} \sum_{\bar{s}, \bar{a} \in B_{\text{ref}}} |c_Q(\theta, \theta', \bar{s}, \bar{a})|$, $\mathcal{C}_\pi(\phi, \phi', B_{\text{ref}}) = \frac{1}{|B_{\text{ref}}|} \sum_{\bar{s} \in B_{\text{ref}}} |c_\pi(\phi, \phi', \bar{s})|$. Without loss of generality, we carry out our analysis mainly regarding $\bar{s}, \bar{a}$ for clarity in the following.

(U.I) How the churns $\mathcal{C}_Q, \mathcal{C}_\pi$ are caused by parameter updates First, we look into the relationship between the Q-value churn $\mathcal{C}_Q$, the policy churn $\mathcal{C}_\pi$ and the network parameter updates $\Delta_\theta = \theta' - \theta, \Delta_\phi = \phi' - \phi$. For $\Delta_\theta, \Delta_\phi$, we use typical TD learning and DPG for demonstration: $\Delta_\theta = \frac{\alpha}{|B_{\text{train}}|} \sum_{s,a \in B_{\text{train}}} \nabla_\theta Q_\theta(s,a) \delta_\theta(s,a)$, and $\Delta_\phi = \frac{\alpha}{|B_{\text{train}}|} \sum_{s \in B_{\text{train}}} \nabla_\phi \pi_\phi(s) \nabla_a Q_\theta(s,a)|_{a=\pi_\phi(s)}$.

Now we characterize $\mathcal{C}_Q$ and $\mathcal{C}_\pi$ as functions of $\Delta_\theta, \Delta_\phi$ with the help of Neural Tangent Kernel (NTK) [Achiam et al., 2019]. For clarity, we use $B_{\text{train}} = \{s, a\}$ and $B_{\text{ref}} = \{\bar{s}, \bar{a}\}$ and abbreviate $B_{\text{train}}, B_{\text{ref}}$ and step size α when context is clear. Concretely,

$$\begin{aligned} c_Q(\theta, \theta') &= \nabla_\theta Q_\theta(\bar{s}, \bar{a})^\top \Delta_\theta + O(\|\Delta_\theta\|^2) \approx \underbrace{\nabla_\theta Q_\theta(\bar{s}, \bar{a})^\top \nabla_\theta Q_\theta(s, a)}_{k_\theta(\bar{s}, \bar{a}, s, a)} \delta_\theta(s, a) \\ c_\pi(\phi, \phi') &= \nabla_\phi \pi_\phi(\bar{s})^\top \Delta_\phi + O(\|\Delta_\phi\|^2) \approx \underbrace{\nabla_\phi \pi_\phi(\bar{s})^\top \nabla_\phi \pi_\phi(s)}_{k_\phi(\bar{s}, s)} \nabla_a Q_\theta(s, a)|_{a=\pi_\phi(s)} \end{aligned} \tag{2}$$

Eq. 2 shows that the value and policy churn are mainly determined by the kernels of the Q-network k_θ and the policy network k_ϕ, along with the TD error and the action gradient. This indicates that churn is determined by both the network's property itself and the learning we performed with the network.

4.3 From Single-step Interplay to The Chain Effect of Churn

In addition to the first piece of understanding **(U.I)** that presents how parameter updates cause the churns, we discuss how the churns affect parameter updates backward with two more pieces of understanding **(U.II)** and **(U.III)**, finally shedding light on a chain effect of churn.

(U.II) How $\mathcal{C}_Q, \mathcal{C}_\pi$ deviates action gradient and policy value First, we introduce two types of deviation derived from the value and policy churn: (1) Action Gradient Deviation ($\mathcal{D}^Q_{\nabla_a}$), the change of action gradient regarding the Q-network for states and actions that are affected by the Q-value churn $\mathcal{C}_Q$; (2) Policy Value Deviation ($\mathcal{D}^\pi_Q$), the change of Q-value due to the action change for states that are affected by policy churn $\mathcal{C}_\pi$. Formally, the two types of deviation are:

$$\begin{aligned} d^Q_{\nabla_a}(\theta, \theta', \bar{s}) &= \nabla_{\bar{a}} Q_{\theta'}(\bar{s}, \bar{a})|_{\bar{a}=\pi(\bar{s})} - \nabla_{\bar{a}} Q_\theta(\bar{s}, \bar{a})|_{\bar{a}=\pi(\bar{s})}. \\ d^\pi_Q(\phi, \phi', \bar{s}) &= Q(\bar{s}, \pi_{\phi'}(\bar{s})) - Q(\bar{s}, \pi_\phi(\bar{s})). \end{aligned} \tag{3}$$

One thing to note is the two types of deviation show the interplay between the value and policy churn, as the value churn derives the deviation in policy ($c_Q \xrightarrow{\text{derive}} d^Q_{\nabla_a}$) and the policy churn derives the deviation in value ($c_\pi \xrightarrow{\text{derive}} d^\pi_Q$), as denoted by the superscripts. This interplay between the policy and value can be shown better with the expressions below (derivation details in Appendix B):

$$d^Q_{\nabla_a}(\theta, \theta') = \nabla_{\bar{a}} c_Q(\theta, \theta')|_{\bar{a}=\pi(\bar{s})}, \quad d^\pi_Q(\phi, \phi') \approx (\nabla_{\bar{a}} Q_\theta(\bar{s}, \bar{a})|_{\bar{a}=\pi_\phi(\bar{s})})^\top c_\pi(\phi, \phi') \tag{4}$$

Since the action gradient and policy value play key roles in parameter updates, the deviations in them naturally incur negative impacts on learning.

The discussion thus far is within a single parameter update. Now, we discuss the implications of these single updates towards a chain of updates to shed light on the long-term effect of churn.

(U.III) How parameter updates are biased by $\mathcal{C}_Q, \mathcal{C}_\pi$ and the deviations $\mathcal{D}^Q_{\nabla_a}, \mathcal{D}^\pi_Q$ Let us consider a segment of two consecutive updates, denoted by $(\theta^-, \phi^-) \to (\theta, \phi) \to (\theta', \phi')$. The churns occurred during the *last update* $(\theta^-, \phi^-) \to (\theta, \phi)$ participate in the *current update* $(\theta, \phi) \to (\theta', \phi')$ about to perform. Concretely, the churns affect the following aspects: (1) Q-value estimate, (2) action selection in both TD error and policy objective, and (3) the gradient of network parameters.

From these aspects, we can deduce the difference between the parameter updates under the impact of the value and policy churn (denoted by $\tilde{\Delta}_\theta, \tilde{\Delta}_\phi$) and the conventional ones $\Delta_\theta, \Delta_\phi$. As a result, we can find that the value and policy churn, as well as the deviations derived, introduce biases in the parameter updates. We provide the complete discussion and derivation in Appendix B.2.

The analysis on the update segment $(\theta^-, \phi^-) \to (\theta, \phi) \to (\theta', \phi')$ can be forwarded, and taking the three pieces of understanding together, we arrive at the chain effect of churn.

> **The Chain Effect of Churn:** **(U.I)** Parameter updates cause the value and policy churn, **(U.II)** which further leads to the deviations in the action gradient and policy value; **(U.III)** the churns and the deviations then bias following parameter updates.

As the cycle illustrated in Figure 2, the value and policy churn and the parameter update bias *accumulate* and can *amplify each other* throughout the learning process. Intuitively, the parameter update chain could derail and fluctuate under the accumulating churns and biases, thus preventing stable and effective learning. We concretize our study on the consequences in the next section.

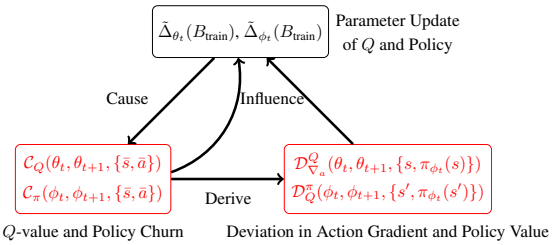


Figure 2: Illustration of the logical cycle of the *chain effect* of the value and policy churn.

5 Reducing Value and Policy Churn in Deep RL

In this section, we show concrete learning issues caused by churn in typical DRL scenarios (Section 5.1), followed by a simple plug-in method to reduce churn and address the issues (Section 5.2).

5.1 Consequences of the Chain Effect of Churn in Different DRL Scenarios

Since churn is involved in most DRL methods, as illustrated by Figure 1, our study focuses on several typical DRL scenarios below.

https://doi.org/10.52202/079017-0490

Greedy action deviation in value-based methods Value-based methods like DQN train a Q-network Q_θ and compute the policy by choosing the greedy action of Q_θ. A consequence of computing the action greedily is that changes in the values will directly cause changes in the action distribution [Schaul et al., 2022]. Similarly to Eq. 4, this deviation can be formalized as: $\mathcal{D}^{Q}_{a^*}(\theta, \theta', B_{\text{ref}}) = \frac{1}{|B_{\text{ref}}|} \sum_{\bar{s} \in B_{\text{ref}}} \mathbb{I}_{\mathcal{A} \setminus \{\arg\max_a Q_\theta(\bar{s}, a)\}} \big(\arg\max_{\bar{a}} Q_{\theta'}(\bar{s}, \bar{a}) \big)$. We suspect that this deviation introduces instability and hinders learning, and we focus on whether reducing churn can improve the performance of value-based methods.

Trust region violation in policy gradient methods Trust region plays a critical role in many policy gradient methods for reliable and efficient policy updates. Proximal Policy Optimization (PPO) [Schulman et al., 2017] uses a clipping mechanism as a simple but effective surrogate of the trust region for TRPO [Schulman et al., 2015]: $\texttt{Clip}\big(r(\phi_{\text{old}}, \phi), 1-\epsilon, 1+\epsilon\big)$ and $r(\phi_{\text{old}}, \phi) = \frac{\pi_\phi(a|s)}{\pi_{\phi_{\text{old}}}(a|s)}$. With respect to policy churn, even though the PPO policy conforms to the trust region for the states in the current training batch, it could silently violate the trust region for other states, including previously updated ones. Consider the policy update $\phi \to \phi'$, it is highly likely to have $r(\phi, \phi') = \frac{\pi_{\phi'}(\bar{a}|\bar{s})}{\pi_\phi(\bar{a}|\bar{s})} \neq 1$ and thus $r(\phi_{\text{old}}, \phi') = \frac{\pi_{\phi'}(\bar{a}|\bar{s})}{\pi_{\phi_{\text{old}}}(\bar{a}|\bar{s})} = r(\phi_{\text{old}}, \phi) r(\phi, \phi') \neq r(\phi_{\text{old}}, \phi)$. Since we have no information about $r(\phi, \phi')$, there is no guarantee for the trust region $1-\epsilon \leq r(\phi_{\text{old}}, \phi') \leq 1+\epsilon$ to be respected after churn. Intuitively, this implicit violation is hazardous and detrimental to learning.

Dual bias of policy value in Actor-Critic methods Deep AC methods interleave the training between the actor-network and the critic-network. Unlike the two scenarios above, where either the value churn or the policy churn raises a learning stability issue, we present the dual bias of policy value that stems from the bilateral effect of churn. The dual bias exists in the policy value as $Q_{\theta'}(\bar{s}, \pi_{\phi'}(\bar{s})) \neq Q_\theta(\bar{s}, \pi_\phi(\bar{s}))$. In the context of AC methods, the policy value is used for the target computation of the critic $r_t + \gamma Q_{\theta'}(s_{t+1}, \pi_{\phi'}(s_{t+1}))$ and the optimization objective of the actor $\nabla_\phi Q_{\theta'}(s, \pi_{\phi'}(s))$. Thus, the dual bias steers the training of the actor and the critic.

Given these negative consequences of churn, a question is raised naturally: how can we control the level of churn to mitigate the issues without introducing complex trust regions or constraints?

5.2 A Regularization Method for Churn Reduction

In this section, we propose a regularization method to reduce value and policy churn, called Churn Approximated ReductIoN (**CHAIN**). To combat the prevalence of churn's negative influence on DRL, our method should be simple to implement and easy to use with different RL methods.

Based on the definitions of the value churn ($\mathcal{C}_Q$) and the policy churn ($\mathcal{C}_\pi$) in Section 4.2, we propose two corresponding loss functions L_{QC} and L_{PC} for churn reduction. Formally, for parameterized networks $Q_{\theta_t}, \pi_{\phi_t}$ at time t and a reference batch B_{ref} sampled from replay buffer, we have:

$$L_{\text{QC}}(\theta_t, B_{\text{ref}}) = \frac{1}{|B_{\text{ref}}|} \sum_{\bar{s}, \bar{a} \in B_{\text{ref}}} \big(Q_{\theta_t}(\bar{s}, \bar{a}) - Q_{\theta_{t-1}}(\bar{s}, \bar{a}) \big)^2 \tag{5}$$

$$L_{\text{PC}}(\phi_t, B_{\text{ref}}) = \frac{1}{|B_{\text{ref}}|} \sum_{\bar{s} \in B_{\text{ref}}} d_\pi(\pi_{\phi_t}(\bar{s}), \pi_{\phi_{t-1}}(\bar{s})) \tag{6}$$

where d_π is a policy distance metric, and we use mean square error or KL divergence for deterministic or stochastic policies. Ideally, the regularization should be imposed on the post-update network parameters θ_{t+1}, ϕ_{t+1}. Since they are not available at time t, we regularize θ_t, ϕ_t and use θ_{t-1}, ϕ_{t-1} as the targets for a convenient and effective surrogate.

By minimizing L_{QC} and L_{PC}, we can reduce the value and policy churn and suppress the chain effect further. This allows us to use the churn reduction regularization terms along with standard RL objectives and arrives at DRL with CHAIN finally:

$$\text{minimize}_\theta \; L(\theta_t, B_{\text{train}}) + \lambda_Q L_{\text{QC}}(\theta_t, B_{\text{ref}}) \tag{7}$$

$$\text{maximize}_\phi \; J(\phi_t, B_{\text{train}}) - \lambda_\pi L_{\text{PC}}(\phi_t, B_{\text{ref}}) \tag{8}$$

where $B_{\text{train}}, B_{\text{ref}}$ are two separate batches randomly sampled from D, and λ_Q, λ_π are coefficients that control the degree of regularization. CHAIN serves as a plug-in component that can be implemented with only a few lines of code modification in most DRL methods. The pseudocode is omitted here, and we refer the readers to Algorithm 1 in the Appendix.

Automatic adjustment of λ_Q, λ_π To alleviate the difficulty of manually selecting the regularization coefficients, we add a simple but effective method to adjust λ_Q, λ_π adaptively during the learning process. The key principle behind this is to keep a consistent relative scale (denoted by β) between the churn reduction regularization terms and the original DRL objectives. More precisely, by maintaining the running means of the absolute Q loss $|\bar{L}_Q|$ and the VCR term $|\bar{L}_{\text{QC}}|$, λ_Q is computed dynamically as $\lambda_Q = \beta \frac{|\bar{L}_Q|}{|\bar{L}_{\text{QC}}|}$, which is similar for λ_π. This is inspired by our empirical observations and the recent study on addressing the reward scale difference across different domains [Hafner et al., 2023].

Another thing worth noting is that CHAIN helps to mitigate the loss of plasticity via churn reduction. This connection can be established by referring to the NTK expressions in Eq. 2: reducing churn encourages k_θ, k_ϕ to 0 and thus prevents the empirical NTK matrix from being low-rank, which is shown to be a consistent indicator of plasticity loss [Lyle et al., 2024].

6 Experiments

In the experiments, we aim to answer the following questions: (1) How large is the value and policy churn in practice, and can our method effectively reduce churn? (2) Does our method's reduction of churn address learning issues and improve performance in terms of efficiency and episode return? (3) Does CHAIN also improve the scaling abilities of deep RL?

We organize our experiments into the four subsections below that correspond to the three DRL scenarios discussed in Section 5.1 as well as a DRL scaling setting. Our experiments include 20 online RL tasks from MinAtar, MuJoCo, DMC, and 8 offline RL datasets from D4RL, as well as 6 popular algorithms, i.e., DoubleDQN, PPO, TD3, SAC, IQL, AWAC. We provide the experimental details in Appendix C and more results in Appendix D.

6.1 Results for CHAIN DoubleDQN in MinAtar

We use DoubleDQN (DDQN) [van Hasselt et al., 2016] as the value-based method and MinAtar [Young and Tian, 2019] as the experiment environments. MinAtar is an Atari-inspired testbed for convenient evaluation and reproduction. We build our DoubleDQN based on the official MinAtar code with no change to the network structure and hyperparameters. We implement CHAIN DDQN by adding a few lines of code to apply the value churn reduction regularization in the standard training of the Q-network (Eq. 7). For CHAIN DDQN, λ_Q is set to 50 for Breakout and 100 for the other tasks. For CHAIN DDQN with automatic adjustment of λ_Q (denoted by the suffix **'Auto'**), the target relative loss scale β is set to 0.05 for all the tasks.

First, to answer Question (1), we report the value churn and the greedy action deviation of DDQN in Figure 3. Each point means the average metric across randomly sampled states and the whole learning process. As expected, we can observe that the value churn accumulates as more training updates take place, leading to the growth of greedy action deviation. With CHAIN, the churn and deviation are reduced significantly. We refer the readers to Figure 9 for more statistics on the value churn.

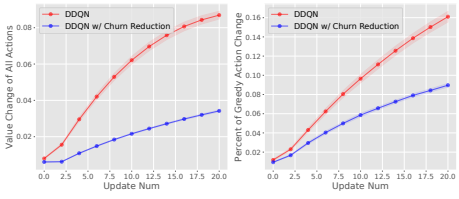


Figure 3: The value churn (*left*) and the greedy action deviation percentage (*right*) in Breakout w/ and w/o CHAIN.

Further, we show the learning curves of CHAIN DDQN regarding episode return in Figure 4. We can see that CHAIN consistently achieves clear improvements over DDQN in terms of both sample efficiency and final scores, especially for Asterix and Seaquest. Moreover, CHAIN (Auto) matches or surpasses the results achieved by manual coefficients, which supports Question (2) positively. In the next subsection, we evaluate how much CHAIN can improve policy gradient-based RL algorithms.

6.2 Results for CHAIN PPO in MuJoCo and DMC

Corresponding to the second DRL scenario discussed in Section 5.1, we focus on the policy churn in Proximal Policy Optimization (PPO) [Schulman et al., 2017] and try to answer the first three questions for policy gradient-based RL algorithms. We build the experiments on the public implementation of PPO from `CleanRL` [Huang et al., 2022] and use the continuous control tasks in MuJoCo and

https://doi.org/10.52202/079017-0490

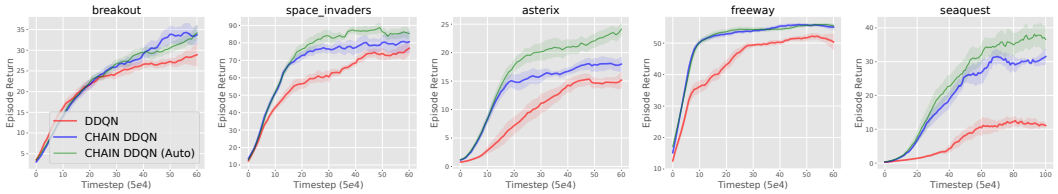


Figure 4: The evaluation of CHAIN DoubleDQN in MinAtar regarding episode return. Curves and shades denote means and standard errors over six random seeds.

DeepMind Control (DMC) as the environments for evaluation. Following the same principle, we implement CHAIN PPO by adding the policy churn reduction regularization to the standard PPO policy training (Eq. 6), with no other modification to the public PPO implementation.

First, to understand the level of churn, we compare PPO and CHAIN PPO with different choices of λ_π in terms of policy churn and episode return. In summary, we observed that PPO also exhibits clear policy churn, and CHAIN significantly reduces it throughout learning, which answers Question (1). Figure 10 shows the details for this analysis. Note that more policy churn makes it more likely to violate the trust region as $\mathcal{C}_\pi \propto r(\phi, \phi')$ discussed in Section 5.1. In turn, we also observed CHAIN PPO consistently outperforms PPO in Ant-v4 and HalfCheetah-v4 across different choices of λ_π.

Further, we aim to answer Question (2) and evaluate whether CHAIN can improve the learning performance of PPO in terms of episode return. For CHAIN PPO (Auto), we set the target relative loss scale β to 0.1 for MuJoCo tasks and 0.02 for DMC tasks. The results for MuJoCo and DMC tasks are reported in Figure 5. The results show that CHAIN PPO outperforms PPO in most cases with higher sample efficiency and final episode return, often significantly. We believe that our results reveal a promising direction to improve more trust-region-based and constraint-based methods in DRL by addressing the issues caused by churn.

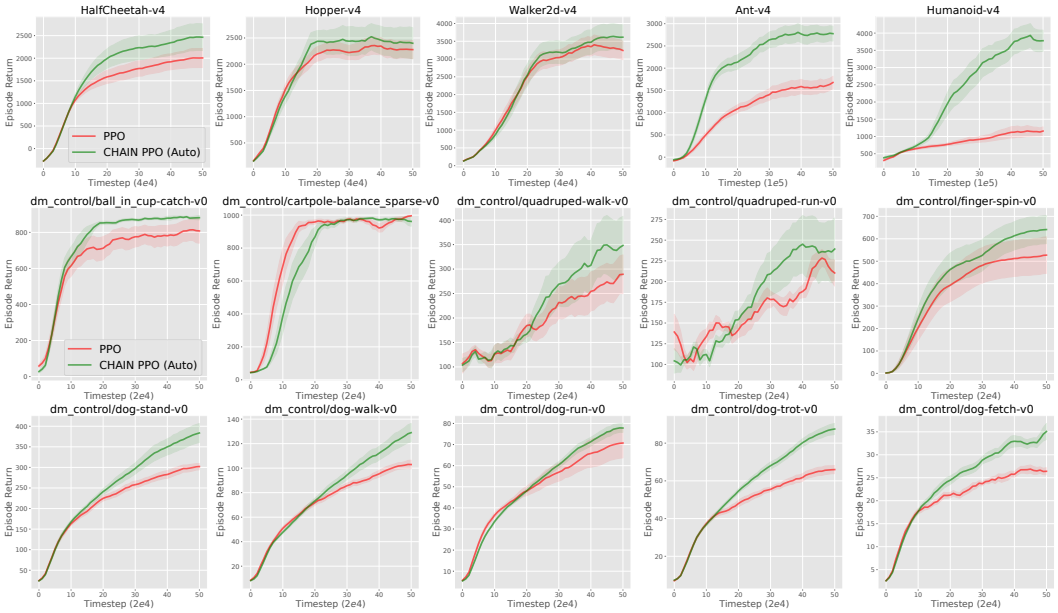


Figure 5: The evaluation of CHAIN PPO in MuJoCo and DeepMind Control (DMC) tasks regarding episode return. Curves and shades denote means and standard errors over twelve random seeds.

6.3 Results for Deep Actor-Critic Methods with CHAIN in MuJoCo and D4RL

Next, we continue our empirical study and evaluate our method for deep actor-critic (AC) methods. We separate our study into online and offline settings, as presented below.

Online AC methods We use TD3 [Fujimoto et al., 2018] and SAC [Haarnoja et al., 2018] and MuJoCo environments based on the public implementation of TD3 and SAC from `CleanRL`. Since AC methods are bilaterally affected by churn, we consider two variants of CHAIN, either of which only applies the value churn reduction (VCR) or the policy churn reduction (PCR).

For Question (1), we again find that both TD3 and SAC exhibit value and policy churn in all environments, and CHAIN-VCR and CHAIN-PCR effectively reduce them respectively in Figure 11 and 12. For Question (2), Figure 6 shows the evaluation results regarding episode return. We can see that CHAIN-PCR often improves the learning performance, especially for Ant-v4; in contrast, CHAIN-VCR improves slightly. We hypothesize that this is because the policy interacts with the environment directly, and the target critic-network also helps to reduce the value churn due to its delayed synchronization with the online critic.

Due to the limitation of space, we refer the readers to Appendix D.3 for more results on churn reduction, the influence of different choices of λ_Q, λ_π, the results of combining VCR and PCR, as well as the effect of auto-adjustment of the regularization coefficient for TD3 and SAC.

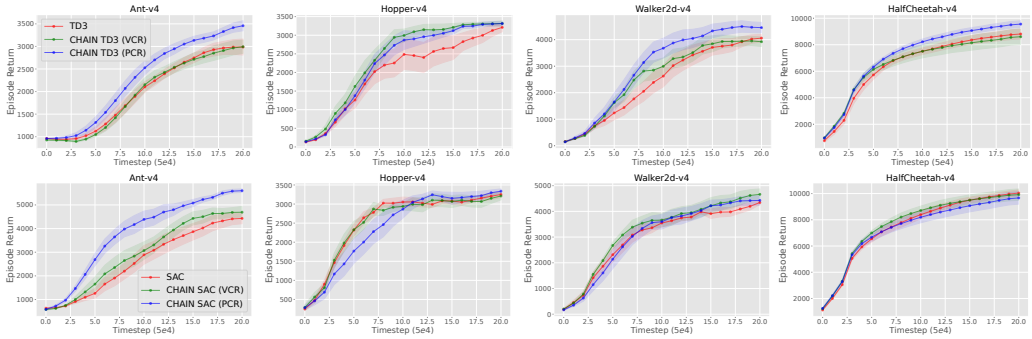


Figure 6: The evaluation of CHAIN TD3 and CHAIN SAC in MuJoCo regarding episode return.

Offline AC methods In Offline RL, a policy is trained over a fixed dataset. We investigate if reducing churn can also improve the convergence of Offline RL. We use IQL [Kostrikov et al., 2022] with D4RL Antmaze dataset [Fu et al., 2020] and AWAC [Nair et al., 2020] with Adroit for our demonstration due to their popularity and good performance in corresponding tasks. Concretely, we use the public implementation and benchmark scores for IQL and AWAC from `CORL`[2]. To apply CHAIN to IQL and AWAC, we implement the regularization for value churn reduction (VCR, Eq. 7) and policy churn reduction (PCR, Eq. 8) separately by adding a couple of lines of code without any other modification. We use $\lambda_\pi = 1e3$ for both CHAIN IQL (PCR) and CHAIN AWAC (PCR); $\lambda_Q = 0.01$ for CHAIN IQL (VCR) and 0.1 for CHAIN AWAC (VCR) across different tasks. The results are summarized in Table 1 and 2.

Table 1: Results for CHAIN IQL in Antmaze, with means and standard errors over twelve seeds.

Task	IQL	CHAIN IQL (PCR)	CHAIN IQL (VCR)
AM-umaze-v2	77.00 ± 5.52	$\mathbf{84.44 \pm 3.19}$	83.33 ± 2.72
AM-umaze-diverse-v2	54.25 ± 5.54	62.50 ± 3.75	$\mathbf{71.67 \pm 7.23}$
AM-medium-play-v2	65.75 ± 11.71	$\mathbf{72.50 \pm 2.92}$	70.00 ± 3.33
AM-medium-diverse-v2	73.75 ± 5.45	$\mathbf{76.67 \pm 4.51}$	66.67 ± 3.79
AM-large-play-v2	42.00 ± 4.53	$\mathbf{50.00 \pm 4.56}$	43.33 ± 4.14
AM-large-diverse-v2	30.25 ± 3.63	26.67 ± 3.96	$\mathbf{31.67 \pm 2.31}$

Table 2: Results for CHAIN AWAC in Adroit, with means and standard errors over twelve seeds.

Task	AWAC	CHAIN AWAC (PCR)	CHAIN AWAC (VCR)
pen-human-v1	81.12 ± 13.47	$\mathbf{99.72 \pm 2.04}$	97.37 ± 3.51
pen-cloned-v1	89.56 ± 15.57	95.49 ± 2.34	$\mathbf{96.66 \pm 2.54}$

We observe that both CHAIN PCR and CHAIN VCR improve the scores for IQL and AWAC in most Antmaze and Adroit tasks. We hypothesize that CHAIN suppresses churn in the training of value and policy networks, thus reducing the bias caused by churn in parameter updates. One thing here that differs from TD3 and SAC considered in the online setting is that the policy network of IQL has no impact on the training of the value networks since the value networks (i.e., Q and V) are trained purely based on in-sample data without accessing $a' = \pi_\phi(s')$. Thus, although IQL does not exhibit a chain effect explicitly, the policy and value networks of IQL still have churns, which are reduced by CHAIN in this case. We provide a further empirical study in Appendix D.4.

[2] `https://github.com/tinkoff-ai/CORL`

https://doi.org/10.52202/079017-0490

6.4 Scaling DRL Agents with CHAIN

It is widely known that scaling DRL agents up is challenging. Naively scaling DRL agents by widening or deepening the conventional MLP networks straightforwardly often fails and could even lead to collapse. Here, we investigate the relationship between churn and scale for DRL agents, as well as the effect of CHAIN on boosting scaling performance, to answer Question (3). We take PPO and MuJoCo tasks as the exemplary setting and scale up both the policy and value networks by a scale-up ratio within $\{2, 4, 8, 16\}$ via *widening* or *deepening*. Note that the default network architecture (i.e., when the scale-up ratio equals one) for both the policy and value networks is a two-layer MLP with 256 neurons for each layer, followed by an output layer.

As expected, we observed that the performance of PPO degraded severely as the scale-up ratio increased, as shown by the solid gray lines in Figure 7. Inspired by the prior study [Obando-Ceron et al., 2024], we found using a decreased learning rate as `lr / sqrt(scale-up ratio)` alleviates the degradation of PPO scaling to some degree (shown by the solid red lines). We then use the learning rate setting below by default. From the perspective of churn, we also observed that scaling PPO escalates the scale of the policy churn in PPO. More results can be found in Appendix D.5. Therefore, we then evaluate the effect of CHAIN in this scaling setting. The results are shown in Figure 7 with dashed lines. By comparing the lines in the same color, we found that CHAIN improves the learning performance of PPO across almost all scale-up ratios and the two learning rate settings.

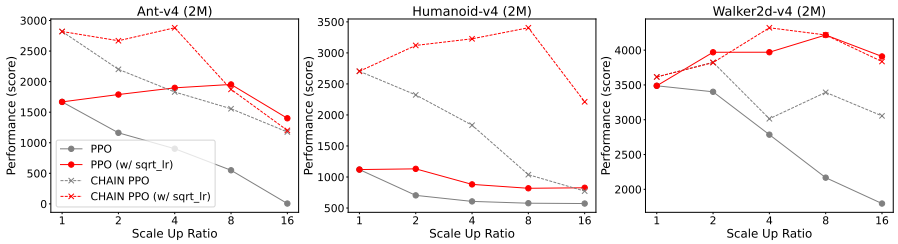


Figure 7: The results regarding episode return for scaling PPO via widening. CHAIN helps to scale almost across all the configurations. Similar results can be found for widening scaling in Figure 20.

In addition, we extend the training horizon from 2M to 10M for Ant, Humanoid, and Walker2d. The results are reported in Table 3. For both widening or deepening cases, CHAIN helps to scale up PPO and achieves clear improvement in terms of episode return. Comparatively, scaling by widening slightly outperforms deepening, which echoes the observation in [Ota et al., 2020] to some extent.

Table 3: Scaling PPO with CHAIN. Means and standard errors of final episode return over six seeds.

Alg. (scale)	Ant (10M)	Human. (10M)	Walker2d (10M)
PPO	2238.45 ± 256.07	1620.45 ± 212.10	3316.77 ± 269.42
PPO (4x wider)	3013.95 ± 223.77	2998.18 ± 237.95	3795.05 ± 208.17
CHAIN PPO (4x wider)	**4916.66** ± 109.01	**4830.58** ± 231.42	**4668.16** ± 234.45
PPO (4x deeper)	2777.62 ± 136.78	3489.47 ± 166.28	2845.68 ± 260.02
CHAIN PPO (4x deeper)	**3760.46** ± 158.29	**4090.80** ± 187.01	**3242.83** ± 173.54
PPO (8x wider)	4235.63 ± 209.68	3198.01 ± 344.71	4335.17 ± 123.12
CHAIN PPO (8x wider)	**5592.94** ± 205.23	**5246.15** ± 130.73	**5161.42** ± 343.40
PPO (8x deeper)	3364.46 ± 173.25	2780.67 ± 304.10	3057.89 ± 299.61
CHAIN PPO (8x deeper)	**4278.76** ± 122.61	**4434.48** ± 144.92	**3324.29** ± 255.88

Our results indicate that uncontrolled churn could be a possible reason for the scaling issue of DRL agents, and CHAIN improves scaling by reducing churn effectively. Though appealing, CHAIN does not fully address the scaling issue per se, and achieves sub-linear scaling on DRL agents.

7 Conclusion

In this paper, we conduct a formal study of churn in a general view and present the chain effect of value and policy churn. The chain effect indicates a compounding cycle, which biases parameter updates throughout learning. We propose an easy-to-implement method for value and policy churn reduction. Our experimental results demonstrate the effectiveness of our method in reducing churn and improving learning performance over a range of DRL environments and algorithms.

https://doi.org/10.52202/079017-0490

Acknowledgements

We want to acknowledge funding support from NSERC, FQRNT and CIFAR and compute support from Digital Research Alliance of Canada, Mila IDT, and NVidia.

References

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning. arXiv preprint, arXiv:2303.07507, 2023.

J. Achiam, E. Knight, and P. Abbeel. Towards characterizing divergence in deep q-learning. arXiv preprint, arXiv:1903.08894, 2019.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. NeurIPS, 2021.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In ICML, 2017.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In ICML, 2020.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. arXiv preprint, arXiv:1606.01540, 2016.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In ICLR, 2021.

Zaiwei Chen, John-Paul Clarke, and Siva Theja Maguluri. Target network and truncation overcome the deadly triad in q-learning. arXiv preprint, arXiv:2203.02628, 2022.

Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In ICML, 2021.

W. Dabney, A. Barreto, M. Rowland, R. Dadashi, J. Quan, M. G. Bellemare, and D. Silver. The value-improvement path: Towards better representations for reinforcement learning. In AAAI, 2021.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Parash Rahman, Richard S. Sutton, and A. Rupam Mahmood. Maintaining plasticity in deep continual learning. arXiv preprint, arXiv:2306.13812, 2023.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron C. Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In ICLR, 2023.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. arXiv preprint, arXiv:2004.07219, 2020.

S. Fujimoto, H. v. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In ICML, 2018.

Sina Ghiassian, Banafsheh Rafiee, Yat Long Lo, and Adam White. Improving performance in reinforcement learning by breaking generalization in neural networks. In AAMAS, 2020.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In ICML, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. arXiv preprint, arXiv:2301.04104, 2023.

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

 https://doi.org/10.52202/079017-0490

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

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In ICLR, 2021.

Steven Kapturowski, Victor Campos, Ray Jiang, Nemanja Rakicevic, Hado van Hasselt, Charles Blundell, and Adrià Puigdomènech Badia. Human-level atari 200x faster. In ICLR, 2023.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In ICLR, 2022.

A. Kumar, R. Agarwal, D. Ghosh, and S. Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In ICLR, 2021.

A. Kumar, R. Agarwal, T. Ma, A. Courville, G. Tucker, and S. Levine. DR3: value-based deep reinforcement learning requires explicit regularization. In ICLR, 2022.

Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. In NeurIPS, 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry P. Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In ICML, 2020.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In ICLR, 2020.

Qingfeng Lan, Yangchen Pan, Jun Luo, and A. Rupam Mahmood. Memory-efficient reinforcement learning with value-based knowledge consolidation. Transaction on Machine Learning Research, 2023.

Jongmin Lee, Wonseok Jeon, Byung-Jun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In ICML, 2021.

Boyan Li, Hongyao Tang, Yan Zheng, Jianye Hao, Pengyi Li, Zhen Wang, Zhaopeng Meng, and Li Wang. Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. In ICLR, 2022.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In ICLR, 2015.

Vincent Liu, Adam White, Hengshuai Yao, and Martha White. Towards a practical measure of interference for reinforcement learning. arXiv preprint, arXiv:2007.03807, 2020.

Vincent Liu, Han Wang, Ruo Yu Tao, Khurram Javed, Adam White, and Martha White. Measuring and mitigating interference in reinforcement learning. In ICML, 2023.

Tyler Lu, Dale Schuurmans, and Craig Boutilier. Non-delusional q-learning and value-iteration. In NeurIPS, 2018.

C. Lyle, M. Rowland, and W. Dabney. Understanding and preventing capacity loss in reinforcement learning. In ICLR, 2022a.

Clare Lyle, Mark Rowland, Will Dabney, Marta Kwiatkowska, and Yarin Gal. Learning dynamics and generalization in deep reinforcement learning. In ICML, 2022b.

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

Y. Ma, H. Tang, D. Li, and Z. Meng. Reining generalization in offline reinforcement learning via representation distinction. In NeurIPS, 2023.

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

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. arXiv preprint, arXiv:1912.02074, 2019.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. arXiv preprint, arXiv:2006.09359, 2020.

Michal Nauman, Michal Bortkiewicz, Mateusz Ostaszewski, Piotr Milos, Tomasz Trzcinski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. arXiv preprint, arXiv:2403.00514, 2024.

E. Nikishin, M. Schwarzer, P. D'Oro, P. Bacon, and A. C. Courville. The primacy bias in deep reinforcement learning. In ICML, Proceedings of Machine Learning Research, 2022.

Johan S. Obando-Ceron, Aaron C. Courville, and Pablo Samuel Castro. In deep reinforcement learning, a pruned network is a good network. arXiv preprint, arXiv:2402.12479, 2024.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. In NeurIPS, 2021.

Kei Ota, Tomoaki Oiki, Devesh K. Jha, Toshisada Mariyama, and Daniel Nikovski. Can increasing input dimensionality improve deep reinforcement learning? In ICML, 2020.

Yangchen Pan, Kirby Banman, and Martha White. Fuzzy tiling activations: A simple approach to learning sparse representations online. In ICLR, 2021.

Alexandre Piché, Valentin Thomas, Joseph Marino, Rafael Pardinas, Gian Maria Marconi, Christopher Pal, and Mohammad Emtiyaz Khan. Bridging the gap between target networks and functional regularization. Transactions on Machine Learning Research, 2022.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In ICML, 2021.

T. Schaul, A. Barreto, J. Quan, and G. Ostrovski. The phenomenon of policy churn. arXiv preprint, arXiv:2206.00730, 2022.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In ICLR, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In ICML, 2015.

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

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. Deterministic policy gradient algorithms. In ICML, 2014.

G. Sokar, R. Agarwal, P. S. Castro, and U. Evci. The dormant neuron phenomenon in deep reinforcement learning. arXiv preprint, arXiv:2302.12902, 2023.

Dijia Su, Jayden Ooi, Tyler Lu, Dale Schuurmans, and Craig Boutilier. Conqur: Mitigating delusional bias in deep q-learning. In ICML, 2020.

R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. IEEE Transactions on Neural Networks, 16:285–286, 1988.

H. Tang, Z. Meng, J. Hao, C. Chen, D. Graves, D. Li, C. Yu, H. Mao, W. Liu, Y. Yang, W. Tao, and L. Wang. What about inputting policy in value function: Policy representation and policy-extended value function approximator. In AAAI, 2022.

 https://doi.org/10.52202/079017-0490

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. P. Lillicrap, and M. A. Riedmiller. Deepmind control suite. arXiv preprint, arXiv:1801.00690, 2018.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In AAAI, 2016.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. arXiv preprint, arXiv:1812.02648, 2018.

Guowei Xu, Ruijie Zheng, Yongyuan Liang, Xiyao Wang, Zhecheng Yuan, Tianying Ji, Yu Luo, Xiaoyu Liu, Jiaxin Yuan, Pu Hua, Shuzhen Li, Yanjie Ze, Hal Daumé III, Furong Huang, and Huazhe Xu. Drm: Mastering visual reinforcement learning through dormant ratio minimization. In ICLR, 2024.

Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. In ICLR, 2022.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for more efficient reinforcement learning experiments. arXiv preprint, arXiv:1903.03176, 2019.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In ICLR, 2021a.

Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. In ICML, 2020.

Shangtong Zhang, Hengshuai Yao, and Shimon Whiteson. Breaking the deadly triad with a target network. In ICML, 2021b.

A Limitations

Our work is limited in several directions below and we expect further studies on these points in the future.

- First, the theoretical analysis of the chain effect under concrete assumptions remains to be explored. Other perspectives that may influence churn, such as network structure, representation learning, and experience replay are not considered in this work, which are worthwhile to study in the future.
- Moreover, although we provided a simple method to adjust the regularization coefficients dynamically throughout learning by keeping a consistent relative loss scale, it is not sufficient for us to use the same hyperparameter for different domains. We believe that using normalization techniques to unify the scales in different domains is necessary to address this point, similar to the work in [Hafner et al., 2023].
- Besides, for algorithms that involve the learning of both policy and value (e.g., deep AC methods), the implementation of CHAIN in different problems faces the question of choosing the best option among PCR, VCR and DCR. For this, we expect to develop a better method to integrate the effect of PCR and VCR in the future.
- Another remaining problem is the lack of an in-depth understanding of the positive and negative effects of churn on the generalization of DRL agents, which could drive new methods that better leverage the potential of churn.

B Additional Formal Analysis

B.1 The NTK Expressions of Two Types of Deviation

The NTK expression for the action gradient deviation $\mathcal{D}^{Q}_{\nabla_a}$ is straightforward to obtain by plugging in the NTK expression for the Q-value churn:

$$\begin{aligned}
\mathcal{D}^{Q}_{\nabla_a}(\theta, \theta') &= \nabla_{\bar{a}} Q_{\theta'}(\bar{s}, \bar{a})|_{\bar{a}=\pi(s)} - \nabla_{\bar{a}} Q_{\theta}(\bar{s}, \bar{a})|_{\bar{a}=\pi(\bar{s})} \\
&= \nabla_{\bar{a}} \underbrace{(Q_{\theta'}(\bar{s}, \bar{a}) - Q_{\theta}(\bar{s}, \bar{a}))}_{\mathcal{C}_Q(\theta, \theta')}|_{\bar{a}=\pi(\bar{s})} \\
&\approx \nabla_{\bar{a}} (k_\theta(\bar{s}, \bar{a}, s, a)\delta_\theta(s, a))|_{\bar{a}=\pi(\bar{s})} \\
&\approx \left(\frac{\partial^2 Q_\theta(\bar{s}, \bar{a})}{\partial \theta \partial \bar{a}} \right) \Delta_\theta|_{\bar{a}=\pi(\bar{s})}
\end{aligned}$$

And for the policy value deviation $\mathcal{D}^{\pi}_{Q}$, the NTK expression is obtained by performing Taylor expansion of $\pi_\phi(\bar{s})$ and plugging in the NTK expression for the policy churn:

$$\begin{aligned}
\mathcal{D}^{\pi}_{Q}(\phi, \phi') &= (\nabla_{\bar{a}} Q_\theta(\bar{s}, \bar{a})|_{\bar{a}=\pi_\phi(\bar{s})})^\top \underbrace{(\pi_{\phi'}(\bar{s}) - \pi_\phi(\bar{s}))}_{\mathcal{C}_\pi(\phi, \phi')} + O(\| \underbrace{\pi_{\phi'}(\bar{s}) - \pi_\phi(\bar{s})}_{\mathcal{C}_\pi(\phi, \phi')} \|^2) \\
&\approx (\nabla_{\bar{a}} Q_\theta(\bar{s}, \bar{a})|_{\bar{a}=\pi_\phi(\bar{s})})^\top k_\phi(\bar{s}, s) \nabla_a Q_\theta(s, a)|_{a=\pi_\phi(s)} \\
&\approx (\underbrace{\nabla_\phi \pi_\phi(\bar{s}) \nabla_{\bar{a}} Q_\theta(\bar{s}, \bar{a})|_{\bar{a}=\pi_\phi(\bar{s})}}_{\text{DPG of } \pi_\phi \text{ at } \bar{s}})^\top \Delta_\phi
\end{aligned}$$

The NTK expression of the two types of deviation above indicates that action gradient deviation and policy value deviation are mainly influenced by the second-order partial derivatives of Q_θ and the deterministic policy gradient (DPG) of π_ϕ at the state $\bar{s}$ considered, in addition to the parameter update. An interesting observation here is, policy value deviation of policy churn has an *implicit optimization* effect (while altered by Δ_ϕ) for the reference states (i.e., $\bar{s}$) although the parameter update $\phi \to \phi'$ is performed for the training states (i.e., s). We observe that $\mathcal{D}^{\pi}_{Q}$ is positive in overall in our empirical investigation later (Appendix D). However, careful considerations are needed in the future because the implicit optimization could be delusional as churn also occurs in Q_θ through the learning process.

https://doi.org/10.52202/079017-0490

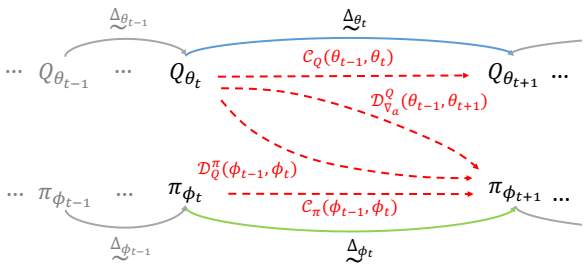


Figure 8: An illustration of the parameter update chain with the value and policy churn. Blue and green arrows denote iterative updates of the value network and policy network. The value and policy churn and their deviations are marked in red.

B.2 Complete Discussions and Derivations for Section 4.3

Recall conventional gradients $\Delta_\theta, \Delta_\phi$ of value network and policy network:

$$\Delta_\theta(s,a) = \nabla_\theta Q_\theta(s,a)\delta_\theta(s,a)$$
$$\Delta_\phi(s) = \nabla_\phi \pi_\phi(s) \nabla_a Q_\theta(s,a)|_{a=\pi_\phi(s)}$$

where $\delta_\theta(s,a) = r + \gamma Q_\theta(s', \pi_\phi(s'))$.

Now let us re-consider the gradients by taking into consideration the value and policy churn, denoted by $\tilde{\Delta}_\theta, \tilde{\Delta}_\phi$. Our purpose is to formulate the difference between $\tilde{\Delta}_\theta, \tilde{\Delta}_\phi$ and $\Delta_\theta, \Delta_\phi$ as functions of the policy and value churns, as well as their derivatives.

More derivatives of $\mathcal{C}_Q, \mathcal{C}_\pi$ Before looking into the gradients $\tilde{\Delta}_\theta, \tilde{\Delta}_\phi$, we need three more definitions for network parameter gradient deviation caused by value and policy churn, and action gradient deviation caused by policy churn, during parameter update $\theta \to \theta', \phi \to \phi'$.

- Q-network Gradient Deviation of Value Churn: $\mathcal{D}^Q_{\nabla_\theta}(\theta, \theta', \{\bar{s}, \bar{a}\}) = \nabla_{\theta'} Q_{\theta'}(\bar{s}, \bar{a}) - \nabla_\theta Q_\theta(\bar{s}, \bar{a})$.
- Policy Network Gradient Deviation of Policy Churn: $\mathcal{D}^\pi_{\nabla_\phi}(\phi, \phi', \{\bar{s}\}) = \nabla_{\phi'} \pi_{\phi'}(\bar{s}) - \nabla_\phi \pi_\phi(\bar{s})$.
- Action Gradient Deviation of Policy Churn: $\mathcal{D}^\pi_{\nabla_a}(\phi, \phi', \{\bar{s}\}) = \nabla_{\bar{a}'} Q(\bar{s}, \bar{a}')|_{\bar{a}'=\pi_{\phi'}(\bar{s})} - \nabla_{\bar{a}} Q(\bar{s}, \bar{a})_{\bar{a}=\pi_\phi(\bar{s})}$

Note $\mathcal{D}^\pi_{\nabla_a}(\phi, \phi', \{s\})$ denotes the action gradient caused by the policy churn for the Q function affected by the churns $\underset{\sim}{Q}_\theta$ rather than Q_θ. $\mathcal{D}^\pi_{\nabla_a}$ is a further consequence of policy value deviation caused by the policy churn $\mathcal{D}^\pi_Q(\phi^-, \phi, \{s\})$.

To shed light on the long-term effect of churn, we depict a typical iterative update scenario in Figure 8, where the Q-network and policy network update with corresponding gradients in a chain. Different from conventional analysis, we explicitly consider the value and policy churn and study how they affect the chain of parameter updates.

As in Section 4.3, we focus on the segment $(\theta^-, \phi^-) \to (\theta, \phi) \to (\theta', \phi')$ on the chain of update. The churns occurred during the past update $(\theta^-, \phi^-) \to (\theta, \phi)$, which should further affect the update about to perform at present $(\theta, \phi) \to (\theta', \phi')$. Concretely, the churns and the deviations affect the following aspects: **(1)** Q-value estimate and **(2)** action selection in both TD error and policy objective, and **(3)** the gradient of network parameter. Now we are ready to deduce the parameter update under the effect of the value and policy churn below. Note that we use $\underset{\sim}{\ }$ under the terms that are affected by

value and policy churn:

$$\tilde{\Delta}_\theta(s,a) = \underbrace{\nabla_\theta Q_\theta}(s,a)\tilde{\Delta}_\theta(s,a) = \underbrace{\nabla_\theta Q_\theta}(s,a)(r + \gamma \underbrace{Q_\theta}(s', \underbrace{a'}) - \underbrace{Q_\theta}(s,a))$$

$$= \Big(\nabla_\theta Q_\theta(s,a) + (\underbrace{\nabla_\theta Q_\theta}(s,a) - \nabla_\theta Q_\theta(s,a))\Big)\Big(r + \gamma\Big[Q_\theta(s',a') + (\underbrace{Q_\theta}(s', \underbrace{a'}) - \underbrace{Q_\theta}(s',a'))$$

$$+ (\underbrace{Q_\theta}(s',a') - Q_\theta(s',a'))\Big] - \Big[Q_\theta(s,a) + (\underbrace{Q_\theta}(s,a) - Q_\theta(s,a))\Big]\Big)$$

$$= \Big(\nabla_\theta Q_\theta(s,a) + \mathcal{D}^Q_{\nabla_\theta}(\theta^-, \theta, \{s,a\})\Big)\Big(\delta_\theta(s,a) + \gamma(\mathcal{D}^\pi_Q(\phi^-, \phi, \{s', \underbrace{\pi_\phi}(s')\}) + \mathcal{C}_Q(\theta^-, \theta, \{s', \pi_\phi(s')\}))$$

$$- \mathcal{C}_Q(\theta^-, \theta, \{s,a\})\Big)$$

$$= \nabla_\theta Q_\theta(s,a)\delta_\theta(s,a)$$

$$+ \gamma \nabla_\theta Q_\theta(s,a)(\mathcal{D}^\pi_Q(\phi^-, \phi, \{s', \underbrace{\pi_\phi}(s')\}) + \mathcal{C}_Q(\theta^-, \theta, \{s', \pi_\phi(s')\})) - \nabla_\theta Q_\theta(s,a)\mathcal{C}_Q(\theta^-, \theta, \{s,a\})$$

$$+ \mathcal{D}^Q_{\nabla_\theta}(\theta^-, \theta, \{s,a\})\Big(\delta_\theta(s,a) + \gamma(\mathcal{D}^\pi_Q(\phi^-, \phi, \{s', \underbrace{\pi_\phi}(s')\}) + \mathcal{C}_Q(\theta^-, \theta, \{s', \pi_\phi(s')\})) - \mathcal{C}_Q(\theta^-, \theta, \{s,a\})\Big)$$

$$\tilde{\Delta}_\phi(s) = \underbrace{\nabla_\phi \pi_\phi}(s)\underbrace{\nabla_a Q_\theta}(s, \underbrace{a})|_{\underbrace{a=\pi_\phi}(s)}$$

$$= \Big(\nabla_\phi \pi_\phi(s) + (\underbrace{\nabla_\phi \pi_\phi}(s) - \nabla_\phi \pi_\phi(s))\Big)\Big(\nabla_a Q_\theta(s,a) + (\underbrace{\nabla_a Q_\theta}(s, \underbrace{a}) - \nabla_a \underbrace{Q_\theta}(s,a))$$

$$+ (\nabla_a \underbrace{Q_\theta}(s,a) - \nabla_a Q_\theta(s,a))\Big)|_{\underbrace{a=\pi_\phi}(s), a=\pi_\phi(s)}$$

$$= \Big(\nabla_\phi \pi_\phi(s) + \mathcal{D}^\pi_{\nabla_\phi}(\phi^-, \phi)\Big)\Big(\nabla_a Q_\theta(s,a) + \mathcal{D}^\pi_{\nabla_a}(\phi^-, \phi, \{s\}) + \mathcal{D}^Q_{\nabla_a}(\theta^-, \theta, \{s,a\})\Big)|_{a=\pi_\phi(s)}$$

$$= \nabla_\phi \pi_\phi(s)\nabla_a Q_\theta(s,a)|_{a=\pi_\phi(s)} + \nabla_\phi \pi_\phi(s)\Big(\mathcal{D}^\pi_{\nabla_a}(\phi^-, \phi, \{s\}) + \mathcal{D}^Q_{\nabla_a}(\theta^-, \theta, \{s,a\})\Big)|_{a=\pi_\phi(s)}$$

$$+ \mathcal{D}^\pi_{\nabla_\phi}(\phi^-, \phi)\Big(\nabla_a Q_\theta(s,a) + \mathcal{D}^\pi_{\nabla_a}(\phi^-, \phi, \{s\}) + \mathcal{D}^Q_{\nabla_a}(\theta^-, \theta, \{s,a\})\Big)|_{a=\pi_\phi(s)}$$

As a result, we can find that the value and policy churn, as well as the deviations derived, introduce biases in the parameter updates.

Recall that the parameter updates cause the churns constantly, the analysis on the update segment $(\theta^-, \phi^-) \to (\theta, \phi) \to (\theta', \phi')$ can be forwarded and leads to the cycle illustrated in Figure 2: (1) parameter update causes the value and policy churn $\mathcal{C}_Q, \mathcal{C}_\pi$, which (2) further deviates the action gradient and policy value $\mathcal{D}^\pi_Q, \mathcal{D}^Q_{\nabla_a}$ (and the other deviations); (3) the churns and the deviations then bias following parameter updates with $\tilde{\Delta}_\theta, \tilde{\Delta}_\phi$. Consequently, the value and policy churn and the parameter update bias *accumulate* and can *amplify each other* throughout the learning process.

Apparently, the long-term chain effect is intricate as the training process is stochastic and the churns are influenced by various factors, e.g., network structure, learning objective, and data distribution. We leave theoretical studies under concrete assumptions in the future.

C Experimental Details

C.1 Compute Resources and Time Cost

We use Nvidia V100 GPU for our experiments. The additional computation introduced by CHAIN lies in the sampling and training with a second batch of data for the churn reduction regularization. In essence, it just increases the batch size by 2 to use CHAIN, resulting in a constant change in complexity with respect to the default DRL algorithm. This is a small price to pay to reduce the churn and increase policy performance. In practice, we also observe similar wall-clock time cost for standard DRL methods and their CHAIN versions.

More importantly, since CHAIN often brings higher sample efficiency, i.e., achieving the same level of score with fewer interaction steps, this implies that CHAIN accelerates learning and achieves good performance earlier. Concrete examples can be found in our experimental results.

https://doi.org/10.52202/079017-0490

C.2 Empirical Metrics for the Investigation of the Chain Effect

To investigate the extent of the value churn and the policy churn, we compare the output changes between current networks (θ_t, ϕ_t) and past network versions $(\theta^-, \phi^-) \in \{(\theta_{t-i}, \phi_{t-i})\}_{i=1}^{N}$.

For policy-based methods, we study TD3 and SAC for MuJoCo. We compute the value churn ($\hat{\mathcal{C}}_{Q_{sa}}, \hat{\mathcal{C}}_{|Q_{sa}|}$), the policy churn ($\hat{\mathcal{C}}_{\pi}$) and the value deviation of policy churn ($\hat{\mathcal{D}}^{\pi}_{Q}$) throughout learning.

For value-based methods, we compute the percentage of the greedy action deviation ($\hat{\mathcal{D}}^{Q}_{a^\star}$), the value churn of greedy action ($\hat{\mathcal{C}}_{Q_{a^\star}}$) and the value churn of all actions ($\hat{\mathcal{C}}_{Q_s}, \hat{\mathcal{C}}_{|Q_s|}$) These metrics are summarized in Table 4.

Table 4: Churn and deviation metrics used in our experiments. θ_t, ϕ_t and θ^-, ϕ^- are current networks and previous networks. The metrics are averaged over $\bar{s}, \bar{a}$ in a reference buffer.

The Metrics for Policy-based Methods	
$\hat{\mathcal{C}}_{Q_{sa}}(\theta^-, \theta_t)$	$Q_{\theta_t}(\bar{s}, \bar{a}) - Q_{\theta^-}(\bar{s}, \bar{a})$ (for TD3, SAC)
$\hat{\mathcal{C}}_{\lvert Q_{sa}\rvert}(\theta^-, \theta_t)$	$\lvert Q_{\theta_t}(\bar{s}, \bar{a}) - Q_{\theta^-}(\bar{s}, \bar{a})\rvert$ (for TD3, SAC)
$\hat{\mathcal{C}}_{\pi}(\phi^-, \phi_t)$	$\lVert \pi_{\phi_t}(\bar{s}) - \pi_{\phi^-}(\bar{s}))\rVert_1$ (for TD3, PPO[3])
	$\mathrm{KL}(\pi_{\phi_t}(\cdot\lvert\bar{s}), \pi_{\phi^-}(\cdot\lvert\bar{s}))$ (for SAC)
$\hat{\mathcal{D}}^{\pi}_{Q}(\phi^-, \phi_t)$	$Q_{\theta_t}(\bar{s}, \pi_{\phi_t}(\bar{s})) - Q_{\theta_t}(\bar{s}, \pi_{\phi^-}(\bar{s}))$
The Metrics for Value-based Methods	
$\hat{\mathcal{C}}_{\lvert Q_s\rvert}(\theta^-, \theta_t)$	$\frac{1}{\lvert A\rvert}\sum_{a\in A} \lvert Q_{\theta_t}(\bar{s}, a) - Q_{\theta^-}(\bar{s}, a)\rvert$
$\hat{\mathcal{C}}_{Q_{a^\star}}(\theta^-, \theta_t)$	$\max_{a'} Q_{\theta_t}(\bar{s}, a') - \max_a Q_{\theta^-}(\bar{s}, a)$
$\hat{\mathcal{D}}^{Q}_{a^\star}(\theta^-, \theta_t)$	$\mathbb{I}_{\{\arg\max_a Q_{\theta^-}(\bar{s}, a)\}}(\arg\max_{a'} Q_{\theta_t}(\bar{s}, a'))$

We use $N = 50$ and $N = 20$ for MuJoCo and MinAtar environments and compute the metrics at an interval of 1k parameter updates. For each type of metric in Table 4 and update number $i \in \{1, \ldots, N\}$, we compute the mean of the quantities throughout learning. We refer the readers to Figure 9, 11 and 12 for the results in Appendix D.

C.3 Code Implementation

We use the public implementations of PPO, TD3 and SAC in `CleanRL`[4] as our codebase. The actor and critic networks are two-layer MLPs with 256 units for each layer. For DoubleDQN, we modified the DQN implementation provided in the official code of MinAtar paper[5] with no change to the network structure, recommended hyperparameter choices, etc. Hyperparameters are listed in Table 5, Table 6 and Table 7. For the experiments in the offline RL setting, we use the public implementation and benchmark scores for IQL and AWAC from `CORL`[6].

One thing to note is that the data in the training batch and the regularization batch should be non-overlapping in principle, but we found simply sampling two random batches from the replay buffer independently works well. From the probabilistic perspective, the overlap could happen at a low probability, which is determined by the batch size and the size of the replay buffer.

We make no change to the state/observation and reward of MuJoCo and MinAtar environments. No additional tricks like state normalization, reward normalization are used in our experiments.

C.4 Other Discussions

Discussion on the Data Batches Used in Training and Anlysis We use separate sets of data for churn reduction regularization (i.e., the regularization set) and churn investigation/evaluation (i.e., the actual reference set), and they are randomly sampled at each network update. In other words, if count in the regular batch for training, we have three separate batches in total: (1) a regular training

[4] `https://github.com/vwxyzjn/cleanrl`
[5] `https://github.com/kenjyoung/MinAtar`
[6] `https://github.com/tinkoff-ai/CORL`

batch for standard RL loss computation, (2) a regularization batch for churn reduction loss, and (3) a reference batch for churn evaluation (optional) throughout the learning process. Note that the reference batch is only used for churn evaluation and does not influence learning.

Table 5: Hyperparameters of DoubleDQN used in MinAtar environments. The values of conventional hyperparameters are taken from the recommended values in [Young and Tian, 2019].

DoubleDQN Hyperparameters	
Learning Rate	$3e^{-4}$
Training Interval	1 step
Discount Factor (γ)	0.99
Hard Replacement Interval	1000 steps
Replay Buffer Size	0.5M
Batch Size	32
Initial ϵ	1.0
End ϵ	0.1
ϵ Decay Steps	0.5M
Initial Random Steps	10k
Value Churn Reduction Hyperparameter	
Value Regularization Coefficient (λ_Q)	50 for Breakout 100 for others
Target Relative Loss β for Auto λ_Q	0.05

Table 6: Hyperparameters of PPO used in MuJoCo environments. The values of conventional hyperparameters are taken from the recommended values in `CleanRL`.

PPO Hyperparameters	
Learning Rate	$3e^{-4}$
Training Interval	2048 steps
Discount Factor (γ)	0.99
GAE Parameter (λ)	0.95
Num. of Minibatches	32
Update Epoch	10
Clipping Range Parameter (ϵ)	0.1 for MuJoCo tasks 0.2 for DMC tasks
Policy Churn Reduction Hyperparameter	
Policy Regularization Coefficient (λ_π)	5000 for Ant-v4 50 for other MuJoCo tasks
Target Relative Loss β for Auto λ_π	0.1 for MuJoCo tasks 0.02 for DMC tasks

https://doi.org/10.52202/079017-0490

Table 7: Hyperparameters of TD3 and SAC used in MuJoCo environments. The values of conventional hyperparameters are taken from the recommended values in `CleanRL`. '-' means 'not applicable'.

TD3 & SAC Hyperparameters		
Hyperparameters	TD3	SAC
Actor Learning Rate	$3e^{-4}$	$3e^{-4}$
Critic Learning Rate	$3e^{-4}$	$3e^{-4}$
Actor Training Interval	2 steps	1 step
Critic Training Interval	1 step	1 step
Exploration Noise	$\mathcal{N}(0, 0.1)$	-
Target Action Noise	$\mathcal{N}(0, 0.2)$	-
Target Action Noise Clip	0.5	-
Discount Factor (γ)	0.99	0.99
Soft Replacement Ratio	0.005	0.005
Initial Random Steps	5k	5k
Replay Buffer Size	1M	1M
Batch Size	256	256
Optimizer	Adam	Adam
Churn Reduction Hyperparameters (refer to the study in Fig. 6, 11, 12)		
Policy Regularization Coefficient (λ_π)	$\{0.1, 1, 20\}$	$\{1e^{-4}, 5e^{-4}, 1e^{-3}\}$
Value Regularization Coefficient (λ_Q)	$\{0.1, 0.5, 1.0\}$	
Target Relative Loss β for Auto λ_π	$5e^{-5}$	

Algorithm 1 Deep RL with Churn Approximated ReductIoN (**CHAIN**).

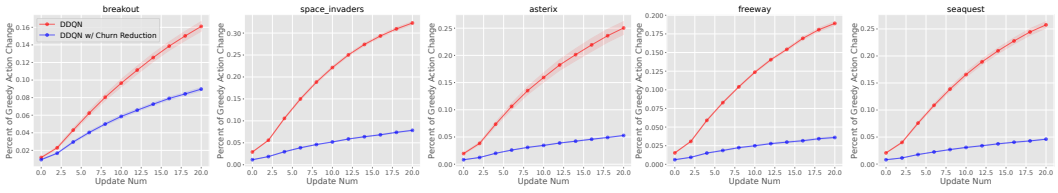
Initialize the policy network π_ϕ and the value network Q_θ with parameters ϕ, θ if they exist
Initialize an empty buffer D
set churn reduction hyperparamters λ_Q, λ_π
(Optional) Set target relative loss scale β for auto-adjustment of λ_Q, λ_π
for iteration $t = 1, 2, 3, ...$ **do**
  // Interact with the environment and collect samples
  Rollout the policy with π_ϕ or Q_θ and store interaction data in D
  // Perform value and policy network learning
  if time to update **then**
    Sample a training batch B_{train} and a reference batch B_{ref} from D
    Update Q_θ with $L(\theta, B_{\text{train}})$, $L_{\text{QC}}(\theta, B_{\text{ref}})$ by Eq. 5, 7
    Update π_ϕ with $L(\phi, B_{\text{train}})$, $L_{\text{PC}}(\phi, B_{\text{ref}})$ by Eq. 6, 8
    (Optional) Re-calculate λ_Q, λ_π according to β (refer to Section 5.2)
  end if
end for

D Complete Results

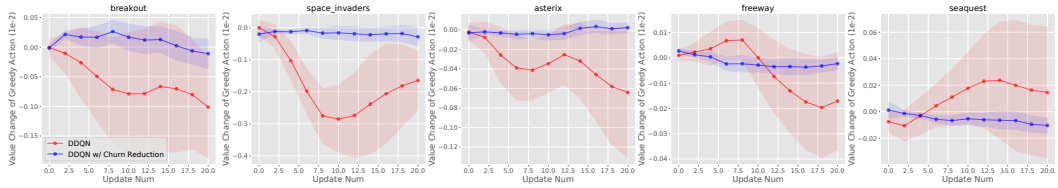
D.1 More Empirical Analysis on the Value Churn in DoubleDQN

In Figure 9 shows the statistics of the three metrics defined in Table 4. Horizontal axes show the number of parameter updates after which the statistics are computed. Each point is obtained by averaging all the quantities throughout learning. Curves and shades denote means and standard errors across six random seeds.

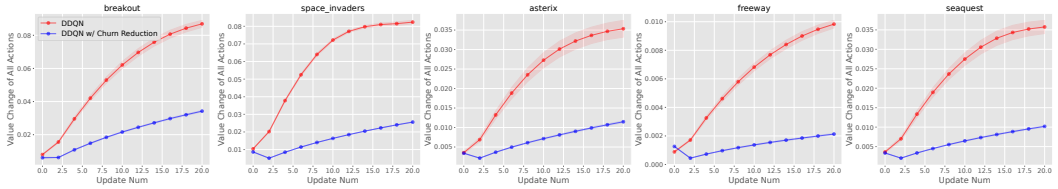
We can observe that the amount of value churn ($\hat{\mathcal{C}}_{|Q_s|}$) accumulates as the update number increases. Although there does not exist an explicit policy network, the percentage of greedy action deviation ($\hat{\mathcal{C}}_{Q_{a^\star}}$) goes up to over 20%. However, different from the case for TD3 and SAC, the value of greedy action ($\hat{\mathcal{D}}^Q_{a^\star}$) decreases and the value churn ($\hat{\mathcal{C}}_{Q_s}$) exhibits a similar trend.



(a) Percentage of Greedy Action Change



(b) Value Change of Greedy Action



(c) Value Change of All Actions

Figure 9: Different statistics on value churn of DoubleDQN in MinAtar. Horizontal axes are numbers of parameter updates after which the statistics are computed. Each point is obtained by averaging all the quantities throughout learning. Curves and shades denote means and standard errors across six random seeds.

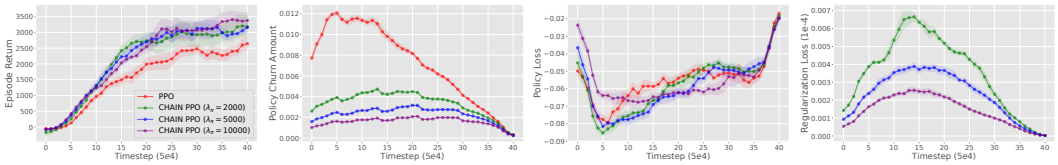
D.2 More Results of CHAIN PPO

Figure 10 provides the learning performance of CHAIN PPO on four MuJoCo tasks. Moreover, we also report the conventional PPO policy loss and the regularization loss $L_{\text{PC}}(\phi)$ during the learning process. Figure 10 also shows the results for different choices of the hyperparameter λ_π.

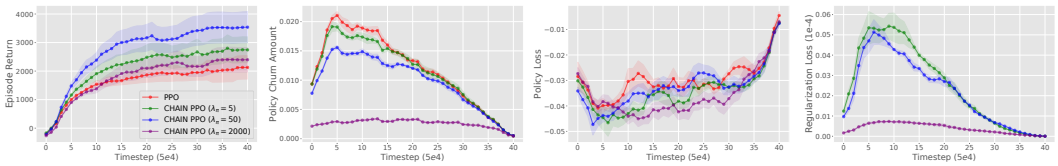
From the results, we can observe: (1) PPO exhibits clear policy churn (the second column). One thing to note is that the fading of the policy churn is due to the decay of the learning rate in `CleanRL`'s PPO implementation. CHAIN (2) reduces the policy churn of PPO effectively, and (3) shows performance improvement in Ant and HalfCheetah and comparable performance in Hopper and Walker2d. For the hyperparameter choice, in practice, we found the churn reduction coefficient between 1e3 and 1e4 works well for Ant, while 50 works best in HalfCheetah.

We hypothesize that the choice of λ_π is likely to be related to the scale difference between policy loss $-J(\phi)$ (the third column) and regularization term $L_{\text{PC}}(\phi)$ (the fourth column). This motivates the proposal of our method for automatic adjustment of the regularization coefficient presented in Section 5.2.

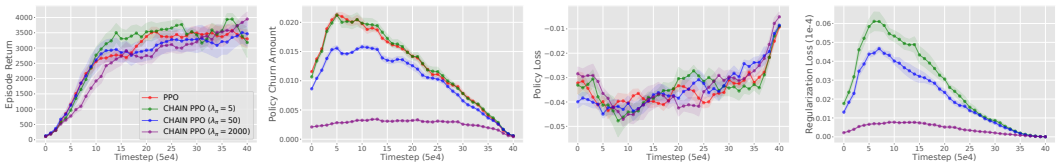
https://doi.org/10.52202/079017-0490



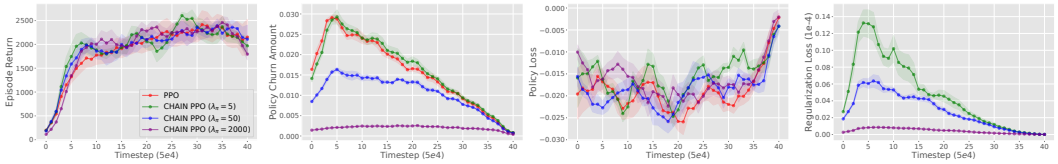
(a) Ant



(b) HalfCheetah



(c) Walker2d



(d) Hopper

Figure 10: Results for CHAIN PPO, including the learning performance, the amount of the policy churn, the loss of conventional policy training, and the loss of regularization.

D.3 More Results of CHAIN for Deep AC Methods

We refer the readers to the following figures and tables for concrete additional results:

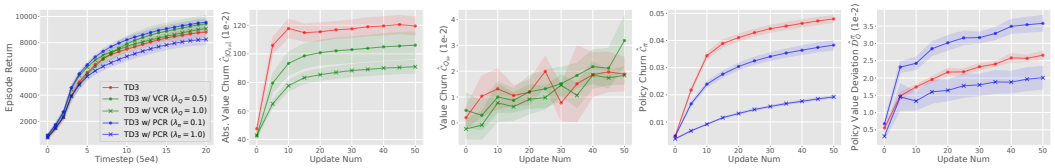
- Figure 11 and Figure 12 show how the value churn reduction (VCR) and policy churn reduction (PCR) take effect during the learning process of TD3 and SAC. The four metrics defined in Table 4 are used.
- Figure 13 and Figure 14 show the results of different choices of λ_Q and λ_π when VCR, PCR, or VCR+PCR (i.e., denoted by DCR) in CHAIN TD3 and CHAIN SAC.
- Figure 15 shows an overall comparison among VCR, PCR, DCR when using either of them in CHAIN TD3 and CHAIN SAC. We also provide the comparison with *Reliable* Metrics[7] [Agarwal et al., 2021] in Figure 17 and Figure 18.
- Figure 16 shows the evaluation of the automatic adjustment method of λ_π for CHAIN TD3 in four MuJoCo tasks.

The Effect of CHAIN-VCR and CHAIN-PCR in Reducing Churn For TD3 and SAC, the amount of value churn ($\hat{\mathcal{C}}_{|Q_{sa}|}$) and policy churn ($\hat{\mathcal{C}}_\pi$) increases as the update number i increases. This is expected as the chain effect indicates the accumulation of churn and deviation. Besides, the churn saturates after a sufficient number of updates on the networks. We hypothesize the target network helps cap the amount of churn.

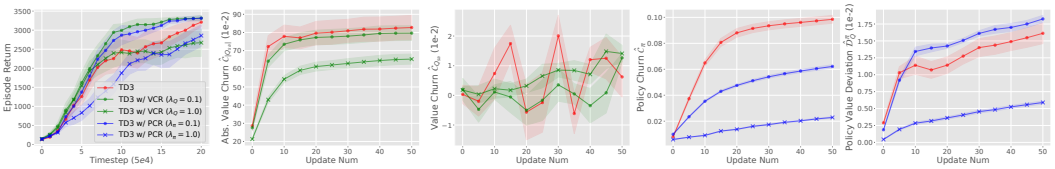
Moreover, we observe a positive policy value deviation of action churn ($\hat{\mathcal{D}}_Q^\pi$), which matches our discussion on *implicit optimization* effect of the policy churn in Appendix B; in contrast, the value churn ($\hat{\mathcal{C}}_{Q_{sa}}$) fluctuates above and below 0.

[7] `https://github.com/google-research/rliable`

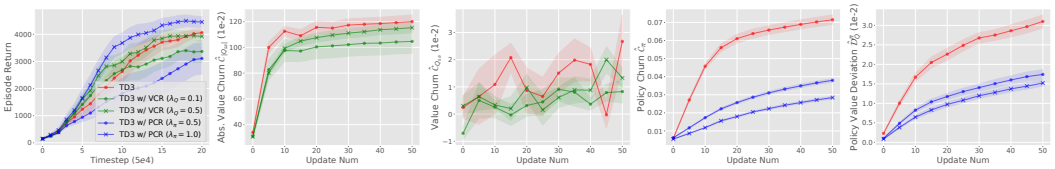
According to the relative scale, TD3 and SAC show less value and policy churn than DoubleDQN. This is due to the structure independence of actor and critic networks. Another possible reason is that MuJoCo locomotion may have smoother underlying problem dynamics than the MinAtar.



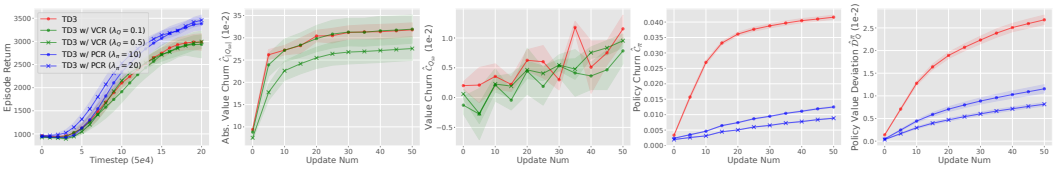
(a) HalfCheetah-v4



(b) Hopper-v4



(c) Walker2d-v4



(d) Ant-v4

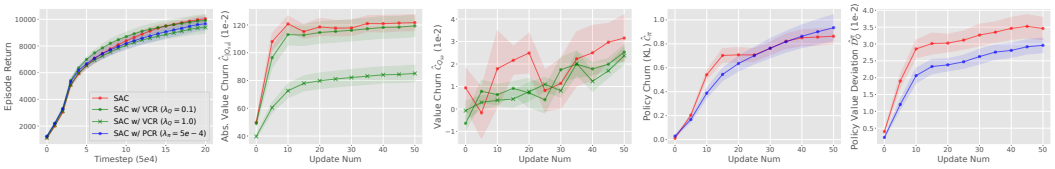
Figure 11: Churn reduction of TD3 in MuJoCo. The 2*nd*-4*th* columns report the value of the four metrics defined in Table 4. Horizontal axes are numbers of parameter updates after which the statistics are computed. Each point is obtained by averaging all the quantities throughout learning.

Discussion on the Effect of CHAIN-VCR, -PCR, -DCR in Improving Episode Return In addition to the two variants of CHAIN introduced in the main body of this paper, we introduce the third one, i.e., Double Churn Reduction (DCR), which corresponds to applying VCR and PCR at the same time. The related results are reported in Figure 13, 14, 15.

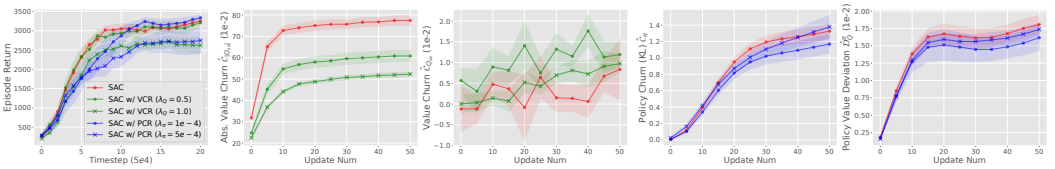
For CHAIN-VCR and CHAIN-PCR, we found that CHAIN-PCR often improves the learning performance, especially for Ant-v4. In contrast, CHAIN-VCR improves slightly. We hypothesize that this is because policy interacts with the environment directly and the target critic-network also helps to cap the value churn. Between TD3 and SAC, CHAIN-PCR works better for TD3 rather than SAC. We hypothesize that the variation (regarding the scale and range) is higher in optimizing the Maximum-Entropy objective and KL-based PCR term together for SAC than in optimizing the Q objective and L2-based PCR term for TD3. Another hypothesis is that the Maximum-Entropy nature of SAC prefers the encouragement of more stochasticity in policy.

For CHAIN-DCR, we found that it is not easy to gain an immediate additive improvement regarding episode return when using the same hyperparameter choices λ_Q, λ_π from both sides. We suggest that this reflects the intricate nature of the chain effect. As mentioned in Section 7, this points out the limitation of this work in aspects like the in-depth theoretical analysis of the long-term chain effect, the lack of automatic coefficient adjustment, and the finer-grained understanding of the positive and negative effects of churn. We leave these directions for future work.

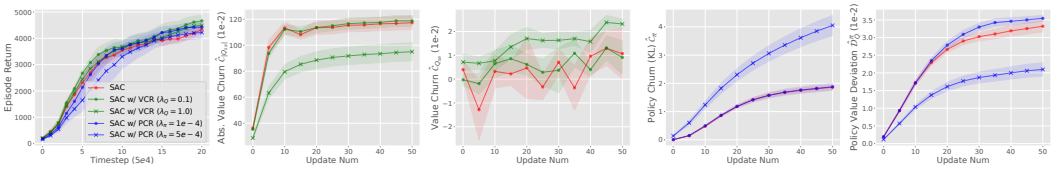
https://doi.org/10.52202/079017-0490



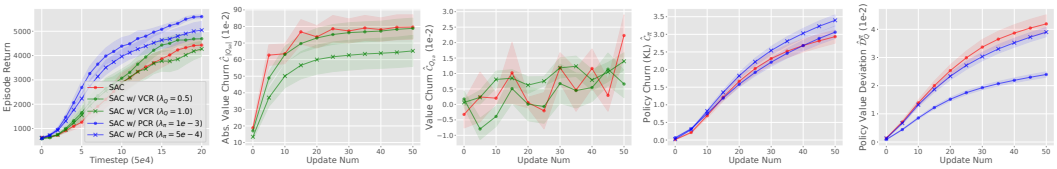
(a) HalfCheetah-v4



(b) Hopper-v4

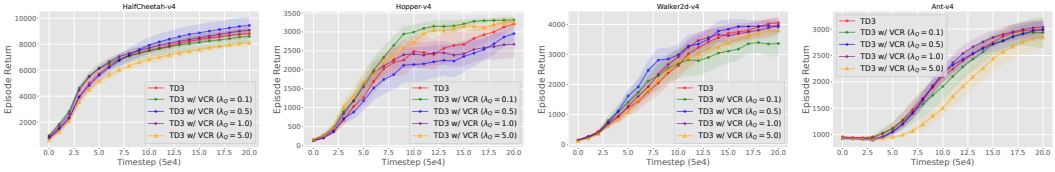


(c) Walker2d-v4

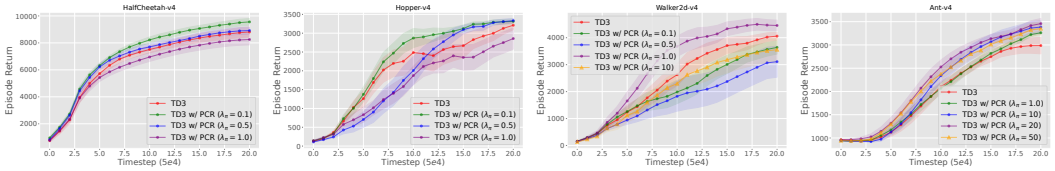


(d) Ant-v4

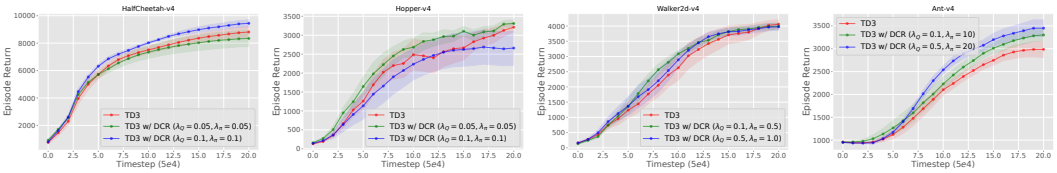
Figure 12: Churn reduction of SAC in MuJoCo. The 2*nd*-4*th* columns report the value of the four metrics defined in Table 4. Horizontal axes are numbers of parameter update after which the statistics are computed. Each point is obtained by averaging all the quantities throughout learning.



(a) Value Churn Reduction (VCR)

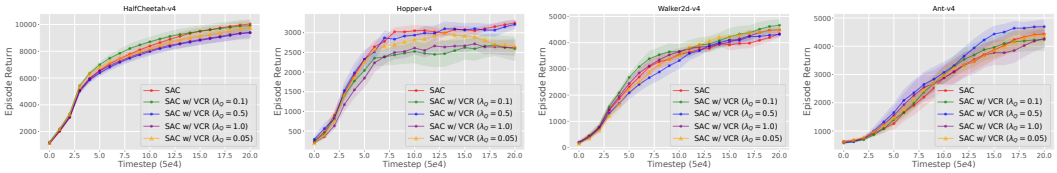


(b) Policy Churn Reduction (PCR)

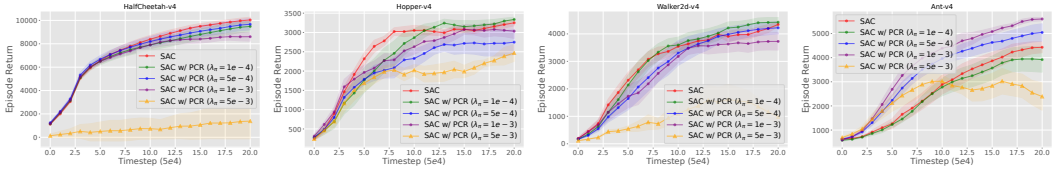


(c) Double Churn Reduction (DCR = VCR + PCR)

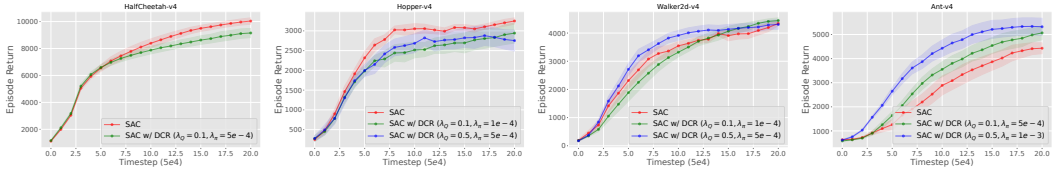
Figure 13: Hyperparameter choices for the value and policy churn reduction regularization for TD3 in MuJoCo. Curves and shades denote means and standard errors across six random seeds.



(a) Value Churn Reduction (VCR)



(b) Policy Churn Reduction (PCR)



(c) Double Churn Reduction (DCR = VCR + PCR)

Figure 14: Hyperparameter choices for the value and policy churn reduction regularization for SAC in MuJoCo. Curves and shades denote means and standard errors across six random seeds.

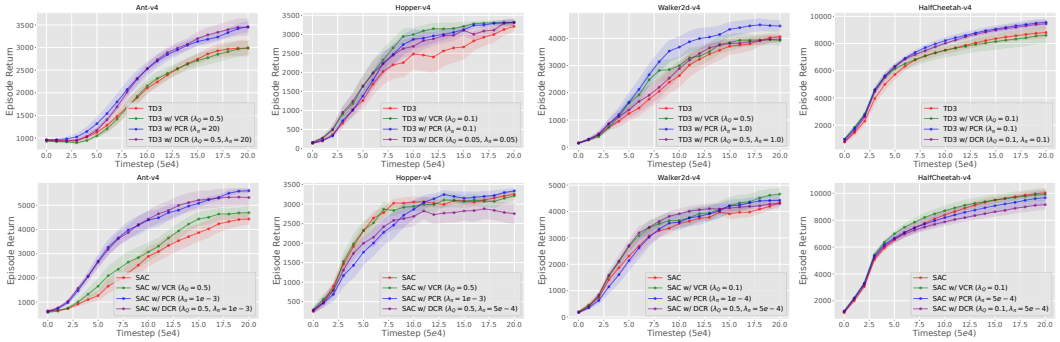


Figure 15: An overall comparison among using value churn reduction, using policy churn reduction, and using them both for TD3 and SAC.

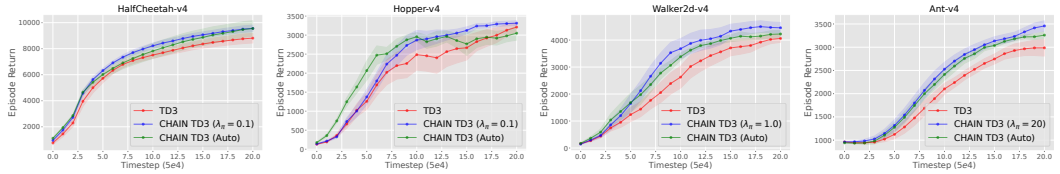
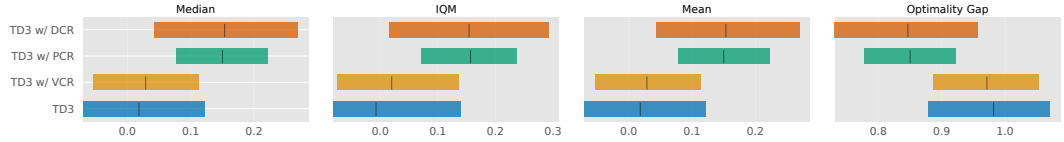


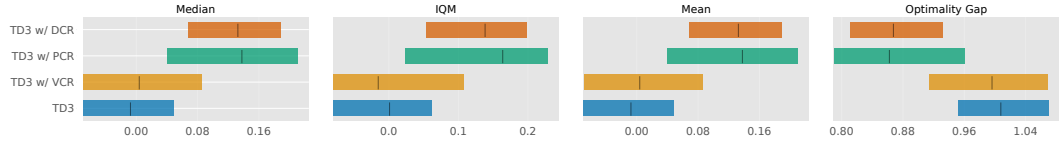
Figure 16: The results for auto-adjustment of λ_π for CHAIN TD3 in four MuJoCo tasks, with means and standard errors across six seeds. The target relative loss scale β is set to 5e-5. CHAIN (Auto) achieves comparable performance with manually selected coefficients and improves TD3.

https://doi.org/10.52202/079017-0490

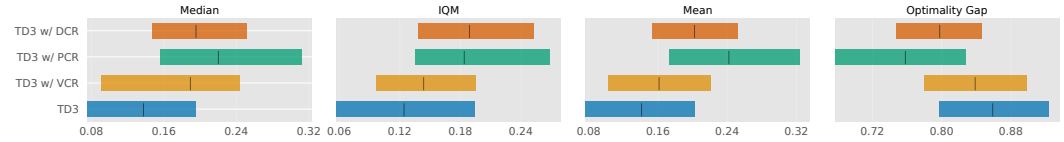
The Effect of Automatic Adjustment of λ_π Figure 16 shows the evaluation of the automatic adjustment method of λ_π for CHAIN TD3 in four MuJoCo tasks. We can observe that CHAIN (Auto) achieves comparable performance with manually selected coefficients and improves TD3, similarly to the conclusions we found in Figure 4 and 5.



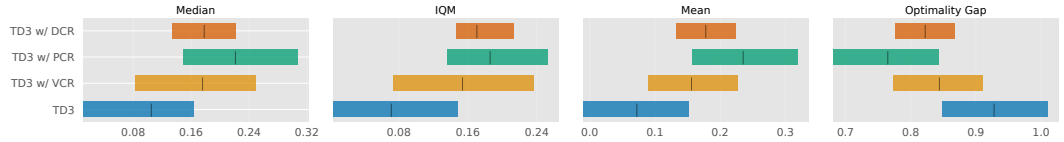
(a) Final scores in Ant



(b) AUC scores in Ant

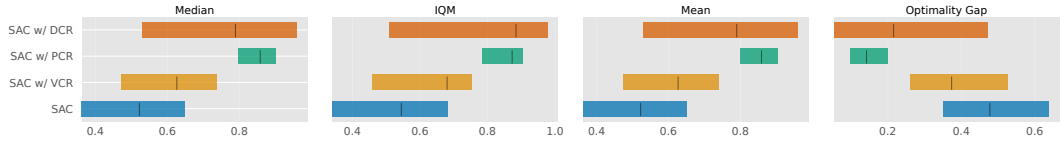


(c) Final scores over all MuJoCo tasks

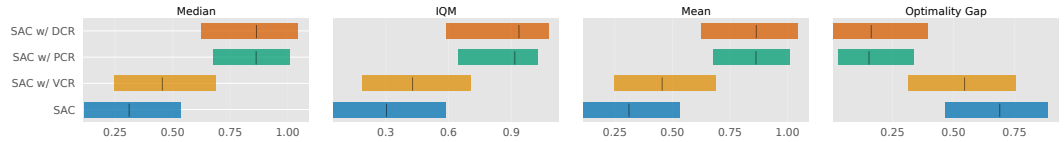


(d) AUC scores over all MuJoCo tasks

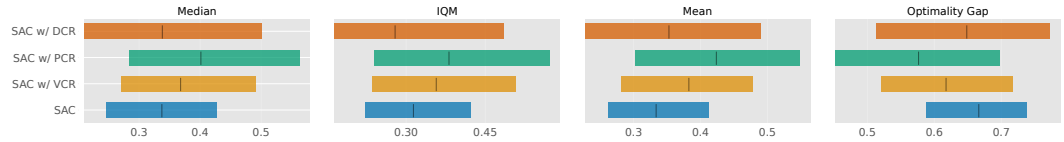
Figure 17: *Reliable* [Agarwal et al., 2021] metrics for TD3 with different churn reduction options.



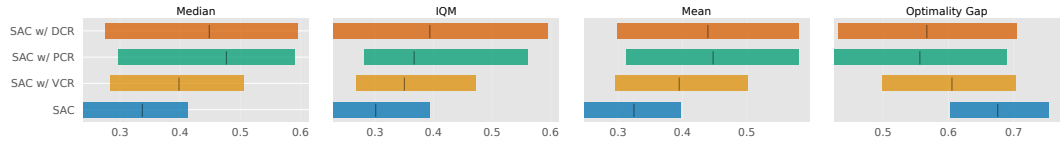
(a) Final scores in Ant



(b) AUC scores in Ant



(c) Final scores over all MuJoCo tasks



(d) AUC scores over all MuJoCo tasks

Figure 18: *Reliable* [Agarwal et al., 2021] metrics for SAC with different churn reduction options.

D.4 More Results of CHAIN for IQL with Sequential Training

As mentioned in Section 6.3, the policy network of IQL has no impact on the training of the value networks, since the value networks (i.e., Q and V) are trained purely based on in-sample data without accessing $a' = \pi_\phi(s')$. Thus, although the policy and value networks of IQL still have churns, the chain effect of churn does not apply in this case.

The default implementation of IQL follows the fashion of iterative training between the policy network and the value network(s). Since the training of the value networks of IQL is independent of the policy, one natural idea is to first fully train the value networks for a sufficient budget and then train the policy value with the well-trained and frozen value networks. We call this *actor-trained-against-final-frozen-critic* fashion as **sequential** training of IQL.

We slightly modified the training process of the CORL implementation of IQL to realize the sequential training of IQL: (1) First train the value network and Q network for 1M steps; (2) Then train the policy network for 1M steps with the value network and Q network frozen; (3) We do not modify any other implementation detail and use the same hyperparameters; (4) We check the learning curves of the policy network and the final scores. We call this variation **IQL (sequential)**. The total number of gradient step is the same as the default IQL implementation where the critic and actor are trained iteratively.

We report the final scores of IQL (sequential) with means and standard errors over 12 seeds in Table 8. The results show that IQL (sequential) performs worse than IQL in 5 of 6 tasks. The difference between IQL (sequential) and IQL can be fully attributed to the difference in the training dynamics, mainly on the policy network. This also means exposing the actor network to the training process of the Q network is helpful compared to training the actor based on a sufficiently trained and frozen Q network. To provide some possible explanation, we guess that this is because the final Q network is a product of accumulated value churns, and the difference in the training dynamics of the policy network of IQL results in a further difference in the policy output on out-of-sample states. However, it is somewhat tricky to explain the difference between IQL (sequential) and IQL. The dynamics is beyond the scope of the chain effect of churn mechanism studied in our work.

Table 8: Results for IQL, IQL (sequential) and CHAIN IQL in Antmaze.

Task	IQL	IQL (sequential)	CHAIN IQL (PCR)	CHAIN IQL (VCR)
AM-umaze-v2	77.00 ± 5.52	60.00 ± 3.91	$\mathbf{84.44 \pm 3.19}$	83.33 ± 2.72
AM-umaze-diverse-v2	54.25 ± 5.54	55.00 ± 5.46	62.50 ± 3.75	$\mathbf{71.67 \pm 7.23}$
AM-medium-play-v2	65.75 ± 11.71	52.50 ± 3.36	$\mathbf{72.50 \pm 2.92}$	70.00 ± 3.33
AM-medium-diverse-v2	73.75 ± 5.45	53.33 ± 5.93	$\mathbf{76.67 \pm 4.51}$	66.67 ± 3.79
AM-large-play-v2	42.00 ± 4.53	17.5 ± 4.10	$\mathbf{50.00 \pm 4.56}$	43.33 ± 4.14
AM-large-diverse-v2	30.25 ± 3.63	5.83 ± 2.19	26.67 ± 3.96	$\mathbf{31.67 \pm 2.31}$

D.5 More Results for Scaling PPO with CHAIN

We take PPO and MuJoCo tasks as the exemplary setting and widen both the policy and value networks by a scale-up ratio within $\{2, 4, 8, 16\}$. Note that the default network architecture (i.e., when the scale-up ratio equals one) for both the policy and value networks is a two-layer MLP with 256 neurons for each layer, followed by an output layer.

Inspired by the prior study [Obando-Ceron et al., 2024], in addition to directly scaling PPO up ('direct'), we use two variants that use different learning rate settings for comparison: (1) 'linear' means using a decreased learning rate as `lr / scale-up ratio`, and (2) 'sqrt' means using `lr / sqrt(scale-up ratio)`. The results of scaling PPO with different learning rate settings are shown in Figure 19 by different colors.

As expected, we observed that the performance of PPO degrades as the increase of the scale-up ratio severely. We found using a decreased learning rate with 'linear' or 'sqrt' alleviates the degradation of PPO scaling to some degree.

The evaluation results of the effect of CHAIN in this scaling setting are shown in Figure 7, Figure **??** and Table 3 as discussed in the main body of this paper.

https://doi.org/10.52202/079017-0490

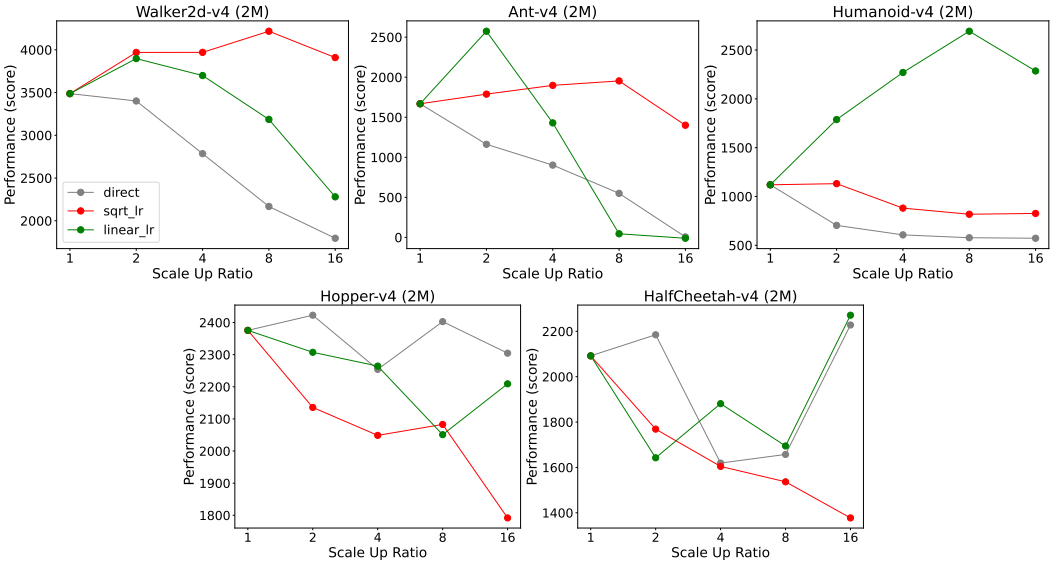
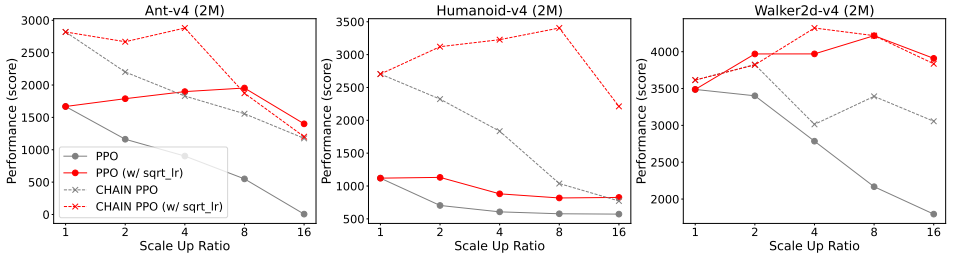
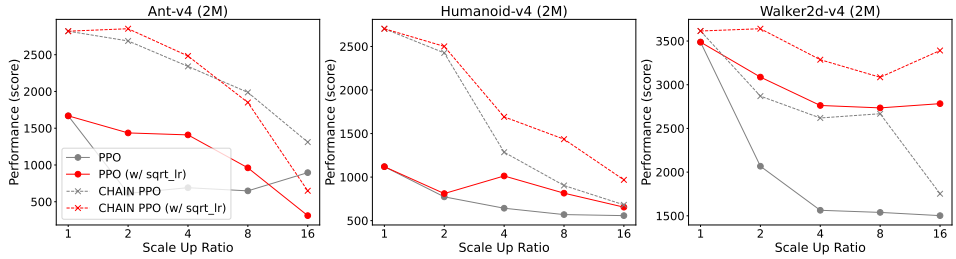


Figure 19: The results for PPO scaling by *widening* with different learning rate settings. 'direct' means using the default learning rate, 'linear' means using a decreased learning rate as `lr / scale-up ratio`, and 'sqrt' means using `lr / sqrt(scale-up ratio)`.



(a) PPO Scaling by *Widening*



(b) PPO Scaling by *Deepening*

Figure 20: The results in terms of episode return for scaling PPO with CHAIN. CHAIN helps to scale almost across all the configurations.

D.6 More Results for CHAIN v.s. Slowing Down Learning

Since churn accompanies each time mini-batch training performed for the networks, a natural question is: whether slowing down learning can alleviate the issue of churn. In principle, using smaller learning rates or target network replacement rates should lead to less churn. This is because churn is positively related to the parameter update amount (as shown by the NTK expressions in Eq. 2) and a slower target network also slows the churn that occurs instantly in each training more when computing the target value to fit for the critic-network.

Empirically, we ran DDQN in Asterix and Freeway and TD3 in Walker2d and Ant, with different learning rates and target network replacement rates. The results are summarized in Table 9. We can observe that either reducing learning rate or target network replacement rate often leads to worse

performance, especially for TD3. To some extent, this also matches the common knowledge in the RL community.

Table 9: Different learning rates (*lr*) and target network replacement rates (*trr*), e.g., "/ 2" means "divided by 2". Mean final episode returns over six seeds are reported.

Alg. - Task	Walker2d	Ant	Alg. - Task	Asterix	Freeway
TD3	**4059.45**	3069.14	DDQN	15.05	49.21
TD3 (*lr* / 2)	2774.87	2408.75	DDQN (*lr* / 2)	**19.26**	**55.83**
TD3 (*lr* / 10)	1314.12	1023.78	DDQN (*lr* / 10)	16.20	49.20
TD3 (*trr* / 5)	3170.18	**3375.33**	DDQN (*trr* / 5)	11.36	50.76
TD3 (*trr* * 5)	4057.33	2743.25	DDQN (*trr* * 5)	18.26	54.33

This indicates that the issue of churn cannot be addressed by reducing learning rate or target network replacement rate (which usually slows down the learning process). Churn is a “by-product” of the training of DRL agents and should be addressed separately.

https://doi.org/10.52202/079017-0490

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction summarize our contributions and are supported by our formal analysis in Section 4 and our experiment results in Section 6.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations in Appendix A.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide sufficient experiment and implementation details in Section 6 and Appendix C.

   Guidelines:

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

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

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

https://doi.org/10.52202/079017-0490

Answer: [Yes]

Justification: Our code can be found at `https://github.com/bluecontra/CHAIN`.

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We provide necessary experimental details in Section 6 and Appendix C.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: The results in our paper are accompanied by error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provide the details of compute resources in Appendix C.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: This work conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA]

    Justification: This paper presents work whose goal is to advance the field of Deep Reinforcement Learning (DRL). No specific real-world application is concerned. Our study on the churn phenomenon may help us better understand and control the behaviours of DRL agents, which can have a positive impact in general on the applicability and safety of DRL techniques.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-0490

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All the environments, data and codes of baseline methods used in this paper are publicly available on Github. We cited the original papers and provided the URLs to the assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [Yes]

    Justification: We provide the necessary details for implementing our proposed method in this paper. We will provide a README document alongside the code to release after the review process.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

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

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

https://doi.org/10.52202/079017-0490

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.