AltNet: Addressing the Plasticity-Stability Dilemma in Reinforcement Learning

Anonymous Author(s)
Submission Id: «submission id»

ABSTRACT

Neural networks have shown remarkable success in supervised learning when trained on a single task using a fixed dataset. However, when neural networks are trained on a reinforcement learning task, their ability to continue learning from new experiences declines over time. This decline in learning ability is known as plasticity loss. To restore plasticity, prior work has explored periodically resetting the parameters of the learning network, a strategy that often improves overall performance. However, such resets come at the cost of a temporary drop in performance, which can be dangerous in real-world settings. To overcome this instability, we introduce AltNet, a reset-based approach that restores plasticity without performance degradation by leveraging twin networks. The use of twin networks anchors performance during resets through a mechanism that allows networks to periodically alternate roles: one network learns as it acts in the environment, while the other learns off-policy from the active network's interactions and a replay buffer. At fixed intervals, the active network is reset and the passive network, having learned from prior experiences, becomes the new active network. AltNet restores plasticity, improving sample efficiency and achieving higher performance, while avoiding performance drops that pose risks in safety-critical settings. We demonstrate these advantages in several high-dimensional control tasks from the DeepMind Control Suite, where AltNet outperforms various relevant baseline methods, as well as state-of-the-art reset-based techniques.

KEYWORDS

Plasticity, Stability, Plasticity-Stability Dilemma, Continual Learning, Reinforcement Learning, Network Resets

ACM Reference Format:
Anonymous Author(s). 2026. AltNet: Addressing the Plasticity-Stability Dilemma in Reinforcement Learning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 11 pages.

1 INTRODUCTION

Deep learning systems are often designed to learn and converge on a single task. In non-stationary environments, however, the goal being optimized by the model evolves over time. Success in such settings requires continual adaptation rather than the ability to identify a single solution. This need motivates the field of continual learning or lifelong learning, where an agent updates, accumulates, and exploits knowledge throughout its lifetime [5]. A central obstacle in continual learning is *plasticity loss*—the progressive decline in an agent's ability to learn from new data over time [6, 15, 16, 22]. We say that a network has lost plasticity if it can no longer optimize its objective as effectively as a freshly initialized counterpart [17]. Plasticity loss has been observed in non-stationary settings. For instance, Achille et al. [2] showed that pre-training on blurred CIFAR images impaired subsequent learning of the original dataset. Similarly, Ash and Adams [3] found that pre-training on half of a dataset and using the resulting model as a starting point when tackling a supervised learning task reduced accuracy compared to training on the full dataset from scratch. More broadly, Dohare et al. [7] demonstrated that when neural networks are trained sequentially on multiple tasks, their ability to learn new tasks declines with each additional task.

Reinforcement learning (RL) compounds the difficulty of maintaining plasticity over time because, even when the task itself is stationary, RL agents face inherent sources of non-stationarity. First, agents collect their own data; as policies evolve, the distribution of encountered states and actions shifts, producing *input non-stationarity*. Second, many RL algorithms such as DQN, A2C, PPO, and SAC [10, 19, 20, 24] rely on bootstrapping, where predictions of future rewards serve as learning targets. As these predictions evolve, the targets themselves change, creating *target non-stationarity*. Together, these factors require agents to continually adapt to shifting data distributions even when tackling a single task, thereby amplifying plasticity loss.

To mitigate plasticity loss, various approaches have been proposed (Section 2). Among these, a particularly promising family of methods is based on periodically resetting network parameters [6, 14, 22, 25]. Resets are effective because they restore the network to a well-conditioned, highly plastic initialization that is gradually lost during training. As networks adapt to specific tasks or data distributions, they accumulate pathologies—such as dormant neurons, growing weight magnitudes, and reduced rank—that impair their ability to learn from new data [6]. Resetting the parameters removes these accumulated effects and reinitializes the network to conditions resembling its original, plastic initialization (see supporting analysis in Appendix E). Nikishin et al. [22] empirically demonstrated that resetting a network can substantially improve performance by renewing its ability to learn and exploit data. Although effective, full network resets come at a cost: they erase all information embedded in the network and cause immediate performance collapses (see Figure 2, orange curve). This makes Standard Resets [22] impractical for real-world deployment. The central challenge we address in this paper is how to retain the benefits of full network resets in restoring plasticity while avoiding the performance instability they induce.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

To address the plasticity-stability dilemma, we introduce *AltNet*, a reset-based alternating network approach that preserves plasticity without inducing recurring performance drops. AltNet maintains two networks that periodically switch roles. At any given time, the *active network* interacts with the environment, while the *passive network* learns off-policy from the active agent's experience and a shared replay buffer. At fixed intervals, the active network is reset and the passive network, having learned from prior experiences, becomes the new active network. This alternating structure anchors performance across resets and prevents performance collapse. Importantly, AltNet successfully leverages resets without any performance instability even at a low replay ratio of 1; in these cases, by contrast, Standard Resets [22] fail (see Figure 2, orange curve) and more sophisticated methods such as Resets with Deep Ensembles (RDE) [14] still exhibit sharp post-reset performance drops (see Figure 2, blue curve). To understand which factors contribute to AltNet's superior performance, we systematically evaluate aspects such as model capacity, number of networks, replay ratio, buffer size, and reset duration (subsection 4.2). Finally, we show that AltNet also improves performance in on-policy settings, as demonstrated by comparisons with the on-policy baseline, PPO [24] (subsection 4.3).

2 RELATED WORK

Plasticity. Prior work uses the term *plasticity* to refer to the degree to which a network generalizes to unseen data [4] or to refer to its ability to continue improving performance on its training objective over time [1, 15, 17, 21]. In this paper, we adopt the latter meaning. We say that a network has lost plasticity if it can no longer optimize its objective as effectively as a freshly initialized counterpart.

Plasticity loss in reinforcement learning. Several prior works point to the same underlying challenge: neural networks in reinforcement learning often lose their ability to adapt as training progresses. Lyle et al. [16] observe a gradual loss of capacity to fit evolving targets even in the single task, while Kumar et al. [15] attribute a similar effect to implicit under-parameterization. Nikishin et al. [22] introduce the term *primacy bias*, referring to the tendency of agents to overfit to early experiences, which hinders subsequent learning. Although framed in different ways, these studies describe facets of the same phenomenon—plasticity loss. We choose the terminology *plasticity loss* because it captures the common thread: the gradual decline in an agent's ability to adapt to new information.

Causes of plasticity loss. The precise cause of plasticity loss remains unknown. Several correlates have been identified, such as inactive neurons, growth of the network's average weight magnitude, decrease in the expressivity of the network, and changes in the curvature of the loss landscape [7, 15, 18, 25]. None of the proposed correlates, however, provide a consistent explanation across settings. For example, Lyle et al. [18] show that for any proposed correlate, counterexamples can be constructed where the correlation disappears or even reverses. Since the underlying cause of plasticity loss has not been identified, it is difficult to directly determine whether a system has retained or lost plasticity. Following prior work [14, 21, 22], we use performance as a proxy for plasticity.

A wide range of strategies have been proposed to mitigate plasticity loss. Broadly, these fall into two families: methods based on regularization, which constrain or perturb weights to preserve plasticity, and methods based on resets, which periodically reinitialize parts or the entire network. Below, we briefly review each in turn:

(1) **Regularization-based strategies.** Prior work has explored regularization-based strategies to maintain plasticity. While L2 penalties can slow down weight growth, they sometimes aggravate rank collapse by biasing weights toward the origin [7, 18]. To address this, methods such as Shrink-and-Perturb [3] and L2 Init [15] have been proposed, which encourage weight updates toward high-plasticity initializations while preserving feature diversity.

(2) **Reset-based strategies.** Another family of methods directly resets the network in whole or in part. Continual Backprop [6] and ReDO [25] reset subsets of neurons selected for low utility or persistent inactivity. Igl et al. [12] propose distilling a trained policy into a newly initialized network, which can be seen as a form of reset with distillation as the transfer mechanism. Nikishin et al. [22] propose periodic full network resets, relying on the replay buffer to transfer knowledge, but these cause sharp performance drops. In this paper, we refer to this approach as Standard resets. Reset Deep Ensembles (RDE) [14] leverages full network resets by maintaining an ensemble of networks, with each network reset in turn to induce plasticity. Actions are chosen through a Q-value–weighted voting scheme, where each proposed action is weighted by the critic of the oldest network in the ensemble. Although RDE (Figure 2, blue curves) improves stability over Standard Resets (Figure 2, orange curves) [22], it still suffers from significant post-reset performance drops because a freshly reset, untrained network can still act in the environment.

3 ALTNET

In this section, we introduce AltNet, a dual-network reset-based architecture designed to restore plasticity while maintaining stability in reinforcement learning.

Central Hypothesis. Prior work has shown that full resets can restore plasticity [22], but they also cause sharp performance collapses when the reset policy acts immediately (see Figure 2, orange curve). We hypothesize that leveraging two insights can reconcile this plasticity-stability dilemma: (i) resetting a neural network initializes it to a highly plastic state, from which it may be possible to learn a better policy, as compared to a trained network, and (ii) using well-trained networks for interaction with the environment prevents performance drops. To combine the benefits of both, AltNet introduces a dual-network architecture that allows resets to occur while avoiding performance instability.

Architecture. AltNet is composed of two networks that alternate roles at a fixed interval, `ResetFreq` (Figure 1). At any given

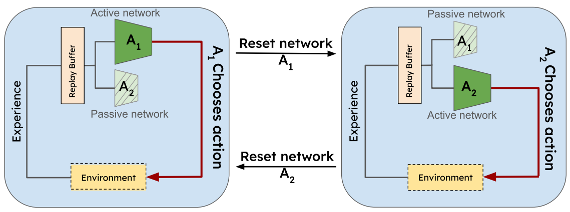


Figure 1: AltNet maintains two networks, A_1 and A_2, which share a replay buffer and alternate roles over time. Initially, A_1 (dark green) is active and collects experience by directly interacting with the environment, while A_2 (light green) remains passive and undergoes off-policy updates. At every `ResetFreq` steps, the active network is reset and becomes passive, while the previously passive network becomes active. This cyclic alternation enables frequent resets to maintain plasticity without sacrificing stability.

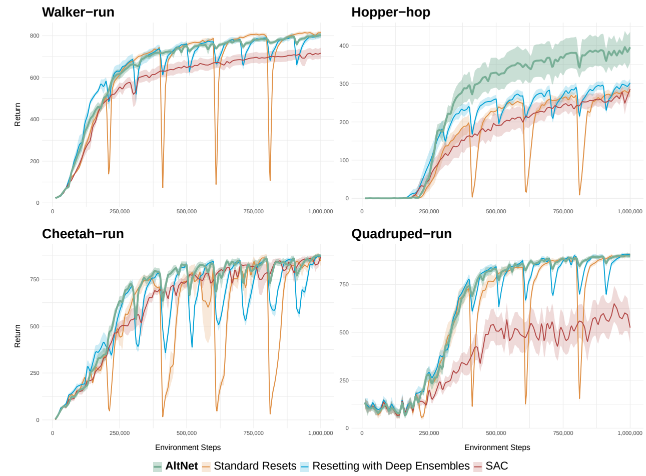


Figure 2: Learning curves in four DMC environments when replay ratio = 1. Results are averaged over 10 seeds; shaded regions indicate ±1 standard error. AltNet (green) avoids post-reset drops and achieves higher returns compared to SAC (red), Standard Resets (orange), and RDE (blue).

time, the active network interacts with the environment, while the passive network learns off-policy from the experiences of the active network and a replay buffer. The replay buffer is shared between the twin networks. Every `ResetFreq` steps the active network is reset and becomes the passive network and vice versa. This alternating cycle ensures that resets occur frequently enough to counter plasticity loss, yet performance remains stable because only trained networks interact with the environment.

Key Innovation. AltNet makes a structural departure from prior reset-based approaches. It prevents recently-reset networks from acting in the environment until they have received sufficient training. In contrast, Standard Resets [22] expose the reset network directly to the environment, making immediate performance collapse inevitable. RDE [14] employs ensembles with a Q-value–weighted gating policy to reduce the likelihood that a reset agent acts prematurely, but still allows recently reset networks to act. AltNet, on the other hand, guarantees that only trained networks interact with the environment. In AltNet, reset networks first train passively before taking over. Empirically, the result is stronger and simpler: AltNet avoids post-reset performance drops across replay ratios, and achieves higher and more stable returns (see Figure 2). More broadly, AltNet shows that even with full network resets, plasticity and stability can be simultaneously achieved.

4 RESULTS AND ANALYSIS

In this section, we present empirical evidence that AltNet effectively addresses the plasticity–stability dilemma. We begin by contextualizing its performance through comparisons with established reset-based and non-reset baselines in diverse continuous-control environments. Next, we analyze the contributions of replay buffer preservation, network alternation, and periodic resets to higher performance. Finally, we extend our investigation to on-policy settings, where replay buffers are absent, to examine how alternating resets influence learning dynamics in these regimes.

We now evaluate AltNet in stationary continuous-control tasks from the DeepMind Control Suite [26], with a focus on the off-policy setting, where the underlying algorithm for all methods is Soft Actor–Critic (SAC) [10], which also serves as our baseline. SAC leverages a replay buffer to reuse past experiences. Later in subsection 4.3, we extend our experiments to include an on-policy reinforcement learning algorithm, PPO, which does not rely on a replay buffer, to assess whether AltNet's benefits persist under these different regimes.

All agents are trained for 1M environment interactions. The reset interval is fixed at 200,000 gradient updates (U). We vary the replay ratio $RR \in \{1, 4\}$, defined as the number of gradient updates per environment step. To ensure a fair comparison, we adopt the reset frequency normalization proposed by Kim et al. [14], which accounts for both the replay ratio (RR) and the number of networks (N), as shown in (1). We compare AltNet against the non-reset-based baseline SAC [10], as well as state-of-the-art full network reset-based methods: Standard Resets [22] and Resets with Deep Ensembles (RDE) [14].

$$\texttt{ResetFreq}_{\text{(env steps)}} = \frac{U}{RR \times N} \tag{1}$$

As shown in Figure 2, when $RR = 1$, the performance of Standard Resets collapses almost immediately, while RDE exhibits sharp post-reset drops in return. In contrast, AltNet avoids these failures by anchoring performance during resets through a twin network that assumes control after each reset. Consequently, AltNet achieves

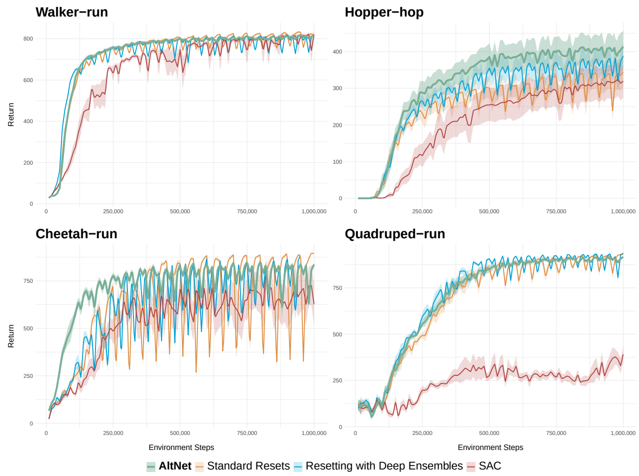


Figure 3: Learning curves in four DMC environments when replay ratio = 4. Results are averaged over 10 seeds; shaded regions indicate ±1 standard error. AltNet (green) avoids post-reset drops and achieves higher returns compared to SAC (red), Standard Resets (orange), and RDE (blue).

higher average returns and smoother learning curves across tasks.

When $RR = 4$, AltNet continues to outperform SAC and Standard Resets and match or exceed the performance of RDE, while maintaining stable training dynamics and avoiding post-reset performance drops (see Figure 3).

We also report the *normalized area under the learning curve (AUC)* (Table 1), which summarizes the overall learning trajectory of an agent (see Appendix B for details of the AUC calculation). The AUC integrates performance over time, rewarding both higher returns and greater stability: agents that maintain strong performance throughout training achieve a higher AUC. We normalize the AUC to rescale the values into a consistent range, making comparisons more interpretable. As shown in Table 1, AltNet achieves the highest average AUC across environments and replay ratios, outperforming SAC by approximately 38%, SR by 12%, and RDE by 6% on average.

4.1 AltNet enhances Sample Efficiency

In this section, we evaluate whether AltNet improves performance and sample efficiency as compared to an SAC baseline. We also compare AltNet against SAC trained at higher replay ratios—a common strategy to improve sample efficiency.

In reinforcement learning, agents learn through direct interaction with the environment, which is often slow and expensive in real-world domains such as robotics or healthcare applications. This makes *sample efficiency*—learning as much as possible from limited interactions—a central concern. A common strategy to improve sample efficiency is to increase the *replay ratio (RR)*, defined as

Table 1: Normalized AUC of different methods across DMC environments. The best method in each environment is highlighted in bold. AltNet achieves the highest normalized AUC, outperforming SAC by approximately 38%, SR by 12%, and RDE by 6% on average.

Environment	AltNet	RDE	SAC	SR
Cheetah (RR=1)	**658.27**	596.62	616.12	529.94
Hopper (RR=1)	**248.68**	189.79	156.69	154.52
Quadruped (RR=1)	**619.12**	609.36	377.27	568.36
Walker (RR=1)	**645.76**	643.22	570.08	617.06
Cheetah (RR=4)	**721.85**	619.15	535.80	595.13
Hopper (RR=4)	**313.78**	278.29	205.00	248.94
Quadruped (RR=4)	703.74	**717.24**	240.93	687.43
Walker (RR=4)	**728.49**	723.64	653.82	725.44
Average (RR=1)	**542.96**	509.75	430.04	467.47
Average (RR=4)	**616.97**	584.61	408.89	565.24

the number of gradient updates performed per environment step [8, 9, 27]. Higher replay ratios allow agents to reuse past experiences more extensively, thereby extracting additional learning signals from limited data. However, increasing RR also linearly increases computational cost and, beyond a point, can degrade performance due to overfitting to outdated experiences [8, 22]. As shown in Figure 4, increasing the replay ratio from 1 to 8 improves SAC's performance by allowing more updates per sample, but performance is substantially lower when $RR = 32$. AltNet, on the contrary, achieves superior performance even at $RR = 1$ (see Figure 8) and $RR = 4$ (see Figure 4), surpassing the SAC trained with much higher replay ratios. This shows that unlike SAC, AltNet achieves higher performance and greater sample efficiency with lower replay ratios, and consequently reduced computational overhead. We further establish that AltNet's benefits are not derived from an increase in capacity due to an additional network: reducing the number of parameters between both networks to match a single SAC network yields nearly identical performance to AltNet with larger networks(Figure 5).

To quantify sample efficiency directly, we measure the *return at fixed interaction budgets* (100k, 300k, 500k steps). AltNet consistently achieves higher returns at every fixed budget and reaches the target threshold substantially earlier than SAC across replay ratios (Table 2). AltNet outperforms the best SAC baselines at fixed budgets by **52**× (100k), **1.8**× (300k), and **1.3**× (500k). These results indicate that AltNet makes more effective use of each environment interaction and provides a more computationally efficient path to sample-efficient learning than simply increasing the replay ratio.

4.2 What accounts for AltNet's success?

Having demonstrated AltNet's advantages, we now investigate the mechanisms behind its success: first ruling out model capacity as a primary factor, and then isolating the contributions of alternating networks, resets and preservation of replay buffer.

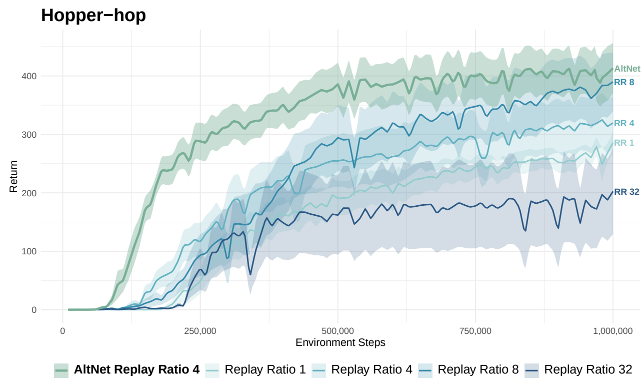


Figure 4: Learning curves of SAC in the `hopper-hop` environment (DMC) under different replay ratios (RR = 1, 4, 8, 32) and AltNet at RR = 4. Curves show mean episodic return over 10 seeds, with shaded regions denoting ±1 standard error. For SAC, performance improves as RR increases up to 8, but degrades at RR = 32. AltNet achieves highest performance at RR = 4 and is the most sample efficient.

Table 2: Fixed-budget returns at 100k, 300k, and 500k environment steps. AltNet achieves higher returns at every interaction budget, indicating greater sample efficiency compared to baseline SAC at various Replay Ratios.

Method	100k Steps	300k Steps	500k Steps
AltNet (RR = 4)	**43.7**	**312.8**	**385.7**
SAC (RR = 1)	0.14	106.8	190.7
SAC (RR = 4)	0.84	171.2	254.6
SAC (RR = 8)	0.56	79.9	294.3
SAC (RR = 32)	0.12	120.9	161.9

RQ1. Does the AltNet advantage arise from the increase in model capacity? A natural hypothesis is that AltNet's advantage arises simply from its increased number of trainable parameters, since it uses two networks. To test this, we reduced the size of the networks in AltNet to match the total number of trainable parameters to that in the single-network baseline (SAC). AltNet retained its performance (Figure 5, orange curve).

RQ2. Can scaling the number of networks in AltNet provide additional benefits? While the alternating network setup in AltNet is primarily designed to anchor performance during resets, it may also encourage exploration by maintaining two policies that learn in parallel and take turns acting in the environment. This raises the question: can increasing the number of networks beyond two further improve performance? To test this, we modify AltNet to use four networks and adjust the reset frequency accordingly, as specified in (1). As shown in Figure 5 (brown curve), scaling beyond two networks does not yield additional gains. Therefore, we conclude that policy diversity is not a primary factor in AltNet's gains.

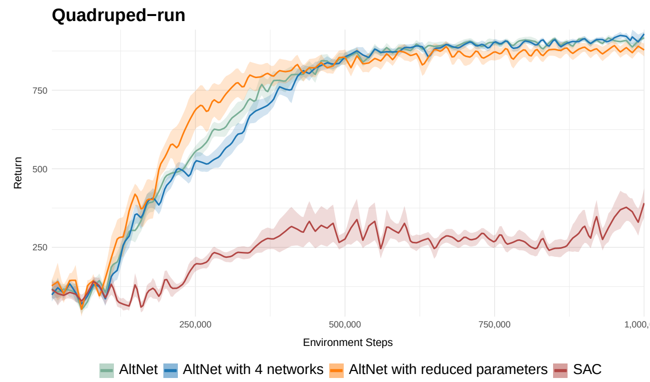


Figure 5: We compare standard AltNet (green), a reduced-parameter version matching SAC's parameter count (orange), and a scaled variant with four alternating networks (blue), against the SAC baseline (red). Curves report mean episodic return over 5 seeds, with shaded regions indicating ±1 standard error. Despite reduced capacity or increased number of networks, AltNet achieves similar performance, ruling out model size and network count as drivers of its gains.

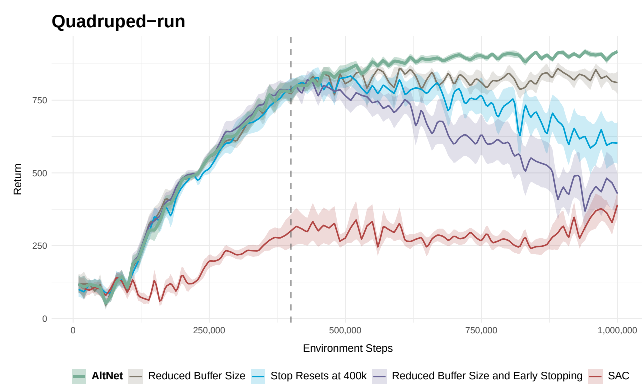


Figure 6: We compare standard AltNet (green), reduced buffer size to 400k (grey), resets halted after 400k steps (blue), both interventions combined (red), and the SAC baseline (purple). Curves report mean episodic return over 10 seeds, with shaded regions indicating ±1 standard error. Both preserving the full replay buffer and maintaining resets are essential for sustaining AltNet's stability and performance.

This rules out model size and number of networks as the primary driver of AltNet's gains and shifts attention to its defining mechanisms. Specifically, AltNet's performance when applied to SAC relies on two interacting processes: (i) preservation of a shared replay buffer that preserves knowledge across resets, and (ii) periodic alternating resets that restore network plasticity. To examine the role of each component, we perform experiments that disrupt them individually and in combination (Figure 6).

RQ3. Is the preservation of the replay buffer critical to AltNet's stability? The replay buffer plays a critical role in AltNet by providing continuity of learning in the resets. If both the network and buffer were reset simultaneously, training would effectively restart from near-scratch, eliminating the benefits of alternating resets. In the experiments described so far, we have retained every agent interaction in the replay buffer. However, most off-policy algorithms do not preserve the full replay buffer indefinitely. To test how much buffer preservation is essential for AltNet, we reduced the replay buffer capacity from the default 1M transitions to 400k, replacing old samples in a first-in, first-out manner. As shown in Figure 6 (gray curve), reducing replay buffer capacity led to a noticeable decline in performance, compared to runs with the full buffer. These results indicate that preserving the full replay buffer is critical for stabilizing resets and maintaining AltNet's advantage.

RQ4. Are alternating resets necessary once performance plateaus? AltNet provides substantial performance gains during the early stages of training, after which progress naturally slows as performance approaches the task's achievable optimum. For example, in `Quadruped-run`, most improvements occur within the first 500k steps, after which performance stabilizes (see Figure 6). This raises a natural question: once learning has plateaued, can a single network—without alternating resets—maintain both plasticity and stability, continuing to perform well over time? To test this, we stopped the resets after 400k steps while preserving the full replay buffer. As shown in Figure 6 (blue curve), performance declined once resets were discontinued. This indicates that resets remain essential even after apparent convergence, as stopping them leads to a gradual loss of plasticity, where it is no longer possible to reinforce or adapt the behavior of a network effectively. Network alternation, in contrast, preserves both plasticity and stability, allowing AltNet to maintain high performance over time.

RQ5. What happens when both mechanisms (buffer preservation and resets) are disrupted simultaneously? Finally, we examine the combined effect of disrupting both mechanisms. If resets and buffer preservation are essential for plasticity and stability, then removing both should compound performance degradation. As expected, stopping resets while simultaneously reducing replay buffer size produced the lowest returns, other than baseline SAC (see Figure 6, purple curve).

To verify that these findings are robust to training hyperparameters and reset timing, we repeated the experiment with a different learning rate and a different timestep to stop resets and discard experiences, observing the same qualitative pattern (see Appendix D). These findings demonstrate that AltNet's gains are not driven by increased model capacity; it maintains its advantages even when the total number of parameters is matched to that of the baseline SAC. In addition, increasing the number of networks does not yield any further gains. Rather, AltNet's improvements arise from the interplay between alternating resets and replay buffer preservation. Resets restore plasticity by erasing accumulated pathologies such as neuron inactivity, rank collapse, or inflated weight norms (see Appendix E); it reinitializes the network into a more plastic and learnable state, allowing it to better exploit the increasingly representative replay buffer as training progresses. Resets also allow diverse policies to sequentially interact in the environment, which may help with exploration. Alternating networks and preservation of the replay buffer maintain the stability and continuity of learning across resets in SAC. We find that disabling either of these components, alternating resets or replay buffer preservation, damages performance when paired with SAC, and both must be maintained throughout training. Next, we examine whether AltNet's benefits extend to the on-policy methods that lack a replay buffer.

4.3 AltNet in on-policy settings

On-policy reinforcement learning provides a critical test of a method's generality and robustness. In on-policy RL, agents collect trajectories by following their current policy, and updates are performed only on those freshly collected samples. Agents do not have access to a replay buffer to use past data.

Prior full network reset methods, such as Standard Resets [22] and RDE [14], have been developed and evaluated primarily in off-policy settings. However, Juliani and Ash [13] show that methods effective in off-policy reinforcement learning often fail to provide benefits in on-policy algorithms, where optimization dynamics differ substantially. Therefore, we investigate whether AltNet's benefits can extend to on-policy settings that lack a replay buffer.

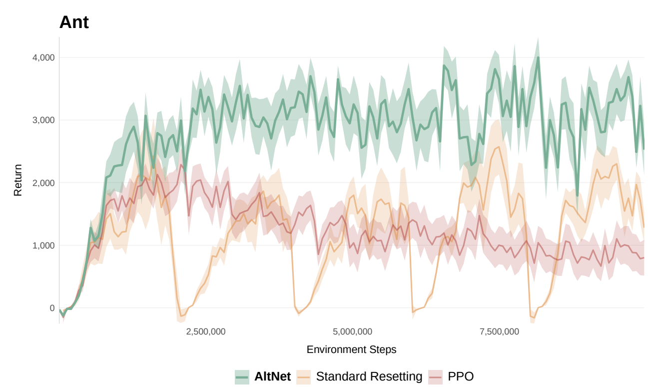


Figure 7: Training performance of PPO (red curve), PPO augmented with Standard Resets (orange curve) and AltNet (green curve) in the MuJoCo `Ant` environment. Curves report mean episodic return over 10 seeds, except for Standard Resets, which were averaged over 5 seeds; shaded regions denote ±1 standard error. AltNet provides consistent gains, demonstrating its efficacy in on-policy settings.

We evaluate AltNet in MuJoCo's Ant environment for 10M steps, a long-horizon setting where standard PPO suffers severe plasticity loss: after reaching moderate reward, performance declines as the agent fails to sustain learning (see Figure 7, red curve). In contrast, AltNet achieves nearly twice the baseline performance and maintains it indefinitely (see Figure 7, green curve). AltNet's benefits persist due to an implicit form of knowledge transfer between its networks. While one network interacts with the environment, the other recently reset network learns in parallel from the same trajectories, enabling it to recover useful representations. This parallel learning anchors performance across resets and allows AltNet to maintain stability and plasticity in the on-policy regime.

We further compare AltNet with Standard Resets [22], which, without a replay buffer, suffer post-reset performance collapse (see Figure 7, orange curve). Note that RDE [14] is not included in these experiments, as it is not designed for on-policy reinforcement learning. It relies on Q-value scoring—ranking actions by expected future reward using an off-policy critic and replay buffer—whereas PPO's critic estimates only state values, not action values. AltNet thus emerges as the only full network reset method capable of restoring plasticity and achieving consistent performance gains across both off-policy and on-policy settings.

5 DISCUSSION

While our work focuses on mitigating plasticity loss, it is important to situate plasticity within the broader set of attributes of an effective reinforcement learning agent. Such an agent should remain plastic, retaining the capacity to update its predictions over time, adapt rapidly when distributions shift, make full use of past data, and achieve strong performance with limited interactions. An ideal RL agent should preserve existing capabilities and ensure performance stability. Plasticity underpins these goals; without the capacity to change, rapid adaptation, data efficiency, and stable long-term performance are negatively impacted. Viewed through this lens, AltNet addresses more than plasticity loss. It also enables rapid adaptation, sustained exploitation of replay buffers, and efficient use of data. And, by anchoring performance through a twin network, it learns stably and without performance degradation. Together, preserving plasticity and stability, achievable through AltNet, are foundational aspects for the broader qualities that characterize an ideal reinforcement learning agent.

6 LIMITATIONS AND FUTURE WORK

Although AltNet demonstrates strong empirical gains and stability across a range of continuous control tasks, our study has limitations. First, we focus our experiments on problems from the DeepMind Control Suite which are representative of challenging control problems used to evaluate state-of-the-art RL methods. That said, extending the evaluation to more diverse environments could provide further insights. Second, AltNet relies on the choice of reset frequency, which currently follows a standard schedule in our experiments. Developing an adaptive scheduling mechanism to automatically determine the optimal reset frequency for each environment and replay ratio is relevant future work.

To support reproducibility and further research, we will release the implementation and training scripts upon acceptance.

REFERENCES

[1] Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. 2023. Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*. PMLR, 620–636.

[2] Alessandro Achille, Matteo Rovere, and Stefano Soatto. 2017. Critical learning periods in deep neural networks. *arXiv preprint arXiv:1711.08856* (2017).

[3] Jordan Ash and Ryan P Adams. 2020. On warm-starting neural network training. *Advances in neural information processing systems* 33 (2020), 3884–3894.

[4] Tudor Berariu, Wojciech Czarnecki, Soham De, Jorg Bornschein, Samuel Smith, Razvan Pascanu, and Claudia Clopath. 2021. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042* (2021).

[5] Zhiyuan Chen and Bing Liu. 2018. *Lifelong machine learning*. Morgan & Claypool Publishers.

[6] Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. 2024. Loss of plasticity in deep continual learning. *Nature* 632, 8026 (2024), 768–774.

[7] Shibhansh Dohare, Richard S Sutton, and A Rupam Mahmood. 2021. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325* (2021).

[8] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. 2022. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop NeurIPS 2022*.

[9] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. 2020. Revisiting fundamentals of experience replay. In *International conference on machine learning*. PMLR, 3061–3071.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. Pmlr, 1861–1870.

[11] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. 2022. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research* 23, 274 (2022), 1–18.

[12] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. 2020. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826* (2020).

[13] Arthur Juliani and Jordan Ash. 2024. A study of plasticity loss in on-policy deep reinforcement learning. *Advances in Neural Information Processing Systems* 37 (2024), 113884–113910.

[14] Woojun Kim, Yongjae Shin, Jongeui Park, and Youngchul Sung. 2023. Sample-efficient and safe deep reinforcement learning via reset deep ensemble agents. *Advances in Neural Information Processing Systems* 36 (2023), 53239–53260.

[15] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. 2020. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498* (2020).

[16] Clare Lyle, Mark Rowland, and Will Dabney. 2022. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560* (2022).

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

[18] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. 2023. Understanding plasticity in neural networks. In *International Conference on Machine Learning*. PMLR, 23190–23211.

[19] Volodymyr Mnih et al. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning*. 1928–1937.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[21] Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. 2023. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems* 36 (2023), 37142–37159.

[22] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. 2022. The primacy bias in deep reinforcement learning. In *International conference on machine learning*. PMLR, 16828–16847.

[23] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of machine learning research* 22, 268 (2021), 1–8.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[25] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. 2023. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 32145–32168.

[26] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690* (2018).

[27] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando De Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224* (2016).

A HYPERPARAMETERS USED

The hyperparameters for AltNet and baseline agents follow standard practice in continuous-control reinforcement learning, with modifications only where required for resets. For reproducibility, we report the exact values used in the DMC environment.

The hyperparameters for all algorithms follow standard practice in continuous-control reinforcement learning. The PPO implementation is based on the `CleanRL` [11], extended with our alternating reset mechanism (AltNet). The Soft Actor–Critic (SAC) and its reset-based variants (Standard Resets, RDE, and AltNet) are implemented using the `Stable-Baselines3` framework [23], following the exact configuration used by RDE [14], on top of which we build our extensions. Only modifications related to resets and network alternation were introduced; all other settings follow default implementations.

B AUC DETAILS

We use the *area under the learning curve (AUC)* to quantify overall learning efficiency. This measure reflects both how quickly and how effectively an agent learns, combining stability and final performance into a single scalar. Higher AUC indicates agents that not only reach strong performance but also maintain it consistently throughout training.

Table 3: Hyperparameters for PPO

Hyperparameters	Value
Environment	`Ant-v4` (MuJoCo)
Total timesteps	1×10^7
Seed	1
Number of environments	1
Algorithm	PPO (CleanRL)
Number of networks (AltNet)	2 (active / passive)
Reset frequency (environment steps)	2×10^5
Optimizer	Adam
Optimizer: learning rate	0.0003
Optimizer: ϵ	1×10^{-5}
Learning rate annealing	False
Discount factor (γ)	0.99
GAE λ	0.95
Rollout length (per env)	2048 steps
Number of minibatches	32
Number of update epochs	10
Batch size	2048
Minibatch size	64
Advantage normalization	True
Clip coefficient (ϵ)	0.2
Value loss coefficient (c_v)	0.5
Entropy coefficient (c_e)	0.0
Max gradient norm	0.5
Target KL divergence	None
Network architecture	MLP (64–64)
Trainable log standard deviation	Initialized to 0

Table 4: Hyperparameters for SAC and variants

Hyperparameters	Value
# of network (AltNet and RDE)	2
# of network (Baseline and Standard Reset)	1
Training steps	1×10^6
Discount factor	0.99
Warm up period	5000
Minibatch size	1024
Optimizer	Adam
Optimizer : learning rate	0.0003
Networks : activation	ReLU
Networks : n. hidden layers	2
Networks : neurons per layer	1024
Initial Temperature	1
Replay Buffer Size	1×10^6
Updates per step (Replay Ratio)	(1, 4)
Target network update period	1
τ (Polyak update)	0.005
Reset Frequency (gradient steps) for all	2×10^5
β (action select coefficient) for RDE	50

To summarize each agent's overall learning trajectory, we compute the *area under the learning curve (AUC)* from training returns. For a learning curve defined by episodic returns R_t recorded at discrete training steps $t \in [t_0, t_T]$, the AUC is given by

$$\mathrm{AUC}_{\mathrm{raw}} = \sum_{i=1}^{T-1} \frac{R_{t_i} + R_{t_{i+1}}}{2} (t_{i+1} - t_i) \quad (2)$$

which approximates the integral of performance over time using the trapezoidal rule. For readability and comparability, we report the *normalized* AUC, obtained by dividing $\mathrm{AUC}_{\mathrm{raw}}$ by the total number of training steps ($t_T - t_0 = 1\mathrm{M}$). The resulting values therefore represent the average return over the training horizon rather than the raw cumulative sum. Normalization does not affect relative rankings—since all runs share the same training horizon—but yields interpretable magnitudes suitable for cross-environment comparison, as reported in Table 1.

C ADDITIONAL RESULTS ON SAMPLE EFFICIENCY

This section provides supplementary results related to subsection 4.1, which examines how AltNet improves performance and sample efficiency compared to SAC. While the main paper (subsection 4.1) focuses on AltNet at $RR = 4$, here we include additional learning curves for AltNet trained at $RR = 1$ to provide a complete

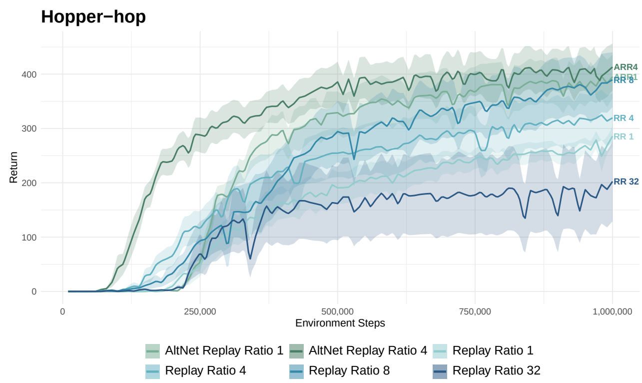
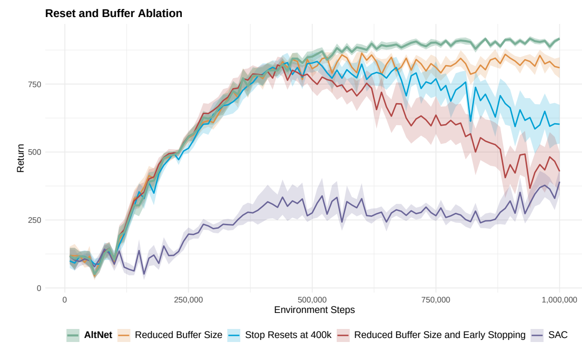
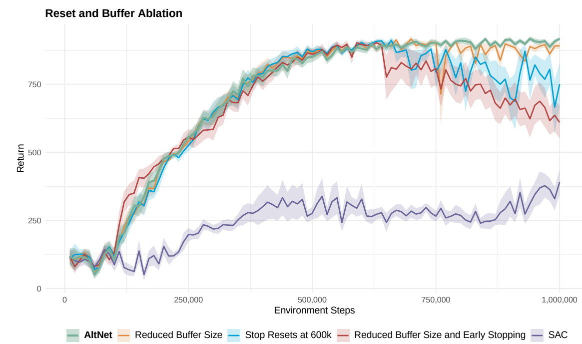


Figure 8: Learning curves of SAC in the `hopper-hop` environment (DMC) under different replay ratios (RR = 1, 4, 8, 32) and AltNet at RR = 1 and RR = 4. Curves show mean episodic return over 10 seeds, with shaded regions denoting ±1 standard error. For SAC, performance improves as RR increases up to 8, but degrades at RR = 32. AltNet achieves highest performance at RR = 4 and is the most sample efficient.



(a) Resets halted after 400k steps. Buffer size reduced to 400k.



(b) Resets halted after 600k steps. Buffer size reduced to 600k.

Figure 9: Analysis in the `Quadruped-run` environment (DMC). Curves show mean episodic return over 10 seeds, with shaded regions denoting ±1 standard error. We compare standard AltNet (green), reduced buffer size (orange), resets halted (blue), both interventions combined (red), and the SAC baseline (purple). Results demonstrate that both preserving the full replay buffer and maintaining resets are essential for AltNet's stability.

view of its behavior across replay ratios.

Figure 8 extends Figure 4 from the main text by showing AltNet's performance at both $RR = 1$ and $RR = 4$, alongside SAC trained at $RR \in \{1, 4, 8, 32\}$. The results confirm the same qualitative pattern observed earlier: as the replay ratio increases, SAC initially improves but eventually overfits and degrades at very high RR, while AltNet achieves consistently strong performance even at the lowest replay ratio.

Together, these additional curves reinforce the findings of subsection 4.1: AltNet achieves higher performance and greater sample efficiency without requiring aggressive replay schedules or the associated computational overhead.

D ADDITIONAL EXPERIMENTS

This section provides supplementary analyses supporting the results discussed in subsection 4.2 (What accounts for AltNet's success?). There, we examined how AltNet's performance depends on two interacting mechanisms: alternating resets and replay-buffer preservation. To verify that those findings are robust to training hyperparameters and specific reset timings, we repeat the same ablation experiments under modified conditions.

Specifically, we vary reset timing, halting resets at 600k steps instead of 400k to test whether AltNet's dependence on resets is sensitive to the cutoff point, and Learning rate, reducing it from 0.0003 to 0.0002 to confirm that the observed trends are not optimizer-specific.

Figure 9 compares AltNet, reduced-buffer, halted-reset, and combined interventions in the *Quadruped-run* environment for both 400k and 600k step halts. The qualitative pattern remains unchanged: performance declines sharply when resets or buffer preservation are removed, and most severely when both are disrupted.

Figure 10 repeats the same experiment at the lower learning rate and again observes consistent behavior.

Together, these additional experiments reinforce the conclusions of subsection 4.2: AltNet's advantages are not artifacts of specific hyperparameter choices or reset schedules. Network alternation and replay-buffer preservation remain essential for maintaining both plasticity and stability across training conditions.

E PLASTICITY LOSS AND RESETS

Plasticity loss refers to a network's declining ability to learn from new data. A network is said to have lost plasticity when it can no longer optimize its objective as effectively as a freshly initialized counterpart [17]. Prior work has shown that this decline is not always evident from return curves alone, and researchers have

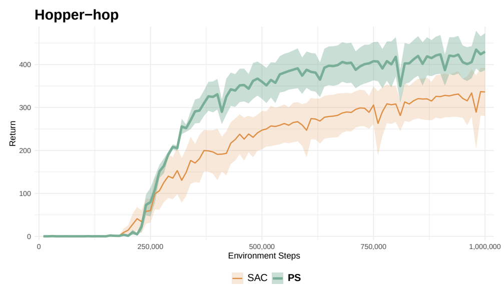


Figure 11: Episodic return over time for SAC and AltNet in the `hopper-hop` environment. While both methods start with similar performance, SAC (purple) exhibits stagnation and eventual decline, indicative of plasticity loss. In contrast, AltNet (pink) maintains and improves performance throughout training, consistent with sustained plasticity.

Figure 12: Evolution of average ℓ_2 weight norm during training. SAC shows steady growth in weight magnitude over time, reflecting accumulation of representational drift and instability. AltNet resets periodically restore weight norms to lower, well-conditioned values, preventing runaway weight growth and preserving learnability.

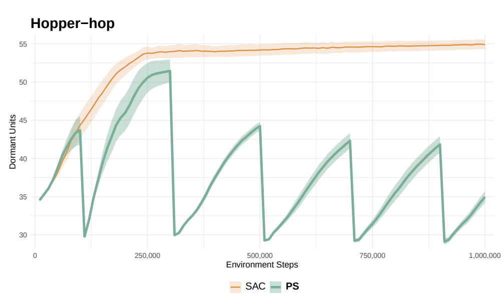


Figure 13: Proportion of dormant (inactive) units over training steps. In SAC, the number of inactive neurons increases steadily, indicating loss of functional capacity in the network. AltNet resets prevent this accumulation by refreshing neuron activations, thereby preserving functional plasticity throughout training.

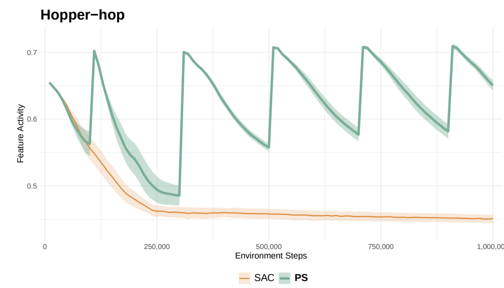


Figure 14: Stable rank of hidden layer activations during training. Stable rank provides a proxy for representational diversity. In SAC, stable rank decreases over time, suggesting a collapse in feature expressivity. AltNet counteracts this trend by reinitializing feature representations, maintaining high diversity across training.

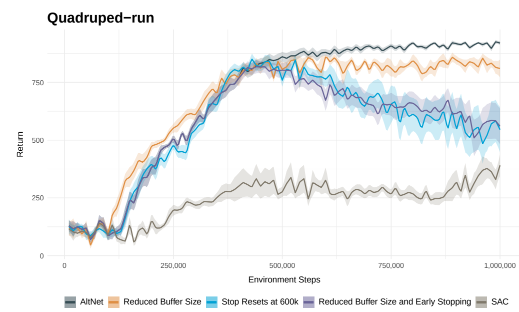


Figure 10: Same buffer–reset ablation experiment as Figure 9(a), conducted at a lower learning rate (0.0002) to test the robustness of AltNet's performance to optimizer settings. We compare standard AltNet (black), reduced buffer size (orange), halted resets (blue), both interventions combined (purple), and the SAC baseline (brown).

therefore developed internal metrics that correlate with plasticity loss, including: increasing weight norms, the emergence of dormant (inactive) units, and a reduction in representational diversity as measured by stable rank or activation sparsity [6, 18, 25]. These metrics serve as diagnostic signals of representational collapse.

In our experiments, we track these same metrics throughout training. Consistent with prior findings, we observe that standard reinforcement learning agents—such as SAC—gradually accumulate internal pathologies: weight magnitudes grow over time (Figure 12), the number of dormant units increases (Figure 13), and representational diversity declines (Figure 14). Together, these trends signal a loss of plasticity, even when external performance initially appears stable. We then examine how resets affect these plasticity-correlated metrics. As shown in Figure 12–Figure 13, each full reset in AltNet restores the network to a state with lower weight norms, fewer dormant units, and higher feature activity. This supports the hypothesis that resets reinitialize the network to a well-conditioned, highly plastic state. Over repeated resets, AltNet avoids the accumulation of representational pathologies seen in single-network baselines, helping to maintain learning capacity over time.