SPEQ: Offline Stabilization Phases for Efficient Q-Learning in High Update-To-Data Ratio Reinforcement Learning

Anonymous authors
Paper under double-blind review

Keywords: Reinforcement Learning, UTD, computational efficiency

Summary

Reinforcement learning (RL) algorithms that employ high update-to-data (UTD) ratios have demonstrated significant improvements in sample efficiency by performing multiple updates per environment interaction. However, this strategy comes at a considerable computational cost that can render it impractical for large-scale or real-world applications where efficiency is paramount. In this work we propose Offline Stabilization Phases for Efficient Q-Learning (SPEQ), a novel RL algorithm that interleaves low-UTD online training with periodic offline stabilization phases. During these phases, Q-functions are fine-tuned with very high UTD ratios while keeping the replay buffer fixed, reducing redundant gradient updates on suboptimal data. To mitigate the overestimation bias problem due to the multiple and consecutive updates, SPEQ implements dropout regularization only for critics. This approach improves computational efficiency without compromising learning effectiveness. Empirical results on the MuJoCo continuous control benchmark demonstrate that SPEQ significantly reduces computational overhead while achieving performance comparable to state-of-the-art high UTD ratio methods.

Contribution(s)

1. We propose **SPEQ**, a novel reinforcement learning algorithm that integrates periodic offline stabilization phases to balance computational and sample efficiency.
   Context: Our method contrasts with traditional high UTD ratio approaches by strategically concentrating computational resources in stabilization phases rather than uniformly distributing Q-function updates.
2. We empirically demonstrate that SPEQ requires from **40%** to **99%** fewer gradient updates and from **27%** to **78%** less training time compared to state-of-the-art, high UTD ratio reinforcement learning methods while maintaining competitive performance.
   Context: This highlights the computational advantages of structured training schedules over conventional high UTD ratio strategies.
3. We show that SPEQ outperforms simple reductions in UTD ratio, demonstrating that periodic stabilization phases provide a **more effective way** to optimize learning efficiency.
   Context: This distinction is crucial for designing more scalable and efficient reinforcement learning algorithms.

SPEQ: Offline Stabilization Phases for Efficient Q-Learning in High Update-To-Data Ratio Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

High update-to-data (UTD) ratio algorithms in reinforcement learning (RL) improve sample efficiency but incur high computational costs, limiting real-world scalability. We propose Offline Stabilization Phases for Efficient Q-Learning (SPEQ), an RL algorithm that combines low-UTD online training with periodic offline stabilization phases. During these phases, Q-functions are fine-tuned with high UTD ratios on a fixed replay buffer, reducing redundant updates on suboptimal data. This structured training schedule optimally balances computational and sample efficiency, addressing the limitations of both high and low UTD ratio approaches. We empirically demonstrate that SPEQ requires from 40% to 99% fewer gradient updates and 27% to 78% less training time compared to state-of-the-art high UTD ratio methods while maintaining or surpassing their performance on the MuJoCo continuous control benchmark. Our findings highlight the potential of periodic stabilization phases as an effective alternative to conventional training schedules, paving the way for more scalable reinforcement learning solutions in real-world applications where computational resources are constrained.

1 Introduction

Reinforcement learning (RL) (Sutton & Barto, 2018; Arulkumaran et al., 2017) has gained significant attention due to its ability to solve complex decision-making tasks through interactions with environments (OpenAI et al., 2018; Schwarzer et al., 2023). However, one of the primary challenges in RL is sample efficiency, which is the ability to learn effectively from a limited number of environment interactions. Typically, RL requires millions of interactions with the environment to achieve strong performance, which becomes impractical in real-world applications where such interactions are expensive, time-consuming, or risky (Arulkumaran et al., 2017).

Traditional off-policy RL algorithms perform a limited number of optimization updates per interaction stored in the replay buffer, leaving much of the potential learning signal unused (D'Oro et al., 2023). Recent studies have proposed increasing the Update-To-Data (UTD) ratio – the number of optimization steps performed per environment interaction – as a simple yet effective strategy to address this issue (Chen et al., 2021; Hiraoka et al., 2021). By performing more updates for each experience sampled from the environment, this approach allows the agent to extract more value from each interaction, thereby improving sample efficiency.

However, the sample efficiency of high-UTD algorithms comes at a significant computational cost. As illustrated in Figure 1, these approaches typically require significantly more gradient updates, which increases the overall computational cost and training time (Hiraoka et al., 2021). SAC (Haarnoja et al., 2018), a low-UTD approach, converges at around one million gradient updates, achieving good performance, while high-UTD solutions such as DroQ (Hiraoka et al., 2021) and RedQ (Chen et al., 2021) require one to two orders of magnitude more gradient updates, re-

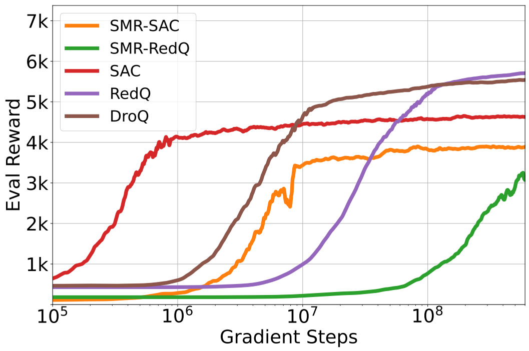


Figure 1: Comparison of state-of-the-art high UTD ratio RL approaches and SAC. This plot shows the performance averaged over four MuJoCo environments as a function of the total number of gradient steps (averaged over 5 random seeds). While high-UTD methods achieve strong final performance, they require significantly more gradient updates (and training time) compared to SAC. In contrast, SAC converges rapidly with far fewer updates, but its final performance remains limited.

spectively, to converge to a better solution than SAC. The substantial increase in required gradient updates translates into the substantial amount of time needed for training with high UTD ratios.

This issue becomes particularly critical in scenarios involving robotic agents interacting with the real world, where computational efficiency is essential (Tang et al., 2024; Paduraru et al., 2021; Li et al., 2025; Huang et al., 2023; Luo et al., 2024). The trade-off between sample efficiency and computational cost represents a fundamental bottleneck in the scalability of reinforcement learning for real-world applications. The primary challenge lies in striking an optimal balance between fully exploiting high UTD training strategies and maintaining computational feasibility.

In this paper we propose an alternative update schedule to improve the computational efficiency of off-policy RL through an increase of the *performance per gradient step* value (see Figure 4). Our approach, which we refer to as Offline **S**tabilization **P**hases for **E**fficient **Q**-Learning (SPEQ), combines the SAC (Haarnoja et al., 2018) algorithm with UTD $= 1$ with periodic offline stabilization phases during which we interrupt online interaction with the environment, thus fixing the replay buffer, and fine-tune only the Q-functions (see Figure 2). To mitigate the problem of overestimation bias, caused by the consecutive updates during offline stabilization, we incorporate dropout regularization (Srivastava et al., 2014; Hiraoka et al., 2021) which has been demonstrated to be more computationally efficient than large ensemble networks (Chen et al., 2021).

In summary, the key contributions of this work are:

- We propose Offline Stabilization Phases for Efficient Q-Learning (SPEQ), a SAC variant that uses offline stabilization phases scheduled periodically during online training to achieve competitive performance while minimizing gradient updates and thus overall computational cost.
- We evaluate SPEQ on the MuJoCo benchmark (Todorov et al., 2012) and compare with state-of-the-art, high UTD ratio methods. Our experimental results show that SPEQ is significantly more efficient, performing from 40% to 99% fewer gradient updates and requiring from 27% to 78% less training time, while maintaining or surpassing the high-UTD state-of-the-art.
- We show that SPEQ outperforms simple reductions in UTD ratio, demonstrating that periodic stabilization phases are more effective than high-UTD reinforcement learning approaches.

2 Related Work

The potential of high UTD ratios for off-policy reinforcement learning has been gaining interest from the RL research community. Model-Based Policy Optimization (MBPO) is a model-based

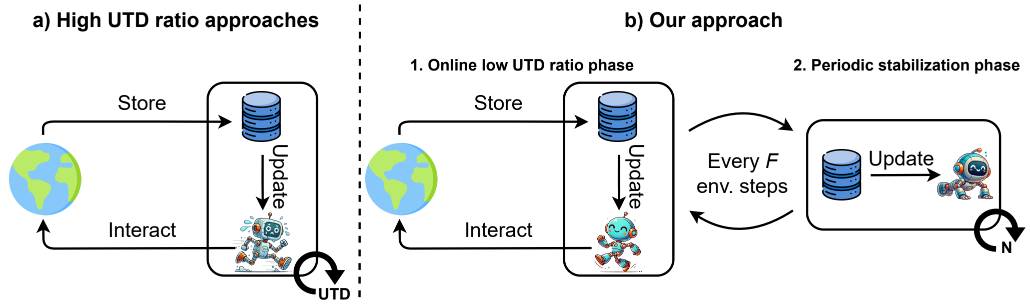

a) High UTD ratio approaches
Store
Update
Interact
UTD
b) Our approach
1. Online low UTD ratio phase
Store
Update
Interact
Every F env. steps
2. Periodic stabilization phase
Update
N

Figure 2: Overview of SPEQ. (a) Classical online RL training with high UTD ratios. For each environment interaction, the agent is trained UTD times on the replay buffer. (b) Our approach (SPEQ) which separates the training of the agent into two distinct phases. In the online interaction phase (b.1), we update the agent only once before moving to the next environment step (equivalent to UTD $= 1$). Every F environment steps we switch to an offline stabilization phase (b.2) in which we fine-tune the agent Q-functions for N optimization steps on the current replay buffer.

algorithm that uses a mix of real and synthetic data along with a large UTD $\gg 1$, achieving higher sample efficiency compared to standard model-free algorithms (Janner et al., 2019).

Randomized Ensemble Double Q-Learning (RedQ) is a model-free high UTD ratio approach using a large ensemble of Q-functions (Chen et al., 2021). Through careful selection of the size of the ensemble, as proposed by Lan et al. (2020), and using a random subset of the ensemble to estimate target values, Chen et al. (2021) showed that their approach is able to minimize the expected difference between the predicted Q-values and the target Q-values (defined as the Q-function *bias*). The authors showed that high-UTD algorithms reach sub-optimal performance because they are unable to cope with this bias. RedQ is independent of the underlying optimization algorithm and can be implemented on top of any other model-free approach, such as Soft Actor-Critic (SAC) (Haarnoja et al., 2018), Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), or Twin-Delayed DDPG (TD3) (Fujimoto et al., 2018). Despite its sample efficiency, the large ensemble renders the approach expensive from a computational efficiency perspective. In contrast, our method is able to alleviate the problem of increasing bias in high-UTD scenarios by using only two critics with their corresponding targets, as in classical Double Q-Learning (Van Hasselt et al., 2016) and thus avoiding a large ensemble and consequently further increasing computational efficiency.

Through the combination of dropout regularization (Srivastava et al., 2014) and layer normalization (Ba et al., 2016), Dropout Q-Functions (DroQ) is able to leverage a smaller ensemble of Q-functions than RedQ to improve computational efficiency (Hiraoka et al., 2021). Nevertheless, the total number of gradient steps required for convergence is unchanged with respect to RedQ, thus leaving room for improvement in terms of computational efficiency.

Sample Multiple Reuse (SMR) is one of the latest state-of-the-art approaches proposed to increase sample efficiency in model-free, off-policy RL (Lyu et al., 2023). SMR applies multiple gradient steps using the same batch of transitions while avoiding overfitting thanks to the moving targets in Q-value estimation. Similarly to RedQ, SMR can be applied on top of different optimization algorithms, such as SAC and RedQ. However, the main drawback is the overall computational efficiency: in the RedQ algorithm the UTD ratio is set to 20, and, in combination with SMR, 5 more gradient steps are performed for each sampled batch. In addition, as we will see in Section 4, SMR is computationally less efficient than SPEQ due to the larger number of gradient updates needed.

Algorithm 1 SPEQ

```
Input: Period of offline stabilization phases $F$, number of stabilization iterations phases $N$
Initialize policy parameters $\theta$, Q-function parameters $\phi$ and empty replay buffer $\mathcal{D}$.
for $m = 1, \ldots, M$ do
    Take action $a_m \sim \pi_\theta(\cdot|s_m)$. Observe reward $r_m$, next state $s_{m+1}$.
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_m, a_m, r_m, s_{m+1})$
    if $(m \bmod F) = 0$ then $G \leftarrow N$ else $G \leftarrow 1$.
    for $g = 1, \ldots, G$ do
        Sample a mini-batch $\mathcal{B} = \{(s, a, r, s')\}$ from $\mathcal{D}$.
        Update Q-functions $\phi$
    Update policy $\theta$
```

3 Offline Stabilization Phases for Efficient Q-Learning (SPEQ)

D'Oro et al. (2023) observed that, by setting the UTD ratio $\gg 1$, high-UTD approaches start to resemble Offline Reinforcement Learning (Levine et al., 2020; Kumar et al., 2020). The problem with offline learning sessions interleaved between consecutive interactions with the environment – a characteristic of all high-UTD approaches – is that the replay buffer is enriched with only one new experience between one offline learning session and the next. From a computational efficiency perspective, the high-UTD approach may not be optimal due to excessive gradient updates on suboptimal data distributions. Based on these observations, the motivation behind our approach is to allow the replay buffer to grow in order to gather new and potentially more rewarding and informative experiences before investing computational resources in high-UTD learning.

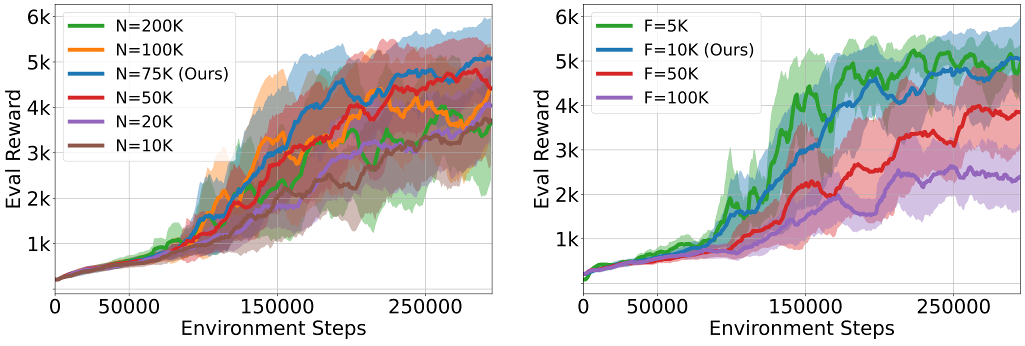
We propose **SPEQ** (Offline Stabilization Phases for Efficient Q-Learning). SPEQ is a variant of SAC that interleaves low-UTD learning during online interactions with the environment with periodic offline stabilization phases. The goal of SPEQ is to optimize the computational expense of high-UTD algorithms by accentuating the offline nature of the learning process.

During the online interactions with the environment we set the UTD $= 1$ to keep a one-to-one ratio between agent updates and addition of new experiences to the replay buffer and to minimize consecutive updates on similar distributions of experiences. This also helps mitigate overfitting to early-stage transitions (Li et al., 2024; Nikishin et al., 2023; D'Oro et al., 2023). After collecting enough new experiences, we switch to an offline stabilization stage that fine-tunes the critics on a fixed replay buffer.

A regularization mechanism for the critic networks is essential in SPEQ due to the very many updates performed in each stabilization phase. Conventional offline reinforcement learning methods typically employ strong regularizers (e.g. behavioral cloning Fujimoto & Gu (2021); Kostrikov et al. (2021); Kumar et al. (2020)) to mitigate overestimation bias. However, these approaches tend to be overly conservative in our setting and can significantly slow training Nair et al. (2020), particularly since additional interactions with the environment are permitted. To address this challenge, we adopt dropout regularization (Srivastava et al., 2014), which has been shown to effectively regulate Q-value estimates during high-UTD online training (Hiraoka et al., 2021). This choice offers a computationally efficient alternative by enabling the use of only two critic networks, as in SAC, in contrast to the large ensemble architectures required in Chen et al. (2021) (see Section 6 in the Supplementary Materials for further analysis). Algorithm 1 gives the pseudocode for SPEQ, with modifications from the standard SAC implementation highlighted in red.

4 Experimental Results

We evaluate our approach on the OpenAI MuJoCo suite (Todorov et al., 2012) in following locomotion environments: Ant, Hopper, Humanoid, and Walker2d. We compare our approach with the following algorithms: SAC (Haarnoja et al., 2018), REDQ (Chen et al., 2021), DroQ (Hiraoka et al., 2021) and SMR (Lyu et al., 2023). All results are averaged over 5 random seeds. To ensure a fair



(**a**) Length of Offline Stabilization Phases (N)

(**b**) Frequency of Offline Stabilization Phases (F)

Figure 3: (a) Results of varying the number of gradient updates N during offline stabilization on the MuJoCo Humanoid task, averaged over 5 random seeds. Offline stabilization phases are performed every $F = 10,000$ environment steps. The plot shows that the performance improves by increasing the number of updates up to about 75K iterations, beyond which further updates result in diminishing returns. (b) Comparison of SPEQ to DroQ with varying UTD ratios. Increasing the UTD ratio in DroQ generally leads to improved performance. However, despite performing approximately the same number of gradient updates as SPEQ, DroQ with a UTD ratio of 9 results in significantly lower performance. These results indicate that reducing the UTD ratio alone significantly impacts DroQ's performance, whereas SPEQ offers a performant and computationally efficient solution.

comparison with published results we use the implementations of DroQ and RedQ from (Hiraoka et al., 2021)[1], and the author implementation of SMR[2].

To verify the effectiveness of SPEQ over a span of roughly one hundred combinations of different frequencies (F) and number of offline steps (N). For brevity and clarity, we only report the most significant results. Given that SPEQ combines elements of both SAC (low UTD ratio) and DroQ (dropout regularization with high UTD updates), through our experimentation we consider SAC as a lower bound and DroQ as an upper bound in terms of computational efficiency.

Our experiments aim to answer the following research questions:

- **Q1:** How do periodic offline stabilization phases affect performance?
- **Q2:** How frequent should offline stabilization phases be?
- **Q3:** How computationally efficient is SPEQ with respect to the interactions with the environment?
- **Q4:** How effective is each gradient step in SPEQ at increasing performance?
- **Q5:** To what extent does SPEQ offline stabilization improve over reducing the UTD ratio?

How do periodic offline stabilization phases affect performance? To address **Q1**, we evaluate the effect of periodic offline stabilization phases in SPEQ by fine-tuning the Q-functions on a fixed replay buffer. We vary the number of gradient updates per stabilization phase while keeping the period fixed at twice the initial exploration phase length. Experiments were conducted on the Humanoid task, averaging results over five seeds. To align with the computational budget of high-UTD methods, we set an upper bound of $N = 200,000$ updates. As shown in Figure 3(a), increasing the number of steps taken during the offline stabilization initially improves performance, but reaches a plateau after $N = 75,000$. Further updates degrade performance due to Q-function overfitting, reducing generalization (D'Oro et al., 2023), and introducing instability rather than improving policy robustness.

[1] https://github.com/TakuyaHiraoka/Dropout-Q-Functions-for-Doubly-Efficient-Reinforcement-Learning.git

[2] https://github.com/dmksjfl/SMR.git

Table 1: Comparison of SPEQ with state-of-the-art algorithms on 300,000 environment interactions with the MuJoCo environments in terms of total gradient steps (in millions), training time (in minutes), and final score. Compared to high UTD ratio baselines, our method significantly reduces the number of gradient steps and training time, highlighting the ability of SPEQ to balance computational efficiency against performance.

	SAC	SMR-SAC	DroQ	RedQ	SMR-RedQ	SPEQ (Ours)
Gradient steps	0.9	9.3	12.3	120	600	5.4
Time	91	640	963	2100	1460	462
Score	2894 ± 1117	3422 ± 1534	4673 ± 982	4923 ± 806	3111 ± 1989	4730 ±871

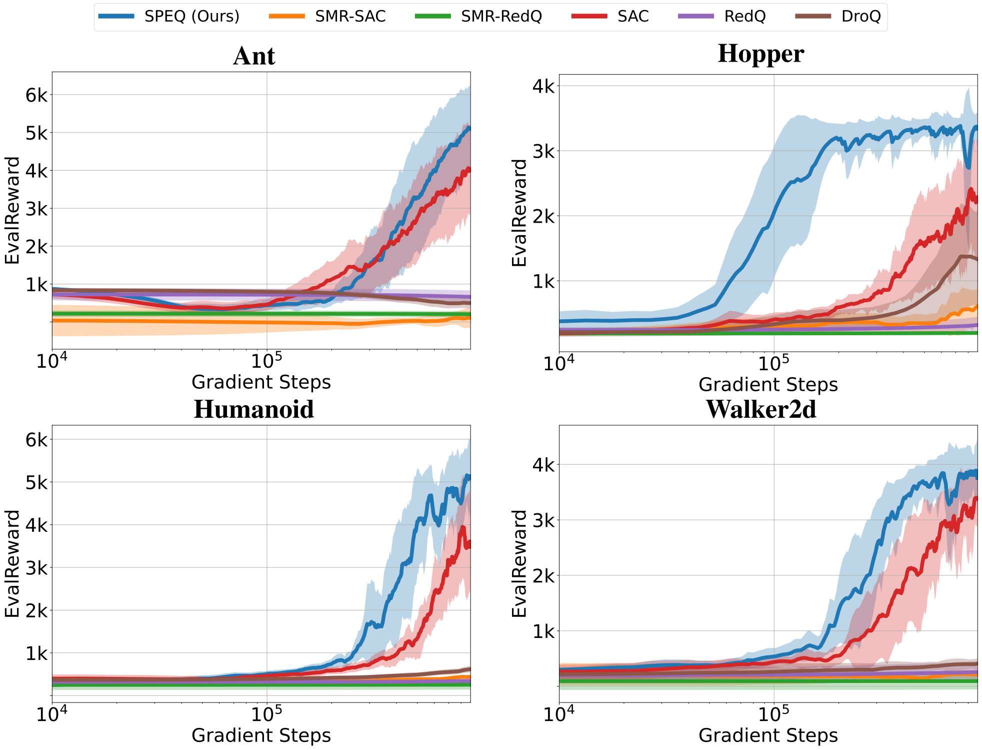


Figure 4: Comparison of SPEQ against baseline and high-UTD methods. Each algorithm performs the same number of gradient updates as SPEQ to evaluate the performance per gradient step value, that is: how effective each gradient step is in increasing performance in a resource-constrained scenario. We observe that high-UTD methods fail to achieve competitive performance when constrained to a limited number of updates. While SAC performs better than the high-UTD approaches, it requires more environment interactions. On the other hand, SPEQ represent the best trade-off.

How frequent should offline stabilization phases be? To address **Q2**, we evaluate the impact of varying the stabilization phase period (F) while keeping the stabilization phase length constant ($N = 75,000$), as identified in the previous experiment. As shown in Figure 3(b), reducing F to 5,000 maintains performance but doubles the computational cost, while increasing F to 50,000 or 100,000 leads to performance degradation because of the significant reduction in the final number of stabilization steps performed. These results indicate that $F = 10,000$ and $N = 75,000$ provide the best trade-off between computational efficiency and learning effectiveness.

Computational Efficiency. To address **Q3**, we evaluate the computational efficiency of SPEQ compared to baseline methods. We analyze the following key factors:

- **Gradient steps:** Total number of gradient updates (in millions), accounting for the number of critics used in each method. This metric is environment invariant and depends only on the algorithm used.
- **Time:** Average runtime per seed (in minutes) across all environments.
- **Score:** Final evaluation reward averaged over all seeds and environments.

Table 1 gives a comparative analysis of these metrics. All experiments were performed on an Intel i7-7800X CPU and an NVIDIA GeForce GTX 1080. While SPEQ's offline stabilization phases increase total gradient steps compared to SAC, it remains significantly more computationally efficient than high-UTD approaches. Notably, SPEQ requires less than half the gradient updates of DroQ – its closest methodological baseline – effectively cutting training time in half. SMR-SAC falls between SPEQ and DroQ in computational cost, but achieves less than half their final performance, demonstrating an unfavorable trade-off between efficiency and performance. RedQ and SMR-RedQ are the most computationally expensive, requiring approximately 60x and 12x more gradient updates than SPEQ, respectively.

Efficiency alone is clearly insufficient without strong performance. As shown in Table 1, SPEQ is the second best-performing method after RedQ, despite using far fewer updates, 95.5% fewer. DroQ achieves a similar final score, but SAC and SMR-RedQ achieve about half the performance of SPEQ. While SMR-SAC may appear computationally efficient, its poor performance highlights the importance of balancing efficiency with learning effectiveness. SPEQ achieves this balance, requiring from 40% to 99% fewer gradient updates and from 27% to 78% less training time while maintaining competitive performance.

Learning effectiveness To address **Q4**, we evaluate the *performance per gradient step value* to assess how efficiently each algorithm utilizes gradient updates when constrained to the same total number of updates as SPEQ. Unlike previous experiments where all methods were trained for 300,000 environment steps, we now standardize the training process by allowing all methods to perform exactly the same number of gradient updates as SPEQ. As shown in Figure 4, conventional high-UTD methods like DroQ, SMR-SAC, SMR-RedQ, and RedQ truggle under this constraint, with performance remaining close to zero. This highlights the inefficiency of high UTD ratios when updates are limited, as these approaches fail to leverage their computational advantage effectively within a restricted gradient budget.

In contrast, SAC, which maintains a more balanced allocation of updates relative to replay buffer expansion, achieves a higher performance per gradient step value than high-UTD approaches, demonstrating better computational efficiency. However, SAC remains less performance than SPEQ, achieving a significant lower final score. These results emphasize the importance of strategically allocating gradient updates based on the agent's accumulated experience.

Comparison with Different UTD Ratios. To address **Q5**, we examine whether SPEQ's computational efficiency can be replicated by simply lowering the UTD ratio. We compare SPEQ with DroQ with UTD values of 2, 3, 9, and 20, the latter being the original DroQ setting (Hiraoka et al., 2021). While DroQ with UTD $= 9$ performs approximately the same number of gradient updates as SPEQ, its performance is significantly inferior, indicating that dropout regularization alone is insufficient to achieve the same tradeoff between efficiency and performance. As shown in Figure 5, increasing the UTD ratio generally improves performance, with DroQ at UTD $= 2$ or UTD $= 3$ performing the worst. Although DroQ with UTD $= 20$ achieves results comparable to SPEQ, it does so *at twice the computational cost*. These findings highlight that SPEQ's efficiency is not merely a result of reducing the UTD ratio, but rather stems from a structured training schedule that strategically allocates computational resources through offline stabilization phases. This demonstrates that our approach is a distinct and more effective solution for optimizing training efficiency.

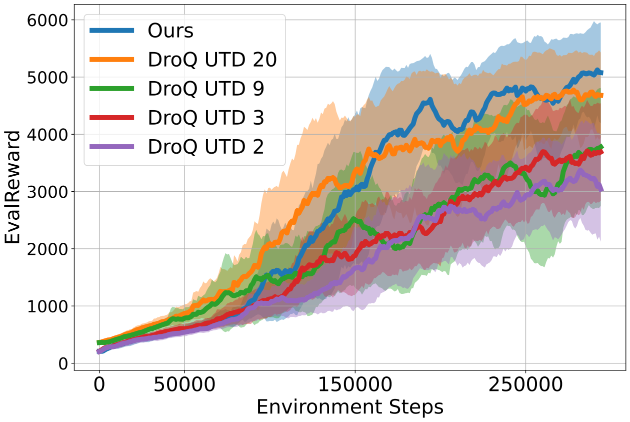


Figure 5: Comparison of SPEQ with DroQ at varying UTD ratios. We see that increasing the UTD ratio in DroQ generally leads to improved performance. However, despite performing approximately the same number of gradient updates as SPEQ, DroQ with UTD $= 9$ results in significantly lower performance. This indicates that reducing the UTD ratio alone significantly impacts DroQ's performance, whereas SPEQ offers a more performant and computationally efficient solution.

5 Conclusions

In this work we introduced SPEQ (Offline **S**tabilization **P**hases for **E**fficient **Q**-Learning), a novel offline RL algorithm designed to improve computational efficiency while maintaining high sample efficiency. Our approach addresses the inefficiencies of conventional high UTD ratio methods by strategically interleaving low UTD online interactions with periodic offline stabilization phases. During these stabilization phases, we fine-tune Q-functions with a high UTD ratio without additional environment interactions, effectively decoupling the trade-off between computational and sample efficiency.

Through extensive empirical evaluations on the MuJoCo continuous control benchmark, we demonstrated that SPEQ significantly reduces computational costs while achieving competitive performance compared to state-of-the-art high UTD ratio methods. Specifically, SPEQ achieves 40% to 99% fewer gradient updates and reduces training time by 27% to 78%, all while maintaining or surpassing the sample efficiency of alternative approaches. Our results further highlight that offline stabilization phases are an effective alternative to simply lowering the UTD ratio, providing a structured and efficient way to allocate computational resources. The relevance of SPEQ lies in its potential to enhance the scalability and practicality of RL. As RL is increasingly applied to real-world tasks that demand computational efficiency, methods like SPEQ provide a viable solution to mitigate excessive training costs while preserving learning effectiveness. Furthermore, our approach offers a new perspective on the design of training schedules, paving the way for more adaptive and efficient RL frameworks.

Future work will focus on developing an automatic stabilization detection mechanism, which monitors relevant training signals, such as the evolution of the Q-value bias, policy improvement rate, or TD-error stability, to decide when and for how long offline stabilization should occur. This approach would eliminate the need for fixed hyperparameters and improve generalization across different tasks and environments. By bridging the gap between computational and sample efficiency, SPEQ offers a promising direction for the application of reinforcement learning agents into real-world problems, such as robotics (Tang et al., 2024; Luo et al., 2024; Paduraru et al., 2021) and autonomous driving (Li et al., 2025; Huang et al., 2023), in which effient solutions are required.

References

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron C. Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *International Conference on Learning Representations*, 2023. URL https://api.semanticscholar.org/CorpusID:259298604.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018. URL https://api.semanticscholar.org/CorpusID:28202810.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *ArXiv*, abs/2110.02034, 2021. URL https://api.semanticscholar.org/CorpusID:238353966.

Zhiyu Huang, Jingda Wu, and Chen Lv. Efficient deep reinforcement learning with imitative expert priors for autonomous driving. *IEEE Transactions on Neural Networks and Learning Systems*, 34 (10):7391–7403, 2023. DOI: 10.1109/TNNLS.2022.3142822.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *ArXiv*, abs/1906.08253, 2019. URL https://api.semanticscholar.org/CorpusID:195068981.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Jingchen Li, Haobin Shi, Huarui Wu, Chunjian Zhao, and Kao-Shing Hwang. Eliminating primacy bias in online reinforcement learning by self-distillation. *IEEE transactions on neural networks and learning systems*, PP, 2024. URL https://api.semanticscholar.org/CorpusID:269837649.

Qifeng Li, Xiaosong Jia, Shaobo Wang, and Junchi Yan. Think2drive: Efficient reinforcement learning by thinking with latent world model for autonomous driving (in carla-v2). In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 142–158, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72995-9.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015. URL https://api.semanticscholar.org/CorpusID:16326763.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16961–16969, 2024. DOI: 10.1109/ICRA57147.2024.10610040.

Jiafei Lyu, Le Wan, Zongqing Lu, and Xiu Li. Off-policy rl algorithms can be sample-efficient for continuous control via sample multiple reuse. *Inf. Sci.*, 666:120371, 2023. URL https://api.semanticscholar.org/CorpusID:258967871.

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

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *ArXiv*, abs/2305.15555, 2023. URL https://api.semanticscholar.org/CorpusID:258887576.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

Cosmin Paduraru, Daniel Jaymin Mankowitz, Gabriel Dulac-Arnold, Jerry Li, Nir Levine, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110:2419 – 2468, 2021. URL https://api.semanticscholar.org/CorpusID:234868359.

Max Schwarzer, Johan Samir Obando-Ceron, Aaron C. Courville, Marc G. Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:258987895.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes, 2024. URL https://arxiv.org/abs/2408.03539.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *Proc. IROS*, pp. 5026–5033. IEEE, 2012.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Supplementary Materials

The following content was not necessarily subject to peer review.

6 Ablation Study

This section analyzes the impact of different design choices in SPEQ. Specifically, we investigate the following key questions:

- **A1:** Is regularization of the Q-functions necessary to mitigate overestimation bias during offline stabilization phases? Furthermore, is dropout regularization the most effective solution?
- **A2:** How does updating both the policy and Q-functions during offline stabilization phases impact performance?

Regularization of the Q-Functions To address **A1**, we evaluate whether dropout regularization is required to maintain stability and prevent overestimation bias in the Q-functions during offline stabilization phases. Additionally, we compare dropout against an alternative regularization strategy that employs a large ensemble of critics, as proposed in RedQ (Chen et al., 2021). We consider the following variants of SPEQ for comparison:

- **SPEQ w/o dropout**: SPEQ without any form of regularization.
- **SPEQ w/ ensemble**: SPEQ regularized using a large ensemble of critics.
- **SPEQ (ours)**: The proposed SPEQ variant with dropout regularization.

For completeness, we also include vanilla implementations of RedQ (UTD $= 20$) and SAC (UTD $=$ 1) to provide a baseline and assess the effects of offline stabilization phases.

As shown in Figure 6, the absence of Q-function regularization (*SPEQ w/o dropout*) leads to performance degradation, performing even worse than SAC without offline stabilization phases. This result confirms that regularization is essential when performing multiple consecutive updates on a fixed replay buffer. Comparing dropout to ensemble-based regularization (*SPEQ w/ ensemble*), we observe that dropout achieves superior performance while significantly reducing computational overhead. The ensemble approach, while effective in mitigating bias, requires a much higher number of gradient updates due to the presence of multiple critics, making it computationally expensive. Specifically, the computational cost of *SPEQ w/ ensemble* scales proportionally to the number of critics (20 in the original RedQ implementation), further highlighting the efficiency of dropout regularization.

Impact of Policy Updates During Stabilization Phases To address **A2**, we evaluate different update strategies during offline stabilization phases by considering the following variants:

- **SPEQ (ours)**: Only the Q-functions are updated during stabilization phases.
- **SPEQ w/ policy update**: Both the policy and Q-functions are updated.
- **SPEQ w/ only policy update**: Only the policy is updated.

The results, presented in Figure 7, reveal two key insights: (i) Updating only the policy during offline stabilization phases leads to a **collapse in performance**, confirming that Q-function updates are crucial for effective learning. (ii) Updating both the policy and Q-functions does not yield additional benefits compared to updating only the Q-functions. Instead, it introduces additional computational overhead, making the approach less efficient.

These findings indicate that the most effective strategy is to exclusively update the Q-functions during stabilization phases, as it optimally balances performance and computational efficiency.

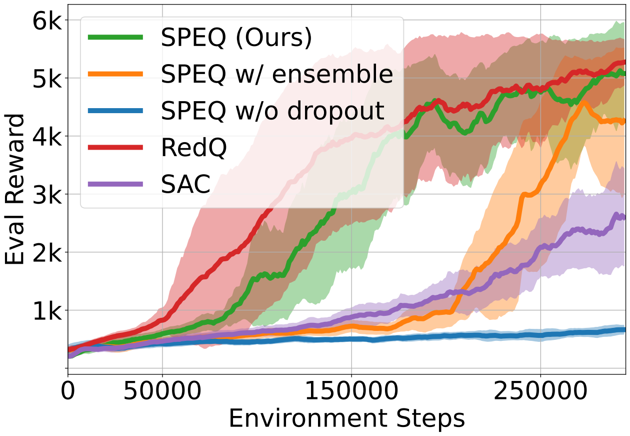


Figure 6: Comparison of different regularization techniques for mitigating overestimation bias during offline stabilization phases on the MuJoCo Humanoid task, averaged over five random seeds. *SPEQ (OURS)* employs dropout regularization for Q-functions, while *SPEQ w/ ensemble* utilizes a large critic ensemble. *SPEQ w/o dropout* does not include any regularization. All SPEQ variants use the following hyperparameters: UTD $= 1, F = 10,000, N = 75,000$. The plot also includes RedQ (UTD $= 20$) and SAC (UTD $= 1$) as baselines, where F and N are set to zero. The results demonstrate that (i) Q-function regularization is necessary when performing multiple consecutive updates and (ii) dropout regularization outperforms ensemble-based regularization while being significantly more computationally efficient.

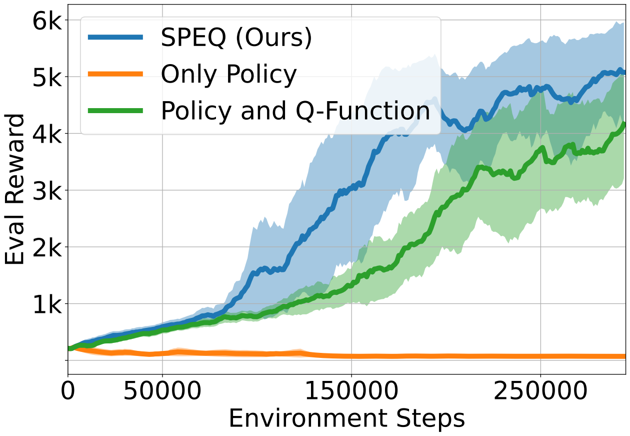


Figure 7: Evaluation of different update strategies during offline stabilization phases on the MuJoCo Humanoid task, averaged over five random seeds. The blue line represents SPEQ (ours), where only the Q-functions are updated. The orange line corresponds to updating only the policy, while the green line represents updating both the policy and Q-functions. The results indicate that updating only the policy leads to performance collapse, while updating only the Q-functions yields the best performance and computational efficiency.