LOSS SMOOTHING FOR CONTINUAL ADAPTATION

Darshan Patil[1,2,3] **Ekaterina Lobacheva**[1,2,3] **Razvan Pascanu**[2] **Sarath Chandar**[1,2,4,5]
[1]Chandar Research Lab [2]Mila – Quebec AI Institute [3]Université de Montréal
[4]Polytechnique Montréal [5]Canada CIFAR AI Chair
Correspondence: darshan.patil@mila.quebec

ABSTRACT

Neural networks are often adapted in nonstationary data distribution settings where the objective is to optimize performance on the current task, and preserving accuracy on previous tasks is not required. As a result, existing methods primarily focus on improving plasticity, while stability is largely studied in the context of continual learning. In this work, we examine whether preserving stability can also be beneficial in model adaptation settings where past-task performance is irrelevant. We propose a simple loss smoothing approach that encourages selective adaptation by preserving task-shared features while modifying task-inconsistent ones. We evaluate our method on continual supervised model adaptation benchmarks and reinforcement learning benchmarks, and show that promoting representational stability during adaptation can improve performance across settings.

1 INTRODUCTION

Modern neural networks are rarely trained exclusively under stationary data distributions. In many practical settings, such as fine-tuning pretrained models (Shi et al., 2025), online learning (Sodhani et al., 2022), and reinforcement learning (RL) (Sutton & Barto, 2018), the data distribution can shift one or multiple times during training. When the data distribution does change, models must now adapt to this new task. The primary goal in such model adaptation settings is to achieve high performance on the new task, rather than to explicitly preserve performance on past tasks.

Adaptation is often framed as a problem of learning new information efficiently, with a strong focus on plasticity, the ability to learn new representations that are compatible with the new data. A wide range of methods, ranging from architectural changes (Lyle et al., 2024; Lee et al., 2025) to optimizer (Dohare et al., 2024; Lee et al., 2024) and regularization techniques (Ash & Adams, 2020; Sokar et al., 2023), have been proposed to improve plasticity and accelerate learning after distribution shifts. Many recent works have shown that improved plasticity can lead to faster and more effective learning in settings such as RL (Nikishin et al., 2022).

In contrast to plasticity, stability represents the model's ability to maintain representations and knowledge learned in previous tasks. It has primarily been studied in the context of preventing catastrophic forgetting in continual learning, where the explicit goal is to retain performance on previous tasks while learning new ones. Continual learning methods therefore emphasize mechanisms such as replay (Rolnick et al., 2019), parameter isolation (Rusu et al., 2022), or regularization (Kirkpatrick et al., 2017) to prevent catastrophic forgetting. The stability-plasticity dilemma is often framed as a tension between these two objectives, with the assumption that improving one necessarily comes at the cost of the other.

In this work, we explore how preserving stability can be beneficial in model adaptation settings, even when past-task performance is irrelevant. We propose a loss smoothing-based adaptation method that uses replay to gradually transition the training objective from the old task to the new one. Unlike standard fine-tuning, this selectively encourages change in task-inconsistent features while preserving task-shared ones, promoting feature-level stability during adaptation without requiring the model to maintain accuracy on previous tasks. We empirically demonstrate, in both supervised and reinforcement learning settings, that standard fine-tuning after a distribution shift can substantially alter all features in the network, including those shared between tasks. This indiscriminate feature drift can interfere with efficient learning by unnecessarily rewriting representations that remain compatible with the new data. We show that leveraging loss smoothing improves adaptation

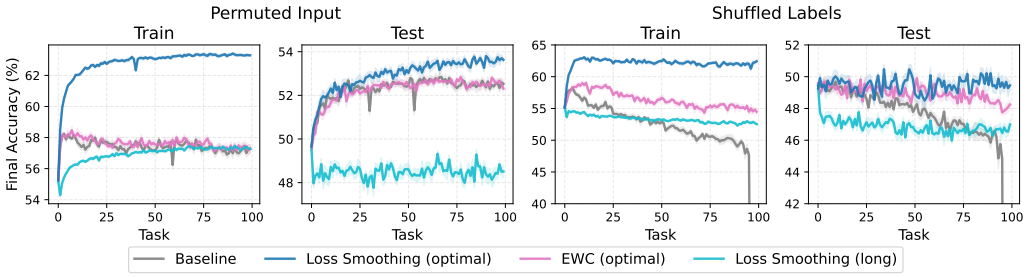


Figure 1: Supervised settings: resulting accuracy on a sequence of tasks.

performance, suggesting that stability and plasticity need not be opposing objectives even outside the continual learning setting.

2 Method and Preliminaries

Preliminaries We consider a continual model adaptation setting where a model with parameters θ is trained on a sequence of K tasks. Unlike traditional continual learning, our primary goal is to maximize performance on the current task rather than to explicitly preserve accuracy on previous tasks. Each task $k \in \{1, \ldots, K\}$ is associated with a data distribution $\mathcal{D}^{(k)}$, from which we sample batches of data during training. The model begins with initial parameters θ_0 and is sequentially trained on each task. The function $l(\theta; B)$ computes the objective on a batch of data B given model parameters θ. In supervised learning settings, this could be the cross-entropy loss, while in reinforcement learning settings, this could be the temporal difference (TD) error. We assume access to a stopping criterion $\textsc{Stop}(k, \theta)$ that determines when the current task ends and the next one begins.

Method Recent work on two stage fine-tuning (Kumar et al., 2022; Trivedi et al., 2022) has shown that directly attempting to finetune a pre-trained model can lead to excessive feature distortion in the network and lead to underperformance on downstream tasks. Rather, it is often more effective to first train the last layer of the model (linear probe) on the new task, and then fine-tune the entire network once the head is more aligned with the new data. While this approach is effective at improving adaptation performance, it also is somewhat inflexible as it requires you to predetermine which part of the network needs to be frozen in the first phase.

Algorithm 1: Loss Smoothing

Input: θ_0, $\{\mathcal{D}^{(k)}\}_{k=1}^{K}$, τ, Opt, $\textsc{Stop}(k, \theta)$
$\theta \leftarrow \theta_0$; $B_{\text{rep}} \leftarrow \emptyset$
for $k \leftarrow 1$ **to** K **do**
  $t \leftarrow 0$
  while ***not*** $\textsc{Stop}(k, \theta)$ **do**
    $B_{\text{cur}} \sim \mathcal{D}^{(k)}$
    $\alpha \leftarrow \min(1, t/\tau)$
    $\mathcal{L}_{cur} \leftarrow l(\theta; B_{\text{cur}})$
    if $B_{rep} \neq \emptyset$ **then**
      $\mathcal{L}_{rep} \leftarrow l(\theta; B_{\text{rep}})$
    else
      $\mathcal{L}_{rep} \leftarrow 0$
    $\mathcal{L} \leftarrow (1-\alpha)\mathcal{L}_{rep} + \alpha\mathcal{L}_{cur}$
    $\theta \leftarrow \text{Opt}(\theta, \nabla_\theta \mathcal{L})$
    $t \leftarrow t + 1$
  $B_{\text{rep}} \leftarrow B_{\text{cur}}$
return θ

Motivated by this, we propose a simple loss smoothing-based continual adaptation method that uses replay to gradually transition the training objective from the old task to the new one. Unlike standard replay from continual learning, we use this replay buffer to interpolate the loss from the previous task to the current one over a fixed number of steps τ. This approach helps to preserve the features shared by both tasks at the beginning of training on the new task, while allowing task-inconsistent features to eventually adapt to the new task. In practice, for our replay, we simply use the last batch of data from the previous task. Our procedure is outlined in Algorithm 1.

3 Supervised Learning

We first consider synthetic supervised learning setups with controlled data distribution shifts. We base our experiments on CIFAR-10 (Krizhevsky, 2009) and use two standard transforma-

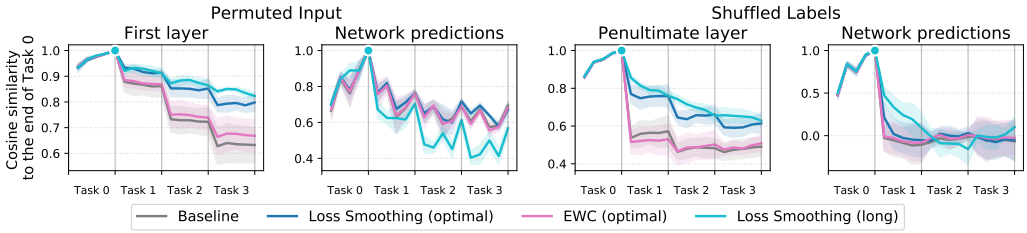


Figure 2: Supervised settings: analysis of the drift in network activations from the final checkpoint of Task 0 (indicated by the round marker) as the model is trained on subsequent tasks.

tions (Patil et al., 2024): *permuted input* and *shuffled labels*. In the permuted-input setting, a randomly generated permutation in the input space, $P_k^x : \mathcal{X} \to \mathcal{X}$, reorders pixel locations. Task k is defined by the transformed dataset $\mathcal{D}_k = \{(P_k^x(x_i), y_i)\}_{i=1}^n$. In the shuffled-label setting, a randomly generated permutation in the output space, $P_k^y : \mathcal{Y} \to \mathcal{Y}$, remaps the labels, and task k is defined as $\mathcal{D}_k = \{(x_i, P_k^y(y_i))\}_{i=1}^n$.

In Figure 1, we compare loss smoothing to a baseline, both with optimal hyperparameters. Loss smoothing outperforms the baseline on the test set, with larger gains on later tasks. Interestingly, training accuracy is also higher, suggesting that the improvement stems from better trainability, despite the additional regularization to the previous task.

Main components To demonstrate the importance of our two main design choices—*how* regularization is applied (via loss smoothing) and *when* it is applied (only at the beginning of training on each task)—we add two additional baselines in Figure 1. First, we compare to an EWC (Kirkpatrick et al., 2017) variant of our method, which follows the same training procedure and regularization coefficient schedule but changes the regularization term from the replay data loss to EWC regularization. This proves substantially less effective in our experiments. We hypothesize that this is because EWC does not differentiate between task-shared and task-inconsistent weights: by penalizing changes to all weights important to the previous task, it not only prevents distortion in task-shared weights but also inadvertently restricts updates to task-inconsistent weights, and therefore fails to rebalance the training dynamics between the two. Second, we compare with a variant of our method with a long decay schedule, where the number of loss smoothing steps τ is equal to the whole length of training, making it closer to standard replay methods from continual learning. The long decay schedule leads to a significant performance degradation in both setups. We attribute this to excessive emphasis on old data, which prevents the model from effectively adapting to the new task.

Method analysis To validate our initial motivation that loss smoothing facilitates adaptation by helping the model identify which components to preserve and which to modify, in Figure 2 we analyze the drift in network activations from the final checkpoint of Task 0 as the model is trained on subsequent tasks. To compare two checkpoints, we take a random subset of examples and compute layer-wise activations for each of them via a forward pass. For each layer, we then compute the average cosine similarity between the corresponding activations. In the permuted-input setting, we apply the permutation associated with the checkpoint's task to the inputs before the forward pass.

In the permuted-input setting, the functions implemented by all layers except the first one can, in principle, be shared across tasks, whereas the first layer cannot due to the different pixel orderings. We observe that the first-layer activations change substantially less under loss smoothing than under the baseline, while the final predictions change to a similar extent for both methods. This suggests that loss smoothing preserves the functions of the deeper layers by primarily adapting the first layer to reproduce the appropriate activations under the new input permutation.

In the shuffled-label setting, all layers except the last one can be shared across tasks, since only the image-label correspondence changes. Here, the penultimate-layer activations change much less with loss smoothing than with the baseline, while the final predictions again change similarly for both methods. This indicates that loss smoothing preserves the shared representation and primarily adapts the last layer, whereas the baseline modifies the network more uniformly.

Additionally, we observe that EWC is ineffective at redistributing changes across the model, while loss smoothing with a long decay schedule alters the final predictions too strongly and prevents the model from learning an appropriate solution for the current task. In Appendix A.2, we further show

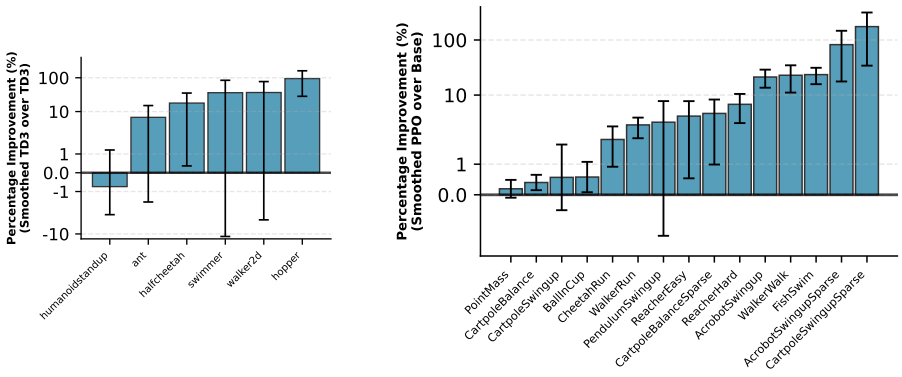


Figure 3: Percentage improvements of smoothed agents over baseline agents for TD3 on Brax environments **(left)** and PPO on DeepMind Control Suite environments **(right)**.

that the positive effect of loss smoothing cannot be explained by implicit learning rate warm-up or increased data per optimization step, and provide additional ablations of hyperparameter effects.

4 Reinforcement Learning

We next evaluate loss smoothing on more practical reinforcement learning settings. RL provides a testbed for evaluating adaptation methods on more natural and complex distribution shifts. Not only does the input data distribution change as the agent learns and explores the environment, but if the agent uses bootstrapping-based methods, then the output distribution also changes as the agent's value function estimates change over time. We evaluate the applicability of our method to both off-policy (TD3 (Fujimoto et al., 2018)) and on-policy (PPO (Schulman et al., 2017)) algorithms, testing them on two suites of continuous control benchmarks: Brax (Freeman et al., 2021) and DeepMind Control Suite (DMC) (Tassa et al., 2018) implemented in the Mujoco Playground (Zakka et al., 2025), respectively. While we could employ loss smoothing on both the policy and critic losses, we focus on the critic in this work since previous work has shown that critic adaptation can be more difficult than policy adaptation (Ma et al., 2023). Since our algorithm requires knowledge of task shifts that are not explicitly defined in RL, we can either use heuristics such as denoting a distribution shift with every new set of policy rollouts or just set a fixed number of training steps for each task, denoted by a hyperparameter n. In our experiments, we use the latter.

While TD3 already has stabilization mechanisms in the replay buffer and target network, our experience replay buffer is separate and stores the previous *outdated* targets directly, rather than recomputing the targets based on the current target and policy networks, enabling a more direct interpolation between the old and new objectives. Similarly, with PPO, we save a batch of previous data and corresponding outdated targets to compute the interpolation. We find that for both algorithms, despite the already existing mechanisms to promote network stability, loss smoothing provides significant improvements over the baseline, with some environments seeing nearly double the performance (Figure 3). For TD3, we find loss smoothing provides improvements on 5 out of 6 Brax environments, and for PPO, we find that it provides improvements on 15 of the 18 DMC environments that we evaluate on (neither agent made significant progress on the 3 Humanoid environments, excluded from the Figure).

5 Conclusion

In this work, we propose loss smoothing, a simple method for continual model adaptation that uses replay to gradually transition the training objective from the old task to the new one. We evaluate our method on both supervised learning and reinforcement learning settings, and find that it provides significant improvements across a variety of continual supervised model adaptation tasks and reinforcement learning environments. Loss smoothing enables better adaptation by automatically stabilizing the features in the network that are shared between the old and new data, while allowing task-inconsistent features to eventually adapt to the new task. Our results suggest that simple, stability-aware approaches can substantially improve adaptation performance, offering a promising path toward more robust learning under distribution shift.

ACKNOWLEDGMENTS

We would like to thank Andrei Mircea for the valuable comments and discussions. We also thank Mila (`mila.quebec`) and its IDT team for providing and supporting the computing resources used in this work. Ekaterina Lobacheva is supported by IVADO and the Canada First Research Excellence Fund. Sarath Chandar is supported by the Canada CIFAR AI Chairs program, the Canada Research Chair in Lifelong Machine Learning, and the NSERC Discovery Grant.

REFERENCES

Jordan Ash and Ryan P Adams. On Warm-Starting Neural Network Training. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3884–3894. Curran Associates, Inc., 2020.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, August 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07711-7.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1587–1596. PMLR, July 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. doi: 10.1073/pnas.1611835114.

A. Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Hojoon Lee, Hyeonseo Cho, Hyunseung Kim, Donghu Kim, Dugki Min, Jaegul Choo, and Clare Lyle. Slow and Steady Wins the Race: Maintaining Plasticity with Hare and Tortoise Networks, June 2024.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical Normalization for Scalable Deep Reinforcement Learning. In *Forty-Second International Conference on Machine Learning*, June 2025.

Jarek Liesen, Chris Lu, and Robert Lange. Rejax, 2024.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks, 2024.

Guozheng Ma, Lu Li, Sen Zhang, Zixuan Liu, Zhen Wang, Yixin Chen, Li Shen, Xueqian Wang, and Dacheng Tao. Revisiting Plasticity in Visual Reinforcement Learning: Data, Modules and Training Stages. In *The Twelfth International Conference on Learning Representations*, October 2023.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The Primacy Bias in Deep Reinforcement Learning. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 16828–16847. PMLR, June 2022.

Darshan Patil, Pranshu Malviya, Maryam Hashemzadeh, and Sarath Chandar. Experimental Design for Nonstationary Optimization. October 2024.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks, October 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual Learning of Large Language Models: A Comprehensive Survey. *ACM Comput. Surv.*, 58(5):120:1–120:42, November 2025. ISSN 0360-0300. doi: 10.1145/3735633.

Shagun Sodhani, Mojtaba Faramarzi, Sanket Vaibhav Mehta, Pranshu Malviya, Mohamed Abdelsalam, Janarthanan Janarthanan, and Sarath Chandar. An Introduction to Lifelong Supervised Learning, July 2022.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The Dormant Neuron Phenomenon in Deep Reinforcement Learning, June 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, October 2018. ISBN 978-0-262-03924-6.

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

Puja Trivedi, Danai Koutra, and Jayaraman J. Thiagarajan. A Closer Look at Model Adaptation using Feature Distortion and Simplicity Bias. In *The Eleventh International Conference on Learning Representations*, September 2022.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. MuJoCo Playground: An open-source framework for GPU-accelerated robot learning and sim-to-real transfer., 2025.

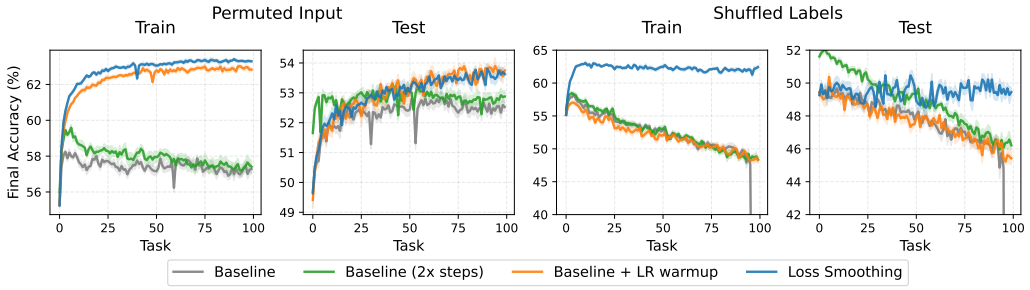


Figure 4: Ablation in supervised settings: comparison to baseline variants using learning rate warm-up or increased data per optimization step.

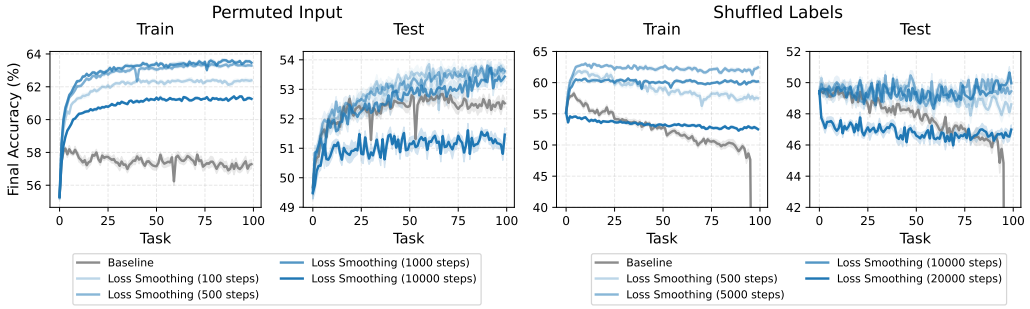


Figure 5: Ablation in supervised settings: effect of the number of loss smoothing steps on model performance.

A Supervised Learning Details

A.1 Experimental Details

We use a simple 3-layer MLP: input layer ($3072 \to 100$) $\to$ ReLU $\to$ hidden layer ($100 \to 100$) $\to$ ReLU $\to$ output layer ($100 \to 10$). The model is trained with SGD, batch size 256, for 20k steps per task. We run all experiments with 10 different random seeds and show average results with standard deviation in all the plots.

We ran baseline, loss smoothing, and EWC experiments across several learning rates and weight decay values, and found that a learning rate of 0.01 and weight decay of 0.001 were optimal for all methods. These values are used in all experiments. For loss smoothing, we additionally search for the optimal number of smoothing steps from $[100, 500, 1000, 5000, 10000, 20000]$, finding it to be 500 for permuted input and 5000 for shuffled labels. A similar search for EWC yields optimal regularization steps of 500 for permuted input and 100 for shuffled labels.

A.2 Ablations

The positive effect of loss smoothing could potentially arise from using a learning rate warm-up at the start of each task or from having more data per optimization step. In Figure 4, we compare to the baseline with these modifications. For the baseline with learning rate warm-up, we use the same schedule for the current-batch loss coefficient as in loss smoothing, but set the replay term to zero. Warm-up helps in the permuted-input setting, achieving results comparable to loss smoothing, but has little effect for shuffled labels. For the baseline with more data, we double the batch size to match the data size used per optimization step in loss smoothing. While this slightly improves optimization, it does not replicate the benefits of loss smoothing.

In Figures 5 and 6, we analyze the effects of two main hyperparameters of loss smoothing: the number of smoothing steps τ and the amount of replay data. For the number of steps, there exists an optimal value that depends on the task (here, 500 for permuted input and 5000 for shuffled

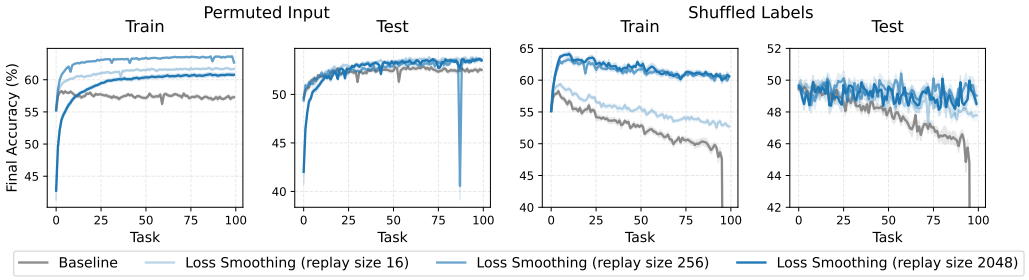
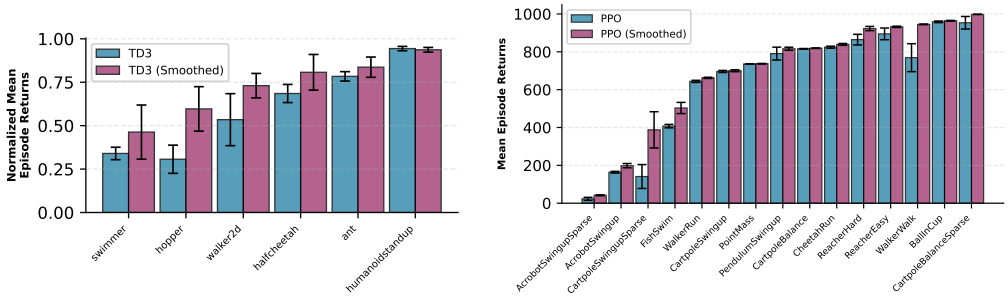


Figure 6: Ablation in supervised settings: effect of the amount of replay data on model performance.



(a) TD3 episode returns on Brax environments, normalized by the maximum return achieved across any seed of any agent.

(b) PPO episode returns on DeepMind Control Suite environments from Mujoco Playground. Maximum return for any environment is 1000.

Figure 7: TD3 and PPO final performance on each task.

labels). Fewer steps reduce the positive effect of loss smoothing, while too many steps degrade performance by overemphasizing older data. For the amount of replay data, a batch of 256 examples is generally sufficient. Using fewer examples can hurt performance, while using more does not provide additional benefits.

B Reinforcement Learning Details

B.1 Additional Results

B.2 Experimental Details

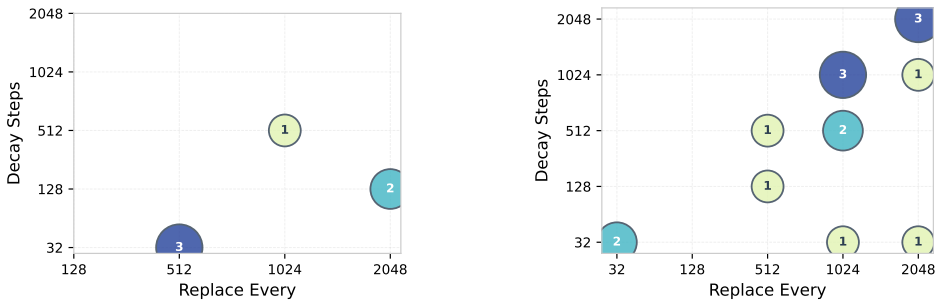
Our code is based on the rejax codebase (Liesen et al., 2024).

TD3 For the base algorithm, we take the hyperparameters present in the rejax codebase, as they have been tuned for good performance on these environments. We then apply loss smoothing to the critic loss, tuning the τ and task length hyperparameters via grid search. Our policy and critic networks are 2 layer MLPs with 256 hidden units and tanh activations. We train for 5 million environment steps on each task. In Figure 8a, we present the frequency of each loss smoothing hyperparameter configuration that we select across environments.

PPO We train for 50 million environment steps, and use the SimbaV2 architecture (Lee et al., 2025) as our base policy and value function network. We tune the agent hyperparameters in stages, first tuning the base PPO hyperparameters such as learning rate and batch size, and then tuning the loss smoothing hyperparameters τ and n. In Table 1, we present the base PPO hyperparameters that we use for all experiments, and in Figure 8b, we present the frequency of each loss smoothing hyperparameter configuration that we select across environments. We see that there are several configurations that are selected quite frequently.

Hyperparameter	Value
Max Gradient Norm	1.0
Total Timesteps	50,000,000
Evaluation Frequency	512,000
Discount Factor (γ)	0.99
GAE Lambda	0.95
Clipping Epsilon	0.2
Entropy Coefficient	0.01
Value Function Coefficient	0.5
Normalize Observations	true
Number of Environments	2048
Number of Steps	10
Learning Rate	0.0001
Number of Epochs	1
Number of Minibatches	32
Entropy Schedule	linear
Starting Entropy Coefficient	1.0

Table 1: PPO base hyperparameters



(a) TD3 Smoothed Hyperparameters Selected

(b) PPO Smoothed Hyperparameters Selected

Figure 8: TD3 and PPO smoothed hyperparameters. Decay steps refers to the number of steps τ over which the loss is smoothed, while Replace Every refers to the number of steps n after which the replay is updated with a new batch.