Local Ridge Regression Resets Mitigate Plasticity Loss

Nitin Jain[1]
nitin.jain@student.uni-tuebingen.de

Georg Martius[1]
georg.martius@uni-tuebingen.de

Marin Vlastelica[1,2]
mvlastelica@ethz.ch

Abstract

Plasticity loss refers to a neural network's diminishing ability to learn in non-stationary environments. In Reinforcement Learning (RL), existing plasticity loss mitigation methods like full network resets [8], Plasticity Injection [21], and ReDo [23] offer partial solutions to this problem, but are limited by issues such as catastrophic performance collapse and computational inefficiency. This paper introduces Ridge Regression Reset ($R3$), a novel approach that maintains output stability while restoring plasticity through an optimization framework. Our experiments show that $R3$ effectively mitigates plasticity loss, avoids catastrophic performance collapses, and provides better sample efficiency.

1 Introduction

Continual learning is one of the cornerstones of intelligence – the ability to adapt based on new experiences and re-use old ones [24, 26, 12, 5]. One of the main challenges that are faced in learning systems is that of maintaining plasticity, i.e. retaining the ability to learn [19]. In particular, Deep Reinforcement Learning (DRL) as a framework for tackling sequential decision-making problems tends to suffer from plasticity loss [20, 18] as a result of the non-stationarity of the objective that is being optimized. Despite having tackled complex problems [9, 17, 15, 3], many pathological cases in DRL have been noted where plasticity loss significantly impacts the final performance of the policy [6, 11, 4, 7].

There have been various approaches proposed to mitigate plasticity loss that are based around regularization of the objective [13, 16], full or selective resampling (resetting) of the parameters [20, 8], adding noise to the parameters [2] or introducing additional parameters to facilitate gradient flow [21]. In the case of parameter resampling, these approaches tend to suffer from performance loss to achieve plasticity gain, but are however very scalable, since no additional parameters are necessary and the objective function that is optimized remains unchanged.

In this work we propose a novel method for mitigating plasticity loss, which we name Ridge Regression Reset ($R3$). $R3$ conducts parameter resets by solving a local convex optimization problem, enabling us to control the deviation of the network's predictions from the original ones while potentially shifting to a more plastic part of the loss landscape. The specific convex problem that we choose admits a closed-form solution, with low computational overhead. We validate the viability of our method in several control experiments.

[1]University of Tübingen
[2]ETH Zurich

Accepted at the Intrinsically Motivated Open-ended Learning workshop at NeurIPS 2024.

2 Related Work

Mitigating and Preventing Plasticity Loss. Various approaches attempt to mitigate plasticity loss, each with unique strategies and specific limitations. Shrink and Perturb [2] shrink the network weights and perturb them with noise to rejuvenate the model, albeit with unpredictable effects on performance. Nikishin et al. [20] reset the last few layers at regular intervals, at the cost of temporary performance drops. Further evolving this concept, D'Oro et al. [8], Zhou et al. [27] propose to fully reset the network at regular intervals. While they are very effective in mitigating plasticity loss, they result in a complete loss of performance immediately after the reset. These methods significantly alter the output of the network, while $R3$ preserves the network output as much as possible by design. Orthogonally, Nikishin et al. [21] introduce Plasticity Injection, which adds two additional heads to the network to inject plasticity. This retains the network output, but increases computational costs and lacks an efficient way to merge new networks. Sokar et al. [23] focus on identifying and resetting dormant neurons periodically, yet our experiments indicate it lacks sample efficiency compared to [8] and can potentially inaccurately identify more dormant neurons in shallow layers when using leaky ReLU (Supplementary C). Abbas et al. [1] show that using the CReLU activation function [22] helps mitigate plasticity loss. Dohare et al. [7] show that weight decay helps mitigate plasticity loss to some extent and propose Continual Backprop, a variation of backpropagation for continual learning. Loss regularization techniques for plasticity loss prevention such as Regenerative L2 [13] and Wasserstein regularization [16] have also been proposed.

Identifying and Measuring Plasticity Loss. Identifying and measuring plasticity loss remains a complex challenge, as the underlying factors are not yet fully understood. Recent literature proposes several techniques to address this issue. Nikishin et al. [21] introduce a method to diagnose plasticity loss: if the network's performance improves following plasticity injection, it suggests that the network was suffering from plasticity loss. Meanwhile, Lyle et al. [19] explore the factors influencing plasticity loss, examining the evolution of the loss landscape throughout training and proposes a metric to quantify plasticity loss. While these methods are valuable for understanding and diagnosing plasticity loss, our work with $R3$ focuses solely on mitigation and does not address its identification or measurement.

3 Method

Addressing plasticity loss in RL requires a solution that is effective, practical, and scalable, especially in high replay ratio settings. Through our evaluation of existing approaches, we observe several limitations that hinder their overall effectiveness.

Desiderata 3.1 (Plasticity Loss Mitigation)**.** Considering the limitations of existing approaches, we identify the following four desiderata of a robust solution to plasticity loss: **(1) Scalability:** A robust solution should be scalable, i.e., it can be applied repeatedly without practical limitations. **(2) Output Retention:** The intervention should minimize or eliminate disruptions to the network's output. **(3) Training Efficiency:** The solution must not introduce significant overhead that slows down training. **(4) Sample Efficiency:** The solution should enable the agent to achieve high performance with fewer interactions with the environment.

3.1 Ridge Regression Reset

Current approaches to mitigate plasticity loss fail to satisfy at least one of the criteria we discussed in Desiderata 3.1, thus lacking comprehensiveness. Here, we propose Ridge Regression Reset ($R3$), a novel approach designed to meet all the mentioned desiderata. In Section 4, we compare $R3$ to some of the existing approaches that we implemented.

The key idea of $R3$ is to adjust the network weights in a manner that minimizes deviation from current output while moving the weights closer to a reset matrix. This is achieved by framing the problem as an Ordinary Least Squares (OLS) optimization with an additional term that controls the extent of deviation from a newly initialized weight matrix, we formulate this as Problem 3.1.

Problem 3.1 (Ridge Regression Reset)**.** In the following, we assume that the bias term is subsumed by all weight matrices. We define the new weight matrix $\hat{\mathbf{W}}$ of a layer as the solution to the problem

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{W_c x} - \mathbf{W x}\|^2\right] + \beta\|\mathbf{W_r} - \mathbf{W}\|_F^2. \tag{1}$$

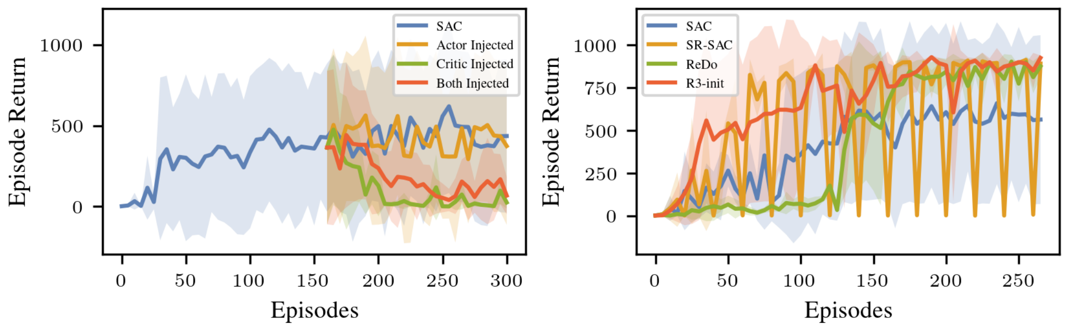

Figure 1: Variations of Plasticity Injection in the hopper-stand environment (RR=16, Left) and comparison of $R3$ using $\mathbf{W_r}$ as the initial weight matrix with baseline, SR-SAC, and ReDo (RR=128, Right) in the hopper-stand environment; (3 runs, mean $\pm$ std).

where $\mathbf{W_c}$ is the current weight matrix, $\mathbf{W_r}$ is a newly initialized weight matrix of same dimensions, $\mathbf{x}$ is the input to the layer drawn from the data distribution (dataset) $\mathcal{D}$, and $\beta > 0$ is a hyperparameter that controls the strength of the penalty for deviation from $\mathbf{W_r}$. Intuitively, the new weight matrix should maintain the same output for the observed data but be as close as possible to the reset matrix.

Proposition 3.1 (Closed-form Solution to Problem 3.1)**.** *Equation* (1) *has the unique closed form solution*

$$\hat{\mathbf{W}} = (\mathbf{W_c}\mathbf{X} + \beta\mathbf{W_r})(\mathbf{X} + \beta\mathbf{I})^{-1}, \tag{2}$$

where $\mathbf{X} = \underset{\mathbf{x}\sim\mathcal{D}}{\mathbb{E}}[\mathbf{x}\mathbf{x}^\top]$.

Proof. A step by step proof is provided in Supplementary E. □

In practice, we scale $\mathbf{W_r}$ by multiplying it by the ratio of the norms of $\mathbf{W_c}$ and $\mathbf{W_r}$. Additionally, a small weight decay term with a coefficient of 10^{-6} is added to handle cases where $\beta = 0$ and $\mathbf{X}$ is degenerate. We apply $R3$ only to the last hidden layer of the agent's networks, as our experiments showed that the first layer does not seem to contribute to plasticity loss.

4 Experiments

We evaluate $R3$ in two experimental settings: (i) a typical RL setting with SAC and mujoco environments and (ii) a controlled setting based on the MNIST dataset [14] with continuously changing labels.

Reinforcement Learning using SAC We conduct our experiments using the Soft Actor-Critic (SAC) algorithm [10] on two environments from the DeepMind Control Suite [25]: hopper-stand and walker-run. We choose SAC for its robustness in continuous action spaces, and these environments due to their complexity, making them suitable testbeds for evaluating plasticity loss. The agent's performance is evaluated every 5 episodes by making the agent act deterministically. We compare results across replay ratios (RR) of 32 and 128. We use ReLU as activation function due to its compatibility with each approach. We perform our experiments on NVIDIA 2080Ti GPUs, with 100,000 environment interactions taking 32 hours at RR=128 and 9 hours at RR=32.

We implement and compare several approaches: SR-SAC [8], which resets the actor and critic networks every 2,560,000 updates; Plasticity Injection [21], applied after 2,56,000 updates in three variations (injecting both actor and critics, only actor, or only critics) at RR=16; ReDo [23], modified to reset every 256,000 updates to align with environment interactions; and our proposed Ridge Regression Reset approach. Note that we do not include Plasticity Injection [21] in the comparisons for higher RRs due to its limitation of not being applicable multiple times. Hyperparameter details for all implementations are provided in Supplementary A for reproducibility.

Continual MNIST To study plasticity loss in a controlled setting, we construct a sequential learning problem using the MNIST dataset. We introduce non-stationarity in the training process through label permutation at fixed intervals of 40 epochs, with each permutation constituting a distinct

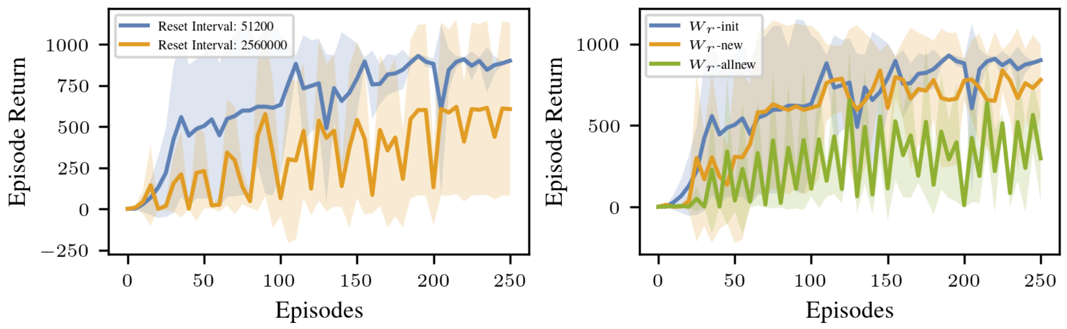

Figure 2: $R3$ on hopper-stand (RR=128) with different reset intervals (network updates, left) and with different choices of $\mathbf{W_r}$(right). (3 seeds, mean $\pm$ std)

task. The model architecture consists of two convolutional layers, followed by two fully connected layers, and incorporates max pooling and dropout. Training proceeds sequentially across tasks, with performance evaluation on the permuted test set after each task completion. We compare $R3$ against a baseline without intervention, applying our method at the beginning of each task (except the first) with $\beta \in \{1.0, 10.0\}$ across 5 seeds.

4.1 Results

Reinforcement Learning

Full network resets in SR-SAC mitigate plasticity loss, especially at high replay ratios where the issue is pronounced. Regular application ensures scalability without significant overhead, maintaining training efficiency. These resets are sample-efficient, enabling swift optimal performance under high replay ratios. However, a significant drawback is the complete performance collapse immediately following each reset, indicating a failure to retain output. We find that Plasticity Injection is effective in SAC only when it is performed on the actor network, while the critics remain unchanged. The agent's performance drops over time in cases where critics are injected while in the case where only the actor is injected with plasticity, the agent is able to maintain performance (Figure 1, left). We hypothesize that this is due to a drastic change in the loss landscape of the critics after injection, which the actor network, even with increased plasticity, cannot compensate for. Despite being excellent at output retention, it is not scalable due to its demand for additional memory every time it is performed. It also slows down training significantly as the gradients have to pass through additional networks. ReDo is able to perform well and retains output effectively by resetting only dormant neurons (Figure 1, right). However, it requires significantly more time to converge. ReDo is scalable, as it is performed at regular intervals, and it does not add significant overhead to the training process.

Figure 1 demonstrates the performance of $R3$ in mitigating plasticity loss compared to other approaches for hopper-stand at RR=128. We provide additional results for hopper-stand (RR=32), and walker-run (RR=32, 128) in Supplementary D. Although $R3$ is slightly less sample-efficient than SR-SAC, we argue that its superior output retention compared to SR-SAC and better performance compared to other methods makes it a more robust solution. $R3$ avoids catastrophic performance collapse and retains output by effectively limiting performance loss, particularly for small values of β. This advantage enables $R3$ to be applied at a much higher frequency than full network resets. Our experiments support this hypothesis, showing that more frequent resets lead to better performance (Figure 2).

We also observe that selecting a new $\mathbf{W_r}$ each time $R3$ is performed results in significant drops in performance. We hypothesize that this may occur because the network is forced to adjust in a different random direction too frequently, hindering its ability to learn. To investigate the impact of $\mathbf{W_r}$ selection, we compare it with scenarios where $\mathbf{W_r}$ is either the initial state of the model ($R3$-init) or a newly sampled random matrix ($R3$-new). We find that this choice varies between environments. While $R3$-init performs better for hopper-stand, $R3$-new performs better for walker-run (Figure 12).

Additionally, $R3$ is not able to reach the optimal performance for walker-run unlike SR-SAC (Figure 12). However, it maintains its advantage in sample efficiency over ReDo. We hypothesize that this is because the optimal value of $\beta = 1$ and reset interval doesn't translate well among environments.

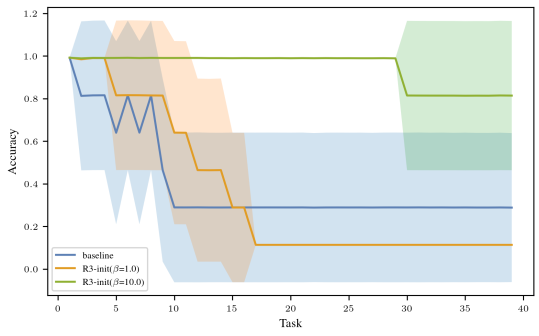

Figure 3: Performance comparison on Continual MNIST between baseline (no intervention) and $R3$ with $\beta \in \{1.0, 10.0\}$. Applying $R3$ mitigates performance degradation over time, with higher β values preserving performance longer. (5 seeds, mean $\pm$ std)

Continual MNIST

The controlled MNIST setting provides clear evidence of $R3$'s ability to mitigate plasticity loss. Figure 3 depicts the effectiveness of $R3$ in mitigating plasticity loss. While the performance begins degrading immediately without intervention, applying $R3$ is able to help in retaining performance for longer. Additionally, a higher value of β preserves performance for longer. This reinforces our hypothesis from the RL setting regarding the sensitivity of $R3$ to the choice of β. While $R3$ helps in retaining plasticity, the optimal value of β appears to be task-dependent, supporting our conclusion about developing an adaptive mechanism for β selection. A detailed analysis of network gradients in this setting (Supplementary B) confirms previous findings that plasticity loss is primarily concentrated in the deeper layers of the network.

5 Conclusion and Future Work

In this paper, we motivate the need for a robust solution to mitigate plasticity loss in Deep Reinforcement Learning and propose Ridge Regression Reset, a novel approach which resets the network weights by efficiently solving a local convex problem, specifically designed to satisfy Desiderata 3.1. We validated our proposed approach with several experiments, comparing to existing plasticity loss mitigation techniques. Results indicate that $R3$ successfully balances performance drop and network plasticity, without the cost of additional parameters.

The main limitation of $R3$ is its sensitivity to the choice of β, as evident from its performance with different β values in Figure 3, and high variation in Figure 1. We believe that an improved version with an adaptive β will alleviate this drawback. Additionally, the choice of reset interval is also non-trivial and can vary among environments based on the sparsity of rewards. Future work on understanding the reasons behind its effectiveness can provide significant insights towards understanding plasticity loss.

6 Acknowledgments

We acknowledge the support from the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039B). Georg Martius is a member of the Machine Learning Cluster of Excellence, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645. This work was supported by the ERC - 101045454 REAL-RL

References

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

[2] Jordan Ash and Ryan P Adams. On warm-starting neural network training. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3884–3894. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/288cd2567953f06e460a33951f55daaf-Paper.pdf.

[3] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[4] Tudor Berariu, Wojciech Czarnecki, Soham De, Jörg Bornschein, Samuel L. Smith, Razvan Pascanu, and Claudia Clopath. A study on the plasticity of neural networks. *CoRR*, abs/2106.00042, 2021. URL https://arxiv.org/abs/2106.00042.

[5] Glen Berseth, Zhiwei Zhang, Grace Zhang, Chelsea Finn, and Sergey Levine. CoMPS: Continual meta policy search. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PVJ6j87gOHz.

[6] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018.

[7] Shibhansh Dohare, Richard S. Sutton, and A. Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness, 2022. URL https://openreview.net/forum?id=86sEVRfeGYS.

[8] 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 *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OpC-9aBBVJe.

[9] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018. ISSN 1935-8237. doi: 10.1561/2200000071. URL http://dx.doi.org/10.1561/2200000071.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://openreview.net/forum?id=HJjvxl-Cb.

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

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proc. of the national academy of sciences*, 2017. URL https://www.pnas.org/content/pnas/114/13/3521.full.pdf.

[13] Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. 2023.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[15] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[16] Alex Lewandowski, Haruto Tanaka, Dale Schuurmans, and Marlos C. Machado. Directions of curvature as an explanation for loss of plasticity, 2024. URL https://arxiv.org/abs/2312.00246.

[17] Long-Ji Lin. *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

[18] Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZkC8wKoLbQ7.

[19] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[20] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron C. Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, 2022. URL https://api.semanticscholar.org/CorpusID:248811264.

[21] Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and Andre Barreto. Deep reinforcement learning with plasticity injection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jucDLW6G9l.

[22] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning*, pages 2217–2225. PMLR, 2016.

[23] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[24] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, 8, 1995.

[25] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

[26] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[27] Hattie Zhou, Ankit Vani, Hugo Larochelle, and Aaron Courville. Fortuitous forgetting in connectionist networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ei3SY1_zYsE.

A Hyperparameter Details

This section of the appendix contains tables with hyperparameter values for all of our implementations for the purposes of reproduction.

Table 1: Common hyperparameters for SAC

Parameter	Value
Initial Collect Steps	10^5
Discount Factor	0.99
Minibatch size	256
Optimizer(All)	Adam
Optimizer(All): learning rate	0.0003
Optimizer(All): β_1	0.9
Optimizer(All): β_2	0.999
Optimizer(All): eps	0.00015
Networks(All): activation	ReLU
Networks(All): hidden units	256
Initial Temperature	1
Replay Buffer Size	10^6
Target network update period	1
τ (EMA coefficient)	0.995

Table 2: Hyperparameters for Ridge Regression Reset

Parameter	Value
Network(All): n. hidden layers	2
Resetting Period(update steps)	51200
β	1.0
Minibatch size for calculating $\mathbf{X}$	2048
Updates per Step (Replay Ratio)	128

Table 3: Hyperparameters for ReDo

Parameter	Value
Network(All): n. hidden layers	2
Recycling Period(update steps)	256000
τ-Dormant	0
Minibatch size for estimating scores	256
Updates per Step (Replay Ratio)	128

Table 4: Hyperparameters for SR-SAC

Parameter	Value
Network(All): n. hidden layers	2
Reset Intervals(gradient steps)	2560000
Layers getting hard reset	All
Updates per Step (Replay Ratio)	variable(up to 128)

Table 5: Hyperparameters for Plasticity Injection

Parameter	Value
Network(All): n. hidden layers	3
N. Hidden Layers(encoder)	2
N. Hidden Layers(heads)	1
Updates per Step (Replay Ratio)	16

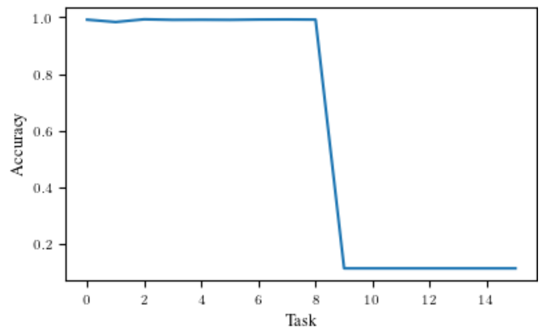

Figure 4: MNIST performance on continuously evolving tasks

B Analysis of Gradient Flow in Continual MNIST

To investigate the mechanisms of plasticity loss, we analyze gradient flow in our Continual MNIST setting by comparing two model checkpoints: one where plasticity loss had occurred (non-plastic) and one where it had not (plastic). Figure 4 shows the performance of the run under investigation across tasks, where we observe that after a certain number of tasks, the network loses its ability to learn. Without updating the weights, we pass the entire dataset through each checkpoint, collecting the following quantities per layer for each batch: (i) norm of the activations, (ii) norm of the gradients, and (iii) norm of the gradients with respect to activations.

The density estimation curves for these quantities are shown in Figure 5, Figure 6, and Figure 7 respectively. While the gradients with respect to activations remain non-zero in the deeper layers, the weight gradients approach zero. This prevents gradient flow to the initial layers, confirming observations from previous works, such as [20], that plasticity loss is concentrated in deeper network layers.

C Limitations of ReDo

Sokar et al. [23] propose a score function for each neuron to identify dormant neurons. In our experiments, this function works well with the ReLU activation function and a threshold of zero. However, with a threshold of zero, one could simply count neurons with zero activations, rendering the score somewhat redundant. To understand more, we take a checkpoint from a training run using leaky ReLU as the activation function and plot the distribution of (i) average activations for each neuron of a minibatch (Figure 8), and (ii) corresponding score values (Figure 9) according to the

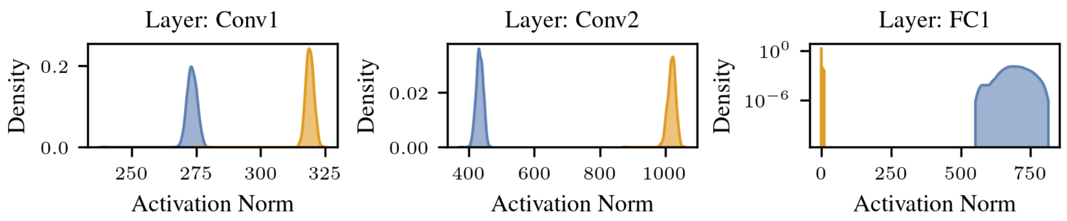

Figure 5: Density plot of Activation Norms for a plastic (blue) and non-plastic (orange) checkpoint of MNIST

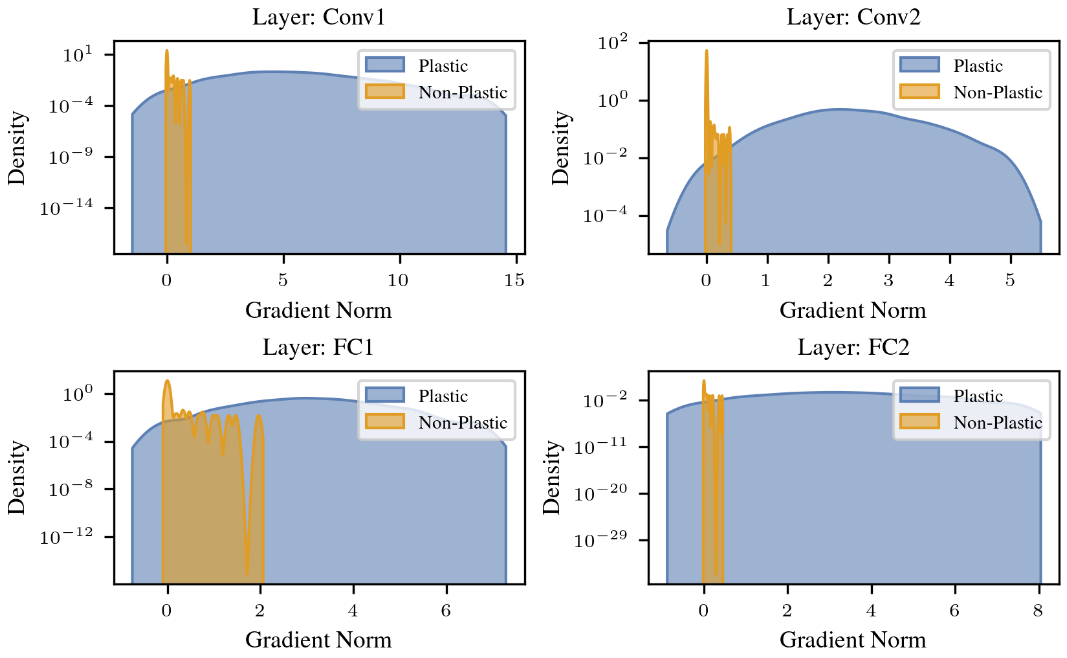

Figure 6: Density Plot of Gradient Norms

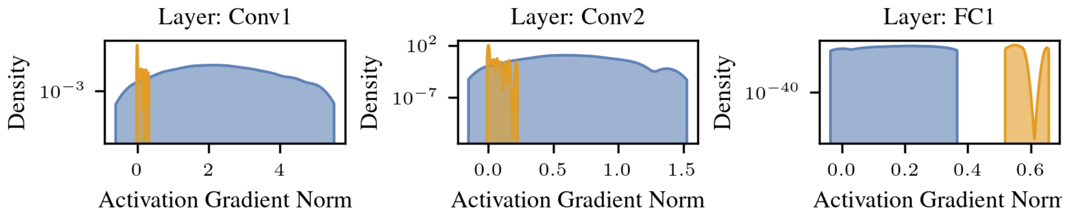

Figure 7: Density plot of norms of Gradients with respect to Activation for a plastic(blue) and non-plastic (orange) checkpoint of MNIST

paper. We observe that any meaningful threshold value here, and in most other checkpoints indicates a higher fraction of dormant neurons in the first hidden layer compared to the second. This observation directly contradicts the findings of other studies, as well as our own results from Supplementary B, which show that plasticity loss is concentrated in the deeper layers of the network.

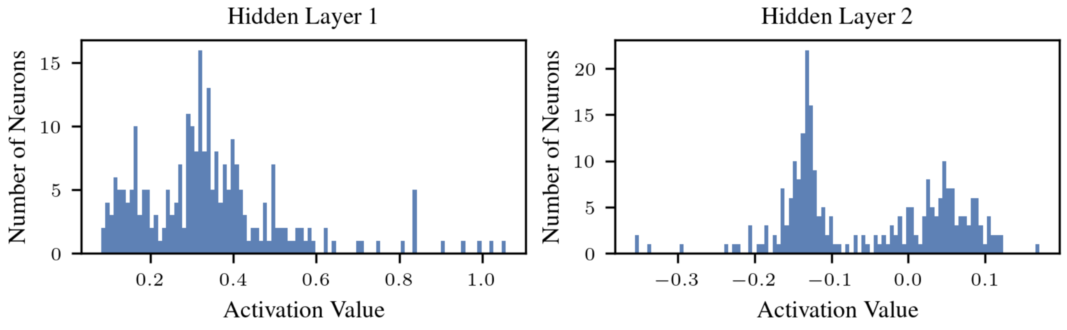

Figure 8: Distribution of mean activations of a minibatch trained on hopper-stand at RR=128 and activation function leaky ReLU.

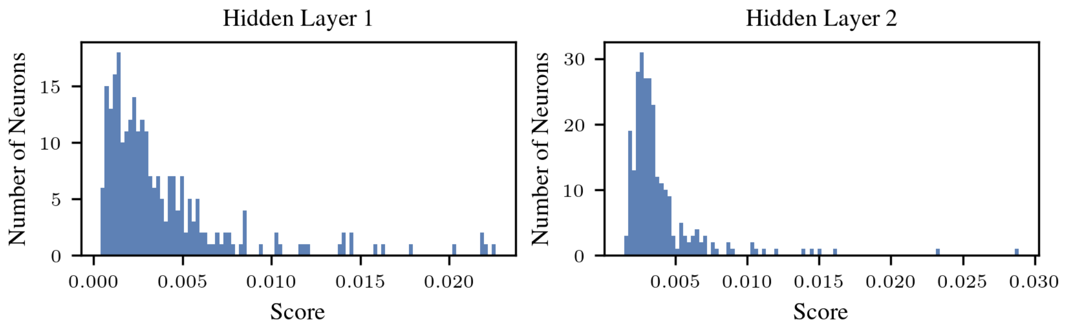

Figure 9: Distribution of score values of a minibatch trained on hopper-stand at RR=128 and activation function leaky ReLU.

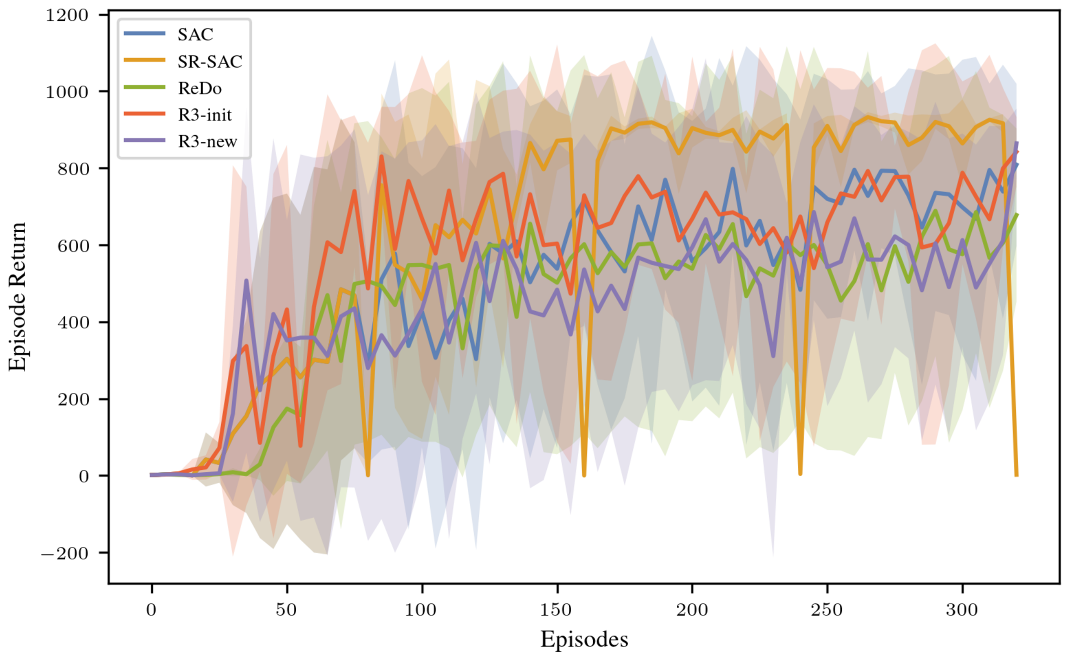

Figure 10: Comparison of 2 variations of $R3$ ($R3$-init and $R3$-new) based on the choice of $\mathbf{W_r}$ with SAC, SR-SAC, and ReDo (RR=32) in the hopper-stand environment (3 runs $\pm$ std).

D Additional Results

In this section we provide additional results comparing Ridge Regression Reset with SAC, SR-SAC, and ReDo. We show the performances of 2 variations of $R3$ based on the choice of $\mathbf{W_r}$: $R3$-init where $\mathbf{W_r}$ is chosen to be the initial weights of the agent, and $R3$-new where $\mathbf{W_R}$ is chosen to be a newly initialized set of weights, kept the same throughout for the whole training runs in both cases. Figure 10 shows this comparison for the task hopper-stand for RR=32. Figure 11 and Figure 12 show this comparison for the task walker-run for RR=32 and RR=128 respectively.

E Proof of Closed form Solution to Ridge Regression Reset

Proposition E.1 (Closed-form Solution to Problem 3.1)**.** *Equation (1) has the unique closed form solution*

$$\hat{\mathbf{W}} = (\mathbf{W_c X} + \beta \mathbf{W_r})(\mathbf{X} + \beta \mathbf{I})^{-1}, \tag{3}$$

where $\mathbf{X} = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}\mathbf{x}^\top]$.

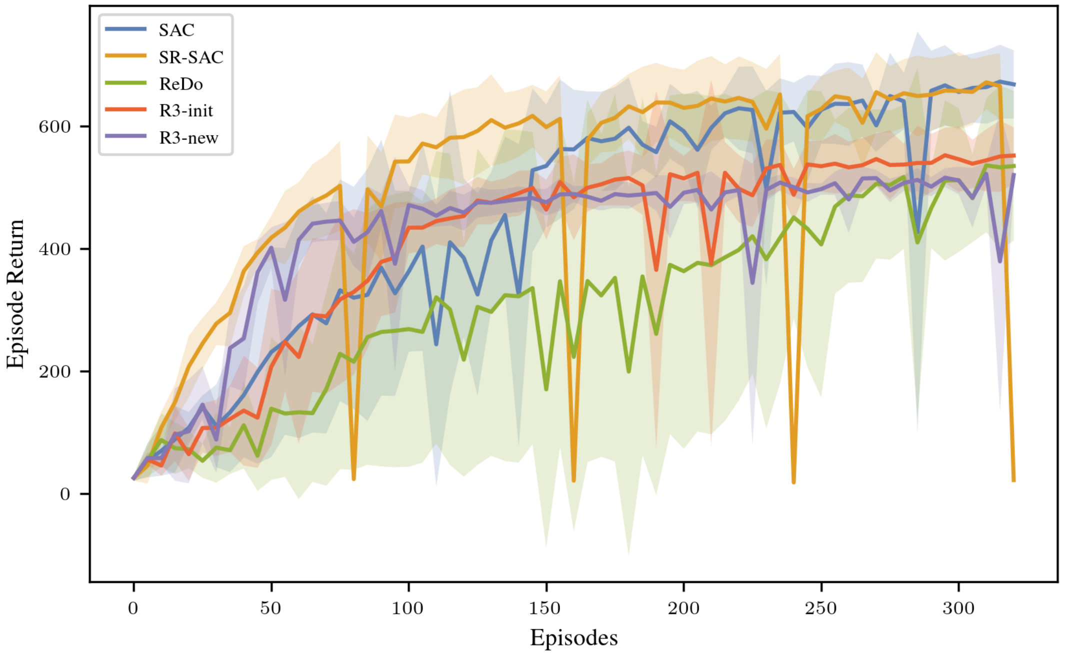

Figure 11: Comparison of 2 variations of $R3$ ($R3$-init and $R3$-new) based on the choice of $\mathbf{W_r}$ with SAC, SR-SAC, and ReDo (RR=32) in the walker-run environment (3 runs ± std).

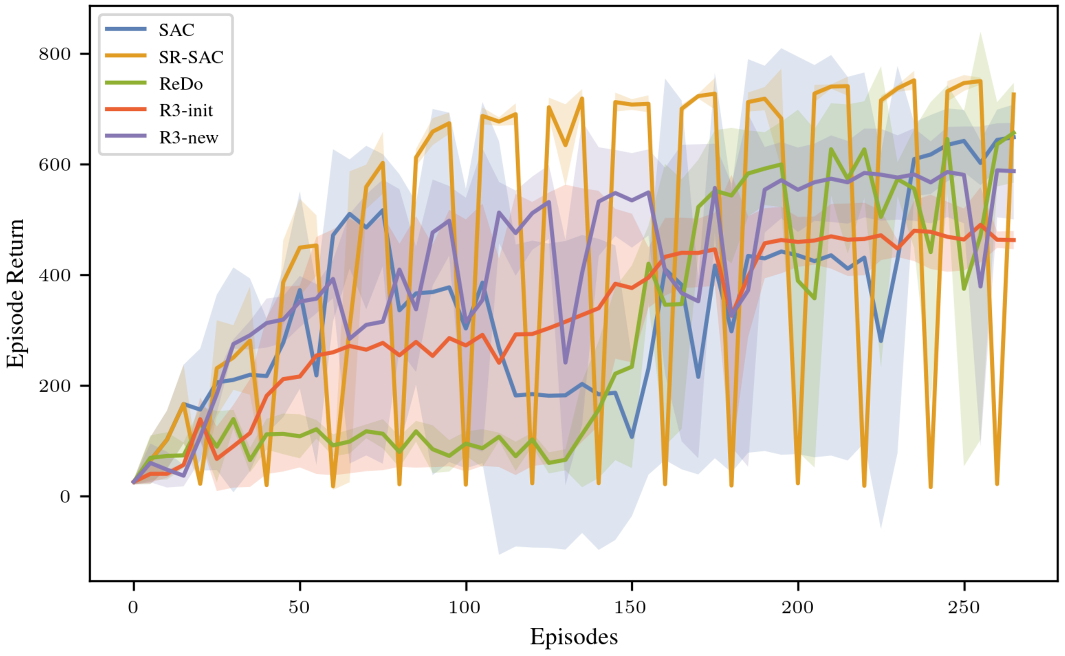

Figure 12: Comparison of 2 variations of $R3$ ($R3$-init and $R3$-new) based on the choice of $\mathbf{W_r}$ with SAC, SR-SAC, and ReDo (RR=128) in the walker-run environment (3 runs ± std).

Proof. **Convexity of the Objective Function**

The objective function can be expressed as:

$$\mathcal{L}(\mathbf{W}) = \hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{W_c x} - \mathbf{Wx}\|^2\right] + \beta\|\mathbf{W_r} - \mathbf{W}\|_F^2.$$

Expanding the first term:

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{W_c x} - \mathbf{Wx}\|^2\right] &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[(\mathbf{W_c x} - \mathbf{Wx})^T(\mathbf{W_c x} - \mathbf{Wx})\right],\\ &= \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[\mathbf{x^T}(\mathbf{W_c^T W_c} - \mathbf{W_c^T W} - \mathbf{W^T W_c} + \mathbf{W^T W})\mathbf{x}\right].\end{aligned}$$

To simplify further, we introduce the trace operator. Using $tr(a) = a$ and $tr(a) = tr(a^T)$ for any scalar quantity a, we get:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{W_c x} - \mathbf{Wx}\|^2\right] = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[tr(\mathbf{x^T}(\mathbf{W_c^T W_c} - 2\mathbf{W_c^T W} + \mathbf{W^T W})\mathbf{x})\right].$$

Using the linearity of expectation and the cyclic property of trace, we can further simplify to:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{W_c x} - \mathbf{Wx}\|^2\right] = tr(\mathbf{W^T W X}) - 2tr(\mathbf{W_c^T W X}) + const,$$

where $\mathbf{X} = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{xx^T}]$.

Since $\mathbf{X}$ is positive semi-definite, $tr(\mathbf{W^T W X})$ is convex in $\mathbf{W}$. The second term of $\mathcal{L}(\mathbf{W})$), $\beta\|\mathbf{W_r} - \mathbf{W}\|_F^2$, is also convex for $\beta > 0$ since it is a quadratic function in $\mathbf{W}$.

Since the sum of convex functions is convex, $\mathcal{L}(\mathbf{W})$ is convex, implying that any local minimizer is also a global minimizer.

Compute the gradient and solve for stationary points

To find the minimizer, we compute the gradient of $\mathcal{L}(\mathbf{W})$ with respect to $\mathbf{W}$:

$$\nabla_{\mathbf{W}}\, \mathcal{L}(\mathbf{W}) = 2\mathbf{WX} - 2\mathbf{W_c X} + 2\beta\mathbf{W} - 2\beta\mathbf{W_r}.$$

Setting the gradient equal to zero to find the stationary points:

$$\mathbf{W}(\mathbf{X} + \beta\mathbf{I}) = \mathbf{W_c} + \beta\mathbf{W_r}$$

Thus, we get the solution:

$$\hat{\mathbf{W}} = (\mathbf{W_c X} + \beta\mathbf{W_r})(\mathbf{X} + \beta\mathbf{I})^{-1}$$

Uniqueness of the solution

Given that $\mathbf{X} + \beta\mathbf{I}$ is a positive definite matrix (since $\mathbf{X}$ is positive semi-definite and $\beta > 0$), the inverse $(\mathbf{X} + \beta\mathbf{I})^{-1}$ exists and is unique. This confirms that the solution $\hat{\mathbf{W}}$ is unique.

Since $\mathcal{L}(\mathbf{W})$is convex and the stationary point is unique, $\hat{\mathbf{W}}$ is the unique global minimum. □