Smart Resets: Dynamic Schedules and Cost-Aware Resets for Faster Autonomous Reinforcement Learning

Shaurye Aggarwal **Brian Park**

Kaustav Mukherjee **Aman Goel** **Santhoshini Gongidi**

Abstract: This paper introduces a novel framework for improving reinforcement learning by incorporating dynamic reset triggers and adaptive reset strategies, which enhance both exploration and safety. The framework builds upon existing methods [1] by employing dynamic reset mechanisms that respond to the context of failures and a policy-switching network that dynamically selects between forward and reset policies. These approaches promote safer exploration and reduce the need for manual resets, thus minimizing human intervention and optimizing the training process. Our experiments demonstrate the effectiveness of this approach in simulated environments, highlighting its potential to increase the efficiency of autonomous learning in more complex and uncertain domains. We show that some of our approaches achieved higher rewards (a maximum increase of 66%) and fewer resets (approximately 33% less). Finally, we discuss future directions to reduce manual intervention and make online reinforcement learning safer. Source code is available at https://github.com/saggarwal46/LeaveNoTraceRLProj.

Keywords: Reset Policy, Early Aborts, Safe RL

1 Introduction

In reinforcement learning (RL), the ability to safely explore and recover from mistakes is critical, particularly in environments where irreversible actions can permanently impact the agent's ability to learn effectively. Eysenbach et al. [1] propose to jointly learn a reset policy along with the forward policy to avoid these issues. The forward policy focuses on completing the primary task, while the reset policy is responsible for bringing the agent back to its starting state. The Q value of the reset policy is also used as a signal to avoid reaching irrecoverable states by aborting the episode early. This setup reduces the need for manual resets which require human interventions to restore the environment to its initial conditions, and hence reduces human effort during training.

The authors reported results on five simulated tasks to demonstrate the strengths of their approach. Across all tasks, learning a reset policy helped reduce the number of manual resets, and for most tasks, the framework achieved rewards comparable to those of a single forward policy. However, this joint training framework has some limitations. Specifically, the authors rely on a static Q-value threshold to trigger the early abort mechanism. We hypothesize that the static thresholding approach causes the agent to make overly conservative choices during exploration, hindering its ability to learn an optimal policy. For instance, in the pusher task, the joint training framework achieves suboptimal rewards compared to a single forward policy approach. Our baseline experiments reproduced similar results.

Moreover, *Leave no Trace* [1] assigns the same cost to all resets, treating minor and severe failures equally. In reality, some hard resets are more expensive than others. For example, a robot falling from a considerable height might cause greater damage to the robot than a robot falling on a ground.

Course Project, 16-831 Introduction to Robot Learning, Fall 2024, CMU

Similarly, for manipulation tasks, breaking a glass object might be more expensive than dropping a cloth. This uniform treatment of resets may hinder the agent's learning process, as it becomes equally cautious across diverse failure scenarios. As a result, the agent may lack sufficient incentive to explore in less risky situations.

To address these limitations, we hypothesize that introducing dynamic reset mechanisms and adaptive reset strategies can strike a better balance between exploration and safety, leading to improved task performance and reduced manual resets. Specifically, we investigate the following questions:

- **Dynamic Reset Triggers:** Can we develop a dynamic strategy to trigger the reset policy, rather than relying on static thresholds? If so, does this enhance exploration and improve learning outcomes for both forward and reset policies?
- **Variable Reset Costs:** Given that resets vary in severity (e.g., minor deviations vs. critical failures), does assigning variable costs based on failure severity allow the agent to explore more and still effectively learn from less severe cases while avoiding catastrophic ones?
- **Policy-Switching Mechanism:** Can we design a policy-choosing network to dynamically switch between forward and reset policies? If so, can this approach further reduce manual resets and optimize overall performance?

2 Previous Work

In recent years, reinforcement learning has been applied with greater frequency to various robotic applications, especially for more complex tasks with greater consequences when there are failures. During online learning, the robot might end up in undesirable states where human supervision is required to reset the robot. This has fostered a search for training schemes requiring less human intervention. [2].

There are a series of current approaches that relate to our work. One such method is multi-task learning to eliminate resets [3, 4], which reframe resets as valuable tasks in a multi-task learning problem, which is not true for all environments. Others include curriculum learning, such as a work by Kim et al. [5] that introduces bi-directional curriculum learning, and a work by Turchetta et al. [6] that introduces a teacher agent that chooses reset policies and designs a curriculum. However, such curriculum learning solutions are challenging to design and implement properly for more complex environments.

The work that most informs our developments is *Leave no Trace* by Eysenbach et al. [1], which simultaneously trains a reset policy along with the forward policy in a safety wrapper. This reset policy learns to return the agent to the starting position. The Q value of this reset policy is used as a signal: when it is below Q_{min}, the agent will use the reset policy to return to the origin, preventing the need for a hard reset. However, this work was limited in the manual selection of its hyper parameters and the rigid reset cost. This was expanded upon in a work by Kim et al. [7] that proposed replacing the Q_{min} threshold with a third "choose policy" that would determine when to reset. This policy would estimate the success probability of the reset, and choose whether or not to reset based on that. However, this method of implementing the choose policy resulted in lower performance for certain environments, warranting further exploration with different implementations.

3 Experiments

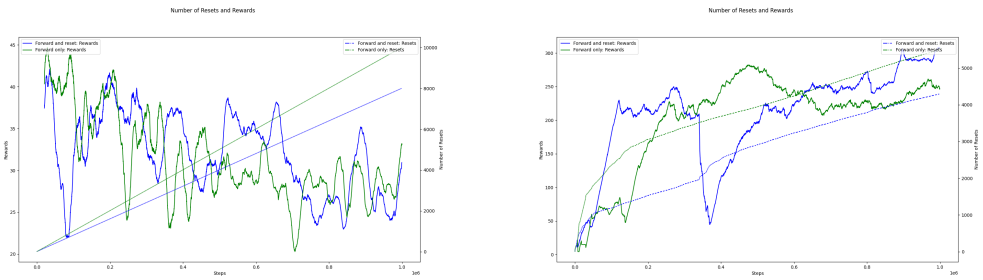
We conducted experiments on the Cliff Walker and Pusher tasks to evaluate the proposed approach. Each experiment was trained for one million iterations using DDPG, following [1] to allow experience sharing between episodes. The parameter controlling the threshold for resetting is denoted as *safety_param* $\in [0, 1]$, where increasing the value from 0 to 1 makes the agent safer.

The tasks are described as follows:

- Cliff Walker: The agent learns to walk on a 6m cliff. The agent is rewarded for moving forward and includes a control penalty. The reset reward is defined to be a combination of the distance from the origin, an indicator of whether the agent was standing, and a control penalty.
- Pusher: The agent pushes a puck to a goal location. The agent's reward is a combination of the distance from the puck to the goal, the distance from arm to the puck, and a control penalty. For the reset reward, the distance from puck to start is used instead of distance from puck to goal.

3.1 Baseline

We conduct baseline experiments to replicate the results in [1] using their default parameters.



(a) Pusher (b) Cliff Walker

Figure 1: Rewards and total resets when training forward only policy vs when training alongside a reset policy

3.1.1 Results

As seen in Figure 1a, for the Pusher task, we get lower rewards and fewer resets when we train both a forward and reset policy, similar to the results in [1]. For the Cliff Walker (Figure 1b) task, as shown in [1], we get higher rewards and fewer resets when training alongside a reset policy.

3.2 Q Signal Scheduler

To learn an optimal policy, the agent should have enough opportunities to explore. We hypothesize that using a static threshold as a signal for early aborting may reduce exploration. Ideally, the agent should be allowed to explore more in the beginning when it starts to learn. This may cause far more manual resets in the beginning. However, a higher tolerance for failure in the beginning will allow the agent to learn which states are undesirable more effectively, thus reducing the overall number of hard resets. Moreover, these increased numbers of hard resets will be limited to the beginning of the training, during which researchers can more easily intervene if necessary.

As training progresses and the agent becomes better at identifying states that would lead to resets, we can increase this threshold and have a lower tolerance for the robot failing.

3.2.1 Set Up

We test this approach by using 3 schedulers (Figure 2) - cosine, linear, and exponential - for the threshold on both tasks. While the optimal threshold obtained by [1] was 0.3, we increased the ceiling of our threshold to 0.4 to have a safer agent towards the end of training.

3.2.2 Results

For the Pusher task (Figure 3), the linear schedule achieves the maximum rewards while minimizing the number of resets by around 2000. The rewards remain largely unaffected when using cosine and

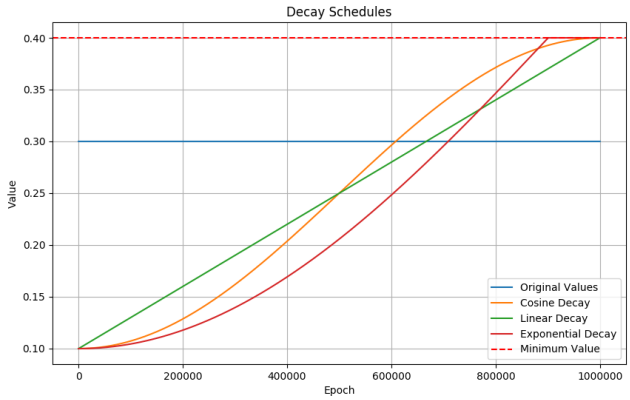


Figure 2: Value of threshold parameter according to various schedules

exponential schedules. The exponential schedule reduces the number of resets by a few hundreds, while the cosine schedule does not affect the number of resets at all.

In the case of Cliff Walker (Figure 4), the cosine schedule results in lowest number of resets, while the linear and exponential schedules lead to more resets compared to the no schedule case. The reward for cosine schedule is a few points below the no schedule case, which we believe might be due to randomness. We plan to run the experiment with more random seeds in future iterations to verify this.

We observe that while schedules are helpful in reducing the number of resets and maximizing the rewards, different schedules work better for different cases: linear schedule for Pusher and cosine schedule for Cliff Walker.

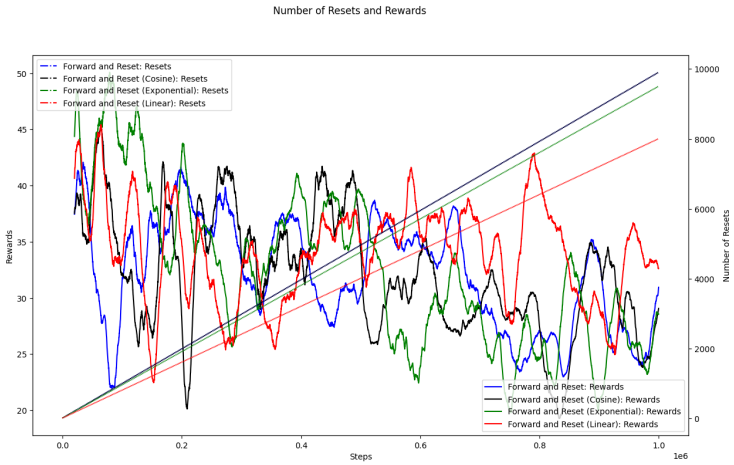


Figure 3: Pusher: Rewards and total resets when reset threshold is varied according to various schedules

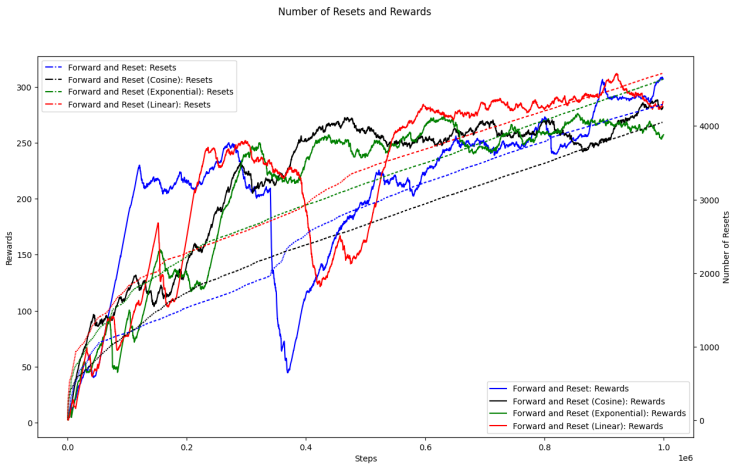


Figure 4: Cliff Walker: Rewards and total resets when reset threshold is varied according to various schedules

3.3 Reset Cost

Leave no Trace [1] treats all resets equally. To test the effects of assigning different costs to resets based on severity, we focus on the cliff walker environment. Here, we define two costs:

- Falling on the cliff: For this less severe reset, we assign a cost of -0.3.
- Falling off the cliff: Falling off the cliff can cause severe damage to the robot, and hence we assign a cost of -1.

Figure 5: (Left) Falling on cliff with a penalty of -0.3. (Right) Falling off cliff scenario with penalty of -1.

We add these costs to the forward and reset policy rewards.

$$r_{forward} = r_{forward} + \textit{reset\_cost}$$
$$r_{reset} = r_{reset} + \textit{reset\_cost}$$

We modified the maximum episode length to 500 in order to see the effects of the agent falling off the cliff.

3.3.1 Results

As seen in Figure 6, the total number of resets for the run with different reset costs is less than the number of resets when all resets are treated equally by 33%. Moreover, with manual cost, the reward received by the agent is 66% higher (around 500) compared to the agent trained without any resets costs (around 300).

Taking a closer look at when the robot warranted a manual reset, we see that with cost-aware resets, the robot falls less often on the cliff compared to the setting where all resets are treated equally (Figure 7). Initially, there are more resets for cost-aware resets. This might be due to the agent exploring more initially and not being afraid to fall into these bad scenarios, which may not be too dangerous.

The robot falls off the cliff 4 times when we use cost-aware resets, which is much more compared to the case where no reset cost is used (Figure 7). Looking at the rewards, we suspect that this might be due to the robot reaching the cliff end more often, and hence the agent falls off more. We believe that over more iterations, this trend will change.

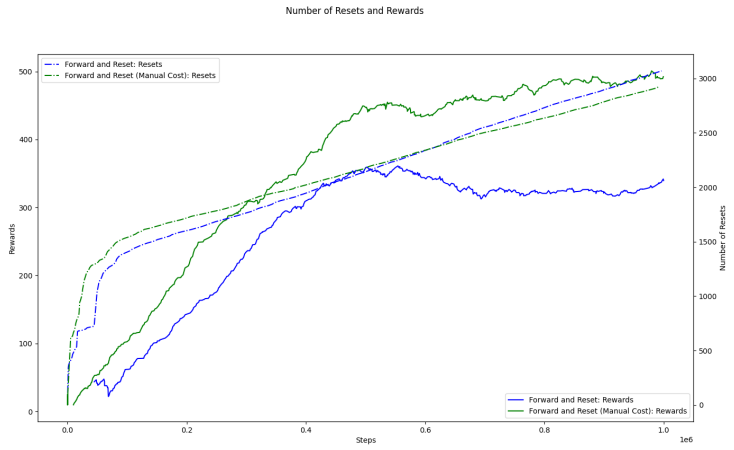


Figure 6: Effect on Rewards when Using Reset Cost on Cliff Walker Environment

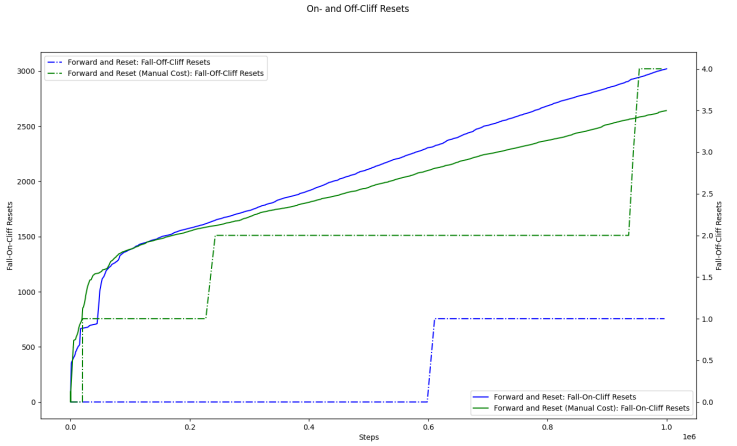


Figure 7: Different Resets when Using Reset Cost on Cliff Walker Environment

3.4 Choose-Policy

Approaches outlined in [1] and our 3.2 and 3.3 treat the threshold as a parameter. We switch between the forward and reset policy depending on how high the Q value from the reset policy is. For choose-policy, we train a new policy using DDQN with 2 actions: run forward policy and run backward policy. The state is the concatenation of the current state of the agent S, Q value estimated by the reset policy, and Q value estimated by the forward policy (Figure 8). We train this new agent until the first 700k steps, during which we use a scheduler as described in 3.2. After 700k steps, this new choose-policy takes over. The rewards for this new policy is defined as follows:

$$r_{choose} = \begin{cases} r_{forward} & \text{, if run forward policy} \\ r_{reset} & \text{, if run reset policy} \end{cases}$$

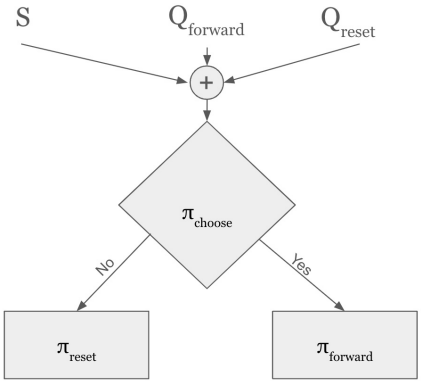
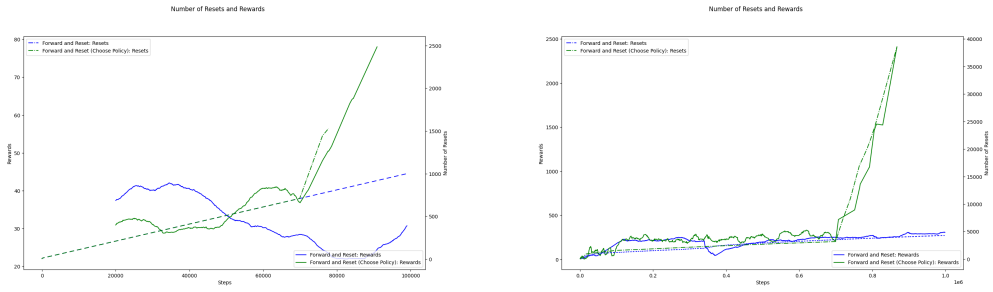


Figure 8: Choose Policy Flowchart

3.4.1 Results

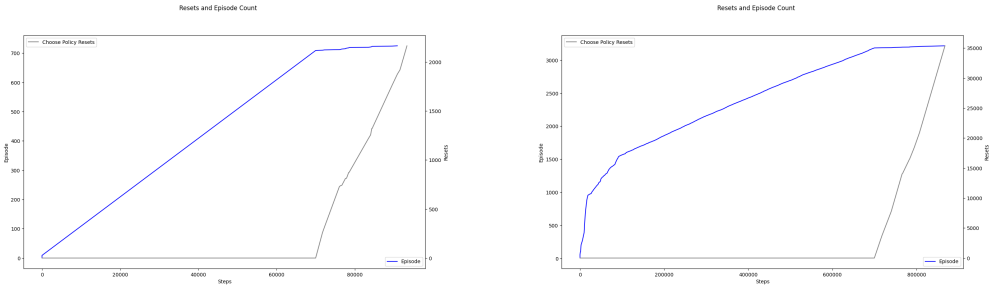
For Cliff Walker's case, after activating the choose policy at iteration 700k, we see that the rewards increase significantly (Figure 9b). This might give an illusion that the choose policy helps learn a better forward policy, where the robot never falls and needs to reset. When we look at the total number of resets per episode we see that, after iteration 700k, for the same episode, the choose policy is resetting approximately 36000 times (Figure 10b). This means that choose policy is reward hacking, which is a phenomenon where the policy tries to maximize rewards without learning the intended outcome of the task. Here, it lets the forward policy run, and then oscillates between reset and forward policy to get maximum rewards.

The results are similar for the pusher task. We see much higher rewards (Figure 9a) after the choose policy is activated compared to the standard threshold approach, with a huge increase in the number of resets chosen by the choose policy (Figure 10a).



(a) Pusher (b) Cliff Walker

Figure 9: Exploding rewards and resets once choose policy is activated.



(a) Pusher (b) Cliff Walker

Figure 10: The number of times choose policy decides to run reset policy per episode is very high, indicating reward hacking.

4 Future Directions and Conclusion

We investigated several directions in improving the *Leave no Trace* framework. In particular, we experimented with implementing dynamic reset triggers, variable reset costs, and a policy-switching mechanism.

The incorporation of dynamic triggers using schedules has allowed our agents to prioritize exploring more freely during the early stages of training and becoming more conservative as time goes on, thereby reducing the number of manual resets required. The implementation of dynamic triggers did increase rewards for the Pusher environment but had no large effect on the resulting training rewards for Cliff Walker environments. Having a constant threshold value yielded similar rewards

compared to a dynamic threshold value in that case. Trying with more random seeds can help to better understand the results in the Cliff walker case.

Our experimentation in assigning manual costs to different scenarios has proven to be able to increase the training reward and decrease the number of manual resets in the Cliff Walker environment. One shortcoming of our method is formulating how to assign manual costs as it may vary for each environment and can be a tedious task. Therefore, a future direction can be to create a framework that automatically assigns reset rewards and manual costs, depending on the environment.

In our implementation, we introduced a policy-choosing mechanism to toggle between forward and reset policies. This approach, however, has inadvertently facilitated reward hacking, where the system prioritizes maximizing immediate rewards over achieving the overarching goals. A future direction is to incorporate dynamic rewards that adjust based on the context, similar to techniques used in Visual Question and Answering. This may prevent reward overfitting specific to the environment.

While the experiments for this paper were done in simulation, where manual resets are inexpensive, the next step would be to apply our algorithm to real robots, where manual resets are costly. For now, our research builds a strong foundation for safer, more efficient reinforcement learning strategies, pointing the way forward for future innovations in the field. The methods developed and tested in this project not only enhance the safety and effectiveness of RL agents but also contribute to the broader understanding of strategic decision-making in complex and uncertain environments.

References

[1] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

[2] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

[3] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *CoRR*, abs/2104.11203, 2021. URL https://arxiv.org/abs/2104.11203.

[4] K. Xu, S. Verma, C. Finn, and S. Levine. Continual learning of control primitives: Skill discovery via reset-games. *Advances in Neural Information Processing Systems*, 33:4999–5010, 2020.

[5] J. Kim, D. Cho, and H. J. Kim. Demonstration-free autonomous reinforcement learning via implicit and bidirectional curriculum, 2023. URL https://arxiv.org/abs/2305.09943.

[6] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal. Safe reinforcement learning via curriculum induction, 2021. URL https://arxiv.org/abs/2006.12136.

[7] J. Kim, J. h. Park, D. Cho, and H. J. Kim. Automating reinforcement learning with example-based resets. *IEEE Robotics and Automation Letters*, 7(3):6606–6613, July 2022. ISSN 2377-3774. doi:10.1109/lra.2022.3173039. URL http://dx.doi.org/10.1109/LRA.2022.3173039.