# ALTNET: ALTERNATING NETWORK RESETS FOR PLASTICITY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Modern deep learning systems struggle in continual learning settings due to plasticity loss—the gradual decline in the ability of a neural network learning system to change its output in response to new information. Although periodically resetting a neural network, in whole or in part, has been shown to restore plasticity, it causes performance to temporarily collapse, which can be dangerous in real-world settings. To address this, we propose a simple and general method: AltNet, a reset-based-alternating network approach. AltNet maintains two neural networks that periodically switch roles: one actively interacts with the environment, while the other learns off-policy from the active agent's interactions and a replay buffer. At fixed intervals, the active network is reset, and the passive network—having learned from recent experience—becomes the new active network. This alternating reset-and-role-switch strategy supports continual learning by preserving both plasticity and performance.

## 1 INTRODUCTION

Humans are lifelong learners, continually acquiring new skills while retaining and refining prior knowledge. In contrast, most machine learning systems are optimized for a single task and then learning is stopped. For agents operating in changing environments, learning must be an ongoing process of adaptation rather than the pursuit of a fixed solution (Abel et al., 2023). This perspective motivates the field of continual learning, which aims to develop agents that can learn effectively throughout their operational lifespan. Continual learning faces two core challenges: catastrophic forgetting—the abrupt loss of previously acquired information when learning new data, and plasticity loss—the observed reduction in agent's ability to learn from new experiences.

Continual Learning is a fundamental aspect of Reinforcement Learning (RL). This paper focuses on addressing plasticity loss in RL, while preserving performance and learning progress. In the next section, we begin by examining certain challenges in deep reinforcement learning, including changes in data distributions caused by an agent's policy updates, design decisions related to replay ratio, and early training dynamics—all of which can contribute to or aggravate plasticity loss. Building on insights and observations from this analysis, we introduce AltNet: a novel method designed to mitigate plasticity loss while preserving both performance and learning progress.

### 1.1 NON-STATIONARITY AND ITS IMPACT ON PLASTICITY

Deep learning methods have shown remarkable success in supervised learning, when training from fixed datasets and stationary environments (Hadsell et al., 2020). However, when a neural network is trained on multiple tasks presented sequentially, its ability to learn progressively declines with each additional task (Dohare et al., 2024; Lyle et al., 2022). This highlights a core limitation of many deep learning methods: they are less effective in the face of non-stationary data sources. This gradual decline in learning ability, exacerbated by training on non-stationary data, is known as plasticity loss. In other words, while a freshly initialized network might be trained to solve a new task, a fully or partially trained network may become too rigid to learn from new data (Dohare et al., 2024).

Reinforcement learning (RL) shares many similarities with non-stationary supervised learning settings. RL agents learn by interacting with the environment, gradually improving their policies. As policies evolve, so does the agent's behavior, thus shifting the distribution of experiences it encounters. In addition to the non-stationarity affecting the distribution of inputs experienced by the agent, the learning process itself introduces non-stationarity: many RL algorithms, for instance, use bootstrapping—a process by which an agent learns from its own evolving predictions of expected future rewards. As learning progresses, these predictions—which serve as learning targets—also change, thereby introducing target non-stationarity. Due to the non-stationarity of both inputs and targets, the agent continually adapts to a changing distribution of data, even within a single task, and this non-stationarity can reveal and exacerbate plasticity loss. Empirically, these effects have been observed in deep RL settings where agent's learning

capabilities often become less effective compared to agents with freshly initialized networks under similar conditions (Nikishin et al., 2022).

### 1.2 Replay Ratio and Plasticity Loss

In RL, agents learn directly from interaction with the environment, which is a lengthy and costly process, particularly in real-world domains like robotics. To maximize the information gained from each interaction, practitioners often use experience replay buffers and increase the replay ratio—the number of gradient updates performed after each interaction with the environment (Fedus et al., 2020; Wang et al., 2016). This allows agents to repeatedly learn from past experiences, which has been compared to the brain replaying memories in sleep to consolidate learning (Wilson & McNaughton, 1994).

High replay ratios have been shown to significantly improve sample efficiency (D'Oro et al., 2022). However, when using a high replay ratio, more updates are performed after each new experience, which accelerates plasticity loss (D'Oro et al., 2022), reducing the agent's ability to learn (Nikishin et al., 2022). Moreover, because the agent gathers data incrementally, early experiences are statistically more likely to be replayed than newer ones—an issue amplified by higher replay ratios. This results in *Primacy Bias*, a phenomenon where the agent overfits to early experiences (Nikishin et al., 2022). While increasing the replay ratio improves sample efficiency, it can undermine long-term learning by reducing the agent's adaptability. As a result, there is a point after which further increasing the replay ratio leads to decrease in performance (see Figure 1, black curve).

To address this, Nikishin et al. (2022) propose fully resetting networks to induce plasticity and allow the use of higher replay ratios. However, such resetting causes sudden behavioral change and often leads to a drop in performance. Several authors (Ash & Adams, 2020; Dohare et al., 2021; Kim et al., 2023) have proposed methods that increase plasticity without incurring such dramatic changes in performance during the learning process.

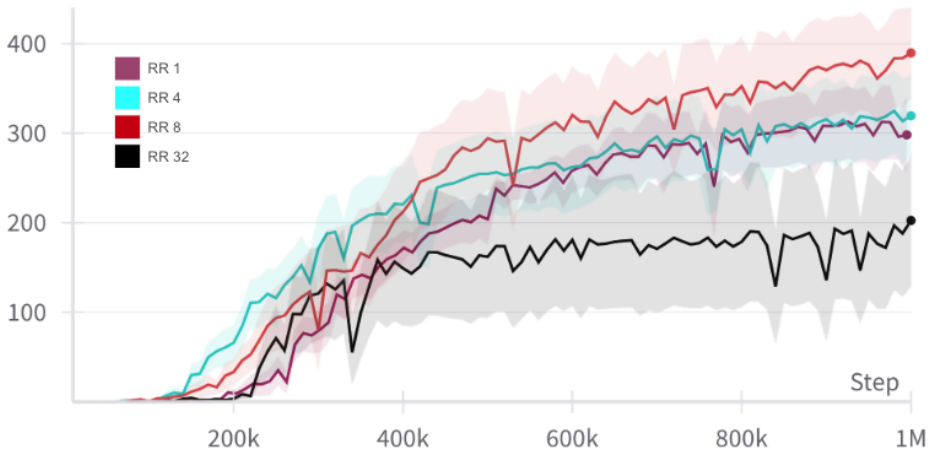



Figure 1: Learning curves of SAC in `hopper-hop` domain in DeepMind Control Suite (DMC) (Tassa et al., 2018) when using various replay ratios. Increasing the replay ratio (RR) improves performance (RR from 1 to 8); performance degrades for higher replay ratios; e.g. RR = 32. Shaded regions represent standard error across 10 seeds.

### 1.3 The Impact of Early Training Dynamics on Long-Term Adaptability

Neural networks are highly sensitive to the data they are trained on shortly after initialization. Achille et al. (2017) showed that a neural network initially trained on a reduced-quality dataset—specifically blurred, low-quality images—struggles to achieve high performance even when later exposed to high-quality images. They concluded that internal representations acquired early in the training are difficult to unlearn and thus constrain future adaptability to new data.

This phenomenon has important consequences in reinforcement learning, where agents are initialized with no prior knowledge and must explore the environment through trial and error. As a result, the data they collect earlier in training is typically generated by a primitive, poor-quality policy (i.e. an unskilled agent) and may limit the potential learning capabilities of the agent later in the training process. While initial experiences shape learning, Kumar et al. (2020)

showed that repeated bootstrapping on updates based on suboptimal value estimates can limit the expressivity of the network and harm long-term performance even if they occur later in training.

## 2 Related Work

Many neural network initialization methods produce highly plastic networks (Glorot & Bengio, 2010; Sutskever et al., 2013). However, as mentioned earlier, when neural networks are trained on a non-stationary objectives they exhibit plasticity loss (Dohare et al., 2021; Lyle et al., 2022). Nikishin et al. (2022) introduced periodic resets (network re-initialization) during training to induce plasticity. However, resetting comes at a cost, as it destroys the information embedded in the trained network, thus leading to sharp performance drops (see Figure 3, green curve). Although leveraging experience replay buffer helps to recover from such information loss by allowing recently-reset networks to rapidly train based on historical data, frequent full resets remain impractical for settings where behavioral stability is critical. This observation reflects a central trade-off in continual learning, known as the plasticity vs. stability dilemma (Nakamura et al., 2000). This phenomenon has also been observed in biological networks (Abraham & Robins, 2005).

Several methods have been proposed to balance plasticity and stability without resorting to full network resets. Shrink and Perturb (Ash & Adams, 2020) interpolates between its current parameters and random values drawn from the initial parameter distribution, and can be interpreted as a form of gradual resetting, increasing plasticity and reducing performance degradation. Continual Backprop (Dohare et al., 2021) and ReDo (Sokar et al., 2023) similarly avoid full resets by selectively resetting the weights of "low-utility" neurons. Although empirically effective, these methods incur substantial computational overhead as they individually track the activations of each neuron. Plasticity Injection (Nikishin et al., 2023) does not track any measure of utility and instead resets the last layers of a network in a specific way ensuring not to change its outputs. However, none of these approaches are capable of fully recovering the plasticity achieved by fully resetting a network's parameters.

Reset Deep Ensembles (RDE) (Kim et al., 2023) mitigate reset-induced performance instability by using an ensemble of networks; resetting each network in turn and allowing other networks to preserve information, while inducing plasticity in the overall ensemble. Actions are selected using Q-value-weighted voting process, where each network's proposed action is weighted by the values assigned by the critic of the oldest network—the network that was last reset—in the ensemble. While this improves performance stability, it also adds complexity by requiring additional hyperparameters. Furthermore even though RDE helps mitigate plasticity loss, it does not fully prevent performance drops (see Figure 3a, orange curve).

## 3 Method and Results

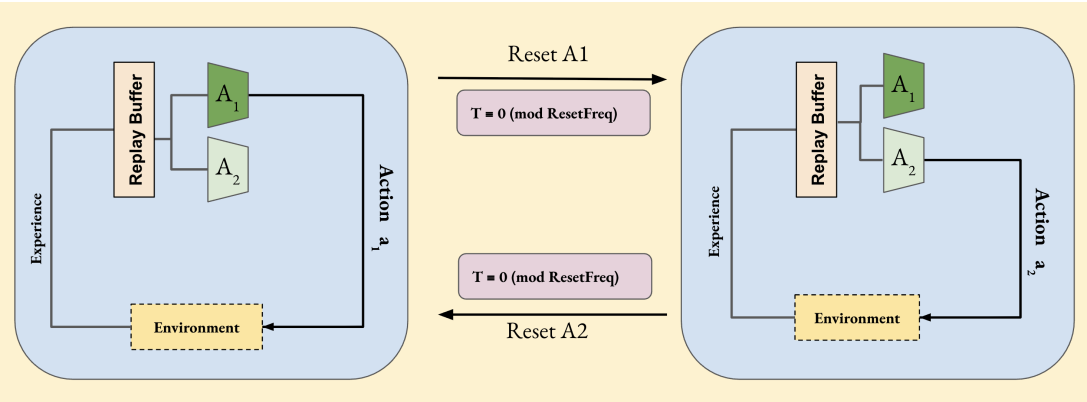



Figure 2: AltNet maintains two networks, $A_1$ and $A_2$, which share a replay buffer and alternate roles over time. Initially, $A_1$ (dark green) is active and collects experience by directly interacting with the environment and updating its parameters based on such experiences and a shared experience replay buffer. During this period $A_2$ (light green) remains passive but undergoes off-policy updates. At every `ResetFreq` steps, the active network is reset and becomes passive, while the previously passive network becomes active and is allowed to directly interact with the environment. This cyclic alternation enables frequent resets to maintain plasticity without sacrificing stability.

AltNet is built on two key insights: that resetting a neural network initializes it to a highly plastic state, from which it may be able to learn a better policy, as compared to a trained network, and that using highly trained networks for

interaction with the environment prevents performance collapse. To combine the benefits of both, AltNet introduces a simple dual-network architecture that allows frequent resets to occur while avoiding performance instability.

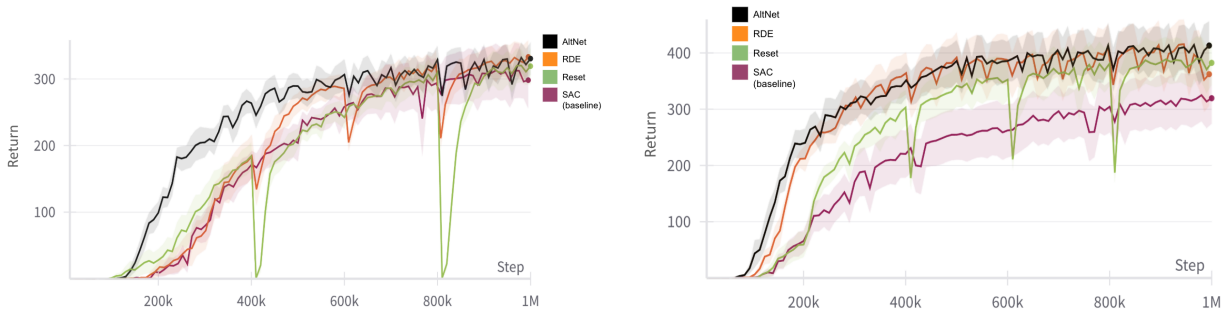
AltNet is composed of two networks that alternate roles at a fixed, predetermined interval, `ResetFreq`. At any given time, the active network interacts with the environment, while the passive network learns off-policy from the experiences of the active network and the replay buffer. Every `ResetFreq` steps in the environment, the active network is reset and becomes the passive network and vice versa. This resetting makes the learning system highly plastic and thus able to incorporate new information, allowing AltNet to maintain plasticity throughout training.

A key idea underlying the AltNet architecture that of *decoupling* recently reset networks from environment interaction. Methods based on full-resetting often suffer sharp drops in performance when resets are applied (see Figure 3, green curve). AltNet mitigates this by making recently-reset networks passive, requiring them to learn before being allowed to act again. When the passive network is promoted to the active role, it has already been trained off-policy and thus acts competently from the outset, ensuring performance stability.

Furthermore, AltNet depends on a *single* hyperparameter, `ResetFreq` which controls network resets and role-switching and avoids additional hyperparameters present in other methods, such as RDE, which in addition to `ResetFreq`, require a hyperparameter to determine how strongly each component of the ensemble influences action selection. Additionally, since only one network acts at a time in AltNet, it does not require complex integration logic and ensemble-based voting, resulting in a simpler and more robust method.

Despite its simplicity, AltNet matches or outperforms the Reset Deep Ensembles, as well as full-reset and Soft Actor Critic (SAC) baselines when evaluated in `hopper-hop` from the DeepMind Control Suite (Tassa et al., 2018). As shown in Figure 3, AltNet (black) learns faster early on, especially at a replay ratio of 1, achieving higher returns more quickly than SAC (purple), Vanilla Reset (green), and RDE (orange).

Our experiments demonstrate that *frequent resetting*, not ensemble diversity, is the primary cause of the performance of multi-network resetting systems. By explicitly decoupling reset networks from direct interaction with the environment, AltNet is able to perform significantly more resetting than other methods. This design leads to faster learning and higher final performance, effectively avoiding the problems with changing data, high replay ratios, and early training dynamics.




(a) Replay Ratio 1

(b) Replay Ratio 4

Figure 3: Performance comparison on the `hopper-hop` domain from the DeepMind Control (DMC) Suite (Tassa et al., 2018). AltNet outperforms state-of-the-art methods like reset (Nikishin et al., 2022) and RDE (Kim et al., 2023) designed to preserve plasticity, as well as a plasticity unaware baseline like SAC. Returns are averaged over 10 random seeds and reported every 10,000 environment steps. Shaded regions show the standard error at each point. AltNet uses a reset frequency of $2 \times 10^5$, while RDE and Reset use $4 \times 10^5$. AltNet benefits from more frequent resets, unlike the other two methods, due to the isolation of reset networks from the environment. Complete hyperparameter tables are provided in Appendix A (Table 1 and Table 2)

## 4 Conclusion and Future Work

We introduce AltNet, a simple and robust reset-based alternating network approach which induces plasticity while maintaining stability. In the future, we plan to extend AltNet to explore more complex domains and real-world environments, develop adaptive reset schedules that respond dynamically to learning and environmental changes, and deepen the theoretical understanding of the interplay between resets, plasticity, and learning dynamics.

## REFERENCES

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:50377–50407, 2023.

Wickliffe C Abraham and Anthony Robins. Memory retention–the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2):73–78, 2005.

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

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

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

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

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

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

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

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

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

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

Hiroshi Nakamura, Akio Ishiguro, and Yoshiki Uchilkawa. Evolutionary construction of behavior arbitration mechanisms based on dynamically-rearranging neural networks. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, volume 1, pp. 158–165. IEEE, 2000.

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

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

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

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

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

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

Matthew A Wilson and Bruce L McNaughton. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679, 1994.

# A Appendix

## A.1 Hyperparameters Used in our Experimental Setup

Table 1: Hyperparameters for RDE and SAC on the `Hopper-Hop` domain.

| **Hyperparameters** | **Value** |
|---|---|
| # of ensemble network | 2 |
| Training steps | $1 \times 10^6$ |
| 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 | $1 \times 10^6$ |
| Updates per step (Replay Ratio) | (1, 4) |
| Target network update period | 1 |
| $\tau$ (Polyak update) | 0.005 |
| Reset Frequency (gradient steps) | $4 \times 10^5$ |
| $\beta$ (action select coefficient) | 50 |

Table 2: Hyperparameters for AltNet and SAC on the `Hopper-Hop` domain.

| **Hyperparameters** | **Value** |
|---|---|
| # of network | 2 |
| Training steps | $1 \times 10^6$ |
| Discount factor | 0.99 |
| Initial collection steps | 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 | $1 \times 10^6$ |
| Updates per step (Replay Ratio) | (1, 4) |
| Target network update period | 1 |
| $\tau$ (Polyak update) | 0.005 |
| Reset Frequency (gradient steps) | see below |
| —Replay ratio 1 | $1 \times 10^5$ |
| —Replay ratio 4 | $2 \times 10^5$ |

Note: We experimented with reset frequencies of $1 \times 10^5$, $2 \times 10^5$, and $4 \times 10^5$ for AltNet and RDE, and report results using the best-performing value. AltNet performs better at higher reset frequencies because the network that is just reset does not interact with the environment before the next reset or role switch occurs. Thus, we want the still-plastic network to become the active network relatively quickly. Moreover, the required reset frequency is sufficient for the passive network to regain stability.