Snapshot Reinforcement Learning: Leveraging Prior Trajectories for Efficiency

Yanxiao Zhao[1,2] **Yangge Qian**[1,2] **Tianyi Wang**[1,2]

Jingyang Shan[1,2] **Xiaolin Qin**[1,2*]

[1] Chengdu Institute of Computer Applications, Chinese Academy of Sciences
[2] School of Computer Science and Technology, University of Chinese Academy of Sciences
{zhaoyanxiao21, qianyangge20, wangtianyi22}@mails.ucas.ac.cn
shanjingyang21@mails.ucas.ac.cn, qinxl2001@126.com

Abstract

Deep reinforcement learning (DRL) algorithms require substantial samples and computational resources to achieve higher performance, which restricts their practical application and poses challenges for further development. Given the constraint of limited resources, it is essential to leverage existing computational work (e.g., learned policies, samples) to enhance sample efficiency and reduce the computational resource consumption of DRL algorithms. Previous works to leverage existing computational work require intrusive modifications to existing algorithms and models, designed specifically for specific algorithms, lacking flexibility and universality. In this paper, we present the Snapshot Reinforcement Learning (SNAPSHOTRL) framework, which enhances sample efficiency by simply altering environments, without making any modifications to algorithms and models. By allowing student agents to choose states in teacher trajectories as the initial state to sample, SNAPSHOTRL can effectively utilize teacher trajectories to assist student agents in training, allowing student agents to explore a larger state space at the early training phase. We propose a simple and effective SNAPSHOTRL baseline algorithm, S3RL, which integrates well with existing DRL algorithms. Our experiments demonstrate that integrating S3RL with TD3, SAC, and PPO algorithms on the MuJoCo benchmark significantly improves sample efficiency and average return, without extra samples and additional computational resources.

1 Introduction

Deep Reinforcement Learning (DRL) has enjoyed numerous accomplishments in game, simulation, and real-world environments. However, the development of powerful agents requires a significant amount of samples and computational resources. For example, AlphaStar [Vinyals et al., 2019] was trained using 16 TPU-v3 for 14 days, during which each agent used the equivalent of 200 years of the real-time StarCraft II game. Similarly, Robotic Transformer 2 (RT-2) [Brohan et al., 2023] utilized demonstration data collected by 13 robots over 17 months in an office kitchen environment. This obstacle prevents researchers who lack necessary resources from reproducing these works, thus limiting the applications and development of these works.

[*]Corresponding author

Automated Reinforcement Learning(AutoRL) workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria.

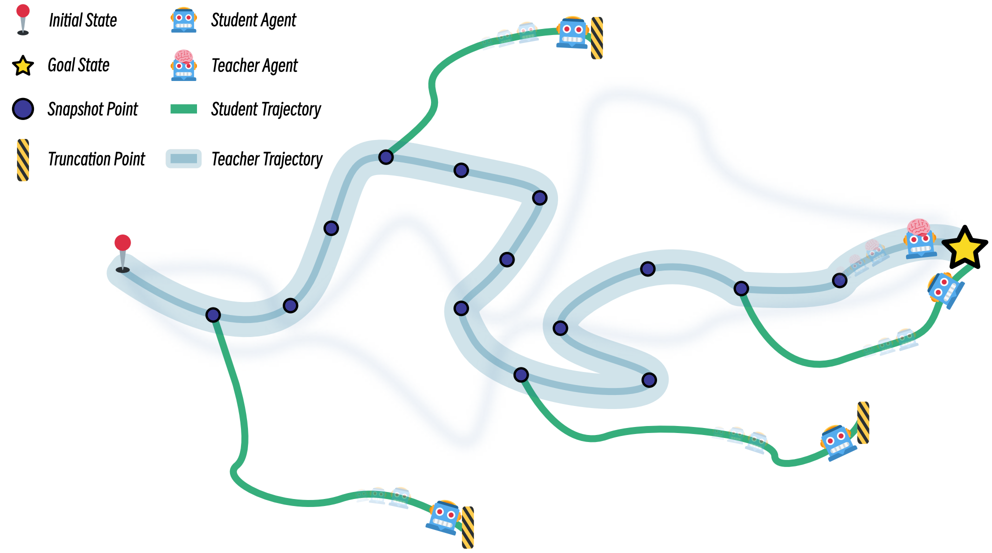

Figure 1: Schematic of S3RL training process. The figure illustrates a teacher trajectory (light blue line with outline) from the initial point (red pin) to the goal point (yellow pentagram). Dark blue dots scattered on this trajectory indicate environment snapshots obtained from the teacher agent's interaction with environment, from which the student agent starts new training represented by green trajectories. Truncation points(black and yellow squares) on the right of three student trajectories signify truncated training implemented to prevent the student agent from deviating excessively from the teacher trajectory. The student trajectory on far right reaches the goal point, demonstrating that the student agent can successfully accomplish tasks. The figure vividly portrays the mechanism and objective of S3RL: to support the training of new agents effectively by leveraging environment snapshots.

In light of this, Reincarnating Reinforcement Learning (RRL) [Agarwal et al., 2022] emerges as a promising research workflow. RRL aims to maximize the utilization of pre-existing computational work, thus releasing researchers from the need for tabula rasa when training agents and ultimately enhancing sample efficiency and reducing computational resource consumption. Previous RRL studies mainly concentrated on reusing pre-existing agent models or replay buffers, to enhance the performance of new agents. For example, seminal works such as those by Vecerík et al. [2017], Nair et al. [2020], Lu et al. [2021], Wu et al. [2022], Nakamoto et al. [2023], Luo et al. [2023] have capitalized on leveraging previously gathered demonstration data for offline pre-training, followed by careful online fine-tuning to refine agent behaviors. In parallel, Pardo et al. [2018], Ross et al. [2011], Agarwal et al. [2022] combined with prior agents to propose special loss functions. Further, Czarnecki et al. [2019], Sun et al. [2018], Zhu et al. [2023] utilize Q-function of teacher agent to compute additional rewards, guiding learning process of student agent.

However, these works usually require intrusive modifications to existing algorithms and models. Such modifications are designed for specific algorithms, lacking flexibility and universality. Researchers need to frequently adjust the design of algorithms and models during experimental studies to verify their ideas. Integrating their newly designed RL algorithms with existing RRL strategies again creates additional workloads, which hardly meet their needs.

We have dubbed our framework Snapshot Reinforcement Learning (SNAPSHOTRL). SNAPSHOTRL can enhance sample efficiency by simply altering environments, without making any modifications to algorithms and models. For simulated environments, the implementation of SNAPSHOTRL merely involves incorporating wrappers that enable the loading of snapshots into the environment, thus avoiding the necessity for extensive code modifications and significantly easing its integration into various RL research works. Environment snapshots preserve complete data of the simulation environment and allow the environment to be restored to a specific previously saved snapshot point. Our main idea is that using snapshots from teacher agent trajectories to assist student agent training allows student agents to choose states within teacher agent trajectories as initial points to begin sampling, leading to a broader exploration of states by student agents during the early training phase. By training with snapshots generated by teacher agents with environment, our framework can effectively leverage the experience accumulated by teacher agents, similar to the practice of endgame training in the game of Go.

In this paper, we first introduce SnapshotRL research framework, propose standardized evaluation suggestions, and analyze the challenges faced by this framework. Subsequently, we designed and introduced SnapshotRL with Status Classification and Student Trajectory Truncation (S3RL), a simple and effective SnapshotRL baseline algorithm developed for these challenges. The schematic process of S3RL training is illustrated in Figure 1. Our experimental results show that, on the Gymnasium MuJoCo benchmark [Todorov et al., 2012, Towers et al., 2023], when integrated with TD3, SAC, and PPO algorithms, S3RL achieves superior performance over baseline methods with just 50% of timesteps, significant improvements sample efficiency. It is important to note that the performance improvement with S3RL was achieved without increasing any computational cost in the learning part and without directly providing additional samples to student agents, which is different from previous RRL works. Without using additional off-policy samples, this makes SnapshotRL a framework that is friendly to on-policy RL algorithms.

2 Preliminaries

In our RL framework based on the concept of a Markov Decision Process (MDP), we consider a process defined by a tuple (S, A, P, R, O, γ), where

- S is the state space, which represents different states of the system.
- A is the action space, which includes all possible actions that can be taken by the agent.
- $P : S \times A \times S \to [0, 1]$ is the state-transition probability function. It quantifies the likelihood of transitioning from one state to another, given a particular action.
- $R : S \times A \times S \to \mathbb{R}$ is the reward function. $R(s, a, s')$ denotes the immediate reward received after transitioning from state s to state s', due to action a.
- O is the observation space, represented by a function $O : S \to \mathbb{O}$, where $\mathbb{O}$ is the set of all possible observations. $O(s)$ denotes the observation when the system is in state s.
- γ is a discount factor. $\gamma \in [0, 1)$

Additionally, we introduce the concept of environment snapshot, which is an extended representation of environment at a certain timestep. An environment snapshot captures not only the current state of the system but also the complete set of parameters defining the MDP. This allows for the possibility of preserving the entire state of the system, including the MDP configuration, facilitating operations such as environment resets to a past state. We denote an environment snapshot as follows:

$$\mathcal{S}_i = \langle s_i \mid (S, A, P, R, O, \gamma) \rangle$$

Here, $\mathcal{S}_i$ includes the current state s_i and the tuple representing the entire MDP configuration.

An agent's purpose in this model is to learn a policy $\pi : S \to A$, which selects an action $a = \pi(s)$ to execute in state s. The aim is to maximize the expected cumulative reward over time.

3 SnapshotRL: A Framework for Leveraging Prior Trajectories

In this section, we introduce a new framework for enhanced sample efficiency in RL algorithms — SnapshotRL. We elucidate the core mechanism of SnapshotRL, including how to capture and store trajectory snapshots, the principles for selecting and applying snapshots, as well as the intuition and expected outcomes behind this mechanism.

For the most straightforward implementation of the SnapshotRL framework in pseudocode, please refer to the parts of Algorithm 1 excluding those marked by 📌.

3.1 Procuring Snapshots

We found that SNAPSHOTRL is highly sensitive to the distribution of snapshots, and this distribution directly impacts algorithm performance. Conventionally, algorithms that leverage environment snapshots tend to manually select snapshots deemed important by experts, a practice that is both random and inflexible, making it difficult to compare and evaluate the performance of different SNAPSHOTRL algorithms. To standardize research on SNAPSHOTRL, we systematically save agent models, and during each student agent training process, we interactively generate multiple trajectories with the environment, saving the environment snapshot of each step in the snapshot collection $\mathcal{D}_{\mathcal{S}}$ for further use.

Our design facilitates the flexible creation of new SNAPSHOTRL algorithms by researchers, who have access to a wealth of information, including Q-values output by agent models. Varying the random seed generates different collections of $\mathcal{D}_{\mathcal{S}}$, which helps us to evaluate our new SNAPSHOTRL algorithms more accurately.

3.2 Weaning off Snapshots

The goal of SNAPSHOTRL is to enhance the sample efficiency of existing reinforcement learning algorithms on existing environments, rather than create environments inherently more favorable for agent training. During training process of SNAPSHOTRL algorithms, the environment used for training is different from the one used for evaluation, and the state distribution of training environment is actively controlled. The ultimate goal is for agents to adapt and perform better on the original environment, a transition that involves progressive reduction of dependence on snapshots. In the algorithm we present later, SNAPSHOTRL is applied only during the first 10% of training timesteps, after which the agent continues training in the unaltered, original environment. Our results indicate that using SNAPSHOTRL only in the initial training phase significantly improves sample efficiency of existing algorithms.

4 S3RL: A simple SNAPSHOTRL baseline

Following the introduction and analysis of SNAPSHOTRL in the previous section, this section will present SNAPSHOTRL with Status Classification and Student Trajectory Truncation (S3RL), a baseline algorithm for SNAPSHOTRL. S3RL consists of two improvement parts: (1) Status Classification (SC) and (2) Student Trajectory Truncation (STT), which are designed to address the challenges of state duplication and insufficient influence within SNAPSHOTRL. Please refer to Algorithm 1 for the pseudocode of S3RL.

4.1 Status Classification

Within our SNAPSHOTRL algorithm, we identified an issue: the snapshot collection $\mathcal{D}_{\mathcal{S}}$ often contains many duplicate or similar snapshots, resulting in an excessively high likelihood of selecting similar snapshots during random sampling processes. Taking the MuJoCo Hopper environment[2] as an example, a well-trained monopedal robot quickly enters a phase of motion marked by distinctive periodic characteristics after it has started. If randomly selected from all snapshots without adjustment, it might focus too much on the periodic phase, neglecting crucial snapshots like those found during the start-up phase.

To address this issue, we have developed a state classification strategy, which is based on Q-value of state in snapshot. Using the standard K-means clustering algorithm, we categorize snapshot according to the Q-values produced by teacher agent and uniformly select snapshot from each category to ensure balanced category coverage. Our work does not delve into which specific snapshots are most conducive to the learning process of student agents. We simply propose a straightforward method of state classification designed to maintain an equilibrium in the significance attributed to various snapshots.

[2] Documentation for Hopper Environment: https://gymnasium.farama.org/environments/mujoco/hopper/

Algorithm 1 S3RL: SnapshotRL with Status Classification and Student Trajectory Truncation

Input: a environment E, a collection of environment snapshots $\{(\mathcal{S}_1, q_1), (\mathcal{S}_2, q_2), \cdots, (\mathcal{S}_N, q_N)\}$, maximum length of student agent trajectory T, a RL algorithm Alg such as TD3.
Initialize policy π from scratch. Initialize snapshot dataset $\mathcal{D}_\mathcal{S} \leftarrow \{(\mathcal{S}_1, q_1), (\mathcal{S}_2, q_2), \cdots, (\mathcal{S}_N, q_N)\}$.
$\mathcal{D}'_\mathcal{S} \leftarrow \text{KMEANS}(\mathcal{D}_\mathcal{S})$
After applying K-means, partition $\mathcal{D}_\mathcal{S}$ into k disjoint clusters by Q-value, with each cluster C_i containing n_i states.
$\mathcal{D}'_\mathcal{S} = \{C_1, C_2, \ldots, C_k\}$ where $C_i = \{\mathcal{S}_{i,1}, \mathcal{S}_{i,2}, \ldots, \mathcal{S}_{i,n_i}\}, \sum_1^k n_i = N$.
while $number\_of\_iterations \leq max\_iterations$ **do**
    E = RESET(E)
    if in snapshot environment train phase **then**
        C = RANDOMCHOICE($\mathcal{D}'_\mathcal{S}$)
        $\mathcal{S}$ = RANDOMCHOICE(C)
        E = LOADSNAPSHOT($E, \mathcal{S}$)
    end if
    Roll out policy π within the environment E using the exploration method Alg to get a time-limited trajectory $\{(o_1, a_1, r_1), \cdots, (o_t, a_t, r_t)\}$, where the length of the trajectory, indicated by t, will not exceed the predefined limit T.
    Update π by Alg.
end while

4.2 Student Trajectory Truncation

In SNAPSHOTRL framework, only initial states of student agent trajectories is regulated. However, such an approach might not be sufficient for tasks that require long-term foresight. The influence of initial states tends to decrease as student agent trajectories lengthen. This is particularly evident in the early stages of training, when student agents may inadvertently fall into adverse states, quickly diminishing the effect of SNAPSHOTRL.

To address this challenge, we propose Student Trajectory Truncation (STT) strategy. STT prematurely truncates student agent trajectories (e.g., setting the maximum episode length in MuJoCo environments to 100 instead of the default 1000 steps). This strategy increases the frequency with which student agents encounter states within $\mathcal{D}_\mathcal{S}$, aiming to enhance the agent's learning opportunities from the initial states that are controlled by SNAPSHOTRL.

5 Experiments

Our experiments will answer the following questions: (1) How does SNAPSHOTRL affect the learned policies quality? (2) What are the most key components of SNAPSHOTRL? (3) Does SNAPSHOTRL have strong robustness and algorithmic compatibility?

We first train five teacher agents using CleanRL's TD3 implementation, each for 1 million timesteps on MuJoCo benchmark, with five random seeds. Teacher models can be found in Table 4. We select the best performing teacher agent for generating snapshot dataset. To ensure the robustness of our experimental results, we generate a unique snapshot dataset for each run using a teacher agent with varying random seeds. The teacher agent interacts with the environment for ten episodes and saves an environment snapshot into a snapshot dataset every ten timesteps.

Subsequently, we integrate SNAPSHOTRL and S3RL with TD3 and run it on six MuJoCo environments, including `Hopper-v4`, `Walker2d-v4`, `HalfCheetah-v4`, `Ant-v4`, `Swimmer-v4`, and `Humanoid-v4`. Our experimental results are shown in Figure 2 and 9. We use SNAPSHOTRL training only for the first $100,000$ timesteps, after that we use the original environment for training, the highlighted part in figures is SNAPSHOTRL training phase. The results show that the TD3 algorithm using only SNAPSHOTRL cannot achieve better performance than TD3, and even performs worse in some environments. However, when we combine SC and STT strategies with SNAPSHOTRL, sample

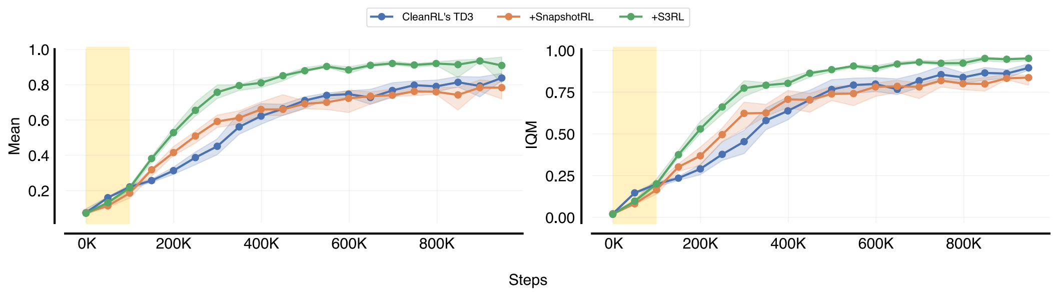

Figure 2: Learning curves sample efficiency comparison of TD3, SNAPSHOTRL+TD3, and S3RL+TD3 on six MuJoCo environments. For individual environment results, see Figure 9.

efficiency and average return of TD3 are significantly improved in all six environments. We also evaluated the performance of S3RL+TD3 under different levels of teacher agents, see Appendix C.1.

To evaluate the compatibility of the S3RL algorithm, we conducted a series of experiments integrating S3RL with SAC and PPO algorithms. For detailed information about these experiments, please refer to Appendices C.2 and C.3. Our results indicate that while S3RL significantly enhances the performance when combined with off-policy algorithms like TD3 and SAC, the performance improvements with the on-policy PPO algorithm are comparatively modest. See Appendix C.3 for analysis and discussion of this phenomenon.

5.1 Ablation Study

We also conducted ablation experiments, and the results are shown in Figure 3. SNAPSHOTRL+SC+STT (S3RL) significantly outperformed its ablation variants (SNAPSHOTRL, SNAPSHOTRL+SC and SNAPSHOTRL+STT) in terms of both sample efficiency and average return. This indicates that SC and STT methods are effective ways to improve the performance of SNAPSHOTRL, and can improve the performance of SNAPSHOTRL whether used alone or in combination.

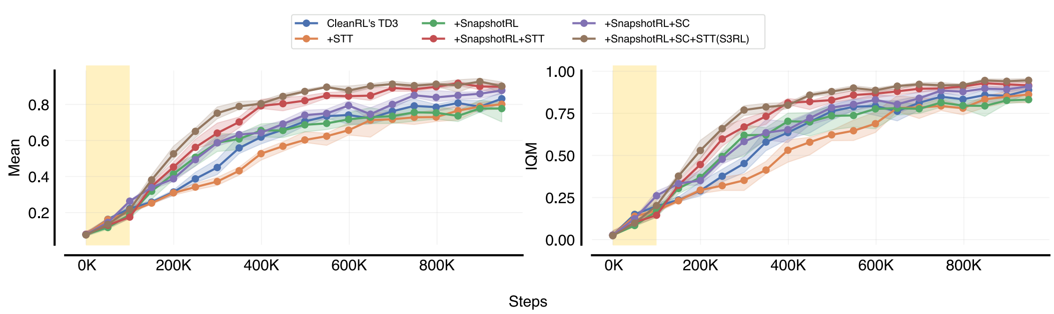

Figure 3: Ablation study results showing the impact of key components on the sample efficiency of S3RL+TD3 on six MuJoCo environments. For individual environment results, see Figure 10.

In addition, Pardo et al. [2018] pointed out that premature truncation can affect algorithm performance. Our ablation experiment TD3+STT sets the truncation step to 100 steps in the first $100,000$ timesteps, which reverts to the default setting of 1000 steps. The results show that without SNAPSHOTRL, STT has a negative impact on the performance of TD3. This result indicates that the performance improvement does not come from the premature truncation effect of STT, but from the fact that STT enhances the impact of SNAPSHOTRL on training.

5.2 Hyperparameter Robustness Study

In S3RL, both SC and STT components possess a hyperparameter each, namely the number of clusters K and the truncation step T, respectively. To demonstrate that the performance improvements obtained with our algorithm are mainly attributable to its design innovations, rather than meticulous parameter optimization, we swept a range of hyperparameters, reporting algorithm performance under these varying conditions. The experimental outcomes, as illustrated in Figure 4 and Figure 5, reveal that our algorithm's performance is not critically dependent on the fine-tuning of the cluster count K, and that the truncation step T exhibits a negative correlation with performance metrics within a certain range.

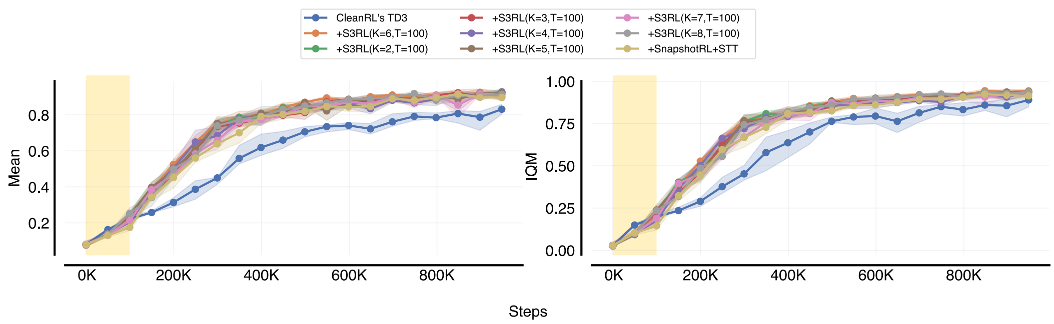

Figure 4: Learning curves sample efficiency sweeps for S3RL+TD3 across K on six MuJoCo environments. For individual environment results, see Figure 11.

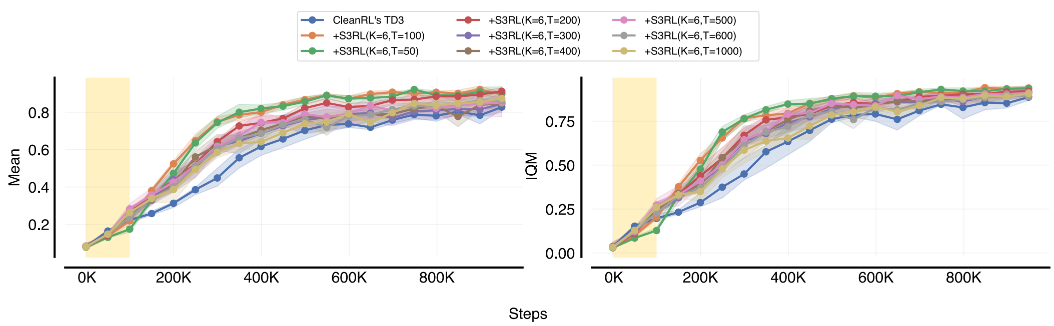

Figure 5: Learning curves sample efficiency sweeps for S3RL+TD3 across T on six MuJoCo environments. For individual environment results, see Figure 12.

These findings bolster our confidence in S3RL: it can achieve performance gains through its innovative design while remaining robust to choices in hyperparameter settings. Specifically, the results show that although the number of clusters K has a minimal impact on performance, it is noteworthy that even a minimal configuration of $K = 2$ results in improvements compared to S3RL without SC. Furthermore, the appropriate selection of the truncation step T can further optimize outcomes. This suggests that fine-tuning the truncation strategy could offer new avenues for improvements in the efficiency and effectiveness of the algorithm in the future. In forthcoming work, we anticipate that adjusting these parameters through adaptive methods or employing advanced parameter search strategies could further enhance the performance of S3RL and streamline its application process.

6 Related Work

In this section, we provide an overview of representative related works in this field, offering a comparative analysis with our contributions.

Hosu and Rebedea [2016], Salimans and Chen [2018], Pinto et al. [2018], Nair et al. [2018] use states in demonstration trajectories as initial states of agents, and demonstration trajectories are obtained by experts interacting with environment or planner solving it. Peng et al. [2018] uses a set of states carefully selected by human experts as a set of initial states of agent. The cost of obtaining these demonstration data is relatively high. It depends on human experts, while our work only uses demonstration data obtained from previous interactions between agent and environment, which is easy to obtain and reproduce. Our work focuses on using suboptimal demonstration data from prior agents, rather than expert demonstration data. Messikommer et al. [2023] saves the previously visited states during the training process, and uses states as initial states in subsequent training. It proposes to use an Embedding Network to extract features from the previously visited states and then use these features to classify states. Different from State Classification proposed in our work, we use Q-values output by prior agents as features required for classification. We believe that prior agents have already learned some helpful information, which is reflected in Q-values, so we do not need to retrain an Embedding Network for classification.

Similar to our work are Jump-Start RL (JSRL) [Uchendu et al., 2023] and Reverse Forward Curriculum Learning (RFCL) Tao et al. [2024]. These approaches respectively utilize a teacher agent and demonstration data to alter the initial state distribution of the student agent, while also providing additional samples to the agent using samples obtained from the teacher agent and offline demonstration data. Our SNAPSHOTRL can be understood as JSRL without rolling in teacher agent samples or reverse curriculum learning without rolling in offline data. We have found that not incorporating additional samples has a significant negative impact on sample efficiency. SNAPSHOTRL aims to provide a more universal RRL training framework, adaptable to existing off-policy and on-policy RL algorithms, without invasive modifications to the existing algorithms. This allows for better integration into the workflow of RL researchers.

7 Conclusion

The contributions of this paper are as follows. (1) We have proposed SNAPSHOTRL framework, which focuses on leveraging prior trajectories to enhance sample efficiency of new agents. (2) We have designed S3RL, a baseline algorithm for SNAPSHOTRL, which consists of two improvement parts, SC and STT, designed to address challenges of state duplication and insufficient influence within SNAPSHOTRL. (3) Experiments were carefully designed to analyze the utility of components of S3RL, assess its robustness, and the performance improvements of integrating S3RL with various RL algorithms.

In future work, we aim to further explore the potential of SNAPSHOTRL, studying how SNAPSHOTRL can be applied to more complex environments and real-world applications. Additionally, we plan to study the integration of SNAPSHOTRL with other methodologies, particularly those that leverage prior computational efforts, to ensure compatibility and more effective utilization.

8 Limitation

There are several limitations in this research. Firstly, our method depends on trajectories provided by teacher agents. Thus, its effectiveness might be limited in environments where teacher agents perform inadequately. If teacher agents cannot provide high-quality demonstrations, this could impact the learning efficacy of student agents. Secondly, our method requires environment snapshots, yet acquiring complete snapshots can be highly challenging, or restoring from a particular state might incur significant costs in some real-world environments. Lastly, our method may experience adverse performance impacts when applied to on-policy algorithms. Our experiments revealed that SNAPSHOTRL+PPO and S3RL+PPO only exhibited satisfactory performance in a limited set of environments, and a detailed analysis of the reasons is provided in Appendix C.3.

Reproducibility Statement

To enhance the reproducibility of our work and support the validation and further research by peers, we have provided a detailed description of our implementation in Section 3, Section 4, and Appendix B, with hyperparameters and models listed in Appendices E and F, respectively. All associated source code, models, and Weights & Biases experiment reports are accessible via sdpkjc.github.io/snapshotrl.

Our experiment results are adapted for comparison with the Open RL Benchmark [Huang et al., 2024], enabling researchers to contrast them with various algorithms without reproducing the experiments.

We invite fellow researchers to use these resources to verify our findings or as a foundation for their investigative efforts.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/ba1c5356d9164bb64c446a4b690226b0-Abstract-Conference.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

Wojciech M. Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M. Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1331–1340. PMLR, 2019. URL http://proceedings.mlr.press/v89/czarnecki19a.html.

Ionel-Alexandru Hosu and Traian Rebedea. Playing atari games with deep reinforcement learning and human checkpoint replay. *CoRR*, abs/1607.05077, 2016. URL http://arxiv.org/abs/1607.05077.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃ£o G.M. AraÃºjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Shengyi Huang, Quentin Gallouédec, Florian Felten, Antonin Raffin, Rousslan Fernand Julien Dossa, Yanxiao Zhao, Ryan Sullivan, Viktor Makoviychuk, Denys Makoviichuk, Mohamad H. Danesh, Cyril Roumégous, Jiayi Weng, Chufan Chen, Md Masudur Rahman, João G. M. Araújo, Guorui Quan, Daniel Tan, Timo Klein, Rujikorn Charakorn, Mark Towers, Yann Berthelot, Kinal Mehta, Dipam Chakraborty, Arjun KG, Valentin Charraut, Chang Ye, Zichen Liu, Lucas N. Alegre, Alexander Nikulin, Xiao Hu, Tianlin Liu, Jongwook Choi, and Brent Yi. Open RL Benchmark: Comprehensive Tracked Experiments for Reinforcement Learning. *arXiv preprint arXiv:2402.03046*, 2024. URL https://arxiv.org/abs/2402.03046.

Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, and Sergey Levine. Aw-opt: Learning robotic skills with imitation and reinforcement at scale. *CoRR*, abs/2111.05424, 2021. URL https://arxiv.org/abs/2111.05424.

Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Peter Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. *CoRR*, abs/2303.17396, 2023. doi: 10.48550/ARXIV.2303.17396. URL https://doi.org/10.48550/arXiv.2303.17396.

Nico Messikommer, Yunlong Song, and Davide Scaramuzza. Contrastive initial state buffer for reinforcement learning. *arXiv preprint arXiv:2309.09752*, 2023.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 6292–6299. IEEE, 2018. doi: 10.1109/ICRA.2018.8463162. URL https://doi.org/10.1109/ICRA.2018.8463162.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *CoRR*, abs/2006.09359, 2020. URL `https://arxiv.org/abs/2006.09359`.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023. URL `https://openreview.net/forum?id=PhCWNmatOX`.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4042–4051. PMLR, 2018. URL `http://proceedings.mlr.press/v80/pardo18a.html`.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143, 2018. doi: 10.1145/3197517.3201311. URL `https://doi.org/10.1145/3197517.3201311`.

Lerrel Pinto, Aditya Mandalika, Brian Hou, and Siddhartha S. Srinivasa. Sample-efficient learning of nonprehensile manipulation policies via physics-based informed state distributions. *CoRR*, abs/1810.10654, 2018. URL `http://arxiv.org/abs/1810.10654`.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org, 2011. URL `http://proceedings.mlr.press/v15/ross11a/ross11a.pdf`.

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *CoRR*, abs/1812.03381, 2018. URL `http://arxiv.org/abs/1812.03381`.

Wen Sun, J. Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=ryUlhzWCZ`.

Stone Tao, Arth Shukla, Tse kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demo efficiency. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=w4rODxXsmM`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL `https://zenodo.org/record/8127025`.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 34556–34583. PMLR, 2023. URL `https://proceedings.mlr.press/v202/uchendu23a.html`.

Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017. URL `http://arxiv.org/abs/1707.08817`.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 31278–31291, 2022.

Deyao Zhu, Yuhui Wang, Jürgen Schmidhuber, and Mohamed Elhoseiny. Guiding online reinforcement learning with action-free offline pretraining. *CoRR*, abs/2301.12876, 2023. doi: 10.48550/arXiv.2301.12876. URL `https://doi.org/10.48550/arXiv.2301.12876`.

A Clarification on Terminology: Snapshot vs. Checkpoint and State

Why do we use the term *snapshot* instead of *checkpoint*? We deliberately use the term *snapshot* to distinguish it from the more commonly used *checkpoint* in machine learning, emphasizing the stored models. In the context of reinforcement learning, *snapshot* is deliberately chosen to represent the comprehensive state of the interaction environment at a specific timestep, encompassing all aspects necessary to replicate an instance of the environment with precise fidelity fully.

What distinguishes a *snapshot* from an RL environment state? A *snapshot* captures a more comprehensive set of information than what is conveyed by the term *state*. In addition to the observable environment state, a snapshot includes hidden variables present in scenarios like Partially Observable Markov Decision Processes (POMDP) and meta-information managed by environment wrappers. This richer data collection ensures that the snapshot can reinitialize the environment, providing interactability that a simple state cannot.

B Experiment Details

We used the CleanRL library's implementations for TD3, SAC, and PPO algorithms in our experiments Huang et al. [2022]. For PPO algorithm, however, we amended CleanRL's original implementation to rectify its incorrect truncation handling, informed by the approach used in Stable Baselines3 Raffin et al. [2021][3].

The implementations of S3RL+TD3, S3RL+SAC, and S3RL+PPO algorithms are all based on modifications of the previously described CleanRL implementations. Every implementation strictly adheres to CleanRL's single-file design philosophy to aid researchers in understanding and replicating our work.

All learning curve figures presented in this paper represent the average of evaluation results. In each run, we conduct an evaluation every 5000 timesteps, with each evaluation comprising three episodes. We then calculate the average of these episodes to determine the evaluation result for that particular timestep.

Our experiments were conducted on machines equipped with NVIDIA 4090 GPUs and Intel 8336C processors. Each individual experiment required approximately one to two hours of execution time.

[3]The correction applied is detailed in Stable Baselines3's pull request 658: `https://github.com/DLR-RM/stable-baselines3/pull/658`

C Additional Experiment

C.1 Sweep of Teacher Models

In this subsection, we evaluated S3RL+TD3 under different performances of teacher agents. We trained five teacher agents using CleanRL's TD3 implementation, each for 1 million timesteps on MuJoCo benchmark, with five different random seeds. These teacher agents were subsequently ranked based on their evaluated performance, detailed in Table 4.

We designed five sets of experiments, where each set uses teacher agents of different performance rankings to conduct the S3RL+TD3 experiment. Our experimental results, as shown in Figures 6 and 13, indicate that S3RL+TD3 shows variability in performance under the guidance of teacher agents with different levels of performance. High-performing teacher agents generate a snapshot dataset that can lead to more significant performance improvements for the student agents, while those with relatively weaker performance offer more limited effects. Notably, even teacher agents with performance below the average performance of TD3 can still enhance student agent's performance, suggesting that S3RL+TD3 can be effective even when high-quality teacher agents are unavailable.

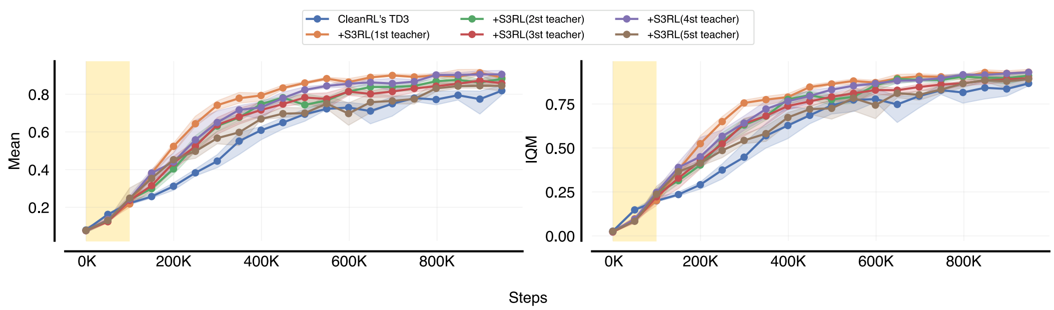

Figure 6: Learning curves sample efficiency sweeps for S3RL+TD3 across teacher models on six MuJoCo environments. For individual environment results, see Figure 13.

C.2 Evaluating S3RL with Soft Actor-Critic

Soft Actor-Critic (SAC) is an advanced off-policy algorithm that optimizes a stochastic policy in an entropy-augmented RL framework, promoting a balance between exploration and exploitation.

Similar to the experiments with TD3, we trained five teacher agents using CleanRL's SAC implementation, and selected the best performing teacher agent for generating the snapshot dataset. SAC teacher models can be found in Table 5. Our results, as shown in Figure 7 and 14, indicate that S3RL+SAC significantly outperforms SAC and SnapshotRL+SAC in terms of sample efficiency.

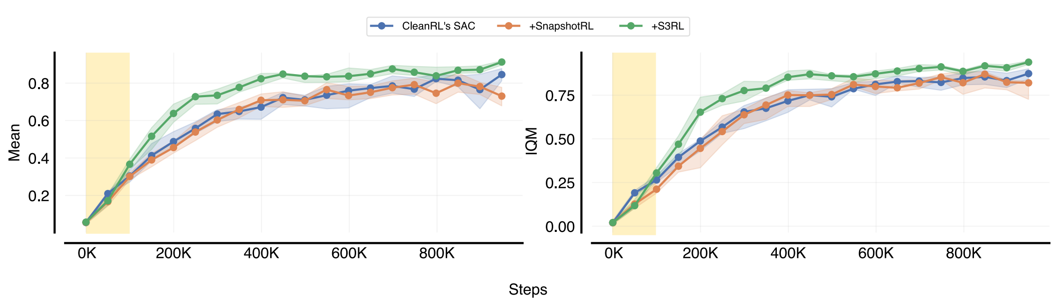

Figure 7: Learning curves sample efficiency comparison of SAC, SnapshotRL+SAC and S3RL+SAC on six MuJoCo environments. For individual environment results, see Figure 14.

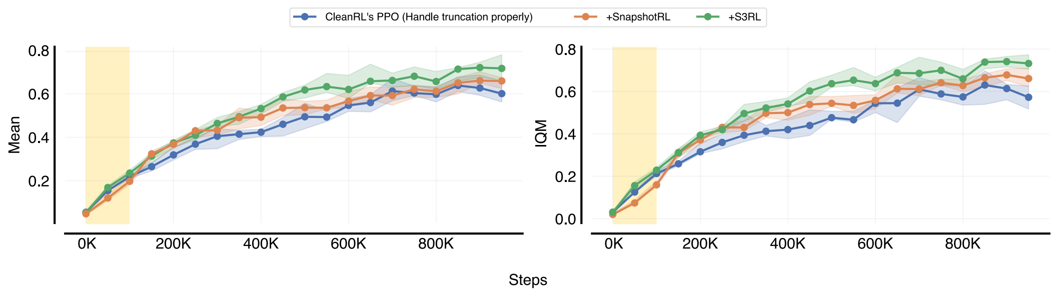

Figure 8: Learning curves sample efficiency comparison of PPO, SNAPSHOTRL+PPO and S3RL+PPO on six MuJoCo environments. For individual environment results, see Figure 15.

C.3 Evaluating S3RL with Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a widely adopted on-policy algorithm that enhances learning stability and efficiency by employing a novel objective function with a clipping mechanism to prevent disruptive policy updates.

Similar to the experiments with TD3, we trained five teacher agents using PPO algorithm, and selected the best performing teacher agent for generating the snapshot dataset. PPO teacher models can be found in Table 6. Our PPO is based on CleanRL's PPO implementation but presents a few implementation differences. For details, please refer to Appendix B. Our results, as shown in Figure 8 and 15, indicate that S3RL+PPO only exhibits satisfactory performance in a limited set of environments.

The performance gains achieved by S3RL+PPO are small compared to S3RL+TD3 and S3RL+SAC, which we analyze for the following reasons:

- In S3RL+TD3 and S3RL+SAC experiments, due to their off-policy attributes, samples collected during the snapshotRL phase are stored in the replay buffer, thus exerting a continuous influence on subsequent learning phases. However, S3RL+PPO, being an on-policy algorithm, does not retain samples from the snapshotRL phase in the replay buffer, which consequently weakens their impact on future learning stages.
- PPO employs Generalized Advantage Estimation (GAE), and the early handle truncation operation of STT strategy may affect the calculation of GAE.
- PPO normalized observations and rewards, but training environment during SNAPSHOTRL phase may alter the distribution of observations and rewards, affecting training after weaning off snapshots.
- Across the MuJoCo benchmarks, the performance of PPO teacher agents is generally lower when compared with TD3 and SAC teacher agents.

D Additional Curves

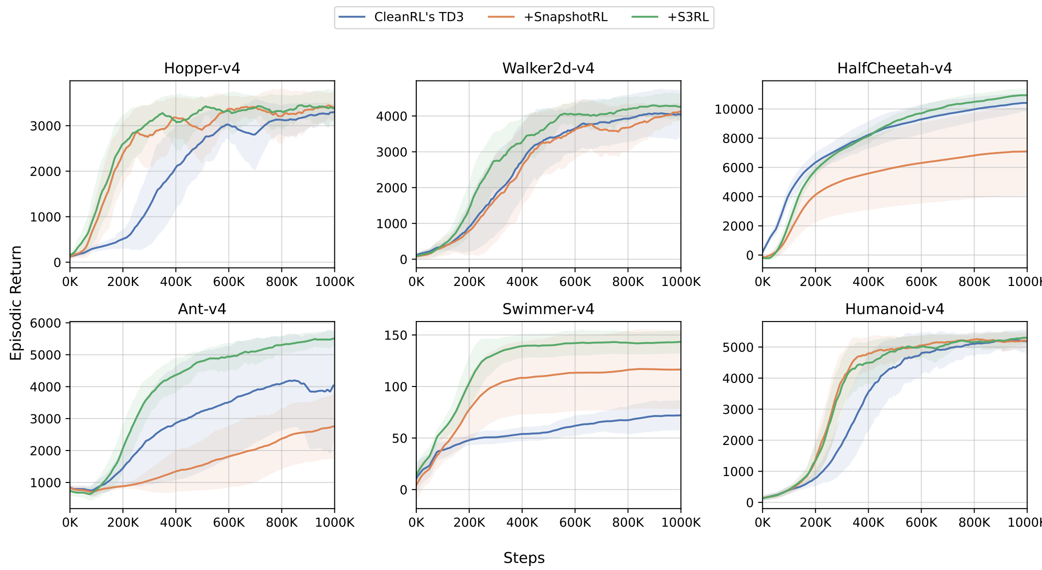

Figure 9: Detailed learning curves for TD3, SNAPSHOTRL+TD3, and S3RL+TD3 on each of six MuJoCo environments. Each subplot illustrates performance variance in sample efficiency across environments.

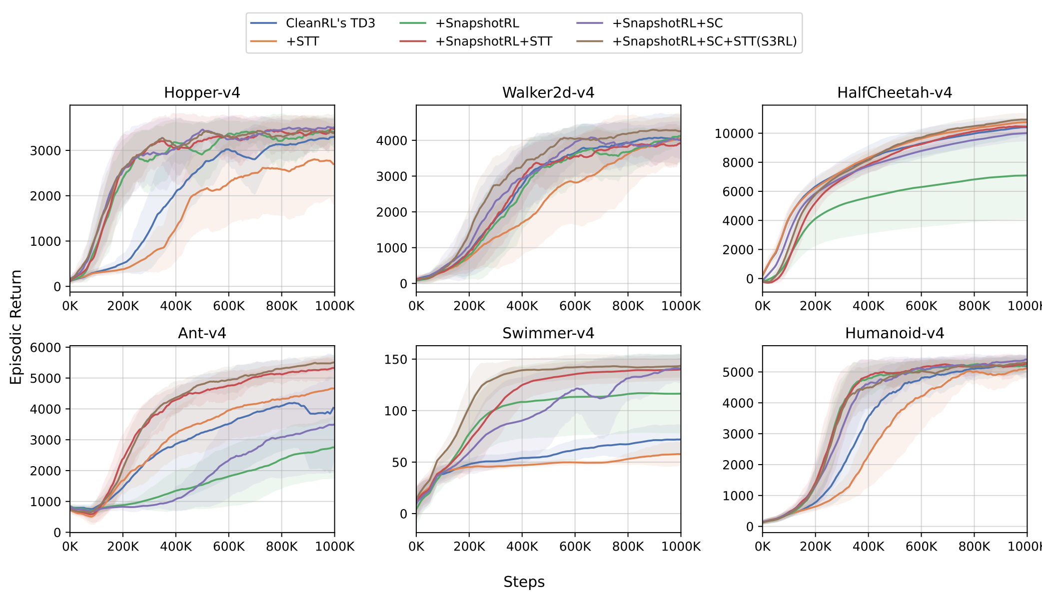

Figure 10: Ablation study details of S3RL+TD3, showing the impact of key components on sample efficiency on each of six MuJoCo environments. Each subplot illustrates performance variance in sample efficiency across environments.

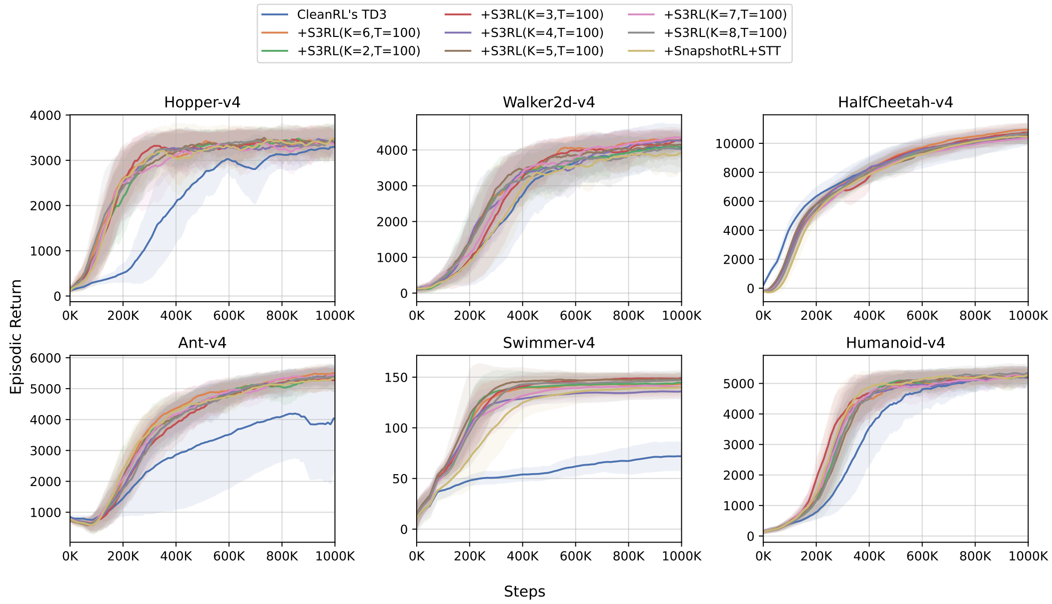

Figure 11: Learning curves sample efficiency sweeps for S3RL+TD3 across K on each of six MuJoCo environments. Each subplot illustrates performance variance in sample efficiency across environments.

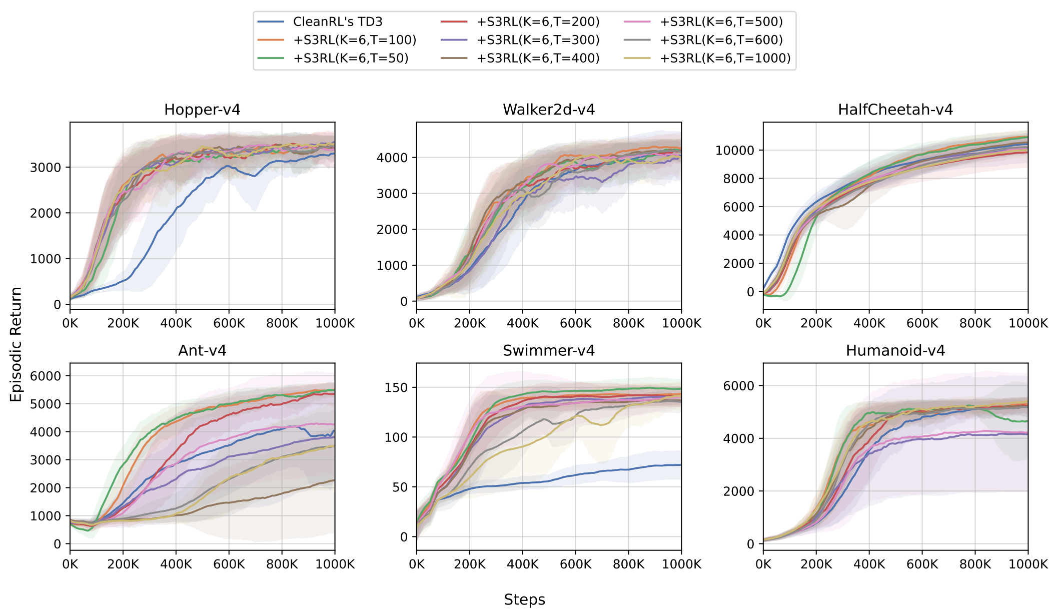

Figure 12: Learning curves sample efficiency sweeps for S3RL+TD3 across T on each of six MuJoCo environments. Each subplot illustrates performance variance in sample efficiency across environments.

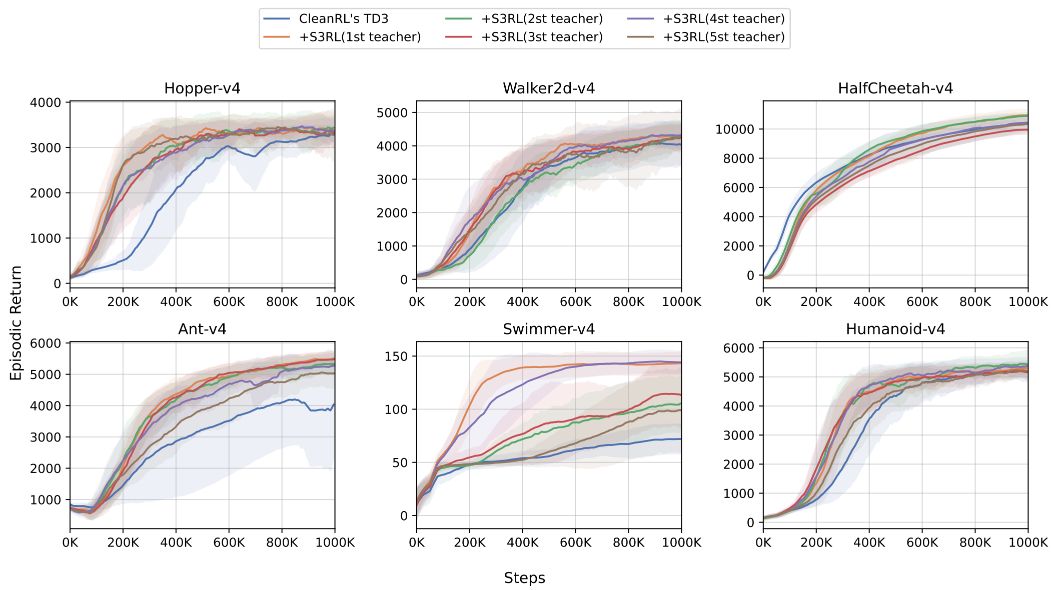

Figure 13: Learning curves sample efficiency sweeps for S3RL+TD3 across teacher models on each of six MuJoCo environments. Each subplot illustrates performance variance in sample efficiency across environments.

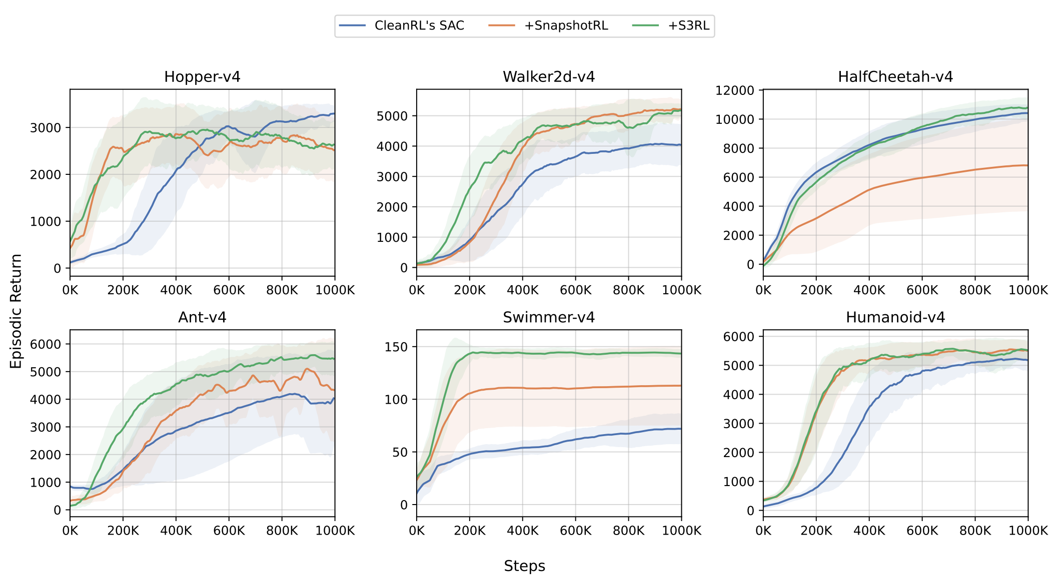

Figure 14: Detailed learning curves for SAC, SNAPSHOTRL+SAC and S3RL+SAC on each of six MuJoCo environments. Each subplot illustrates performance variance in sample efficiency across environments.

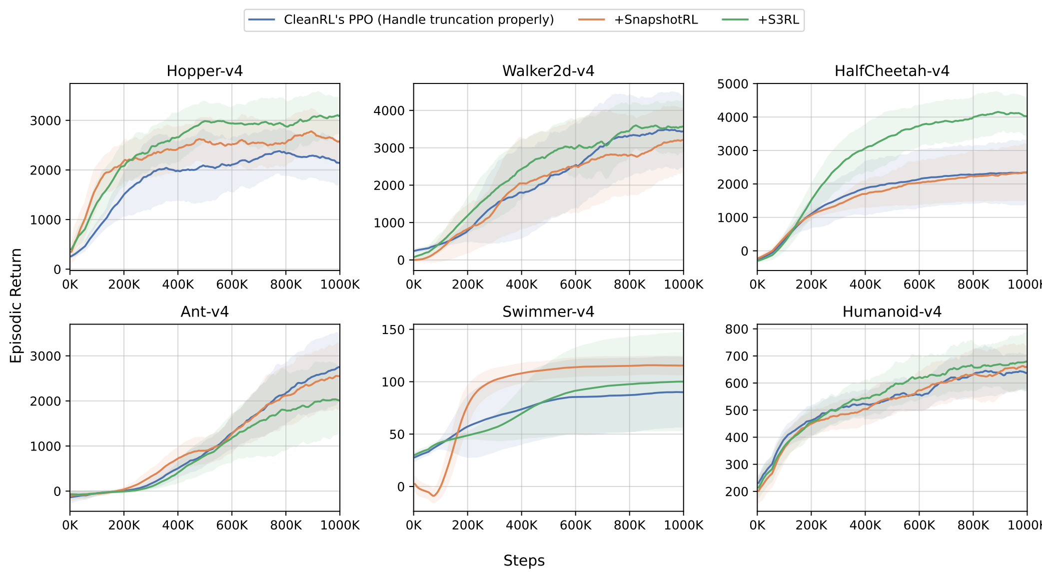

Figure 15: Detailed learning curves for PPO, SNAPSHOTRL+PPO and S3RL+PPO on each of six MuJoCo environments. Each subplot illustrates performance variance in sample efficiency across environments.

E Hyperparameter

The hyperparameters for the implementations of S3RL+TD3, S3RL+SAC, and S3RL+PPO can be found in Table 1, 2, and 3, respectively. Each table is structured such that the standard hyperparameters correspond to the standard settings of each algorithm, while the latter parameters represent additional hyperparameters introduced for SnapshotRL with Status Classification and Student Trajectory Truncation (S3RL).

Parameter Names	Parameter Values
N_{total} Total Time Steps	1,000,000
α Learning Rate	0.0003
N_{buffer} Replay Memory Buffer Size	1,000,000
γ Discount Factor	0.99
τ Target Smoothing Coefficient	0.005
N_{batch} Batch Size	256
Policy Noise Scale	0.2
Exploration Noise Scale	0.1
Time Steps Before Learning	25,000
Training Policy Frequency (Delayed)	2
Noise Clip Parameter for Target Policy Smoothing Regularization	0.5
Optimizer	Adam
N_{tep} Number of Teacher Episodes	10
N_{sostp} Number of Steps in Snapshot Training Phase	100,000
K for KMeans in State Classification	6
T in Student Trajectory Truncation	100

Table 1: TD3 and SNAPSHOTRL+TD3 hyperparameters.

Parameter Names	Parameter Values
N_{total} Total Time Steps	1,000,000
α_{policy} Policy Network Learning Rate	0.0003
α_{Q} Q Network Learning Rate	0.001
N_{buffer} Replay Memory Buffer Size	1,000,000
γ Discount Factor	0.99
τ Target Smoothing Coefficient	0.005
N_{batch} Batch Size	256
Policy Noise Scale	0.2
Exploration Noise Scale	0.1
Time Steps Before Learning	5,000
Policy Training Frequency (Delayed)	2
Target Networks Update Frequency	1
Noise Clip Parameter for Target Policy Smoothing Regularization	0.5
Entropy Regularization Coefficient	0.2
Entropy Coefficient Auto-Tuning	True
Optimizer	Adam
N_{tep} Number of Teacher Episodes	10
N_{sostp} Number of Steps in Snapshot Training Phase	100,000
K for KMeans in State Classification	6
T_{stu} in Student Trajectory Truncation	100

Table 2: SAC and SNAPSHOTRL+SAC hyperparameters.

Parameter Names	Parameter Values
N_{total} Total Time Steps	1,000,000
α Learning Rate	0.0003
N_{envs} Number of Parallel Environments	1
N_{steps} Number of Steps per Environment	2048
γ (Discount Factor)	0.99
λ (for GAE)	0.95
N_{mb} Number of Mini-batches	32
K (Number of PPO Update Iteration Per Epoch)	10
ε (PPO's Clipping Coefficient)	0.2
c_1 (Value Function Coefficient)	0.5
c_2 (Entropy Coefficient)	0.0
ω (Gradient Norm Threshold)	0.5
Value Function Loss Clipping	True
Optimizer	Adam
N_{tep} Number of Teacher Episodes	10
N_{sostp} Number of Steps in Snapshot Training Phase	100,000
K for KMeans in State Classification	6
T_{stu} in Student Trajectory Truncation	100

Table 3: PPO and SNAPSHOTRL+PPO hyperparameters.

F Teacher Models

Table 4, 5, and 6 detail teacher models used in our experiments, ranked by their performance in terms of the mean evaluation score minus the standard deviation. The model that ranks highest in each environment is indicated by a 📌 icon. Our main experimental analysis relies solely on these top-ranked models.

Environment	Model Name (Click to go to repo)	Evaluation Score	Commit
	📌`sdpkjc/Hopper-v4-td3_continuous_action-seed3`	3577.72 ± 18.84	`77fccc4`
	`cleanrl/Hopper-v4-td3_continuous_action-seed1`	3244.59 ± 8.55	`1e14f8f`
Hopper-v4	`sdpkjc/Hopper-v4-td3_continuous_action-seed2`	3162.70 ± 400.28	`e3219bd`
	`sdpkjc/Hopper-v4-td3_continuous_action-seed5`	3161.47 ± 427.71	`754ff16`
	`sdpkjc/Hopper-v4-td3_continuous_action-seed4`	3094.02 ± 807.61	`398b843`
	📌`cleanrl/Walker2d-v4-td3_continuous_action-seed1`	3964.51 ± 9.70	`51752b6`
	`sdpkjc/Walker2d-v4-td3_continuous_action-seed5`	3678.97 ± 340.29	`089a235`
Walker2d-v4	`sdpkjc/Walker2d-v4-td3_continuous_action-seed3`	3314.10 ± 12.34	`614767a`
	`sdpkjc/Walker2d-v4-td3_continuous_action-seed4`	3624.09 ± 539.63	`dbf05cb`
	`sdpkjc/Walker2d-v4-td3_continuous_action-seed2`	3527.67 ± 746.96	`fdf3439`
	📌`cleanrl/HalfCheetah-v4-td3_continuous_action-seed1`	10762.42 ± 84.09	`8547754`
	`sdpkjc/HalfCheetah-v4-td3_continuous_action-seed4`	10653.27 ± 75.24	`0f60f2f`
HalfCheetah-v4	`sdpkjc/HalfCheetah-v4-td3_continuous_action-seed3`	11443.36 ± 933.39	`0c33876`
	`sdpkjc/HalfCheetah-v4-td3_continuous_action-seed2`	10185.49 ± 107.47	`ec9624a`
	`sdpkjc/HalfCheetah-v4-td3_continuous_action-seed5`	10204.25 ± 139.05	`39939c8`
	📌`sdpkjc/Ant-v4-td3_continuous_action-seed4`	5473.45 ± 118.94	`9a956a6`
	`sdpkjc/Ant-v4-td3_continuous_action-seed3`	5211.38 ± 428.70	`14610c5`
Ant-v4	`cleanrl/Ant-v4-td3_continuous_action-seed1`	5240.79 ± 730.24	`3bd17bc`
	`sdpkjc/Ant-v4-td3_continuous_action-seed5`	2802.61 ± 163.65	`074ff1a`
	`sdpkjc/Ant-v4-td3_continuous_action-seed2`	2606.88 ± 36.88	`ad845c9`
	📌`sdpkjc/Swimmer-v4-td3_continuous_action-seed4`	113.19 ± 18.53	`1161fa1`
	`sdpkjc/Swimmer-v4-td3_continuous_action-seed2`	88.97 ± 19.63	`d6ad4b1`
Swimmer-v4	`sdpkjc/Swimmer-v4-td3_continuous_action-seed5`	82.71 ± 14.26	`69dfa47`
	`cleanrl/Swimmer-v4-td3_continuous_action-seed1`	60.09 ± 9.06	`6eab7d2`
	`sdpkjc/Swimmer-v4-td3_continuous_action-seed3`	62.38 ± 12.78	`380bcd0`
	📌`sdpkjc/Humanoid-v4-td3_continuous_action-seed3`	5279.53 ± 35.43	`e9dd75c`
	`sdpkjc/Humanoid-v4-td3_continuous_action-seed5`	5189.38 ± 27.99	`c015a50`
Humanoid-v4	`sdpkjc/Humanoid-v4-td3_continuous_action-seed2`	5038.18 ± 130.45	`5f196ad`
	`cleanrl/Humanoid-v4-td3_continuous_action-seed1`	5303.39 ± 514.14	`0450bee`
	`sdpkjc/Humanoid-v4-td3_continuous_action-seed4`	4880.24 ± 1187.43	`873f6ab`

Table 4: TD3 Models Evaluation Scores and Links

Environment	Model Name (Click to go to repo)	Evaluation Score	Commit
	📌sdpkjc/Hopper-v4-sac_continuous_action-seed4	2862.20 ± 972.12	1ee692e
	sdpkjc/Hopper-v4-sac_continuous_action-seed3	2493.92 ± 609.06	4015a94
Hopper-v4	sdpkjc/Hopper-v4-sac_continuous_action-seed1	2274.04 ± 605.18	63ba003
	sdpkjc/Hopper-v4-sac_continuous_action-seed5	1598.77 ± 492.69	bf93082
	sdpkjc/Hopper-v4-sac_continuous_action-seed2	1555.12 ± 279.93	d6b664e
	📌sdpkjc/Walker2d-v4-sac_continuous_action-seed4	5350.98 ± 89.84	c8561ff
	sdpkjc/Walker2d-v4-sac_continuous_action-seed3	5237.31 ± 942.48	2bb35b1
Walker2d-v4	sdpkjc/Walker2d-v4-sac_continuous_action-seed1	5192.85 ± 85.73	29d35a9
	sdpkjc/Walker2d-v4-sac_continuous_action-seed5	4731.36 ± 28.52	7a7f631
	sdpkjc/Walker2d-v4-sac_continuous_action-seed2	3678.91 ± 523.03	49501ed
	📌sdpkjc/HalfCheetah-v4-sac_continuous_action-seed4	11623.83 ± 156.02	bf0622e
	sdpkjc/HalfCheetah-v4-sac_continuous_action-seed2	11615.36 ± 1484.63	f5122c3
HalfCheetah-v4	sdpkjc/HalfCheetah-v4-sac_continuous_action-seed3	11543.00 ± 122.49	a8c2810
	sdpkjc/HalfCheetah-v4-sac_continuous_action-seed1	11211.47 ± 972.19	19da4f5
	sdpkjc/HalfCheetah-v4-sac_continuous_action-seed5	8187.18 ± 676.54	6816d88
	📌sdpkjc/Ant-v4-sac_continuous_action-seed3	5735.30 ± 989.07	b1126bf
	sdpkjc/Ant-v4-sac_continuous_action-seed4	5517.12 ± 1143.23	83b4537
Ant-v4	sdpkjc/Ant-v4-sac_continuous_action-seed2	5511.89 ± 1041.57	514f6d2
	sdpkjc/Ant-v4-sac_continuous_action-seed1	5314.44 ± 1159.54	b32f853
	sdpkjc/Ant-v4-sac_continuous_action-seed5	3544.68 ± 2044.81	be8c365
	📌sdpkjc/Swimmer-v4-sac_continuous_action-seed3	148.97 ± 5.85	6c0875a
	sdpkjc/Swimmer-v4-sac_continuous_action-seed2	76.70 ± 25.53	cf113b4
Swimmer-v4	sdpkjc/Swimmer-v4-sac_continuous_action-seed1	74.85 ± 27.64	d9fd594
	sdpkjc/Swimmer-v4-sac_continuous_action-seed4	50.26 ± 2.03	40ca421
	sdpkjc/Swimmer-v4-sac_continuous_action-seed5	46.46 ± 1.08	94560c4
	📌sdpkjc/Humanoid-v4-sac_continuous_action-seed4	5604.16 ± 404.34	316b06c
	sdpkjc/Humanoid-v4-sac_continuous_action-seed5	5570.79 ± 750.60	6e3b960
Humanoid-v4	sdpkjc/Humanoid-v4-sac_continuous_action-seed3	5328.96 ± 1015.76	204ee92
	sdpkjc/Humanoid-v4-sac_continuous_action-seed2	5306.36 ± 466.78	72f53bc
	sdpkjc/Humanoid-v4-sac_continuous_action-seed1	5220.03 ± 212.43	6f2042f

Table 5: SAC Models Evaluation Scores and Links

Environment	Model Name (Click to go to repo)	Evaluation Score	Commit
Hopper-v4	📌`Hopper-v4-ppo_fix_continuous_action-seed3`	2515.99 ± 807.22	`3d317e2`
	`Hopper-v4-ppo_fix_continuous_action-seed5`	2444.71 ± 794.51	`3f3fd61`
	`Hopper-v4-ppo_fix_continuous_action-seed2`	1990.14 ± 683.73	`54a25d8`
	`Hopper-v4-ppo_fix_continuous_action-seed4`	1917.18 ± 681.46	`2322d58`
	`Hopper-v4-ppo_fix_continuous_action-seed1`	1649.65 ± 559.09	`d27a3d5`
Walker2d-v4	📌`Walker2d-v4-ppo_fix_continuous_action-seed4`	4735.58 ± 1183.56	`9df90bd`
	`Walker2d-v4-ppo_fix_continuous_action-seed2`	4057.75 ± 1062.76	`b25b341`
	`Walker2d-v4-ppo_fix_continuous_action-seed3`	3781.41 ± 1202.34	`907651a`
	`Walker2d-v4-ppo_fix_continuous_action-seed1`	3357.25 ± 1235.64	`28a01f1`
	`Walker2d-v4-ppo_fix_continuous_action-seed5`	2401.69 ± 876.52	`67e3c10`
HalfCheetah-v4	📌`HalfCheetah-v4-ppo_fix_continuous_action-seed1`	4043.23 ± 526.25	`bc83fb6`
	`HalfCheetah-v4-ppo_fix_continuous_action-seed4`	2522.56 ± 537.35	`515348e`
	`HalfCheetah-v4-ppo_fix_continuous_action-seed2`	1866.44 ± 23.70	`871ea55`
	`HalfCheetah-v4-ppo_fix_continuous_action-seed5`	1821.81 ± 27.10	`b007d7f`
	`HalfCheetah-v4-ppo_fix_continuous_action-seed3`	1741.62 ± 30.79	`f696a66`
Ant-v4	📌`Ant-v4-ppo_fix_continuous_action-seed2`	3611.87 ± 747.12	`b88f77d`
	`Ant-v4-ppo_fix_continuous_action-seed3`	2739.20 ± 562.54	`419360f`
	`Ant-v4-ppo_fix_continuous_action-seed4`	2942.98 ± 823.33	`07048f2`
	`Ant-v4-ppo_fix_continuous_action-seed5`	2383.17 ± 1044.23	`3eec78a`
	`Ant-v4-ppo_fix_continuous_action-seed1`	1866.34 ± 766.40	`be0d911`
Swimmer-v4	📌`Swimmer-v4-ppo_fix_continuous_action-seed1`	131.51 ± 2.04	`989c6ba`
	`Swimmer-v4-ppo_fix_continuous_action-seed4`	119.79 ± 2.48	`5057fec`
	`Swimmer-v4-ppo_fix_continuous_action-seed3`	75.22 ± 4.29	`cc81c0e`
	`Swimmer-v4-ppo_fix_continuous_action-seed2`	63.36 ± 1.08	`63be675`
	`Swimmer-v4-ppo_fix_continuous_action-seed5`	60.77 ± 3.35	`4435bb6`
Humanoid-v4	📌`Humanoid-v4-ppo_fix_continuous_action-seed4`	704.90 ± 153.81	`83d57b0`
	`Humanoid-v4-ppo_fix_continuous_action-seed3`	687.42 ± 159.92	`318aafa`
	`Humanoid-v4-ppo_fix_continuous_action-seed2`	645.69 ± 143.65	`b5dcc47`
	`Humanoid-v4-ppo_fix_continuous_action-seed5`	591.69 ± 107.84	`d08d91f`
	`Humanoid-v4-ppo_fix_continuous_action-seed1`	640.32 ± 171.90	`e1edbff`

Table 6: PPO Models Evaluation Scores and Links