The Ladder in Chaos: Improving Policy Learning by Harnessing the Parameter Evolving Path in A Low-dimensional Space

Hongyao Tang[1,2]*, **Min Zhang**[1], **Chen Chen**[3], **Jianye Hao**[1]
[1]College of Intelligence and Computing, Tianjin University
[2]Mila, Université de Montréal
[3]Department of Automation, Tsinghua University

Abstract

Knowing the learning dynamics of policy is significant to unveiling the mysteries of Reinforcement Learning (RL). It is especially crucial yet challenging to Deep RL, from which the remedies to notorious issues like sample inefficiency and learning instability could be obtained. In this paper, we study how the policy networks of typical DRL agents evolve during the learning process by empirically investigating several kinds of temporal change for each policy parameter. In popular MuJoCo and DeepMind Control Suite (DMC) environments, we find common phenomena for TD3 and RAD agents: (1) the activity of policy network parameters is highly asymmetric and policy networks advance monotonically along a very limited number of major parameter directions; (2) severe detours occur in parameter update and harmonic-like changes are observed for all minor parameter directions. By performing a novel temporal SVD along the policy learning path, the major and minor parameter directions are identified as the columns of the right unitary matrix associated with dominant and insignificant singular values respectively. Driven by the discoveries above, we propose a simple and effective method, called Policy Path Trimming and Boosting (PPTB), as a general plug-in improvement to DRL algorithms. The key idea of PPTB is to trim the policy learning path by canceling the policy updates in minor parameter directions, and boost the learning path by encouraging the advance in major directions. In experiments, we demonstrate that our method improves the learning performance of TD3, RAD, and DoubleDQN regarding scores and efficiency in MuJoCo, DMC, and MinAtar tasks respectively.

1 Introduction

Deep Reinforcement Learning (DRL) is far from well understood, although its great potential has been demonstrated with a lot of achievements in different practical problems [Badia et al., 2020, Shah et al., 2022, Fawzi et al., 2022, Degrave et al., 2022, OpenAI, 2022]. Consistent efforts are made to gain a better understanding of the learning dynamics of RL agents. Different from the studies with tabular [Sutton and Barto, 1988] and linear approximation [Ghosh and Bellemare, 2020], understanding the learning dynamics of DRL agents is challenging. The difficulty comes from the complex interplay between Deep Learning models and RL algorithms, and further escalates when only limited online interactions or offline logged data are considered.

Recently, there have been a few works that study the learning dynamics of DRL agents from different perspectives. A major stream of works among them studies the co-learning dynamics between representation and DRL functions (usually the value network) [Dabney et al., 2021, Kumar et al.,

*Corresponding author: Hongyao Tang (tang.hongyao@quebec.mila).

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

2021, Lyle et al., 2022, Nikishin et al., 2022, Tang et al., 2022b]. The focus of this stream is that the DRL agent over-shapes its representation toward early experiences and objectives, and becomes less capable of learning for the later learning process. This degradation finally leads to myopic convergence or even divergence. A related work [Sokar et al., 2023] presents a phenomenon called Dormant Neuron in DRL. It reveals that the neurons of typical value networks gradually become inactive during the learning process, leading to the loss of network expressivity. From another angle, a phenomenon called Policy Churn is discovered by [Schaul et al., 2022]. It shows that the greedy policy of a typical value network changes on about 10% of the states after only a single update. These discoveries provide useful insights for understanding the learning behavior of the DRL agent, based on which new methods can be proposed to improve the learning performance of DRL.

In this paper, we aim to unveil the learning dynamics of the policy network during the training process of typical DRL agents. Specifically, we focus on how the parameters of the policy network evolve along the *policy learning path*, i.e., the sequence of historical policy networks. Taking MuJoCo [Brockman et al., 2016] and DeepMind Control Suite (DMC) [Tassa et al., 2018] as typical DRL environments, we conduct a series of empirical investigations on the policy learning path of TD3 [Fujimoto et al., 2018] and RAD [Laskin et al., 2020] agents respectively. From the perspective of accumulated absolute parameter change and a novel temporal SVD view, we present four commonly observed phenomena on the policy learning path. The main message from the phenomena is that the policy networks of DRL agents advance nearly monotonically along only very few major parameter directions and show harmonic-like oscillations on almost all other minor ones.

This drives us to ask the question: *can we make the DRL agent focus on the policy learning along major directions and suppress the oscillation in minor directions?* To this end, we propose a new method called **Policy Path Trimming and Boosting (PPTB)**. The key idea of PPTB is to trim the policy learning path by canceling the policy parameter updates in minor dimensions, and boost the learning path by encouraging the advance in major directions. The method is a simple and general plug-in improvement for DRL agents, which can be implemented with only a few lines of code modification to most DRL methods. Finally, we evaluate the effectiveness of PPTB based on TD3, RAD, and DoubleDQN [van Hasselt et al., 2016] in MuJoCo, DMC, and MinAtar [Young and Tian, 2019] environments.

Key contributions of this work are summarized below: (1) We propose a temporal analysis view to study the policy learning path of typical DRL agents. (2) We present four common phenomena from our empirical investigations, which reveal the policy evolving path in a low-dimensional space. (3) We propose an easy-to-implement and general improvement for general DRL algorithms, which is demonstrated to improve the learning performance of TD3, RAD, and DoubleDQN regarding scores and efficiency in MuJoCo, DMC, and MinAtar environments.

2 Preliminaries

Markov Decision Process (MDP) Consider a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho_0, T \rangle$, defined with a state set $\mathcal{S}$, an action set $\mathcal{A}$, the transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, 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 its policy $\pi : \mathcal{S} \to P(\mathcal{A})$ that defines the distribution over all actions for each state. 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 $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi(s_t)$, $s_{t+1} \sim \mathcal{P}(s_{t+1} \mid s_t, a_t)$ and $r_t = \mathcal{R}(s_t, a_t)$. The state-action value function Q^π is defined as $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$ for all $s, a \in \mathcal{S} \times \mathcal{A}$.

Deep Reinforcement Learning (DRL) With function approximation based on deep neural networks, an RL agent is able to deal with large and continuous state-action space. Conventionally, Q^π can be approximated by Q_ϕ with parameters ϕ typically through minimizing Temporal Difference loss [Sutton and Barto, 1988], i.e., $L(\phi) = \frac{1}{2} \left[Q_\phi(s, a) - \mathbb{E}_{a' \sim \pi(s')} \left(r + \gamma Q_{\bar{\phi}}(s', a') \right) \right]^2$. A parameterized policy π_θ, with parameters θ, can be updated by taking the gradient of the objective, i.e., $\theta' \leftarrow \theta + \eta \nabla_\theta J(\pi_\theta)$ with a learning rate η. Therefore, starting from a initial policy π_{θ_1}, the DRL agent proceeds along the learning process and obtains a sequence of policies $\{\pi_{\theta_1}, \pi_{\theta_2}, \dots\}$. For two representative DRL algorithms, Deterministic Policy Gradient (DPG) theo-

rem [Silver et al., 2014] is often used to update a deterministic policy with the gradient: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^{\pi_\theta}}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q_\phi(s,a)|_{a=\pi_\theta(s)}\right]$, where ρ^{π_θ} is the discounted state distribution under policy π_θ; Soft Actor-Critic (SAC) [Haarnoja et al., 2018] updates a stochastic policy with the gradient: $\nabla_\theta \hat{J}(\pi_\theta) = \mathbb{E}_{s\sim\rho^{\pi_\theta}}\left[\nabla_\theta \log \pi_\theta(a|s) + (\nabla_a \log \pi_\theta(a|s)) - \nabla_a Q_\phi(s,a))\nabla_\theta f_\theta(\epsilon;s))|_{a=f_\theta(\epsilon;s)}\right]$ (with noise ϵ and implicit function f_θ for re-parameterization), based on the maximum-entropy objective $\tilde{J}(\pi_\theta) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t + \xi \mathcal{H}(\pi_\theta(\cdot|s)]$ where ξ is the temperature of entropy term.

3 Phenomena on DRL Policy Learning Path

In this section, we conduct empirical investigations on the typical DRL policy learning process. In specific, we use official codes of TD3 [Fujimoto et al., 2018] and RAD [Laskin et al., 2020] for OpenAI MuJoCo [Brockman et al., 2016] continuous control environments and visual-input DeepMind Control (DMC) Suite [Tassa et al., 2018] environments respectively. Both of them use a policy architecture of two-layer MLP (where the output layer is viewed as *Layer 3*). Note that we exclude the convolution layers for image representation in RAD here and focus on the policy part [Chung et al., 2019]. All the experimental details can be found in Appendix D.

First of all, we call the sequence of policies $\{\pi_1, \pi_2, \ldots, \pi_n\}$ or policy parameters $\{\theta_1, \theta_2, \ldots, \theta_n\}$ obtained during the learning process of a typical DRL algorithm as *policy learning path* or *policy parameter path* throughout this paper. The policy parameters θ_i is an m-dimensional vector $\theta_i = [\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,m}] \in \mathbb{R}^m$. In the following, we investigate how policy network parameters evolve along the policy learning path from different angles.

3.1 Policy Parameter Change and Detour

We start by tracking the absolute change amount of each parameter in policy network. Given any $j \in \{1, 2, \cdots, m\}$, we first introduce three quantities for our following investigations.

- Accumulated Parameter Change: $\Delta_j^{\text{apc}}(\{\theta_i\}_{i=1}^n) \triangleq \sum_{i=1}^{n-1} |\theta_{i+1,j} - \theta_{i,j}|$.
- Final Parameter Change: $\Delta_j^{\text{fpc}}(\{\theta_i\}_{i=1}^n) \triangleq |\theta_{n,j} - \theta_{1,j}|$.
- Parameter Update Detour Ratio: $r_j^{\text{pud}}(\{\theta_i\}_{i=1}^n) \triangleq \frac{\sum_{i=1}^{n-1} |\theta_{i+1,j} - \theta_{i,j}|}{|\theta_{n,j} - \theta_{1,j}|}$.

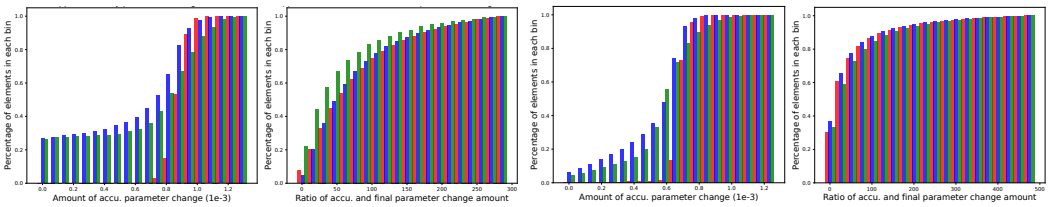
We use $\Delta_j^{\text{apc}}, \Delta_j^{\text{fpc}}, r_j^{\text{pud}}$ as abbreviations when the context is clear and write (Δ_j^{apc}), (Δ_j^{fpc}), (r_j^{pud}) as the vectors consisting of the quantities from all parameters. It can be derived that the larger r_j^{pud} is, the more detours there are in the evolving path of parameter j. For each environment, we run TD3 or RAD for three trials and collect the policies along the learning process at intervals. We then calculate vectors (Δ_j^{apc}), (r_j^{pud}) and plot their CDF histograms in a layer-wise manner (i.e., Layer 1,2,3). The results are shown in Fig. 1. The results are similar in almost all other MuJoCo and DMC environments, and the complete figures can be found in Appendix D.1.

We begin by assessing the magnitude of accumulated changes in the policy parameters. CDF histograms of vector (Δ_j^{apc}) (as shown in left panels of Fig. 1 (a), (b)) indicate that a certain portion of parameters (ranging from 10% to 40% across environments) in the second and output layers experience minor accumulated changes. In contrast, the cumulative changes for nearly all parameters in the first layer are significantly higher, with thresholds around 0.7 for Hopper and 0.6 for walker-walk. This observation marks our initial key finding.

Phenomenon 1.1 (Parameter Change Asymmetry): There is a significant discrepancy in the change amount among policy parameters. The second and the output layers have similar patterns which differ from that of the first layer.

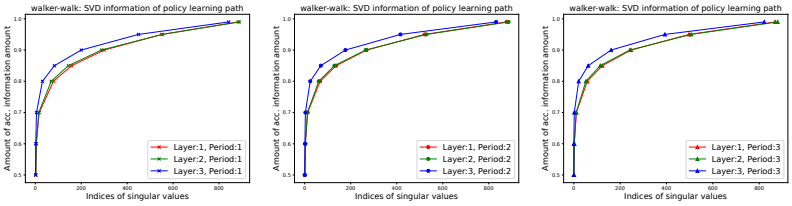
Further discussion about Phenomenon 1.1 can be found in Appendix D.1.

Moving forward, we examine how much each policy parameter detours from its initial value to its final value. The CDF histograms of (r_j^{pud}) (as shown in the right panels of Fig. 1 (a) and (b)) indicate that a significant proportion of parameters exhibit substantial detour ratio, which we identify as the second phenomenon.

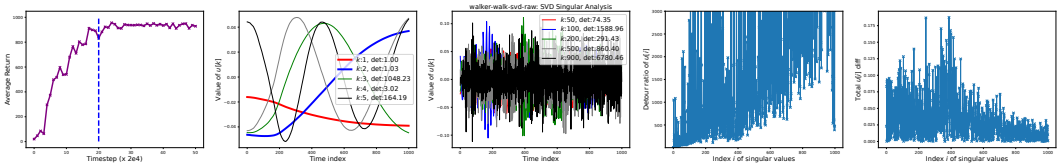


(a) MuJoCo/Hopper (b) DMC/walker-walk

Figure 1: **Policy Parameter Change and Detour** on policy paths obtained by TD3 on MuJoCo/Hopper and RAD on DMC/walker-walk. We use the colors (■ ■ ■) to denote Layer 1,2,3 respectively. Each subfigure contains: (*left*) the CDF histogram of the elements in the vectors $(\Delta^{\text{apc}}_{j\in\text{Layer } k})$, and (*right*) the CDF histograms of the elements in the vectors $(r^{\text{rup}}_{j\in\text{Layer } k})$ for $k \in \{1,2,3\}$. Only upper 80% values according to (Δ^{apc}_j) are taken to plot the histogram for (r^{rup}_j) for meaningful analysis; extreme elements in (r^{rup}_j) are neglected for clarity. See **Phenomenon 1.1** and **2.1** for conclusions.



(a) Singular Value Information Curves



(b) SVD Left Unitary Matrix (U) Investigations

Figure 2: **Temporal SVD Analysis** of policy paths obtained by RAD on DMC/walker-walk. *(a)* Curves of $\mathbb{D}(\beta)$ against various threshold β for the three layers and three periods (i.e., early, middle, later) of the learning process for a temporal view. X-axis represents $\mathbb{D}(\beta)$ and y-axis represents different β. *(b)* The second and third panels depict the curves of $u_{*,k}$, with each curve illustrating the evolving path of the k-th coordinate. The fourth and fifth panels display the curves of detour ratio (r^{pud}_i) and final change (Δ^{fpc}_i). The X-axis represents the index i of the singular values.

> **Phenomenon 2.1 (Parameter Update Detour):** There are severe detours in policy parameter update. All the three layers show similar patterns.

It can be easy to consider that such detours or oscillations in policy parameter updates can be mainly attributed to noisy policy gradients. This also reveals the correlation between policy parameters where most of them can not be updated independently.

3.2 Temporal SVD Analysis of Policy Learning Path

The two phenomena observed above naturally raise further questions: 1) The asymmetry of parameter change shown by Phenomenon 1.1 indicates the imbalanced importance of parameters. Can we *rule the important parameters or parameter directions off from the less important ones*? 2) For the detour shown by Phenomenon 2.1, can we *identify the difference among parameters or parameter directions in their detour behaviors*? And do important parameters or parameter directions detour less?

In this section, we make use of a *temporal* viewpoint on policy path. We introduce Temporal Singular Value Decomposition (SVD) as our main tool for the following investigations. Given a policy path $\{\theta_1, \theta_2, \ldots, \theta_n\}$ with $\theta_i = [\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,m}]$, where m ($\geq 10^5$ usually) is the dimenionality of

policy parameters. We then stack the parameter vectors along the policy path and form a temporal policy parameter matrix, for which standard SVD can be performed:

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_n\end{bmatrix} = \begin{bmatrix}\theta_{1,1} & \dots & \theta_{1,m}\\ \theta_{2,1} & \dots & \theta_{2,m}\\ \vdots & \ddots & \vdots\\ \theta_{n,1} & \dots & \theta_{n,m}\end{bmatrix} = \underbrace{\begin{bmatrix}u_{1,1} & \dots & u_{1,d}\\ \vdots & \ddots & \vdots\\ u_{n,1} & \dots & u_{n,d}\end{bmatrix}}_{U} \underbrace{\begin{bmatrix}\sigma_1 & \dots & 0\\ \vdots & \ddots & \vdots\\ 0 & \dots & \sigma_d\end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix}v_{1,1} & \dots & v_{1,m}\\ \vdots & \ddots & \vdots\\ v_{d,1} & \dots & v_{d,m}\end{bmatrix}}_{V^\top}$$

where U, V are left and right unitary matrices, the singular values are indexed in decreasing order with $d = \min(n, m)$. For the convenience of expression, we use $u_{i,*}$ for the i-th row vector of U and use $u_{*,j}$ for the j-th column vector (and the same way for $V^\top$). The temporal SVD offers us **an angle to view how policy evolves in a lower-dimensional space**: we can now take the row vectors of U as *new d-dimensional coordinates* for policies along the policy path (i.e., $u_{i,*}$ for the i-th policy) regarding the scaling vector $\texttt{Diag}(\Sigma)$ and the parameter subspace spanned by the row vectors of $V^\top$, i.e., $\texttt{Span}(\{v_{1,*}^\top, v_{2,*}^\top, \dots, v_{d,*}^\top\}) \subseteq \mathbb{R}^d$. In the following, we also call $\{v_{i,*}\}$ as *SVD directions*. For any index $j < d$, it is obvious that $u_{*,j}$ is the evolving path of the j-th new coordinate.

To answer the questions listed at the beginning of this subsection, we introduce two more quantities:

- Singular Value Information Amount for a dimensionality number $k \in \{1, \dots, d\}$: $a_k = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{d} \sigma_i}$.
- SVD Major Dimensionality regarding an information threshold $\beta \in (0, 1]$: $\mathbb{D}(\beta) = \min\{k \in \{1, \dots, d\} : \alpha_k \geq \beta\}$.

Note that $d = \mathbb{D}(1)$ and $\mathbb{D}(0.99)$ recovers *approximate rank* used in a few recent works [Yang et al., 2020, Kumar et al., 2021, Lyle et al., 2022]. Additionally, a smaller $\mathbb{D}(\beta)$ implies fewer dominant SVD directions when β is held constant.

We also use the policy path data obtained by TD3 on MuJoCo and RAD on DMC as in Sec. 3.1. First, we plot $\mathbb{D}(\beta)$ against various threshold candidates $\beta \in \{0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.95, 0.99\}$. In addition to a layer-wised manner, we consider three periods (i.e., early, middle, and later) of learning process for a temporal view. The results for DMC/walker-walk are shown in Fig. 2(a). The results are similar in almost all other MuJoCo and DMC environments and the complete figures can be found in Appendix D.2.

The following observations can be made: (i) Within the range of β from 0.5 to 0.8, we observe that $\mathbb{D}(\beta)$ for all three layers in both environments consistently remain near 0. This observation suggests that the policy learning path primarily proceeds within a low-dimensional subspace of the entire parameter space, particularly within the dimensions spanned by the most dominant SVD directions. This provides additional insights into Phenomenon 1.1, elucidating which parameters undergo more significant changes. (ii) The curves of the second layer show a more pronounced steepness near $\beta = 0$, indicating that the parameters in this layer evolve within a subspace characterized by fewer dominant SVD directions compared to those in other layers. This observation also matches the results in Fig. 1, as the second layer is often more over-parameterized than the other layers. (iii) As training advances from period 1 to period 3, the curve's steepness around $\beta = 0$ correspondingly increases, revealing a growing concentration on a diminishing number of dominant singular values. The higher concentration of later periods is easy to understand since policy improvement gradually becomes slower. We can summarize these observations as our third phenomenon.

Phenomenon 1.2 (Singular Value Information Concentration): The singular value information is highly concentrated on the first several singular values. The concentration is more evident for the second layer and later periods.

Further, regarding the second question raised in the beginning, we expect to find some correlation between the policy parameter path in the low-dimensional subspace obtained by SVD and the improvement of policy performance. To this end, in Figure 2(b), we examine the evolving path of the k-th coordinate by plotting coordinate vectors $u_{*,k}$ (panels 2, 3). We further present the curves of detour ratio (r_i^{pud}) (panel 4) and final change (Δ_i^{fpc}) (panel 5) with respect to the singular value index

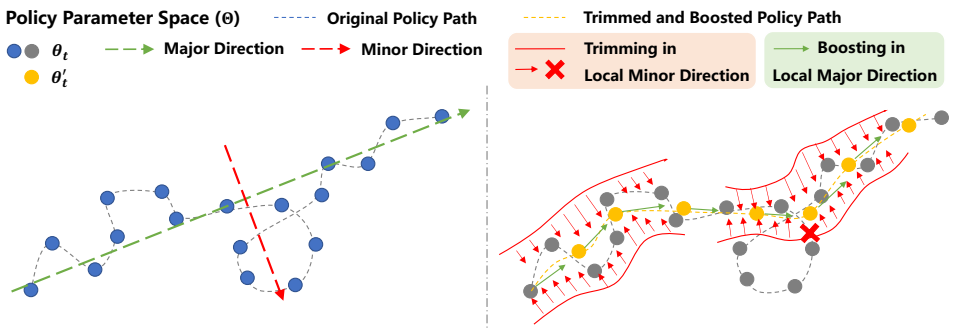


Figure 3: A conceptual illustration of Policy Path Trimming and Boosting (PPTB). The 2D view of an exemplary policy learning path in policy parameter space *(left)* and the corresponding policy learning path improved by PPTB *(right)*.

i. These results are against the policy performance curve (panel 1), and we focus on only the period with significant policy improvement, i.e., the left of the blue dashed vertical line.

According to the results in Fig. 2(b), we observe the final phenomenon.

Phenomenon 2.2 (Policy Path in Major and Minor SVD Directions): The extent of the detour becomes increasingly pronounced as the coordinate index k increases (from the major SVD directions to the minor ones). Specially, $u_{*,1}$ is almost monotonic while $u_{*,k}$ shows harmonic-wave-like changes with overall increasingly higher frequencies for $k \geq 2$.

It is surprising to observe Phenomenon 2.2. Intuitively, it indicates that the policy parameter path proceeds monotonically along one major direction and oscillates in other directions with frequencies inversely proportional to singular values. This empirically supports our hypothesis that important (i.e., major) parameter directions detour less while insignificant directions detour severely. Since typical policy gradients are derived regarding approximate value estimates, we suggest that the dynamics of policy parameters may be closely related to recent studies on the learning dynamics of value function [Lyle et al., 2021, 2022]. We leave it as a major direction of future work.

For a brief summary, till now we have observed that the policy parameter path evolves mainly in a low-dimensional parameter subspace (spanned by $\{v_{i,*}\}$). This is commonly seen in our empirical investigations for typical DRL algorithms in popular environments with proprioceptive or visual observations. A natural idea is: why not let the agent focus on the policy update in the parameter subspace, by *following the major SVD directions and neglecting the insignificant SVD directions*? Moreover, somewhat excitingly, it seems that $u_{*,1}, u_{*,2}$ has a strong correlation to policy performance. We are curious about whether it is possible to *leverage the correlation to boost the learning process*. In the next section, we study on these points along with the proposal of Policy Path Trimming and Boosting.

4 Policy Path Trimming and Boosting

Driven by the phenomena we discovered in the previous section, we propose a simple and effective method, called Policy Path Trimming and Boosting (**PPTB**), as a general plug-in improvement to DRL algorithms. In the following, we introduce the details of the two components of PPTB, i.e., Policy Path Trimming (Sec. 4.1) and Policy Path Boosting (Sec. 4.2), and then the general implementation of DRL with PPTB (Sec. 4.3).

4.1 Policy Path Trimming

As summarized in Phenomenon 1.2, we have observed that the policy path mainly evolves in a low-dimensional parameter subspace with a large proportion of singular value information concentrated in the first several singular values. Our first idea is to truncate the parameter change in minor SVD directions and only keep the change in major ones based on the Temporal SVD of the policy path. We call this method as Policy Path Trimming (PPT).

Given a policy path $\{\theta_1, \theta_2, \ldots, \theta_n\}$ and the number of major directions to remain r_t (usually $\ll d = \min(n, m)$). For a policy with original parameters θ_i, PPT reconstructs policy parameters by only taking into consideration the first r_t SVD directions:

$$\tilde{\theta}_i = \underbrace{[u_{i,1} \quad \ldots \quad u_{i,r_t}]}_{u_{i,1:r_t}} \underbrace{\begin{bmatrix} \sigma_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_{r_t} \end{bmatrix}}_{\Sigma[1:r_t]} \underbrace{\begin{bmatrix} v_{1,*} \\ \vdots \\ v_{r_t,*} \end{bmatrix}}_{V^\top[1:r_t]} \tag{1}$$

Note that we use $\cdot, \cdot$ in subscripts and $[\cdot, \cdot]$ to denote the slices of vector and matrix respectively.

A conceptual illustration of PPT is shown in Fig. 3. As shown by the red arrows and cross, PPT trims the policy parameter update in minor SVD directions and enforces the policy path proceeds in major directions, i.e., in the subspace $\texttt{Span}(\{v_{1,*}^\top, v_{2,*}^\top, \ldots, v_{r_t,*}^\top\})$. Intuitively, this suppresses the detours and oscillations of parameter update (recall the results in Fig. 2(b)). Therefore, we expect PPT to improve the efficiency of policy learning process in this sense.

One may worry about whether the trimmed policy parameters θ_i' still ensure an effective policy. For sanity check, we compare policy performance between θ_i and θ_i' regarding different choices of r_t. We refer the readers to Table 4 to 9 in Appendix D.3. We found that a small r_t (≤ 128) is sufficient to ensure a valid recovery of policy performance, while increasing r_t shows no significant difference. Somewhat surprisingly, we found r_t (≤ 64) often achieves improved performance after reconstruction, which supports the effectiveness of PPT to some extent. In addition, PPT can be viewed as a particular way to do network resetting to alleviate the primacy bias [Nikishin et al., 2022]. More discussion on this can be found in Appendix B.

4.2 Policy Path Boosting

In addition to suppressing the parameter change in minor directions, we are interested in accelerating policy learning by leveraging the dynamics pattern of major SVD directions as noted in Phenomenon 2.2. In particular, we propose to boost the change of $u_{*,1}, u_{*,2}$ since they show near monotonic changes corresponding to the improvement of policy performance in Fig. 2(b). We call this method as Policy Path Boosting (PPB).

For a policy with original parameters θ_i among the policy path $\{\theta_1, \theta_2, \ldots, \theta_n\}$, PPB modifies θ_n by increasing $u_{n,1}, u_{n,2}$ along the temporal direction as follows with a boosting coefficient p_b:

$$\begin{aligned} \hat{u}_{n,*} &= u_{n,*} + p_b(u_{n,*} - u_{1,*}) \\ \hat{\theta}_n &= \underbrace{[\hat{u}_{n,1} \quad \hat{u}_{n,2} \quad u_{n,3} \quad \ldots \quad u_{n,d}]}_{\text{concat}(\hat{u}_{n,1:2}\ ,\ u_{n,3:d})} \Sigma V^\top \end{aligned} \tag{2}$$

PPB only modifies $u_{n,1}, u_{n,2}$ while keeps other parts unchanged. As illustrated by the green arrows in Fig. 3, Intuitively, PPB boosts the policy parameter update in the first two major directions to accelerate the improvement of policy performance.

4.3 DRL with PPTB

Now, we are ready to propose PPTB as a combination of PPT and PPB. Formally,

$$\theta_i' = \text{concat}(\hat{u}_{i,1:2}\ ,\ u_{i,3:r_t})\Sigma[1:r_t]V^\top[1:r_t]. \tag{3}$$

Although we can perform PPTB for any policy on an arbitrary policy path, in practice we consider the policy path that consists of the historical policies within a recent window and the current policy, and we perform PPTB for the current policy.

Apparently, PPTB is algorithmic-agnostic. For almost all off-the-shelf policy-based DRL algorithms, PPTB can be implemented and incorporated by adding the following three steps to the conventional policy update scheme: (1) Initialize a policy buffer and store the parameters of the current policy at intervals along the policy learning process; (2) On certain occasions, retrieve the recent policy parameter path and perform Temporal SVD; then do policy path trimming and boosting for the current policy. (3) Load the consequent policy parameters processed by PPTB back to the current policy.

In our practical implementation used by our experiments, we only make the modifications of about 10-line core codes in the official implementations of TD3 [Fujimoto et al., 2018] and RAD [Laskin et al., 2020]. We refer the readers to Algorithm 1 in the appendix for a pytorch-like pseudocode.

5 Experiments

5.1 Performance Evaluation of PPTB

Setups To evaluate the performance of our proposed method PPTB, we consider the continuous control environments in OpenAI MuJoCo [Brockman et al., 2016] and DeepMind Control Suite (DMC) [Tassa et al., 2018], which cover both proprioceptive and visual observation. We use TD3 [Fujimoto et al., 2018] and RAD [Laskin et al., 2020] as the base algorithms for MuJoCo and DMC respectively, due to their simplicity and effectiveness. We use the official codes of TD3 and RAD and modify them according to Algorithm 1 to implement PPTB with no other change to the original implementation. Our code is available in `https://github.com/bluecontra/PPTB`.

Moreover, we also evaluate the performance of PPTB for value-based RL, although the greedy policy is implicitly derived from the value networks. Specifically, we use three tasks from MinAtar [Young and Tian, 2019] and use DoubleDQN [van Hasselt et al., 2016] as the value-based RL baseline. More implementation details are provided in Appendix C.

Results for AC Methods We train the agent for 1 million timesteps and evaluate the agent every 5k timesteps for each agent-environment configuration. We run each configuration with six random seeds. We consider two evaluation metrics: (1) Score: the *best average* evaluation returns (across multiple runs) over the course of learning, which is used by TD3 [Fujimoto et al., 2018]; (2) AUC: the *mean of average* evaluation returns over the course of learning, which is also used in [Kumar et al., 2022]. The former cares about *effectiveness* while the latter measures *efficiency* and *stability*, which is also significant to the practical use of DRL algorithms.

For MuJoCo environments, we report Score and AUC for 1M time step training; for DMC, we report for 100k, 500k as usually done in prior works [Laskin et al., 2020]. For the convenience of comparing across different return scales, we report the comparative improvement (↗) and the aggregate results by normalizing with a random-agent baseline as 0 and the DRL base algorithm (i.e., TD3 or RAD) as 1. The score of the random-agent baseline is omitted. We report the means and standard deviation errors across six independent trials.

The results for MuJoCo and DMC are reported in Table 1 and 2, respectively. The results show the overall improvements brought by PPTB for both TD3 and RAD, indicating its compatibility and effectiveness. The improvement is more obvious in AUC, which means PPTB improves the learning efficiency and stability of the base algorithms. Somewhat surprisingly, we can observe that PPTB outperforms the base algorithms by a large margin in several environments, e.g., Walker2d and Ant in MuJoCo, and Walker-walk in DMC. To some extent, this reveals the potential of studying and modifying the policy learning path to achieve general improvements to DRL agents.

Table 1: Performance evaluation of Policy Path Trimming and Boosting (PPTB) for TD3 [Fujimoto et al., 2018] in four MuJoCo environments. The learning curves are in Figure 10.

Environments	Metrics	TD3	TD3-PPTB
HalfCheetah	Score	10548 ± 357	11103 ± 60 (5.12%↗)
	AUC	7981 ± 304	8689 ± 76 (8.55%↗)
Hopper	Score	3394 ± 59	3420 ± 54 (0.77%↗)
	AUC	2005 ± 99	2249 ± 114 (12.28%↗)
Walker2d	Score	3406 ± 436	4385 ± 164 (28.76%↗)
	AUC	1977 ± 353	2805 ± 85 (41.92%↗)
Ant	Score	4177 ± 451	5147 ± 449 (22.81%↗)
	AUC	2624 ± 293	3542 ± 392 (34.02%↗)
Aggregate	Score	1.0	1.1436
	AUC	1.0	1.2419

Table 2: Performance evaluation of Policy Path Trimming and Boosting (PPTB) for RAD [Laskin et al., 2020] in four DeepMind Control (DMC) environments. The learning curves are in Figure 11.

Environments	Metrics	RAD (100k)	RAD-PPTB (100k)	RAD (500k)	RAD-PPTB (500k)
finger-spin	Score	553 ± 71	592 ± 55 (7.09%↗)	898 ± 55	970 ± 5 (8.04%↗)
	AUC	201 ± 33	254 ± 40 (26.76%↗)	695 ± 41	767 ± 20 (10.40%↗)
walker-walk	Score	210 ± 48	349 ± 35 (86.87%↗)	923 ± 9	940 ± 6 (1.94%↗)
	AUC	104 ± 22	148 ± 15 (81.48%↗)	583 ± 30	672 ± 11 (16.69%↗)
cartpole-swingup	Score	235 ± 15	268 ± 18 (30.84%↗)	836 ± 12	860 ± 9 (3.38%↗)
	AUC	158 ± 6	189 ± 2 (103.33%↗)	530 ± 18	572 ± 20 (10.44%↗)
cheetah-run	Score	360 ± 8	394 ± 10 (10.11%↗)	574 ± 13	605 ± 10 (5.63%↗)
	AUC	194 ± 5	207 ± 13 (7.64%↗)	428 ± 10	452 ± 6 (5.94%↗)
Aggregate	Score	1.0	1.3372	1.0	1.0474
	AUC	1.0	1.5480	1.0	1.1086

Results for Value-based Methods We run 3M time steps for SpaceInvader and Breakout, and 5M time steps for Seaquest. We reported the means and standard errors of the final episode returns across six seeds in Table 3.

Table 3: Performance evaluation regarding Score of Policy Path Trimming and Boosting (PPTB) for DoubleDQN [van Hasselt et al., 2016] in three MinAtar environments.

Environments	DoubleDQN	DoubleDQN-PPTB
SpaceInvader	77.63 ± 6.85	$\mathbf{99.08 \pm 6.76}$
Seaquest	11.00 ± 1.70	$\mathbf{25.43 \pm 4.38}$
Breakout	28.85 ± 4.01	$\mathbf{32.33 \pm 2.92}$

The results show that PPTB also improves the learning performance of DoubleDQN somewhat significantly in MinAtar tasks with discrete actions. Note that for DoubleDQN, value networks are learned to derive the greedy actions. Intuitively, the learning dynamics of the value learning path and the policy learning path in AC methods are different as typical TD learning adopts bootstrapping and semi-gradient [Sutton and Barto, 1988]. We view our experimental results here as preliminary evidence of the effectiveness of PPTB for value-based methods, which also inspired us to further investigate this point in the future.

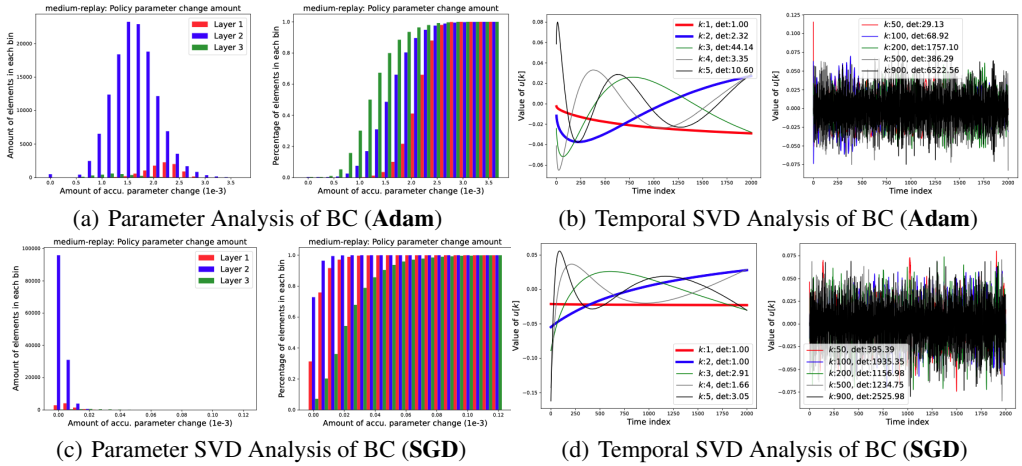
5.2 Investigation on Policy Learning Path of Behavior Clone

The policy learning path is influenced by multiple factors, including, policy learning algorithm, distribution of training data, network architecture, optimizer, etc. To expand our study on typical online RL a bit, we investigate the policy learning path of Behavior Cloning (BC) with Adam and vanilla SGD on halfcheetah-medium-replay, an offline dataset in D4RL [Fu et al., 2020]. This additional setting provides two comparison angles, i.e., RL v.s. BC, Adam v.s. SGD (where using BC and the same offline dataset fixes the influence of data distribution). Similar to our investigation in Sec. 3, we perform parameter analysis and Temporal SVD analysis to the logged policy paths through learning under different configurations. The results are presented in Figure 4 and 12.

For the parameter analysis, we can observe that BC and RL show different distributions of network parameter update amount. Compared with the RL cases reported in Figure 5(a), the parameter update amount is more Gaussian distribution like for BC (Adam) in Figure 4(a). In addition, by comparing the parameter analysis of BC (Adam) and BC (SGD), somewhat surprisingly, we can observe that the asymmetry of parameter update amount turns out to be more severe in BC (SGD). The momentum mechanism used in Adam leads to a relatively more evenly distributed network parameter update amount, while SGD has a large number of minor updates (also reflected by the flat curve in Figure 4(d), the first column) and a long tail in distribution.

For the Temporal SVD analysis, we can observe that BC (Adam), BC (SGD) and the RL cases show different patterns in terms of temporal SVD evolvement. BC policies show wavelet-like oscillations (Figure 4(b)), where the amplitude of the oscillations seems to decrease throughout training; while the oscillations of RL are more harmonic-wave-like (Figure 8(a)).

These observations reveal that low-dimensional policy paths can exist in many policy learning problems with different features influenced by multiple factors. We expect further studies in this direction in the future.



(a) Parameter Analysis of BC (**Adam**) (b) Temporal SVD Analysis of BC (**Adam**)

(c) Parameter SVD Analysis of BC (**SGD**) (d) Temporal SVD Analysis of BC (**SGD**)

Figure 4: The parameter and Temporal SVD analysis for Behavior Cloning with Adam and SGD optimizers in D4RL halfcheetah-medium-replay.

6 Related Works

In the topic of general Deep Learning, there have been a large number of efforts devoted consistently during the past decade for the purpose of understanding the learning dynamics and behaviors of deep neural networks, e.g., Neural Tangent Kernel (NTK) [Jacot et al., 2018] and Lazy Training [Chizat et al., 2018]. Recently, there have been a few works that study the learning dynamics of DRL agents from different perspectives. Among them, a major stream of works study the co-learning dynamics between representation and RL functions (i.e., value network or policy network) [Kumar et al., 2021, Lyle et al., 2022, Nikishin et al., 2022]. The focus of this stream is that the DRL agent over-shapes its representation towards early experiences and objectives (e.g., TD targets of early policies) and becomes less capable for the later learning process. Such a degradation becomes more severe and detrimental gradually due to the non-stationary learning nature, finally leading to myopic convergence or even divergence. In this work, we study the learning dynamics of typical DRL policy networks, mainly from the angle of policy network parameters. To the best of our knowledge, we are almost the first to study how the parameters of practical DRL policy networks evolve.

To our knowledge, there are some works that conduct their study from the angle of the path of learning. Dabney et al. [2021] takes inspiration from the study in [Bellemare et al., 2019] and proposes the concept of the *value improvement path*. With this concept, a new method for representation learning is proposed by approximating the value functions along the path. From another angle, Tang et al. [2022a] studies the generalization along the policy learning path. Through learning a low-dimensional policy embedding, an improved generalized policy iteration [Sutton and Barto, 1988] that leverages policy generalization is proposed. Closely related to our work, a very recent work [Schneider et al., 2024] demonstrates the existence of gradient subspaces in deep policy gradient algorithms with experimental investigations. In this paper, we investigate the dynamics of policy network parameters from the perspective of Temporal SVD. Moreover, we propose a new method that leverages the parameter evolving path in the low-dimensional space, which demonstrates the potential for improving learning. In essence, the policy learning path is a special scenario of stochastic gradient descent (SGD) in DRL, thus this work is related to studies on the dynamics of SGD or optimization [Gur-Ari et al., 2018, Gressmann et al., 2020, Li et al., 2023] in more general speaking.

7 Conclusion

In this paper, we aim to unveil the learning dynamics of policy network over the training course of typical DRL agents, mainly from an empirical perspective. Focusing on the concept of policy learning path, we conduct a series of empirical investigations and summarize four common phenomena, revealing that the policy learning path of typical DRL agents evolves mainly in a low-dimensional space with monotonic and waggling changes in major and minor parameter directions respectively. Driven by our discovery, we propose a simple method, called Policy Path Trimming and Boosting (PPTB), as a plug-in improvement to general DRL algorithms. We demonstrate the effectiveness of PPTB based on TD3 and RAD in a few MuJoCo and DMC environments.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant Nos. 92370132).

References

Jordan T. Ash and Ryan P. Adams. On warm-starting neural network training. In *NeurIPS*, 2020.

A. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. In *ICML*, volume 119, pages 507–517, 2020.

Marc G. Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taïga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. In *NeurIPS*, pages 4360–4371, 2019.

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

L. Chizat, E. Oyallon, and F. R. Bach. On lazy training in differentiable programming. In *NeurIPS*, 2018.

W. Chung, S. Nath, A. Joseph, and M. White. Two-timescale networks for nonlinear value function approximation. In *ICLR*, 2019.

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*, pages 7160–7168, 2021.

J. Degrave, F. Felici, J. Buchli, M. Neunert, B. D. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. A. Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

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*, volume 80, pages 1582–1591, 2018.

D. Ghosh and M. G. Bellemare. Representations for stable off-policy reinforcement learning. In *ICML*, 2020.

Frithjof Gressmann, Zach Eaton-Rosen, and Carlo Luschi. Improving neural network training in low dimensional random bases. In *NeurIPS*, 2020.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint, arXiv:1812.04754*, 2018.

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*, volume 80, pages 1856–1865, 2018.

A. Jacot, C. Hongler, and F. Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, pages 8580–8589, 2018.

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.

M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. In *NeurIPS*, 2020.

Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. Low dimensional trajectory hypothesis is true: Dnns can be trained in tiny subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3411–3420, 2023.

C. Lyle, M. Rowland, G. Ostrovski, and W. Dabney. On the effect of auxiliary tasks on representation dynamics. In *AISTATS*, volume 130, pages 1–9, 2021.

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

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

E. Nikishin, M. Schwarzer, P. D'Oro, P. Bacon, and A. C. Courville. The primacy bias in deep reinforcement learning. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 16828–16847, 2022.

OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. URL `https://openai.com/blog/chatgpt/`.

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

Jan Schneider, Pierre Schumacher, Simon Guist, Le Chen, Daniel F. B. Häufle, Bernhard Schölkopf, and Dieter Büchler. Identifying policy gradient subspaces. *arXiv preprint, arXiv:2401.06604*, 2024.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

D. Shah, B. Osinski, B. Ichter, and S. Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *arXiv preprint, arXiv:2207.04429*, 2022.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, pages 387–395, 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.

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*, pages 8441–8449, 2022a.

Hongyao Tang and Glen Berseth. Improving deep reinforcement learning by reducing the chain effect of value and policy churn. *arXiv preprint, arXiv:2409.04792*, 2024.

Y. Tang, Z. D. Guo, P. H. Richemond, B. Á. Pires, Y. Chandak, R. Munos, M. Rowland, M. G. Azar, C. Le Lan, C. Lyle, A. György, S. Thakoor, W. Dabney, B. Piot, D. Calandriello, and M. Valko. Understanding self-predictive learning for reinforcement learning. *arXiv preprint, arXiv:2212.03319*, 2022b.

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.

Y. Yang, G. Zhang, Z. Xu, and D. Katabi. Harnessing structures for value-based planning and reinforcement learning. In *ICLR*, 2020.

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

A Limitations

In this paper, we only provide empirical investigations on the phenomena we present in Sec. 3. Although we observe relatively consistent results for TD3 and RAD across a variety of MuJoCo and DMC environments, which demonstrates the generality to some degree, we have no theoretical support for the observed phenomena at present. A rudimentary thought on this point is to study the learning dynamics of $\nabla_\theta J(\pi_\theta)$, especially of $\nabla_a Q_\phi(s, a)$. This is because the policy path is the accumulation of policy gradient while policy gradients are significantly determined by the landscape of Q-network, which learns concurrently along the process. We believe that recent studies on the learning dynamics of the value function and representation [Lyle et al., 2021, 2022] can be inspiring references.

Still from the empirical perspective, our work is not complete in the sense that we only consider TD3 and RAD (i.e., SAC inside). On-policy policy-based DRL algorithms like PPO [Schulman et al., 2017] may have a very different policy path. Besides, the empirical investigation for more value-based DRL algorithms beyond DQN [Mnih et al., 2015] is missing in this paper. For typical value-based DRL algorithms, we can also treat the value network as a special form of policy and perform PPTB in the same way. Although we provided some preliminary results of applying PPTB to DoubleDQN in our experiments, this is kind of reckless since the natures of policy and value function are different, nor are their learning dynamics. A recent phenomenon called *Policy Churn* discovered in [Schaul et al., 2022, Tang and Berseth, 2024] reveals that the greedy policy induced by a typical value network changes its actions on about 10% states after one update. We leave in-depth studies on the value function path in the future.

For methodology, we propose very simple methods and we believe that there is great potential for more sophisticated methods to be proposed. First, we use standard SVD as the main tool in both our empirical investigation and our methods. We use no acceleration for SVD or other more advanced alternatives to obtain the major and minor directions of the policy path. Moreover, we use fixed dimensionalities and intervals for policy path trimming and boosting. This paper contains no attempts to design adaptive approaches or propose principled algorithms (which may rely on the advances in theoretical results). In addition, smarter hyperparameter selection methods are expected. One another limitation is that our proposed method is not evaluated in sparse-reward environments.

B More Discussion

Policy Path Trimming in the View of Primacy Bias The policy path trimming (PPT) method proposed in this work can be viewed as a special way to do network resetting to alleviate the primacy bias [Nikishin et al., 2022]. The resetting effect of PPT in the minor update directions is achieved by (1) obtaining the transformed parameter space via performing temporal SVD with the historical policies collected in a sliding window, (2) and then (re-)setting (or dropping) the left singular vector value (i.e., u) to zero. More concretely, this differs from the resetting method in [Nikishin et al., 2022] at two points:

- **Resetting to random initialization v.s., Resetting to Zero in the transformed parameter space**: The resetting method in [Nikishin et al., 2022] resets the network parameters to a set of randomly (re-)initialized parameters. In contrast, our method resets the left singular vector value (i.e., u) to zero for the minor temporal SVD directions, i.e., resetting the parameters to zero in the transformed parameter space. One thing to note is that resetting the left singular vector value (i.e., u) to zero does not mean that the network parameters (in the original parameter space) are necessarily zero, because the basis of space is different. A more in-depth analysis of the correlation between them is worth further study in the future.
- **Global resetting v.s., Local resetting**: The resetting method in [Nikishin et al., 2022] is global, as the network parameters are reset to a set of randomly (re-)initialized parameters. Differently, we do resetting according to the temporal SVD parameter space based on a sliding window that includes recent historical policies (thus being local). In this sense, this is similar to Shrink-and-Perturb [Ash and Adams, 2020], which can be viewed as a soft version of the resetting method, that shrinks the network parameters and adds random parameters with a coefficient.

Moreover, from the perspective of plasticity, our method periodically rolls back the plasticity of the network in the directions that are orthogonal to the ones that represent the effective knowledge learned so far.

The Connection between PPTB and Momentum-based Optimization The idea of momentum adopted in modern optimizers is essentially to make use of the information of historical gradients. In this sense, we can establish a connection between momentum and PPTB proposed in our work as the key idea of PPTB is to make use of the latent structure of the historical policies along the policy update path.

Momentum-based optimizers use different kinds of aggregation quantities of the historical gradients (e.g., $m_{t+1} = \beta m_t + \nabla_t$). In our context, PPTB differs at two points: (1) not all the temporal SVD directions are used, and directions with large oscillations are trimmed (note that both SVD and eigendecomposition are techniques that capture the features of curvature); (2) policy path boosting can be viewed as a type of momentum conducted in the major temporal SVD directions. Another difference to notice is that, both our analysis and experimental evaluation are done with Adam which already leverages momentum (except for the analysis for SGD). Thus, our findings are based on the practical optimization of a momentum-based optimizer.

We believe that a further study on this connection with formal description is worthwhile in the future.

C Experiment Details

Hyperparameter Setting For the maintenance of the policy path, we use a FIFO buffer with the size of 2k and 1k for MuJoCo and DMC respectively, and save the policy parameters every 25 mini-batch gradient updates of the policy network. For the hyperparameter of PPTB, the number of major directions to keep (i.e., r_t in Sec. 4.1) is searched in the set $\{8, 16, 32, 64\}$ for each environment. For PPB, we simply always boost the first two SVD directions (i.e., $u_{*,1}, u_{*,2}$ as described in Sec. 4.2) and choose the boosting coefficient p_b from the set $\{0.1, 0.5, 1.0\}$.

For the experiments in MinAtar, we implemented DoubleDQN-PPTB based on the official code of MinAtar[2] by adding several lines of code to collect the historical network parameters of DoublDQN and perform PPTB at intervals through learning. We made no change to the default hyperparameters of DoubleDQN. For DoubleDQN-PPTB, we use $r_t = 64$ and $p_b = 0.1$.

Compute Resource We use Nvidia 3080ti GPU and V100 GPU for our experiments. For instance, each training trial of MuJoCo experiment requires less than 4G memory and about 8G GPU memory.

Note that the main computation cost added by PPTB is the calculation of SVD. In practice, we perform PPBT at sparse intervals (e.g., $t_p \geq 1000$) for the policy parameter matrix with a size $[2000, \approx 1e5]$, and each SVD computation takes less than 1 second with `torch.linalg.svd`. The practical computation cost in wall-clock time is less than 5% of the total training time, which is worth it compared to the benefits SVD brings. Moreover, the temporal SVD operations remain scalable as matrix size increases, thanks to the use of efficient linear algebra libraries.

PyTorch Code Implementation of PPTB We provide a pytorch-like pseudocode of PPTB implementation in Algorithm 1, where the slices of vector and matrix are also in a pytorch-like style. In our practical implementation used by our experiments, we only make the modifications of about 10-line core codes in the official implementations of TD3 [Fujimoto et al., 2018] and RAD [Laskin et al., 2020].

Code implementation of Baseline Algorithms We build our experiment codes based on the official implementation on GitHub for both TD3 [Fujimoto et al., 2018][3] and RAD [Laskin et al., 2020][4]. We do not change the default algorithm implementation or the recommended hyperparameters.

[2] `https://github.com/kenjyoung/MinAtar`

[3] `https://github.com/sfujim/TD3`

[4] `https://github.com/MishaLaskin/rad`

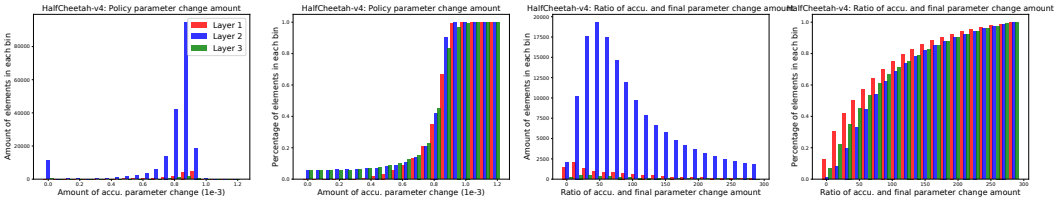
Algorithm 1: DRL with PPTB (Psedudocode in a PyTorch-like style)

```
# env:  environment
# agent:  policy-based DRL agent
# d_p:  policy parameter buffer with size k
# r_t, r_b, p_b:  dimensionality hyperparams of PPTB where r_t ≥ r_b (see Eq. 1 and 2)
# t_s, t_p:  intervals of storing policy and performing PPTB where t_p % t_s == 0

d_p = [agent.policy.params]  # Initialize
for t in range(max_interaction_steps):
  # Typical agent-env interaction (omitted)
  ...
  agent.learn()

  # 1) Store policy parameters at intervals
  If t % t_s == 0
    d_p.append(agent.policy.params)
    d_p = d_p[-k:]

  # 2) Perform Temporal SVD and PPTB
  If t % t_p == 0
    u, sgl, vh = SVD(d_p)
    u_b = (u[-1] - u[0]) * p_b + u[-1]
    u_tb = concat(u_b[:r_b], u[-1,r_b:r_t])
    param_tb = (u_tb * sgl[:r_t]).dot(vh[:r_t])

  # 3) Apply modified params to agent
    agent.policy.load(params_tb)
```

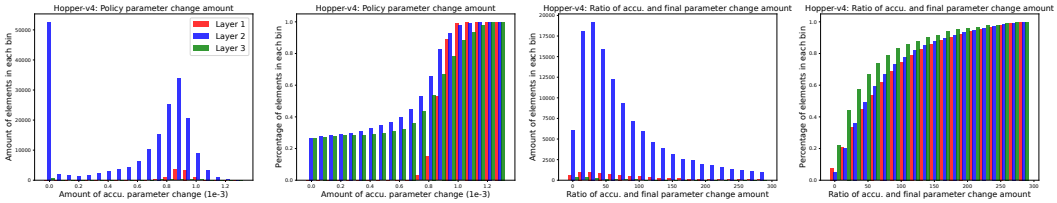
D Additional Experimental Results

D.1 Empirical Investigation on Policy Parameter Change Amount

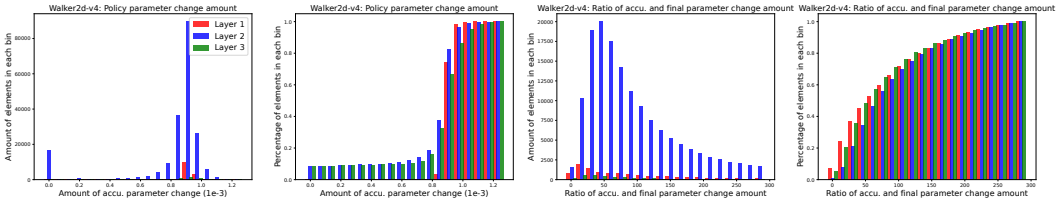
Further discussion about Phenomenon 1.1. One thing to note is that the results reported in Fig. 1 are based on the policy paths of entire learning process, thus being a global view. For a more local view (i.e., a window of recent learning process), the phenomenon of parameter change asymmetry becomes more obvious and more parameters have minor and even no change (i.e., dead parameters). We hypothesize that this could be some practical evidence of lazy training of neural network [Chizat et al., 2018]. Concretely, the phenomenon indicates that a lot of parameters have insignificant gradients $\nabla_a Q(s,a)\nabla_\theta \pi_\theta(s)|_{a=\pi(s)}$ during learning process.



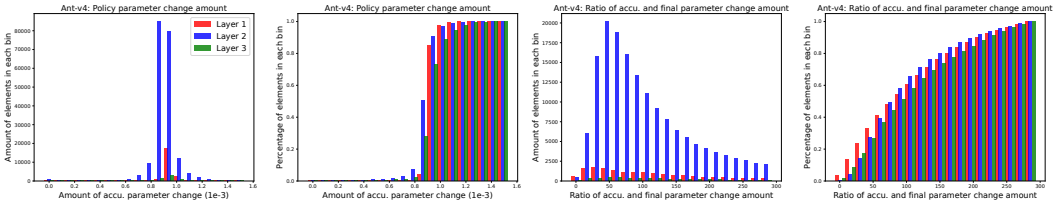
(a) HalfCheetah-v4



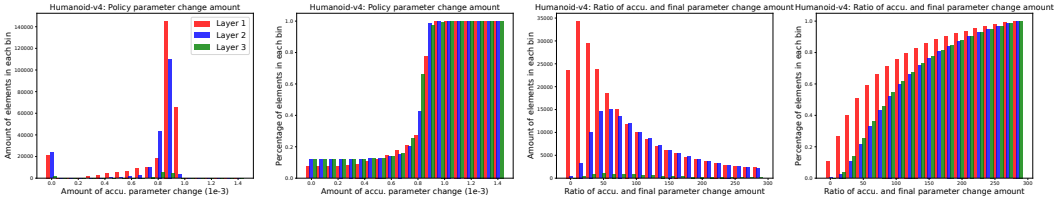
(b) Hopper-v4



(c) Walker2d-v4

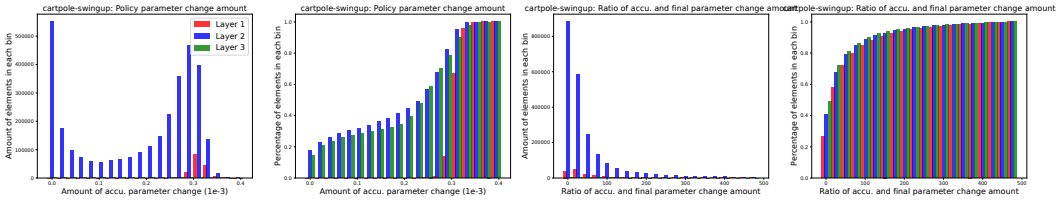


(d) Ant-v4

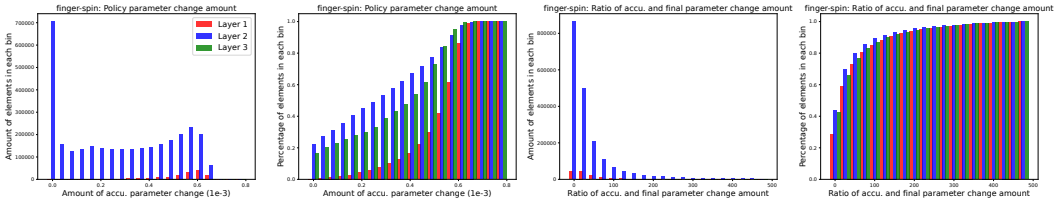


(e) Humanoid-v4

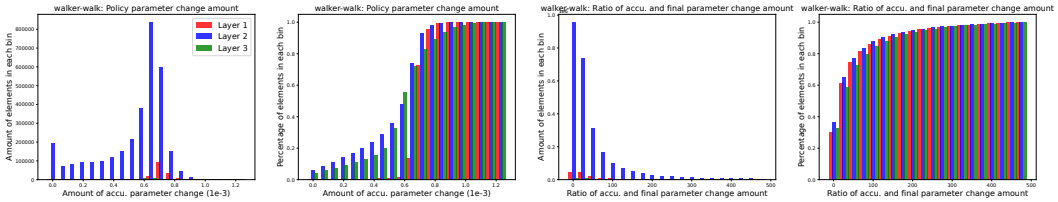
Figure 5: Complete results of empirical investigation on policy parameter change amount in MuJoCo environments.



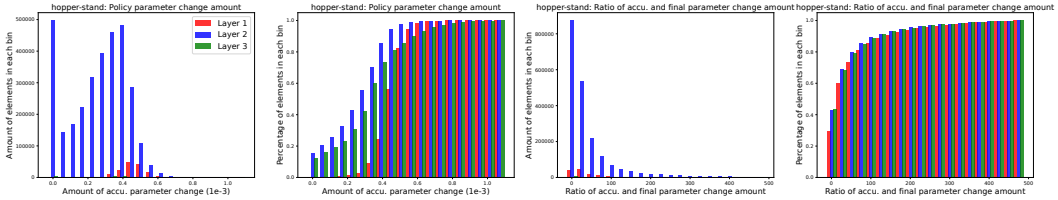
(a) cartpole-swingup



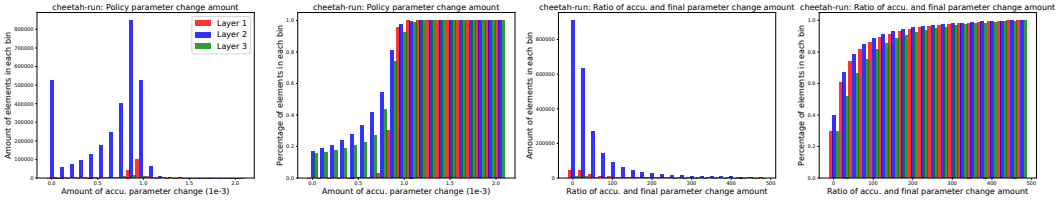
(b) finger-spin



(c) walker-walk



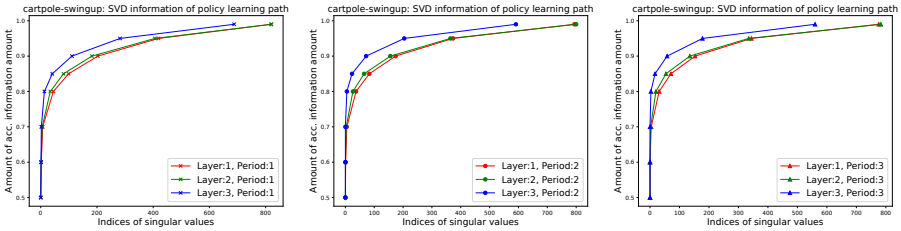
(d) hopper-stand



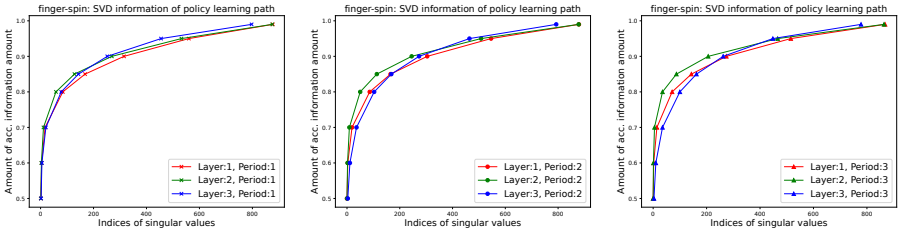
(e) cheetah-run

Figure 6: Complete results of empirical investigation on policy parameter change amount in DMC environments.

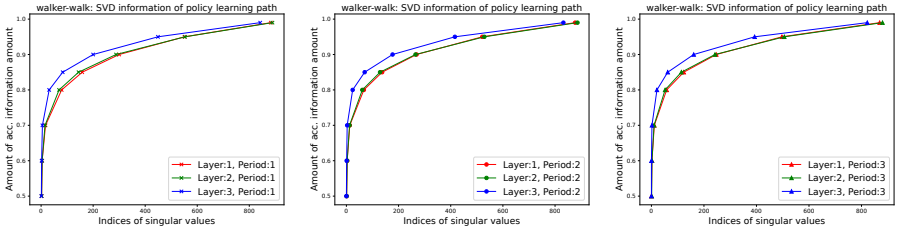
D.2 Empirical Investigation on Policy Learning Path by Temporal SVD



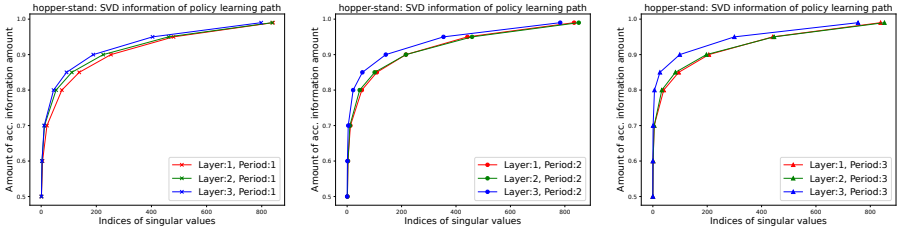
(a) cartpole-swingup



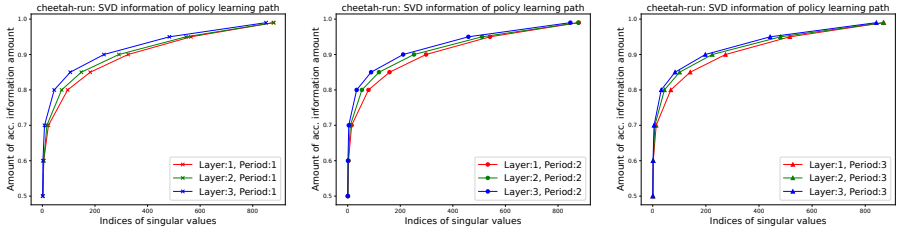
(b) finger-spin



(c) walker-walk

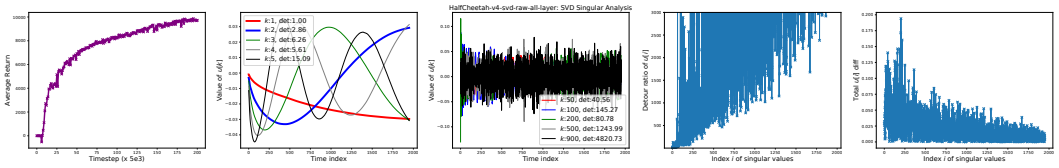


(d) hopper-stand

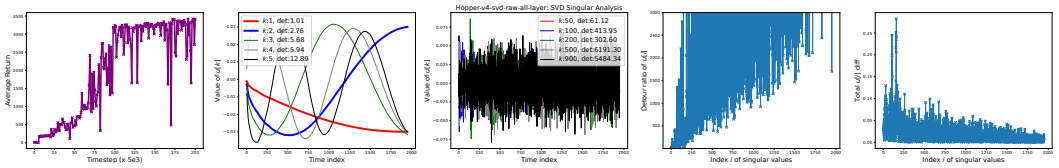


(e) cheetah-run

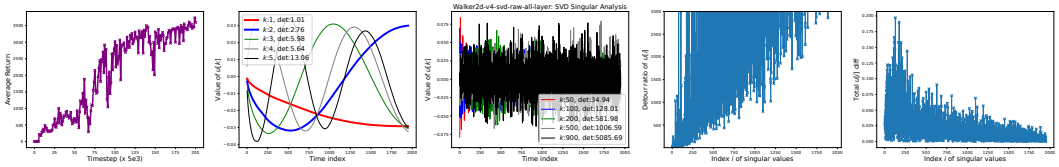
Figure 7: Complete results of the empirical investigation on SVD Information of policy learning path in DMC environments.



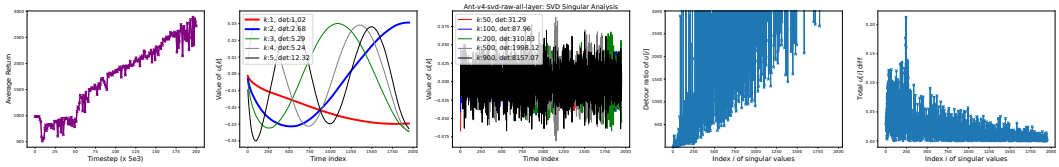
(a) HalfCheetah-v4



(b) Hopper-v4

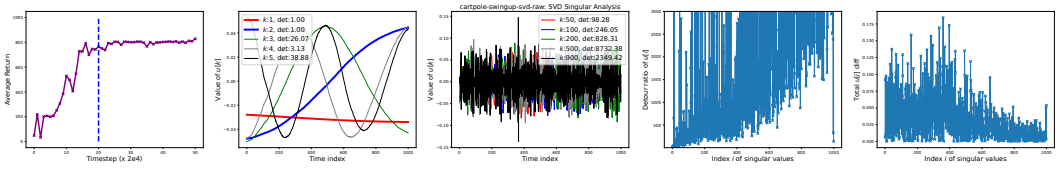


(c) Walker2d-v4

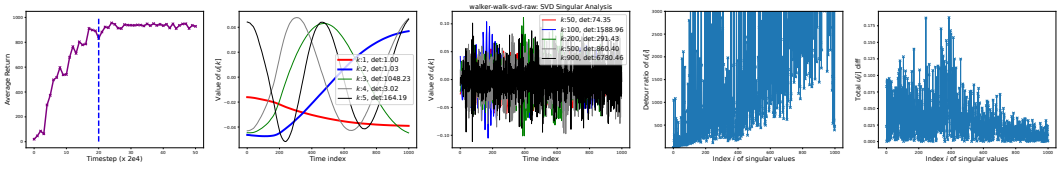


(d) Ant-v4

Figure 8: Complete results of the empirical investigation on SVD left unitary matrix of policy learning path in MuJoCo environments.



(a) cartpole-swingup



(b) walker-walk

Figure 9: Complete results of the empirical investigation on SVD left unitary matrix of policy learning path in DMC environments.

D.3 Empirical Investigation on Temporal SVD Reconstruction of DRL Policies

Table 4: Statistics for Temporal SVD reconstruction of RAD policies in **cartpole-wingup (the first period)** with varying number of major dimensions kept.

NO. OF MAJOR DIM.	AVG($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	AVG($\{\lvert\Delta_{\mathrm{R}}(\theta_i)\rvert\}_i$)	MAX($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	MIN($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)
1	24.24 ± 65.10	43.83 ± 53.90	313.15	-85.48
2	26.68 ± 61.70	37.45 ± 55.83	278.79	-71.03
4	14.47 ± 50.29	34.78 ± 39.10	168.46	-106.50
8	13.86 ± 45.97	32.91 ± 34.96	157.25	-94.24
16	15.06 ± 38.30	27.69 ± 30.45	125.26	-52.95
32	8.98 ± 41.06	29.47 ± 29.97	129.39	-72.51
64	2.15 ± 41.69	28.19 ± 30.78	125.08	-132.06
128	-2.04 ± 45.76	27.95 ± 36.30	120.36	-202.92

Table 5: Statistics for Temporal SVD reconstruction of RAD policies in **cartpole-wingup (the second period)** with varying number of major dimensions kept.

NO. OF MAJOR DIM.	AVG($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	AVG($\{\lvert\Delta_{\mathrm{R}}(\theta_i)\rvert\}_i$)	MAX($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	MIN($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)
1	18.10 ± 41.86	21.01 ± 40.47	265.96	-12.61
2	8.37 ± 24.39	9.70 ± 23.89	167.57	-13.24
4	4.71 ± 21.28	10.07 ± 19.33	129.82	-54.63
8	5.03 ± 24.63	8.28 ± 23.74	164.32	-53.45
16	4.42 ± 18.25	6.96 ± 17.44	123.94	-27.75
32	5.48 ± 35.52	10.54 ± 34.36	238.01	-77.67
64	3.41 ± 18.15	7.35 ± 16.94	116.21	-32.73
128	3.02 ± 11.10	5.51 ± 10.09	65.10	-31.63

Table 6: Statistics for Temporal SVD reconstruction of RAD policies in **finger-spin (the first period)** with varying number of major dimensions kept.

NO. OF MAJOR DIM.	AVG($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	AVG($\{\lvert\Delta_{\mathrm{R}}(\theta_i)\rvert\}_i$)	MAX($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	MIN($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)
1	-61.70 ± 94.46	62.70 ± 93.80	10.00	-393.67
2	2.28 ± 13.87	7.48 ± 11.91	70.00	-15.67
4	0.77 ± 18.79	8.96 ± 16.54	51.33	-106.33
8	0.09 ± 12.15	6.49 ± 10.27	28.33	-62.67
16	-5.06 ± 25.28	9.07 ± 24.13	24.33	-158.00
32	-0.83 ± 7.45	5.77 ± 4.80	20.00	-20.67
64	-0.24 ± 9.50	5.15 ± 7.99	18.67	-52.00
128	1.55 ± 10.02	5.91 ± 8.24	52.33	-22.33

Table 7: Statistics for Temporal SVD reconstruction of RAD policies in **finger-spin (the second period)** with varying number of major dimensions kept.

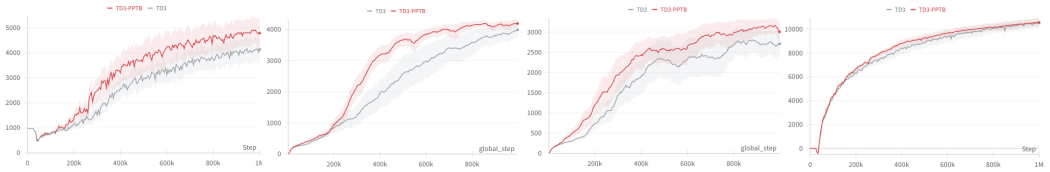
NO. OF MAJOR DIM.	AVG($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	AVG($\{\lvert\Delta_{\mathrm{R}}(\theta_i)\rvert\}_i$)	MAX($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	MIN($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)
1	-10.68 $\pm$ 26.53	19.89 $\pm$ 20.54	45.33	-90.00
2	-5.94 $\pm$ 30.71	11.67 $\pm$ 29.02	26.67	-202.00
4	-0.95 $\pm$ 10.37	6.05 $\pm$ 8.48	23.00	-51.00
8	0.03 $\pm$ 10.39	6.04 $\pm$ 8.46	26.00	-54.33
16	-0.81 $\pm$ 10.20	5.81 $\pm$ 8.42	27.33	-47.00
32	-0.47 $\pm$ 10.63	5.99 $\pm$ 8.80	28.33	-53.67
64	-4.90 $\pm$ 29.45	9.85 $\pm$ 28.19	24.67	-197.00
128	-16.57 $\pm$ 69.42	22.17 $\pm$ 67.84	23.00	-407.67

Table 8: Statistics for Temporal SVD reconstruction of RAD policies in **walker-walk (the first period)** with varying number of major dimensions kept.

NO. OF MAJOR DIM.	AVG($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	AVG($\{\lvert\Delta_{\mathrm{R}}(\theta_i)\rvert\}_i$)	MAX($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	MIN($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)
1	7.20 $\pm$ 116.51	84.63 $\pm$ 80.40	285.06	-298.17
2	61.12 $\pm$ 82.60	72.36 $\pm$ 72.96	347.76	-58.92
4	41.79 $\pm$ 86.74	67.88 $\pm$ 68.29	283.30	-211.50
8	32.80 $\pm$ 77.18	55.46 $\pm$ 62.91	261.98	-144.36
16	25.53 $\pm$ 82.42	62.40 $\pm$ 59.59	233.16	-177.25
32	24.22 $\pm$ 86.47	61.09 $\pm$ 65.81	255.32	-262.19
64	20.31 $\pm$ 77.88	51.96 $\pm$ 61.47	298.37	-225.65
128	12.57 $\pm$ 74.66	56.62 $\pm$ 50.27	240.83	-156.34

Table 9: Statistics for Temporal SVD reconstruction of RAD policies in **walker-walk (the second period)** with varying number of major dimensions kept.

NO. OF MAJOR DIM.	AVG($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	AVG($\{\lvert\Delta_{\mathrm{R}}(\theta_i)\rvert\}_i$)	MAX($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)	MIN($\{\Delta_{\mathrm{R}}(\theta_i)\}_i$)
1	15.03 $\pm$ 52.61	38.35 $\pm$ 39.02	155.56	-149.09
2	21.23 $\pm$ 37.68	30.37 $\pm$ 30.80	145.45	-37.95
4	21.81 $\pm$ 34.98	28.14 $\pm$ 30.12	113.82	-25.32
8	14.12 $\pm$ 44.51	31.12 $\pm$ 34.81	134.84	-163.47
16	6.16 $\pm$ 46.68	32.26 $\pm$ 34.30	100.79	-169.74
32	1.55 $\pm$ 62.75	35.26 $\pm$ 51.93	101.41	-266.15
64	11.91 $\pm$ 38.39	28.99 $\pm$ 27.85	125.69	-75.02
128	14.11 $\pm$ 39.72	28.39 $\pm$ 31.15	124.47	-116.62



(a) Ant-v4 (b) Walker2d-v4 (c) Hopper-v4 (d) HalfCheetah-v4

Figure 10: The learning curves for TD3 (Gray) and TD3-PPTB (Red) in four MuJoCo tasks, with means and standard errors across 6 seeds.

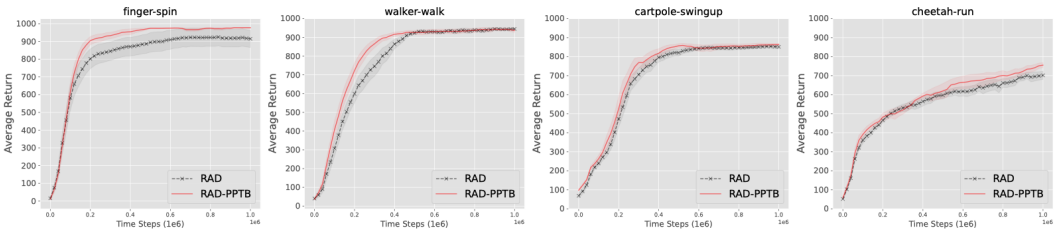
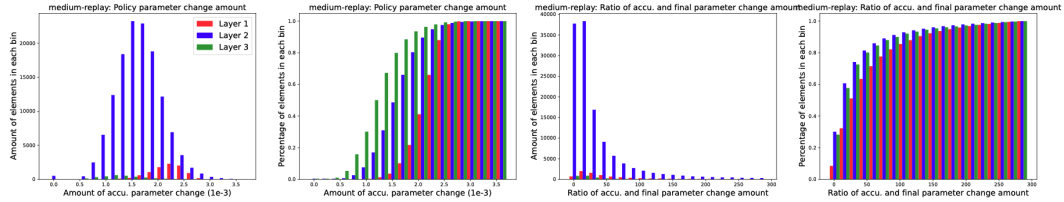
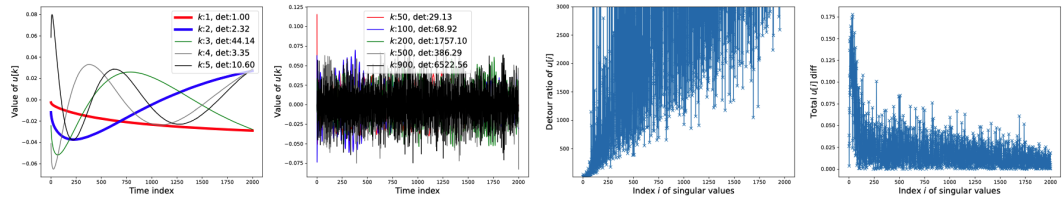


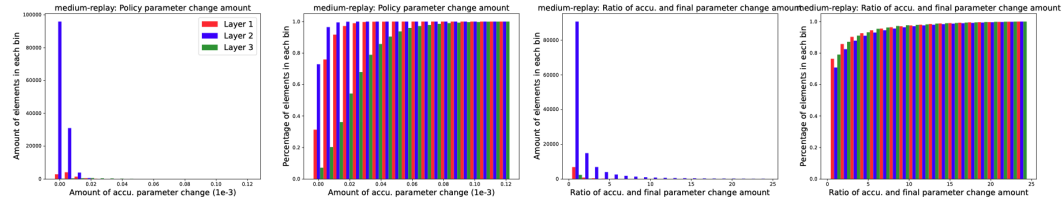
Figure 11: The learning curves for RAD (Gray) and RAD-PPTB (Red) in four DMC tasks, with means and standard errors across 6 seeds.



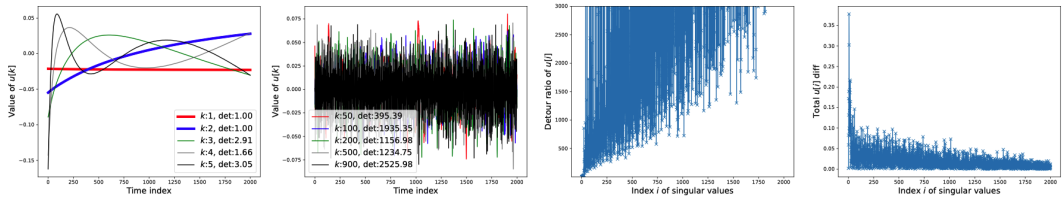
(a) Parameter Analysis of BC (**Adam**)



(b) Temporal SVD Analysis of BC (**Adam**)



(c) Parameter SVD Analysis of BC (**SGD**)



(d) Temporal SVD Analysis of BC (**SGD**)

Figure 12: The parameter and Temporal SVD analysis for Behavior Cloning with Adam and SGD optimizers in D4RL *halfcheetah-medium-replay*. Compared with the RL cases reported in our paper, **the asymmetry of parameter update amount is less observed in BC (Adam) but more severe in BC (SGD); the pattern of temporal SVD evolvement is different for BC (Adam) and BC (SGD).**

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 experiment results in Section 3 and Section 5.

   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 Section 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 5 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?

Answer: [Yes]

Justification: Our code is available in `https://github.com/bluecontra/PPTB`.

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 5 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 study policy network evolvement in a low-dimensional space, which can have a positive impact in general for better understanding the learning dynamics of DRL algorithms. No specific real-world application is concerned and we do not see a specific societal impact of this paper.

    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.

- 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.

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