A Study of Plasticity Loss in On-Policy Deep Reinforcement Learning

Arthur Juliani **Jordan T. Ash**
Microsoft Research NYC
{ajuliani, ash.jordan}@microsoft.com

Abstract

Continual learning with deep neural networks presents challenges distinct from both the fixed-dataset and convex continual learning regimes. One such challenge is *plasticity loss*, wherein a neural network trained in an online fashion displays a degraded ability to fit new tasks. This problem has been extensively studied in both supervised learning and off-policy reinforcement learning (RL), where a number of remedies have been proposed. Still, plasticity loss has received less attention in the on-policy deep RL setting. Here we perform an extensive set of experiments examining plasticity loss and a variety of mitigation methods in on-policy deep RL. We demonstrate that plasticity loss is pervasive under domain shift in this regime, and that a number of methods developed to resolve it in other settings fail, sometimes even performing worse than applying no intervention at all. In contrast, we find that a class of "regenerative" methods are able to consistently mitigate plasticity loss in a variety of contexts, including in gridworld tasks and more challenging environments like Montezuma's Revenge and ProcGen.

1 Introduction

In many important machine learning domains, such as continual learning and online reinforcement learning (RL), training data are not wholly available simultaneously. Rather, in these settings, data arrives sequentially over a long period of time. Irrespective of the available quantity of training data, we would typically like to have the highest performing model possible at any given point. Unfortunately, these sequential learning scenarios are known to present optimization issues for neural networks. Specifically, if a model is naively updated to convergence each time new data arrives, and the amount of training data effectively increases, the resulting model generalizes worse than it would under more standard, non-sequential constraints. This phenomenon has been introduced by Ash and Adams (2020) as the "warm-start problem," where it was found that randomly initialized models generalize well but are expensive to fit, and warm-started models generalize poorly but converge far more quickly.

An adjacent problem to warm-starting is that of *plasticity loss* (Lyle et al., 2023), which presents itself both in the supervised learning and reinforcement learning settings. Like in the warm-start problem, plasticity loss describes a degradation in a model's ability to fit new data as training progresses, ultimately harming both sample efficiency and asymptotic performance. If an agent achieves worse performance when fitting first a source distribution and then a distinct target distribution than it would if instead trained on the target distribution in isolation, the degradation is attributed to plasticity loss.

Whereas the warm-start problem is restricted to test performance, plasticity loss is generally measured as a degradation in performance on training data, often under some form of distribution shift. Despite the typical focus on training data, an ideal solution to plasticity loss would be able to prevent performance degradation both for data seen and unseen when fitting the policy. Importantly, the plasticity loss phenomenon is distinct from overfitting, as resolving plasticity loss should improve both training performance and generalization. In contrast, overfitting describes the opposite, where improving training performance has deleterious effects on generalization.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

This work studies the plasticity loss phenomenon in detail for the on-policy reinforcement learning setting. We introduce three distinct kinds of distribution shift to facilitate our analysis as well as a variety of environments and tasks. While several interventions have already been proposed, we demonstrate that some of the methods which proved successful for addressing plasticity loss in the supervised learning or off-policy reinforcement learning setting fail in the on-policy setting or under our proposed forms of distribution shift. We further highlight several criteria that we believe are necessary to remedy the issue. Based on these insights, we describe several techniques that resolve plasticity loss on the environments we consider.

In summary, the contributions of this paper are:

- We extend studies of plasticity loss and the warm-start problem to the on-policy regime, finding that they present a persistent issue across a variety of model architectures and environmental distribution shift conditions.
- We provide an in-depth analysis of the correlates of these pathologies, studying several types of environmental settings, model architectures, and previously proposed approaches for mitigation in settings related to on-policy reinforcement learning. Importantly, we include generalization trends in our consideration of these phenomena.
- We characterize properties of methods that seem necessary for interventions to be successful at both addressing plasticity loss and ensuring generalization performance is maintained, and provide recommendations along these lines.

2 Preliminaries

An RL algorithm is described as "on policy" if it is trained using data collected from its own policy. While generally more sample inefficient than off-policy alternatives, where the data collection mechanism deviates from the policy being fit, on-policy methods have become more commonly used in practical RL problems because of their simplicity and stability (Andrychowicz et al., 2020).

Proximal Policy Optimization (PPO) is a ubiquitous on-policy deep RL algorithm (Schulman et al., 2017). As a trust-region method, PPO constrains the extent by which a weight update can modify the current policy, quelling instability issues caused by high-variance policy gradients. Despite this robustness, on-policy algorithms still update on data corresponding only to the agent's current experience, which may be sub-optimal for capturing necessary information around solving the underlying MDP. There is some evidence that plasticity loss can be present in on-policy RL learning problems (Dohare et al., 2023b), demonstrating the effect in a somewhat ad hoc manner, we systematically use multiple environments and distribution shift conditions to more thoroughly characterize plasticity loss—and how to resolve it—in the on-policy setting.

2.1 Simulating environmental distribution shift

This paper primarily studies three distinct forms of distribution shift that we can reasonably expect a plasticity-preserving intervention to mitigate. Each experiment is organized into r distinct rounds. In each round we supply k environments from some distribution to the agent, fit a policy for that set of environments, and measure performance at the end of the round. To induce distribution shift, between each round we modify the data according to one of the three strategies described below. The environmental modifications we consider are:

Permute. Input pixels are randomly shuffled from round to round. Each of the k environments in a given round are shuffled in the same way, but distinct from the shuffling in any other round. This method was utilized in Dohare et al. (2023b). We expect that an optimal learning algorithm will be able to achieve equivalent performance on the initial and subsequent rounds.

Window. We supply the agent with k new environments for the same task. Here previously seen environments are no longer accessible, so the agent can only optimize with respect to the current k. This approach was considered by

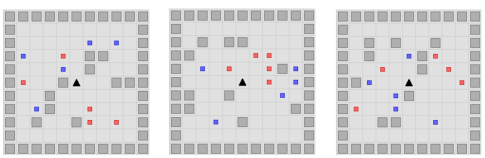

Figure 1: Examples of gridworld environment tasks. The agent (black triangle) begins each episode in the center of the environment. Blue jewels provide *+1* reward, red jewels provide *-1* reward, and dark grey walls prevent movement. Objects are placed randomly.

Abbas et al. (2023) and Lyle et al. (2023). We expect that an optimal learning algorithm will be able to achieve equivalent or better performance on subsequent rounds as compared to the initial round.

Expand. Like in the window modification, k new environments are supplied to the agent. Here, however, they are appended to the total amount of training data, such that the number of available environments in the final round of training is $k \times n$. This is most similar to the warm-start environment studied primarily in Ash and Adams (2020), and revisited in Igl et al. (2020). In this setting we expect training performance to decrease each round, even for an optimal learning algorithm, as available training data grows and becomes more difficult to fit. Correspondingly, we expect the generalization performance to increase as the training distribution expands to better capture the true distribution.

In the case of the latter two modifications, data at each new round are drawn from the exact same distribution as environments from previous rounds. As we demonstrate in Section 4.1, these still induce significant performance depredation in warm-started models. This observation is surprising—we often think of a model's initialization as a sort of "prior" over learned functions, and here we demonstrate that even an initialization obtained by training on data distributed identically to the current round still hinders performance in on-policy learning.

We use a simple gridworld task as our main sandbox for studying plasticity loss. The environment is drawn from the NeuroNav library (Juliani et al., 2022), and the goal of the agent is to collect rewarding blue jewels while avoiding punishing red jewels in a fixed time window (100 time-steps per episode). Each sample of the environment from the distribution of possible tasks changes both the location of the jewels and the walls of the maze. Figure 1 shows environment modification examples.

3 Existing Approaches

A number of methods have been proposed to address plasticity loss in deep neural networks, though none were designed with on-policy RL explicitly in mind. Instead, these methods have been primarily validated in the context of either supervised learning or off-policy RL. Each of these techniques can be divided into one of two groups: Interventions which are performed intermittently during training, and those which are applied continuously, either as part of the model architecture or as part of the model update process. For additional implementation details see Appendix Section A.

3.1 Intermittent Interventions

We consider an intervention to be *intermittent* if it is applied only at specific points during training. Periodically, the interventions in this category are applied, and training proceeds normally otherwise. Most of these are applied each time there is a training distribution change, implying that there must be awareness of when this occurs; in practice this information may be unavailable and difficult to detect.

Resetting final layer. Proposed by Nikishin et al., this method involves periodically replacing the weights of the final layer in the network with newly initialized values. It was demonstrated that this alleviates plasticity loss in certain off-policy RL settings.

Shrink+Perturb. This technique periodically scales the magnitude of all weights in the network by a factor and then adds a small amount of noise (Ash and Adams, 2020). It was demonstrated that this improves performance in batched continual learning settings. In our implementation the weight of these two factors are entangled such that they sum to one.

Plasticity Injection. Plasticity Injection replaces the final layer of a network with a new function that is a sum of the final layer's output and the output of a newly initialized layer subtracted by itself (Nikishin et al., 2023). The gradient is then blocked in both the original layer and the subtracted new layer. It was shown that this method increases performance of off-policy RL agents in the ALE.

ReDo. This technique resets individual neurons within the network based on a dormancy criteria at fixed intervals (Sokar et al., 2023), leading to improvement in performance in the context of off-policy RL agents trained on the ALE. Following Sokar et al., we reset the model parameters more frequently than we apply environmental distribution shifts.

3.2 Continuous Interventions

We characterize a method as *continuous* if it is applied at every step of optimization. Continuous interventions are desirable because they do not require an awareness of when a distribution shift has occurred, which can be challenging in practical scenarios where we want to mitigate plasticity loss.

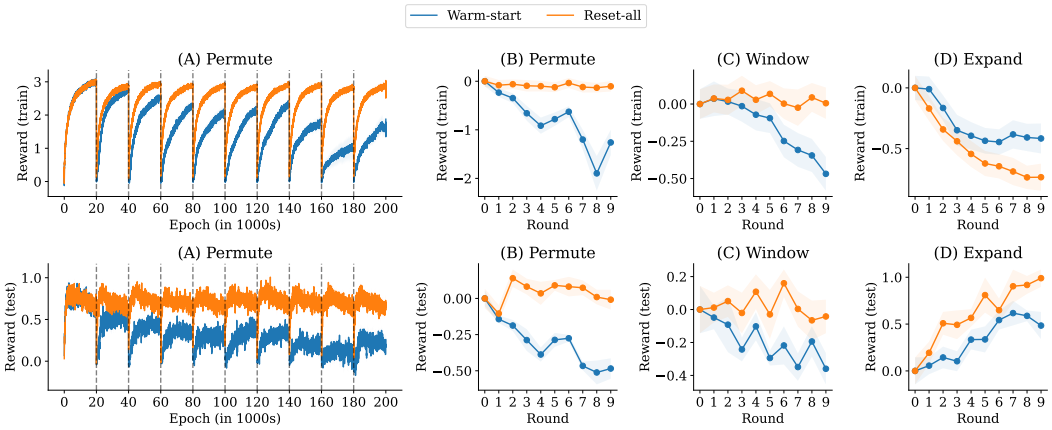

Figure 2: Performance in the gridworld environment under each of the three distribution shift conditions. The degradation between rounds is evidence of plasticity loss. (A): Epoch-level training performance for permute modification. Dotted vertical lines indicate the end of each round, before a new environmental distribution shift is applied. (B): Round-level training performance for the Permute modification. Data points correspond to normalized mean reward in final 50 episodes of round. Shaded regions correspond to standard error. (C, D): Round-level training performance for the Window and Expand conditions. **Top row**: Training performance. **Bottom row**: Test performance.

L2 Norm. Discussed by Lyle et al., this method involves regularizing the network using the L2 norm. It was demonstrated that in certain continual learning settings this reduces plasticity loss.

LayerNorm. LayerNorm (Ba et al., 2016) is a ubiquitous deep learning regularization technique, and was shown useful for mitigating plasticity loss by Lyle et al.. The authors demonstrated a performance improvement in the ALE when using off-policy RL.

CReLU activations. Proposed by Abbas et al. and studied further by Lee et al., this method involves replacing ReLU activations with the CReLU activation function. This ensures that the gradient is non-zero for all units in a given layer, which appears to mitigate plasticity loss when periodically switching between tasks in the ALE with off-policy methods.

Regenerative Regularization. Regenerative regularization uses an L2 penalty to encourage model weights to be near their initial values (Kumar et al., 2023). It has been shown to be effective in a number of continual learning settings.

4 Experiments

Below we present results of a suite of experiments conducted using a 2D gridworld environment (Juliani et al., 2022), the procedurally generated CoinRun environment (Cobbe et al., 2019), and the Atari game Montezuma's Revenge (Bellemare et al., 2013). All models are trained using PPO (Schulman et al., 2017). For the gridworld experiments we use an MLP encoder, and for the CoinRun and Montezuma's Revenge experiments we instead use a convolutional encoder. We set the number of environment instances (k) to 100 for all experiments conducted with the gridworld and CoinRun environments. Additional training details can be found in Appendix Section B.

4.1 Plasticity loss is apparent in on-policy RL

We find evidence of plasticity loss in the on-policy RL setting in the presence of multiple kinds of distribution shift, as presented in Figure 2. For round-level plots like these, reward values are normalized such that the mean reward at the end of the first round is set to zero. As such, positive and negative values correspond respectively to increases and decreases in performance relative to the first round. Decreasing normalized reward indicates a decay in relative performance, implying plasticity loss.

For both the permute and window shift conditions, Figure 2 demonstrates clear degradation in mean episodic reward for the warm-started model as compared to a model which has all of its weights reset between each round. While there is some positive transfer for the warm-start method in the expand shift condition, we find generalization performance suffers in this condition as compared to reset-all. We provide statistical tests comparing the performance of these methods to the warm-start and random initialization baselines in Section D of the appendix.

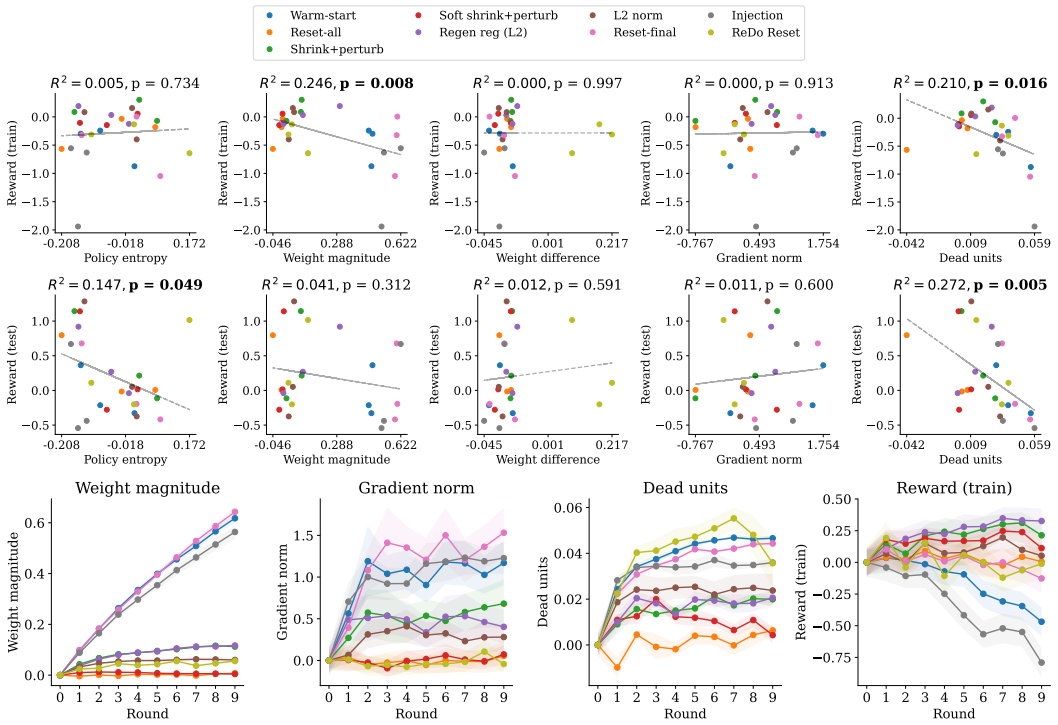

Figure 3: **Top**: Correlation plots of normalized mean reward in the gridworld environment compared against identified metrics. Each point is averaged over five replicates, and shows the final values produced after a ten-round experiment. Values for each measurement are normalized by its baseline level at initialization. Measurements that significantly correlate ($p < 0.05$) with normalized reward are bolded. **First row**: Training distribution performance. **Second row**: Test distribution performance. **Bottom**: Values of the three predictive metrics during the course of training for each intervention and window change condition as compared to training performance. Shaded regions correspond to standard error. Final values of plots like these correspond to a single point in the correlation plots above.

4.2 Predictors of plasticity loss and generalization

Identifying the exact statistical underpinnings of plasticity loss is an active area of research in continual and reinforcement learning. Here we identify several factors which are significantly correlated with plasticity loss as well as a degradation in generalization performance. Figure 3 presents correlation plots of normalized reward with various metrics of interest. Here we normalize rewards (i.e. subtract the initial reward value from the final reward value) to compensate for the fact that many of these algorithms have a regularization effect that improves performance on average but does not necessarily improve performance trends—it is possible to improve generalization, for example, without remedying plasticity loss.

Note that the *LayerNorm* and *CReLU* methods are not included in this analysis, due to their direct effects on the weight magnitude and dead unit counts, respectively. For graphs of the underlying metrics over time during training, see Supplemental Figure 11. We find that weight magnitude and number of dead units are both significantly correlated with the normalized reward [$p < 0.05$]. "Dead units" refer to ReLU activations that never produce non-negative outputs.

We find that neither the gradient magnitude nor the magnitude of weight difference between updates has a significant, linear relationship with either plasticity or generalization. This is in contrast to the findings of Abbas et al. and Nikishin et al., each of which examined plasticity loss in off-policy rather on-policy settings. The former found that weight difference was correlated with loss of plasticity while the latter found that gradient norms were predictive.

We also fit a generalized linear model (GLM) using all five metrics as independent variables and normalized reward as the dependent variable. Here we find that both gradient and weight magnitudes

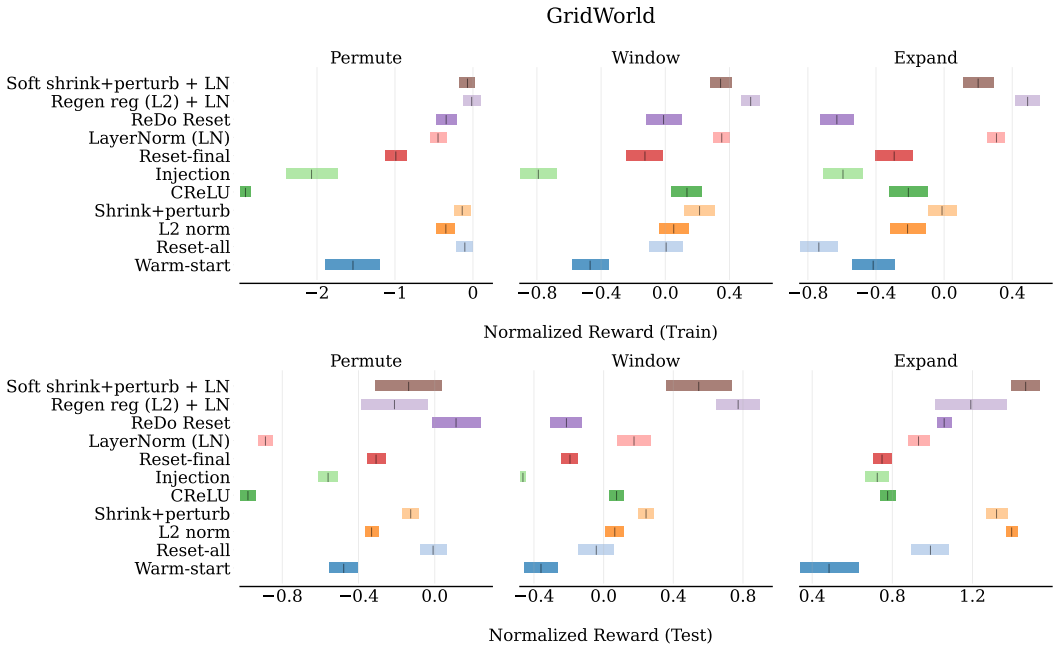

Figure 4: Performance of intervention methods compared to warm-start and reset-all baselines on the *Gridworld* environment. Final round mean reward is normalized by the performance at end of the first round, and interval bars denote standard error. **Top**: Train performance. **Bottom**: Test performance.

are significantly predictive of train performance [$p < 0.01$], while dead unit count no longer is [$p > 0.05$]. This suggests that the predictive power of the dead unit count can be accounted for by the weight magnitude. An additional GLM was fit to the test data, and we find that only the dead unit count remains predictive [$p < 0.05$].

Although these analyses do not establish causality, the robust significance of weight magnitude in predicting plasticity loss suggests that plasticity loss may be a function of how much trainable parameters deviate from their initialization distribution. As such, successful mitigation strategies would induce a sub-linear increase in the weight norm of the network parameters over the course of learning. This has the downstream effect of minimizing the number of dead units in the network. Below we compare the mitigation strategies, finding that those which address this underlying cause directly indeed perform best.

4.3 Novel architectural methods do not fully address plasticity loss

Recently a number of methods have been introduced to specifically address plasticity loss using changes to the network architecture. These methods were previously validated in the off-policy setting only. Here we consider the behavior of two of these methods as it applies to the on-policy setting: *CReLU* Abbas et al. (2023) and *Plasticity injection* Nikishin et al. (2023). We find that both methods fail to address plasticity loss in the permute shift condition. The *CReLU* method is able to address plasticity loss in the window and expand settings only, while the plasticity inject method underperforms even the warm-start baseline in all three contexts (Figure 4).

4.4 Regularization methods address plasticity loss

We next consider the class of methods which regularize the weights of the network, either continuously or intermittently. These methods include *final layer reset* (Nikishin et al., 2022), *shrink+perturb* (Ash and Adams, 2020), a continuous "soft" variant of shrink+perturb (Dohare et al., 2023b), *L2 regularization*, *ReDo* (Sokar et al., 2023), and *regenerative regularization* (Kumar et al., 2023). We find *final layer reset* fails to address plasticity loss in any of the three conditions. In contrast, the other five methods mitigate both plasticity loss and the warm-start problem to some extent in all three shift conditions (Figure 4). Of these methods, *ReDo* performs the worst as it is unable to benefit from the positive transfer which is possible in the cases of the "window" and "expand" conditions,

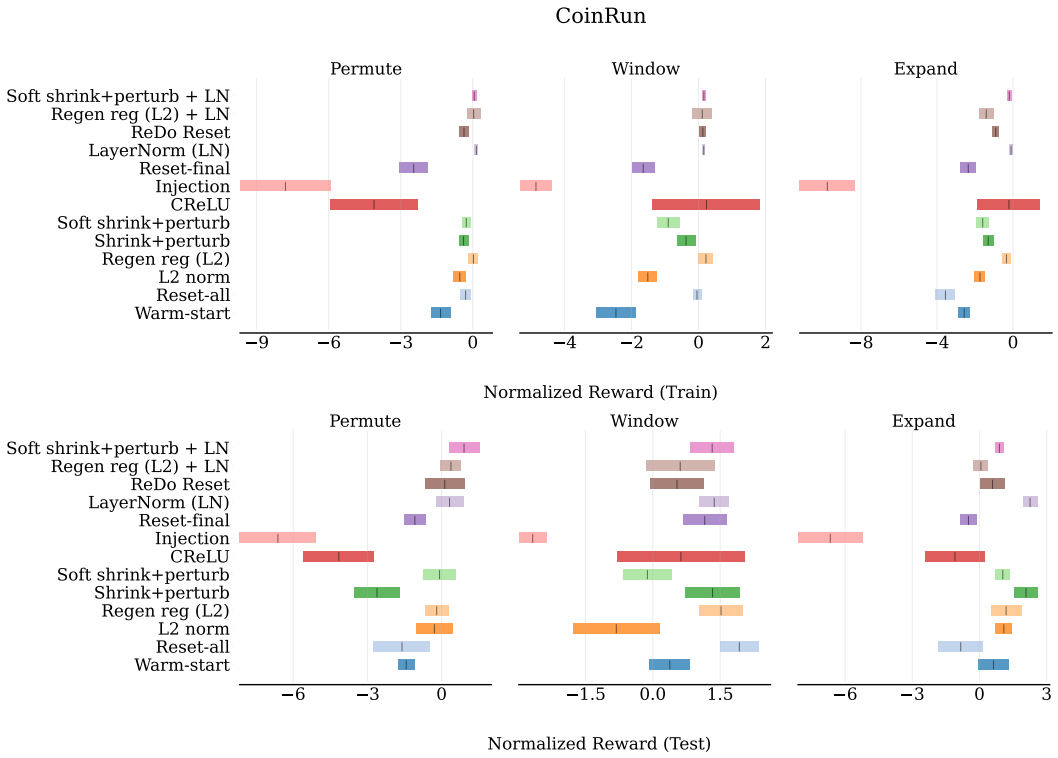

Figure 5: Performance of intervention methods compared to warm-start and reset-all baselines on the *CoinRun* environment. Final round mean reward is normalized by the performance at end of the first round, and interval bars denote standard error. **Top**: Train performance. **Bottom**: Test performance.

as methods that retain information from previous rounds effectively have access to more training data in these conditions. *ReDo* underperforms even the warm-start baseline in the "expand" condition.

4.5 LayerNorm addresses training plasticity loss but not generalization

We also consider the effect of LayerNorm on plasticity loss. It was previously demonstrated in the off-policy setting that LayerNorm was able to mitigate some effects of plasticity loss (Lyle et al., 2023). Given that LayerNorm tends to improve learning across a number of contexts (Ba et al., 2016), it is important to attempt to separate the effects of LayerNorm on performance generally from its effect on plasticity specifically. We find that LayerNorm resolves plasticity loss in terms of training performance, but is inconsistent in its effect on generalization performance. For example, the LayerNorm model significantly underperforms even the warm-start baseline on the test data of the permute shift condition. We find however that combining *soft shrink+perturb* or *regenerative regularization* with LayerNorm results in overcoming both training plasticity loss and deleterious generalization trends (Figure 4).

4.6 Plasticity loss in *CoinRun*

We also use the CoinRun environment from the ProcGen suite of tasks to evaluate the set of candidate interventions Cobbe et al. (2019, 2020). ProcGen is built using procedural generation Cobbe et al. (2020), making it possible to study the same three distribution-shift conditions which were considered in the gridworld experiments. Here we present performance results for each of the intervention methods on the CoinRun environment.

Overall, we find results largely consistent with those of the gridworld task. There is significant plasticity loss present in all three tasks as measured by the gap between the warm-start and reset-all conditions. We also find that *reset-final*, *CReLU*, and *plasticity injection* all unable to fully resolve the loss of plasticity. In contrast, the three most effective methods in the gridworld task (*LayerNorm*, *regenerative regularization*, and *soft shrink+perturb*) are all also most effective in CoinRun as well.

See Figure 5 for plots of the normalized mean rewards over the course of training for all methods. For results on two additional ProcGen environments, see Appendix D. For round-level performance plots of all methods, see Section D in the appendix. We find that relative efficacy of the intervention methods is consistent across ProcGen environments.

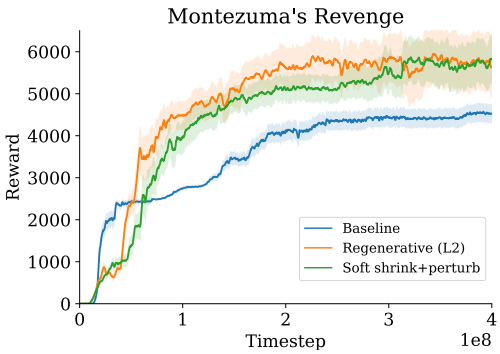

Figure 6: A comparison between RND agents trained with and without a plasticity-loss mitigating intervention. Experiments are done over twenty replicates and shaded regions show standard error.

4.7 Plasticity loss in *Montezuma's Revenge*

Montezuma's Revenge is an Atari game that is often used to benchmark the quality of exploration procedures in RL Bellemare et al. (2013); Salimans and Chen (2018). The environment consists of many separate rooms, some only accessible through a locked door. The agent must avoid obstacles, find and use keys, and traverse several rooms before obtaining non-trivial reward. Because of this reward sparsity, exploration is needed to incentivize the agent to advance beyond the periphery of its experience.

Montezuma's Revenge is designed such that the agent only sees the room it currently occupies. Accordingly, the distribution over input states expands as the agent learns to explore the environment. This introduction of significantly different states induces a level of distribution shift which is unique among ALE environments. Within our framework, the distribution shift of Montezuma's Revenge is most similar to the "expand" condition studied here.

In this experiment we consider two versions of a policy trained using the Random Network Distillation (RND) exploration bonus (Burda et al., 2018). All architectures and hyperparameters are inherited from the original RND paper. Figure 6 compares a typical RND policy to ones that are trained in conjunction with the *soft shrink+perturb* or *regenerative regularization* interventions. The plasticity loss mitigating policy achieves higher reward than a corresponding agent fitted in a standard fashion. Due to computation constraints, we did not evaluate the other intervention methods.

5 Discussion

Plasticity loss has been identified and rigorously studied in the continual learning and off-policy reinforcement learning settings. We find that it is also an issue in the on-policy setting under a variety of different conditions. Methods introduced in the off-policy setting such as *plasticity injection* and *CReLU* appear to be not as consistently effective in the on-policy setting. We hypothesize that this degradation may be due to the violation of the iid assumption which supervised learning and off-policy reinforcement learning rely on. Further, these methods require architectural modifications of the policy and value networks, and are thus less general than regularization-based alternatives.

In contrast, the class of regularization methods, which include *L2 regularization*, *ReDo*, *shrink+perturb*, and *regenerative regularization* all are able to significantly mitigate plasticity loss in our experiments. Of these methods, *soft shrink+perturb* displayed the best generalization performance, and can be easily combined with layer normalization. *LayerNorm*, a now ubiquitous method in supervised learning, was previously introduced to address plasticity loss in the off-policy setting Lyle et al. (2023), and we find that it is also effective in the on-policy setting. Due to its general regularizing effects, *LayerNorm* also increases baseline performance in the absence of distributional shift Ba et al. (2016), suggesting that it is a generally useful method for on-policy reinforcement learning methods such as PPO.

A consistent feature of the class of regularization methods discussed here is that they all normalize the network parameters towards their initial distribution. This has the effect of decreasing the weight magnitude and reducing the number of dead or saturated activation units, as seen in Figure 3. It has recently been proposed that both these metrics may be a proxy for the underlying curvature of the optimization landscape (Lewandowski et al., 2023). Studying this connection more deeply would potentially be a fruitful avenue for future research.

6 Related Work

In the context of generalization in continual learning settings, a related issue to plasticity loss is the warm-start problem (Ash and Adams, 2020). Recent work has shown that the benefits of methods such as shrink+perturb to generalization were dependent on the presence of noise in the labels used for training (Zaidi et al., 2023). In the absence of noisy labels, other regularization methods were found to be competitive with methods that utilize re-initialization such as shrink+perturb.

Another perspective on the problem of loss of plasticity has focused on the content of early learning in a network. Achille et al. (2018) demonstrate that neural networks have critical early periods of sensitivity to training data which are analogous to those in animals (Hensch, 2004). In the biological context this sensitivity is tied directly to neuronal plasticity. This connection to critical period learning was also drawn in work on off-policy RL, in which a "primacy bias" was proposed to explain the loss of plasticity seen in a number of task domains (Nikishin et al., 2022). Utilizing this insight, the periodic soft-resetting of the neural network has enabled significant increases in the sample efficiency of off-policy RL methods (D'Oro et al., 2022; Schwarzer et al., 2023).

In the deep RL domain a similar problem called *ray interference* has been identified (Schaul et al., 2019). Ray interference arises from the conflicting gradient signals related to unique sub-tasks required to solve a single more complex task in online deep reinforcement learning. It is possible that for complex tasks such as CoinRun or Montezuma's Revenge there are unique sub-tasks which are learned and whose gradients may interfere with one another. Exploring the distinction between ray interference and plasticity loss in the RL setting is a promising future direction.

In the off-policy RL setting the problem of plasticity loss has also been studied under the name of "capacity loss" (Lyle et al., 2022). While plasticity loss refers to a property of the network-task interaction, capacity loss is a more generic term to refer to the properties of a network that are task invariant. This work advocates for a solution similar to that of Kumar et al. (2023), and can thus be considered in the class of methods which regularize the network towards an initialization distribution.

Within the domain of on-policy RL there is a related phenomena to plasticity loss called "policy collapse," which refers to a degradation of performance as training progresses, even on a fixed dataset (Dohare et al., 2023c). Although these are distinct phenomena, there is likely overlap in the success of mitigation methods. A novel optimizer strategy "Non-stationary Adam" was proposed to address policy collapse, and although we did not study it, it may provide benefits on the tasks examined here. In terms of direct studies of plasticity loss in on-policy RL, both Dohare et al. (2023a) and Igl et al. (2020) show plasticity loss in the on-policy setting, and propose a method to address it.

As part of their analysis, Dohare et al. (2023b) propose a plasticity-enhancing method called "Continual Backprop," which is another form of regularization towards the initialization distribution, however it is one which is selective on a per-neuron basis. This method is similar to ReDo (Sokar et al., 2023), which we analyze here. It is also related to DrM (Xu et al., 2023), which builds on the DrQ algorithm.

It is finally worth mentioning that notions of plasticity loss have been studied in the neuroscience community as well. These range from classic work demonstrating the inability of animal models to fully accommodate new sensory input after developing cognitively without it (Wiesel and Hubel, 1963), to more recent work characterizing plasticity loss in humans as a component of various psychopathologies (Peled, 2005; Carhart-Harris et al., 2023; Juliani et al., 2024).

7 Conclusion

In this work we studied the problem of plasticity loss in the context of on-policy deep RL. We find that similar to continual learning and off-policy RL, plasticity loss is an issue across a number of different environments and forms of distributional shift. We find that some methods previously proposed to resolve the issue in other problem settings, such as *CReLU* and *plasticity injection* do not transfer to on-policy RL. The methods which best resolve plasticity loss are those which act as continual regularizers as opposed to intermittent interventions. Within this class of methods we find that a soft variant of *shrink+perturb* combined with LayerNorm performs the best across our evaluated settings, providing a simple and general method for addressing plasticity loss.

References

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. (2023). Loss of plasticity in continual deep reinforcement learning. *arXiv preprint arXiv:2303.07507*.

Achille, A., Rovere, M., and Soatto, S. (2018). Critical learning periods in deep networks. In *International Conference on Learning Representations*.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. (2020). What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*.

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

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Carhart-Harris, R., Chandaria, S., Erritzoe, D., Gazzaley, A., Girn, M., Kettner, H., Mediano, P., Nutt, D., Rosas, F., Roseman, L., et al. (2023). Canalization and plasticity in psychopathology. *Neuropharmacology*, 226:109398.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020). Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019). Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR.

Dohare, S., Hernandez-Garcia, J., Rahman, P., Sutton, R., and Mahmood, A. R. (2023a). Loss of plasticity in deep continual learning.

Dohare, S., Hernandez-Garcia, J. F., Rahman, P., Sutton, R. S., and Mahmood, A. R. (2023b). Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*.

Dohare, S., Lan, Q., and Mahmood, A. R. (2023c). Overcoming policy collapse in deep reinforcement learning. In *Sixteenth European Workshop on Reinforcement Learning*.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. (2022). Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop NeurIPS 2022*.

Hensch, T. K. (2004). Critical period regulation. *Annu. Rev. Neurosci.*, 27:549–579.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. (2020). Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*.

Juliani, A., Barnett, S., Davis, B., Sereno, M., and Momennejad, I. (2022). Neuro-nav: a library for neurally-plausible reinforcement learning. *arXiv preprint arXiv:2206.03312*.

Juliani, A., Safron, A., and Kanai, R. (2024). Deep canals: A deep learning approach to refining the canalization theory of psychopathology. *Neuroscience of Consciousness*.

Kumar, S., Marklund, H., and Van Roy, B. (2023). Maintaining plasticity via regenerative regularization. *arXiv preprint arXiv:2308.11958*.

Lee, H., Cho, H., Kim, H., Gwak, D., Kim, J., Choo, J., Yun, S.-Y., and Yun, C. (2024). Plastic: Improving input and label plasticity for sample efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Lewandowski, A., Tanaka, H., Schuurmans, D., and Machado, M. C. (2023). Curvature explains loss of plasticity. *arXiv preprint arXiv:2312.00246*.

Lyle, C., Rowland, M., and Dabney, W. (2022). Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*.

Lyle, C., Zheng, Z., Nikishin, E., Pires, B. A., Pascanu, R., and Dabney, W. (2023). Understanding plasticity in neural networks. *arXiv preprint arXiv:2303.01486*.

Nikishin, E., Oh, J., Ostrovski, G., Lyle, C., Pascanu, R., Dabney, W., and Barreto, A. (2023). Deep reinforcement learning with plasticity injection. *arXiv preprint arXiv:2305.15555*.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. (2022). The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR.

Peled, A. (2005). Plasticity imbalance in mental disorders the neuroscience of psychiatry: implications for diagnosis and research. *Medical hypotheses*, 65(5):947–952.

Salimans, T. and Chen, R. (2018). Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*.

Schaul, T., Borsa, D., Modayil, J., and Pascanu, R. (2019). Ray interference: a source of plateaus in deep reinforcement learning. *arXiv preprint arXiv:1904.11455*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. (2023). Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. (2023). The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pages 32145–32168. PMLR.

Wiesel, T. N. and Hubel, D. H. (1963). Effects of visual deprivation on morphology and physiology of cells in the cat's lateral geniculate body. *Journal of neurophysiology*, 26(6):978–993.

Xu, G., Zheng, R., Liang, Y., Wang, X., Yuan, Z., Ji, T., Luo, Y., Liu, X., Yuan, J., Hua, P., et al. (2023). Drm: Mastering visual reinforcement learning through dormant ratio minimization. *arXiv preprint arXiv:2310.19668*.

Zaidi, S., Berariu, T., Kim, H., Bornschein, J., Clopath, C., Teh, Y. W., and Pascanu, R. (2023). When does re-initialization work? In *Proceedings on*, pages 12–26. PMLR.

A Implementation Details

Below are the specific implementation details for each intervention. Wherever possible we followed the methods provided by the papers which introduced each method. Relevant hyperparameters used are listed in Table 1.

Shrink+perturb. When the intervention is applied all learnable parameters in the network are iterated through and scaled by α. All parameters are then additively combined with newly sampled initialization parameters which are scaled by β. For a set of parameters x, this corresponds to $x_{new} = \alpha x_{current} + \beta x_{init}$, where x_{init} is sampled from the parameter initialization distribution. For all experiments $\alpha = 1 - \beta$.

Soft shrink+perturb. The shrink+perturb procedure is applied after each step of gradient descent instead of only at specific intervals. Specifically, for a set of parameters x, this corresponds to $x_{new} = \alpha x_{current} + \beta x_{init}$, where x_{init} is sampled from the parameter initialization distribution. For all experiments $\alpha = 1 - \beta$.

CReLU. All ReLU activations in the network are replaced with CReLU. The equation for CReLU is: $y = concat(ReLU(x), ReLU(-x))$. We use the method where input parameter sizes are halved compared to the baseline so that output parameters remain consistent for each layer. This results in a network with half the number of parameters for the effected layers.

Regenerative Regularization (L2). An additional loss term is added to the PPO update in each step of gradient descent. This term corresponds to L2 norm of the difference between the current parameters of the network and the network parameters at initialization. This loss is scaled by α.

Plasticity injection. We follow the method described by Nikishin et al. for the initial intervention, only performing plasticity injection on the final layers of the network (i.e. value and policy heads). To ensure computational efficiency, during subsequent interventions we combine the previous parameter values into a single set of weights and biases (i.e. $w_{old} = w_{old} + w_{newa.}w_{newb}$). This ensures that at any given time the plasticity-injected layer is represented by $y = x_{old} + x_{newa.}sg(x_{newb})$.

LayerNorm. We apply the LayerNorm Ba et al. (2016) function before the ReLU activation at each layer.

Reset-final. At each intervention point we replace the weights of the final layers (i.e. value and policy heads) of the network with freshly initialized values.

L2 Norm. We apply an additional loss term to the PPO update which corresponds to the L2 norm of the weights of the network. This loss is scaled by a parameter α.

ReDo. At a fixed epoch interval the procedure from Sokar et al. is performed on all layers of the neural network. A mini-batch of training data is used to calculate the dormancy rate for each unit, and units which are below τ are re-initialized.

B Environment and Model Details

Additional hyperparameters used for Gridworld and CoinRun models are provided in Table 2. All experiments involve running five repetitions with unique random seeds each. All experiments are conducted using either a single P100 or V100 GPU on a cloud machine. We used preliminary exploratory training runs to determine the necessary number of epochs to ensure convergence within the initial single round of training. In these longer training runs we did not find evidence of policy collapse Dohare et al. (2023c), and thus believe that performance degradation in subsequent rounds can be attributed to plasticity loss. Code which can be used to reproduce our results is available at https://github.com/awjuliani/deep-rl-plasticity.

Gridworld. The gridworld environment consisted of an 11x11x4 one-hot encoded observation space. Each observation was flattened into a single vector and provided as input to the model. In the permute environment variation, observations were split into 1x1x4 patches and randomized according to a fixed map within each round. The PPO model consisted of an MLP with two hidden layers, ReLU activations, and a "dual-head" architecture, as described in Schulman et al. (2017). The action space consisted of four possible actions, corresponding to movement in the cardinal directions. All episodes lasted 100 time-steps. Each round of training consists of 20,000 iterations of inference and model updates. Given ten rounds, the total number of iterations is 200,000.

CoinRun. The CoinRun environment consisted of a 64x64x3 observation space. This was provided to the PPO model as a tensor, and processed by a convolutional encoder with four layers and ReLU activations. Each convolutional layer utilized a kernel size of four and a stride of two. In the permute environment variation, images were broken into 8x8x3 patches and the position of patches was randomized according to a fixed map within each round. A "dual-head" architecture was also utilized. The action space consisted of the standard nine possible actions provided by the ProcGen environment Cobbe et al. (2020). Episode length varied based on agent behavior. Each round of training consists of 5,000 iterations of inference and model updates. Given ten rounds, the total number of iterations is 50,000.

Montezuma's Revenge. Like in CoinRun, Montezuma's revenge states are processed into observations of size 64x64x3 and passed through a convolutional encoder. We inherit hyperparameters from Burda et al. (2018), which were previously optimized for final performance. These include 4 randomly masked update epochs per observation, an entropy coefficient of 0.001, γ_E = 0.999 and γ_I =0.99. We use a learning rate of 0.0001 and 128 simultaneous agents simultaneously. All model updates are performed via the Adam optimizer. We use a shrinkage coefficient $1 - 10^{-8}$ and a perturbation coefficient of 10^{-8}.

C Hyperparameters

Method	Gridworld	CoinRun
L2-Norm (α)	1e-3	1e-3
Regen-Reg (α)	1e-4	1e-2
Soft-S+P (β)	1e-6	1e-6
S+P (β)	0.5	0.5
ReDo period	10 epochs	10 epochs
ReDo (τ)	0.025	0.025

Table 1: Optimal hyperparameter values used for different environments and interventions. Chosen values are the result of performing a sweep over possible values using *permute* environment shift condition.

Parameter	Value
Number of hidden units	256
Learning rate	5e-4
Default activation	ReLU
Minibatch size	64
Buffer size	1024
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clip parameter (ϵ)	0.2
Entropy parameter (β)	0.02
Number of update epochs	3

Table 2: Default values used in PPO algorithm for all gridworld and CoinRun experiments.

D Extended Results

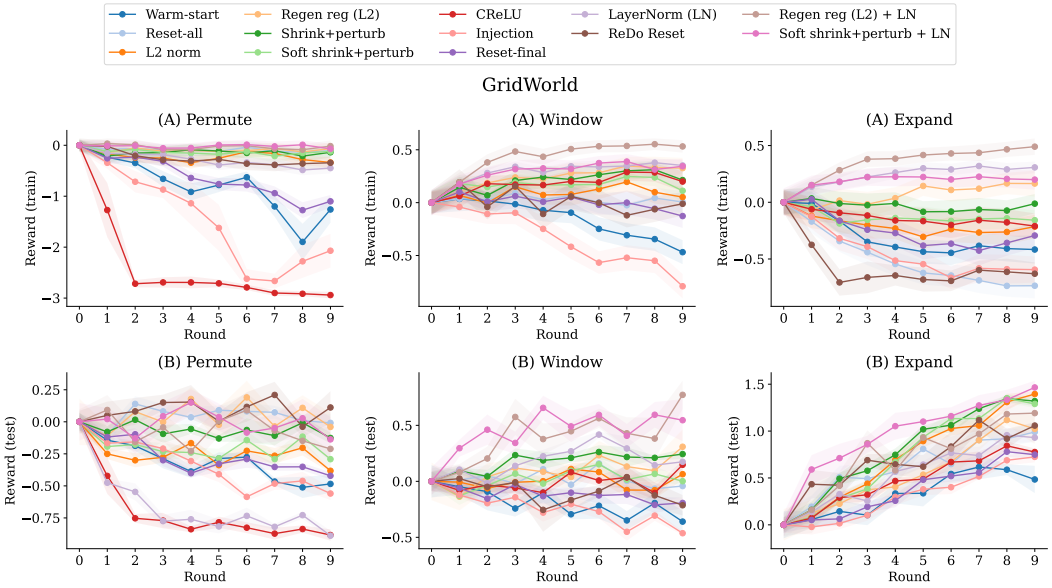

Figure 7: Performance of intervention methods compared to baseline (warm-start) on Gridworld task. Shaded region corresponds to normalized mean episodic reward. **Tow row**: Training distribution performance. **Bottom row**: Test distribution performance.

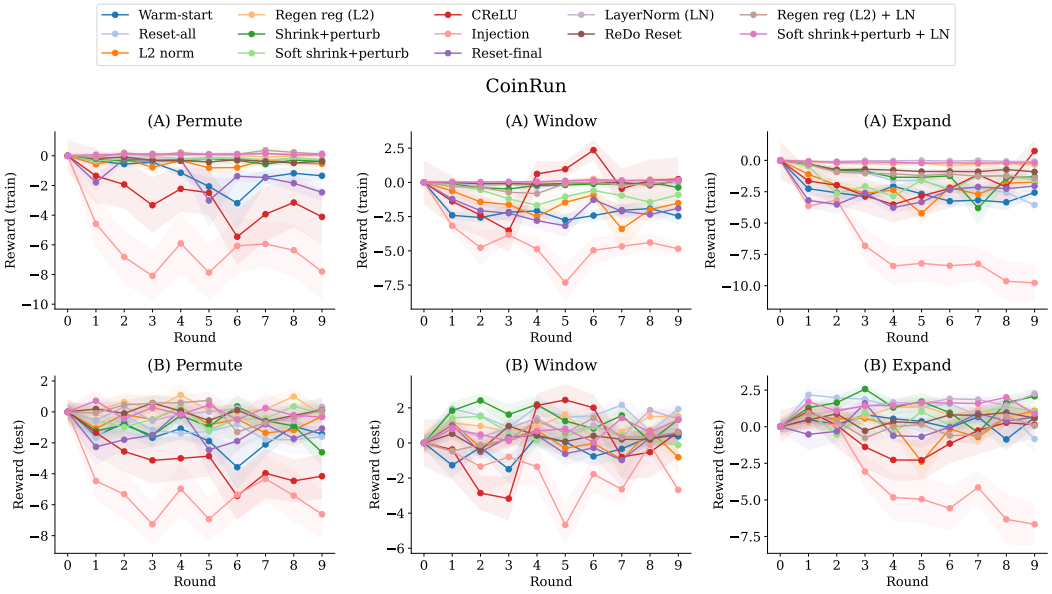

Figure 8: Performance of intervention methods compared to baseline (warm-start) on CoinRun task. Shaded region corresponds to normalized mean episodic reward. **Top row**: Training distribution performance. **Bottom row**: Test distribution performance.

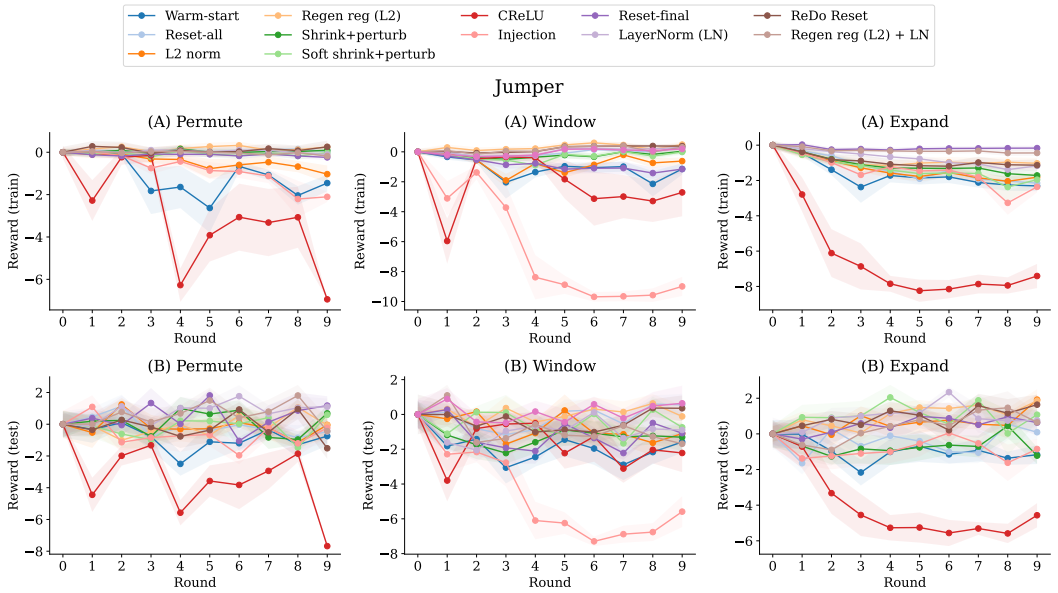

Figure 9: Performance of intervention methods compared to baseline (warm-start) on Jumper task. **Top row**: Training distribution performance. **Bottom row**: Test distribution performance.

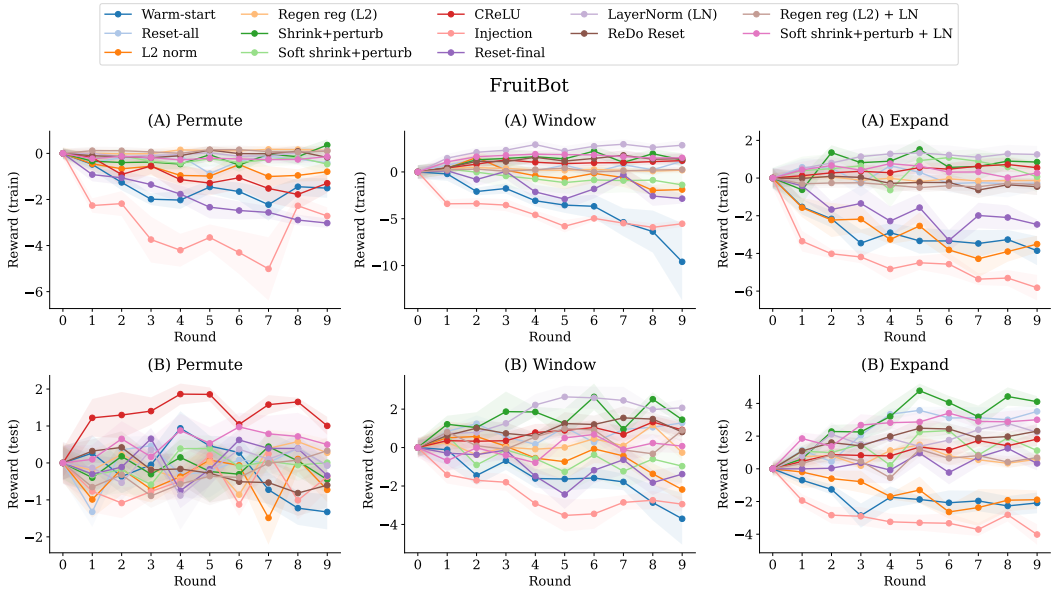

Figure 10: Performance of intervention methods compared to baseline (warm-start) on Fruitbot task. **Top row**: Training distribution performance. **Bottom row**: Test distribution performance.

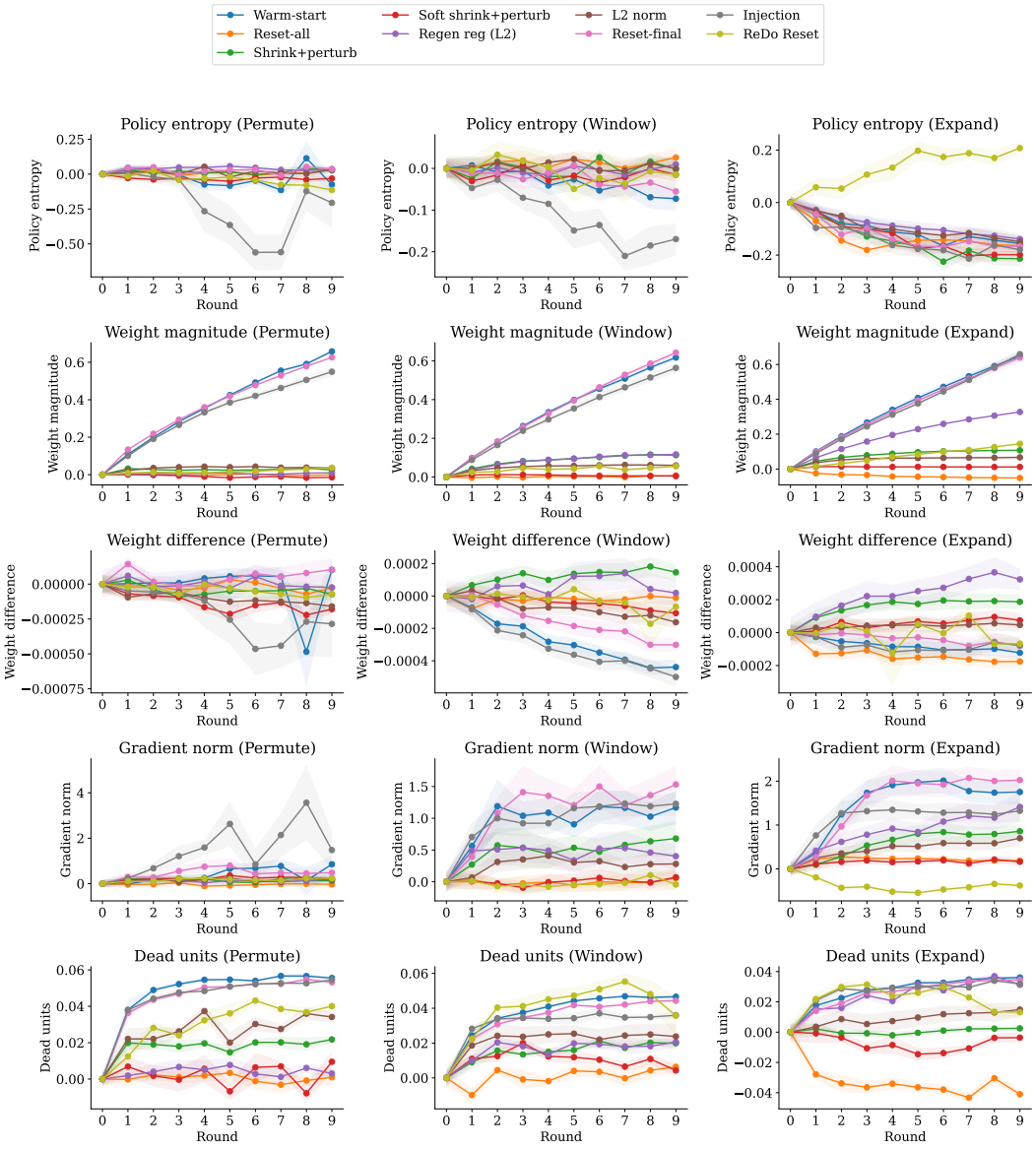

Figure 11: Effect of learning under various conditions on five diagnostic metrics of interest.

Dep. Variable:	train_reward	**No. Observations:**	27
Model:	GLM	**Df Residuals:**	21
Model Family:	Gaussian	**Df Model:**	5
Link Function:	Identity	**Scale:**	0.13264
Method:	IRLS	**Log-Likelihood:**	-7.6473
		Deviance:	2.7855
		Pearson chi2:	2.79
No. Iterations:	3	**Pseudo R-squ. (CS):**	0.5624

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	-0.0920	0.118	-0.783	0.434	-0.322	0.138
entropy	1.6302	0.940	1.734	0.083	-0.213	3.473
weight_mag	-1.8262	0.637	-2.868	0.004	-3.074	-0.578
weight_diff	-35.4734	133.365	-0.266	0.790	-296.865	225.918
grad_norm	0.5742	0.184	3.116	0.002	0.213	0.935
dead_units	0.5074	5.905	0.086	0.932	-11.066	12.081

Table 3: Results of GLM using various diagnostic metrics to predict reward during training.

Dep. Variable:	test_reward	**No. Observations:**	27
Model:	GLM	**Df Residuals:**	21
Model Family:	Gaussian	**Df Model:**	5
Link Function:	Identity	**Scale:**	0.19798
Method:	IRLS	**Log-Likelihood:**	-13.054
		Deviance:	4.1576
		Pearson chi2:	4.16
No. Iterations:	3	**Pseudo R-squ. (CS):**	0.5215

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.3249	0.144	2.263	0.024	0.043	0.606
entropy	-1.4060	1.149	-1.224	0.221	-3.657	0.845
weight_mag	1.1259	0.778	1.447	0.148	-0.399	2.651
weight_diff	327.8923	162.934	2.012	0.044	8.547	647.238
grad_norm	-0.0568	0.225	-0.252	0.801	-0.498	0.384
dead_units	-21.6282	7.214	-2.998	0.003	-35.768	-7.488

Table 4: Results of GLM using various diagnostic metrics to predict reward on the test distribution.

Gridworld (Permute - Train)		
Method	**t-value**	**p-value**
Warm-start	t(7) = 2.524	0.04
CReLU	t(7) = 10.477	0.0
Injection	t(8) = 6.21	0.0
Reset-final	t(7) = 2.444	0.044
L2 norm	t(7) = -0.08	0.939
Shrink+perturb	t(8) = -0.087	0.933
Soft shrink+perturb	t(8) = 0.472	0.65
Regen reg (L2)	t(8) = 0.134	0.897
LayerNorm (LN)	t(8) = 0.787	0.454
Soft shrink+perturb + LN	t(8) = -0.212	0.837
Regen reg (L2) + LN	t(8) = -1.368	0.208

Table 5: Results of statistical tests comparing different methods to baseline of *reset-all* in gridworld (permute) condition. T-values below zero correspond to train distribution performance better than baseline. T-values above zero correspond to performance worse than baseline. P-values above 0.05 correspond to methods which produce mean normalized rewards not significantly different from the baseline.

Gridworld (Window - Train)		
Method	**t-value**	**p-value**
Warm-start	t(8) = 0.759	0.469
CReLU	t(7) = -1.563	0.162
Injection	t(8) = 1.741	0.12
Reset-final	t(8) = -0.017	0.987
L2 norm	t(8) = -1.291	0.233
Shrink+perturb	t(8) = -1.361	0.211
Soft shrink+perturb	t(7) = -0.222	0.831
Regen reg (L2)	t(8) = -1.068	0.317
LayerNorm (LN)	t(7) = -2.107	0.073
Soft shrink+perturb + LN	t(8) = -1.549	0.16
Regen reg (L2) + LN	t(8) = -3.106	0.015

Table 6: Results of statistical tests comparing different methods to baseline of *reset-all* in gridworld (window) condition. T-values below zero correspond to train distribution performance better than baseline. T-values above zero correspond to performance worse than baseline. P-values above 0.05 correspond to methods which produce mean normalized rewards not significantly different from the baseline.

Gridworld (Expand - Train)		
Method	**t-value**	**p-value**
Warm-start	t(8) = -0.926	0.382
CReLU	t(8) = -2.247	0.055
Injection	t(8) = -0.715	0.495
Reset-final	t(8) = -1.301	0.229
L2 norm	t(8) = -2.252	0.054
Shrink+perturb	t(8) = -2.476	0.038
Soft shrink+perturb	t(7) = -1.788	0.117
Regen reg (L2)	t(8) = -2.879	0.021
LayerNorm (LN)	t(8) = -4.098	0.003
Soft shrink+perturb + LN	t(8) = -3.683	0.006
Regen reg (L2) + LN	t(8) = -5.44	0.001

Table 7: Results of statistical tests comparing different methods to baseline of *reset-all* in gridworld (expand) condition. T-values below zero correspond to train distribution performance better than baseline. T-values above zero correspond to performance worse than baseline. P-values above 0.05 correspond to methods which produce mean normalized rewards not significantly different from the baseline.

Gridworld (Permute - Test)		
Method	**t-value**	**p-value**
Warm-start	t(7) = 3.0	0.02
CReLU	t(7) = 6.975	0.0
Injection	t(8) = 3.595	0.007
Reset-final	t(7) = 2.445	0.044
L2 norm	t(7) = 3.009	0.02
Shrink+perturb	t(8) = 0.907	0.391
Soft shrink+perturb	t(8) = 2.177	0.061
Regen reg (L2)	t(8) = 0.007	0.995
LayerNorm (LN)	t(8) = 6.959	0.0
Soft shrink+perturb + LN	t(8) = 0.235	0.82
Regen reg (L2) + LN	t(8) = 0.804	0.445

Table 8: Results of statistical tests comparing different methods to baseline of *reset-all* in gridworld (permute) condition. T-values below zero correspond to test distribution performance better than baseline. T-values above zero correspond to performance worse than baseline. P-values above 0.05 correspond to methods which produce mean normalized rewards not significantly different from the baseline.

Gridworld (Window - Test)		
Method	**t-value**	**p-value**
Warm-start	t(8) = 0.882	0.404
CReLU	t(7) = 0.239	0.818
Injection	t(8) = 1.444	0.187
Reset-final	t(8) = 0.804	0.445
L2 norm	t(8) = 0.216	0.834
Shrink+perturb	t(8) = -0.781	0.458
Soft shrink+perturb	t(7) = -0.046	0.965
Regen reg (L2)	t(8) = -0.187	0.856
LayerNorm (LN)	t(7) = -0.813	0.443
Soft shrink+perturb + LN	t(8) = -2.349	0.047
Regen reg (L2) + LN	t(8) = -1.714	0.125

Table 9: Results of statistical tests comparing different methods to baseline of *reset-all* in gridworld (window) condition. T-values below zero correspond to test distribution performance better than baseline. T-values above zero correspond to performance worse than baseline. P-values above 0.05 correspond to methods which produce mean normalized rewards not significantly different from the baseline.

Gridworld (Expand - Test)		
Method	**t-value**	**p-value**
Warm-start	t(8) = 1.227	0.255
CReLU	t(8) = 0.891	0.399
Injection	t(8) = 1.815	0.107
Reset-final	t(8) = 1.457	0.183
L2 norm	t(8) = -0.512	0.623
Shrink+perturb	t(8) = -1.021	0.337
Soft shrink+perturb	t(7) = -0.527	0.614
Regen reg (L2)	t(8) = 0.333	0.748
LayerNorm (LN)	t(8) = 0.438	0.673
Soft shrink+perturb + LN	t(8) = -1.793	0.111
Regen reg (L2) + LN	t(8) = -0.579	0.579

Table 10: Results of statistical tests comparing different methods to baseline of *reset-all* in gridworld (expand) condition. T-values below zero correspond to test distribution performance better than baseline. T-values above zero correspond to performance worse than baseline. P-values above 0.05 correspond to methods which produce mean normalized rewards not significantly different from the baseline.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract accurately reflects all the findings presented in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed throughout results and discussion sections.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Our paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The appendix sections contain the necessary hyperparameters and implementation details to reproduce our work. Furthermore, we provide code to reproduce the results and figures.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide as a supplemental file the code necessary to reproduce all results and figures excluding the results presented in section 4.7 (Montezuma's Revenge).

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer:[Yes]

   Justification: The results and appendix provides all necessary experimental details to fully understand and interpret our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We use error bars for all figures. Furthermore, we provide statistical tests where relevant in the main body of the text, along with more extended statistical results in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on the compute resources used in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: Our research fully conforms to the code of ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We include an impact statement at the end of the main body of the paper.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: There are no risks associated with our paper or code.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly attribute all relevant software and datasets used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide code to reproduce our results, and this code contains documentation necessary for running it.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: There were no human subjects involved in our research.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: There were no human subjects involved in our research.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.