LOSS SMOOTHING FOR STABLE ADAPTATION UNDER DISTRIBUTION SHIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Over their life cycles, neural networks are often adapted under distribution shift. Standard adaptation methods typically optimize the target objective directly, inducing an abrupt change from the source training objective. This abrupt change can hurt optimization or distort learned representations, including features that may still be useful for the new task. We investigate how softening this transition can improve adaptation. Specifically, we formulate *loss smoothing* as a simple approach that linearly interpolates between the source and target objectives. We show that retaining the source supervision can improve target adaptation even when preserving source-task performance is not a goal. A controlled toy study shows how smoothing sustains learning of features shared between source and target tasks. In pretrained vision and language models, smoothing limits the loss of source information during adaptation, with retention advantages that persist after source supervision ends. Across a broad variety of adaptation settings, including pretrained vision model adaptation, continual supervised learning, fine-tuning "overtrained" large language models (LLMs), and offline-to-online reinforcement learning, loss smoothing improves performance, suggesting that smoother objective transitions are a broadly useful tool for model adaptation.

1 INTRODUCTION

Modern neural networks are rarely trained once and then left unchanged: the training procedure of almost every model involves some nonstationarity. The dominant paradigm, whether in image classification, large language model (LLM) fine-tuning, or reinforcement learning (RL), is to start from a pretrained model and adapt it to the target application. In each case, adaptation begins from a model that already contains diverse useful features. This model is then trained on a new *target* task, which may differ from the *source* task in its inputs, its learning signal, or even its objective.

The simplest, and often default, approach is to directly fine-tune on the target task, but this can cause not only disruption to the knowledge stored in the pretrained network (Kumar et al., 2022), but also suboptimal adaptation to the target task (Springer et al., 2025). We argue that this is often not because the pretrained knowledge is of little use for the target task, but because the transition from the source to the target task is poorly chosen. After a hard switch to the target objective, the model has to reuse existing features and adapt or learn new ones at the same time, and these processes can interfere. As a result, the model can end up ignoring some of the useful structure (Pezeshki et al., 2021).

A natural way to address this is to make the transition gradual: keep the source objective for a while and fade it out as the model adapts. Versions of this idea have been used before, such as annealing the source loss or the share of source data during fine-tuning (Chronopoulou et al., 2019; He et al., 2021), or mixing source data into target training (Rolnick et al., 2019; Ouyang et al., 2022; Kotha & Liang, 2026; Ball et al., 2023). These were proposed separately for each setting, often to reduce forgetting, although recent work shows that replay can also improve the target task (Kotha & Liang, 2026). We study this idea in a general form, which we call *loss smoothing*: we optimize a convex interpolation of the source and target objectives and anneal the interpolation coefficient until ordinary target training is recovered. This only requires an estimate of the source objective, obtained through replay data, stored targets, or the source training data, and applies whether the tasks differ in their inputs, their learning signals, or their loss functions. We demonstrate its wide applicability: the same recipe can

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

improve adaptation to the target task across supervised continual learning, vision and language model fine-tuning, and offline-to-online RL, even when retaining the source task is not a goal.

Our analysis examines how a gradual transition changes the use of pretrained knowledge during adaptation. In a controlled toy setting, a hard switch allows rapid learning of a conflicting feature to suppress further learning of a feature shared by both tasks. Smoothing slows the conflicting part, allowing the model to continue learning the shared feature. What the model learns during this transition persists after smoothing ends. In larger models, we find that improved target performance accompanies greater retention of source features, and that the retention advantage survives subsequent training on the target objective alone. These findings suggest that continued source supervision can guide how pretrained knowledge is used during adaptation. We test this interpretation by comparing smoothing with methods that preserve the pretrained model's weights or predictions, prepare the target head, or reduce the target-loss coefficient. These controls do not reproduce its benefits. Together, the results show how a temporary change to the transition can have lasting effects on what the model learns and retains.

Overall, our contributions are:

- **Target adaptation**: A general loss-smoothing recipe evaluated across four adaptation settings, demonstrating target-performance gains even when source retention is not the goal.
- **What smoothing changes**: A controlled analysis showing that smoothing biases learning towards shared features, and that this bias retention persists after smoothing ends.
- **The role of source supervision**: Evidence that the benefits come from the source signal itself, and not from simpler alternatives.

2 Related Work

Adaptation under distribution shift. Fine-tuning from a pretrained model is a standard approach to adaptation, but several lines of work show that the resulting update can be brittle under distribution shift. Fine-tuning can distort pretrained features and hurt out-of-distribution performance (Kumar et al., 2022), and adaptation can depend strongly on which parts of the network are allowed to change (Lee et al., 2023; Tomihari & Sato, 2024; Trivedi et al., 2023; Chronopoulou et al., 2019). Other works have also explored the idea of softening the transition between distribution shifts (generally in the context of mitigating forgetting, but recently also to improve adaptation) by incorporating target data in the pretraining phase (Feng et al., 2026) or replaying pretraining data in the fine-tuning phase (Kotha & Liang, 2026; Ouyang et al., 2022). Gradual domain adaptation studies learning through creating intermediate distributions between a source and target domain (Kumar et al., 2020). Liu & Mou (2026) also study smoothing task transition boundaries, but rely on distribution or data specific heuristics rather than a general interpolation of the source and target objectives. Other works explore gradually annealing away source supervision when fine-tuning language models, Chronopoulou et al. (2019) by annealing the loss coefficient for LSTM models and He et al. (2021) by annealing the pretraining-fine-tuning data ratio. Building on prior approaches that anneal source supervision during fine-tuning, we develop a unified formulation across changes in data distributions and objective functions, and study when and how this transition improves target adaptation.

Continual Learning. Continual learning studies how models adapt across sequences of tasks without catastrophic forgetting (Sodhani et al., 2022). Classical approaches preserve previous knowledge through parameter regularization (Kirkpatrick et al., 2017), replay (Rolnick et al., 2019), or distillation from an earlier model (Li & Hoiem, 2018). Dark experience replay further stores previous model outputs and replays them as targets during later tasks (Buzzega et al., 2020), making it especially close to settings where loss smoothing estimates an old objective from stored targets. Recent work has also emphasized transient failures during task changes, including the stability gap (Harun & Kanan, 2024).

Learning dynamics The phenomena driving the behavior of loss smoothing are very related to behaviors seen in neural network training in other settings. For example, it has been shown that effects from early learning persist through training (Achille et al., 2018; Ash & Adams, 2020; Nikishin et al., 2022). Another related effect is the notion of gradient starvation (Pezeshki et al., 2021), which happens when the loss is minimized with only a subset of predictive features, preventing the network from learning other potentially useful features.

Offline-to-online reinforcement learning. Offline-to-online RL studies how to initialize an agent from a fixed dataset and then improve it through online interaction. Offline RL methods such as TD3+BC (Fujimoto & Gu, 2021; Tarasov et al., 2023) and SPOT (Wu et al., 2022) regularize policy learning to avoid unsupported actions, while offline-to-online methods decide how much offline data or conservatism should remain during interaction. AWAC uses offline data to accelerate online RL (Nair et al., 2021), Cal-QL calibrates conservative offline pretraining for efficient online fine-tuning (Nakamoto et al., 2023), and RLPD mixes offline and online data during online training (Ball et al., 2023). Other offline-to-online methods modify the pretraining or fine-tuning procedure itself (Shin et al., 2025). These methods address the same source-to-target tension that motivates our work: remaining too close to the offline policy can limit improvement, while switching too abruptly to unconstrained online RL can destabilize learning. Our contribution is orthogonal to these algorithmic choices: we smooth the transition between the offline and online objectives, and in settings where the loss family changes, we also smooth the objective scale.

3 Loss Smoothing

We consider a model f_θ with parameters θ trained on some data source $\mathcal{D}$. This data source can be a supervised learning dataset, a pretraining corpus, or even (in the case RL bootstrapping) a combination of a RL environment and the model itself. A training objective can then be characterized by a combination of a data source and a loss function ℓ: $J(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\ell(\theta; x)\right]$. Thus, adaptation involves taking a model trained on some source objective $J_s(\theta) = \mathbb{E}_{x\sim\mathcal{D}_s}\left[\ell_s(\theta; x)\right]$, and optimizing it on a target objective $J_t(\theta) = \mathbb{E}_{x\sim\mathcal{D}_t}\left[\ell_t(\theta; x)\right]$.

This formulation is broad enough that it encompasses most of the distribution shifts encountered in neural network training. Continual supervised learning generally involves tasks with changing data sources $\mathcal{D}$. Moving from pretraining to fine-tuning involves a change in the data source $\mathcal{D}$ and sometimes a change in the loss function ℓ. Offline-to-Online RL involves both a change in $\mathcal{D}$ (offline dataset to online environment) and loss function (offline RL loss functions are generally different from online RL loss functions).

Standard fine-tuning performs an abrupt switch discarding the source objective once target training begins. Loss smoothing instead replaces this discontinuous switch with a path of intermediate objectives:

$$J_\alpha(\theta) = (1-\alpha)J_s(\theta) + \alpha J_t(\theta), \qquad \alpha \in [0, 1].$$

Depending on the schedule used for α you can get different types of loss smoothing. The method that we refer to as loss smoothing in the rest of the experiments linearly increases α over τ steps of smoothing according to a schedule $\alpha(k) = \min(k/\tau,\, 1)$. Another schedule involves just setting $\alpha(k) = c$, where c is some constant over the smoothing period. In practice, this is simply using a constant replay term throughout the smoothing period, which we argue is another instantiation of loss smoothing. For temporary replay, we set $\alpha(k) = c$ for $k < \tau$ and $\alpha(k) = 1$ thereafter. Persistent replay maintains the constant mixture throughout adaptation.

Loss smoothing is flexible in that it makes no assumptions on what the source training phase was or how it relates to the current phase. The shifts we mention can happen independently and even all at once. The intended role of smoothing is to improve the beginning of adaptation by acting as a mechanism to implicitly upweight and speed up learning of task shared components and slow down the learning of task conflicting components. In our experiments, we estimate the previous objective using replay buffers (offline-to-online RL), direct access to the previous data (LLM fine-tuning and DomainNet ViT adaptation), or a single stored batch of past data (continual supervised learning).

4 Does loss smoothing improve adaptation?

We now examine the effect of loss smoothing in several realistic adaptation settings. Table 1 summarizes the five setups: (1) continual supervised adaptation on CIFAR-10 with shuffled labels and (2) permuted inputs; (3) adapting a pretrained vision transformer to DomainNet; (4) fine-tuning an LLM on instruction tuning datasets; (5) Offline-to-online RL. For more details on the experimental setups, see Appendix A.

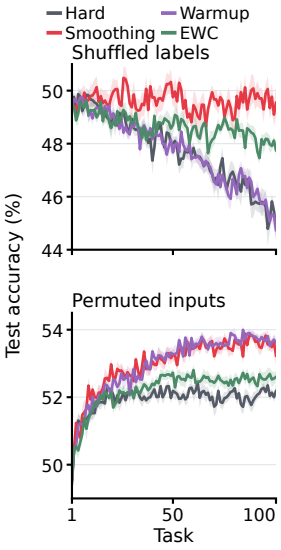


Figure 2: **Continual learning.** CIFAR-10 test accuracy at the end of each task. Curves show means and shaded bands show SEM over ten seeds.

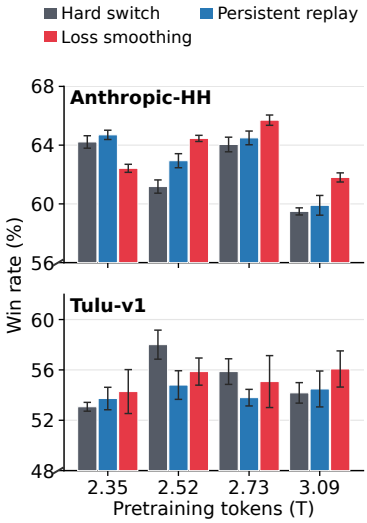


Figure 3: **OLMo-1B.** Win rate after instruction tuning at the final four pretrained checkpoints on Anthropic-HH and Tulu-v1 datasets. Bars show means and SEM over three seeds on 483 evaluation prompts.

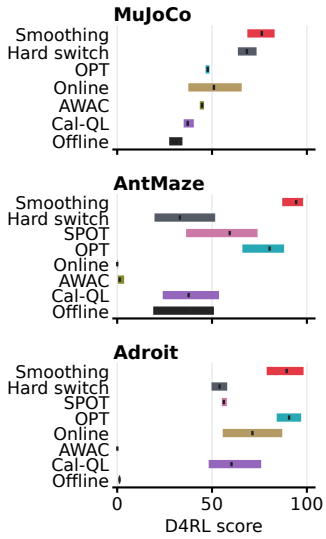


Figure 4: **Offline-to-online RL.** D4RL suite IQMs after 500k environment steps, with 95% bootstrap intervals from five seeds per task. Offline denotes the pretrained policy. Smoothing is the best or tied for best across all 3 suites.

Continual Supervised Adaptation For this setup, we construct sequences of 100 tasks by transforming the CIFAR-10 dataset (Krizhevsky, 2009), sampling a new input permutation or label remapping at each task. The network is trained for 20k updates per task, and test accuracy is reported over 10 random seeds. In Figure 2, we see that for both shuffled label and permuted inputs, loss smoothing results in the best test accuracy across the task sequence. While it is matched by doing a learning rate warmup for permuted inputs, it clearly outperforms it on the shuffled labels shift. Loss smoothing also clearly outperforms a tuned Elastic Weight Consolidation (EWC) baseline, implying that its performance is not just due to higher regularization.

Pre-trained Vision Transformer Adaptation We next move to adapting a pretrained vision model, fine-tuning separately on the six different domains of DomainNet (Peng et al., 2019). Each domain contains the same 345 classes in a different style. We start from an ImageNet trained vision transformer (ViT) and fine-tune for 60 epochs on each domain. Figure 1 shows target-test accuracy, aggregated as an interquartile mean (IQM) across domains and seeds. We compare hard switching with linear loss smoothing and constant 50% replay, varying the initial source-supervision window over 5, 10, 25, 50, and 100% of training. After this window, both methods use only the target objective.

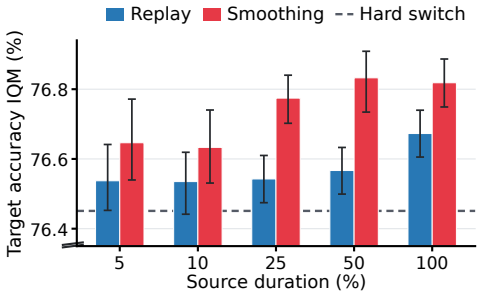


Figure 1: **DomainNet.** Target-test accuracy IQM over six domains and three seeds, with 95% bootstrap intervals. Duration is the fraction of training with source supervision. Across all durations, loss smoothing outperforms replay, and both source supervision methods outperform hard-switch on target adaptation.

Both forms of source supervision improve target performance, with loss smoothing outperforming replay at each matched duration. Replay benefits from longer source supervision throughout the tested range, whereas smoothing reaches its highest target accuracy with a 50% window. The corresponding source-retention results are reported in Table 6.

Large Language Model Finetuning Our next setup considers the overtrained language model setting of Springer et al. (2025). Their results show that longer pretraining can improve the pretrained model while making the resulting checkpoint harder to adapt, a phenomenon they call *catastrophic overtraining*. They attribute this degradation to progressive sensitivity: as pretraining continues, the model's parameters become increasingly sensitive to downstream updates. We take checkpoints along the pretraining trajectory of OlMo1B (Groeneveld et al., 2024) and fine-tune them on downstream instruction tuning datasets Anthropic-HH Bai et al. (2022) and Tulu-v1 (Wang et al., 2023) (see Appendix A.4 for specific details on model checkpoints and data curation). We then evaluate each fine-tuned checkpoint to a reference checkpoint AlpacaEval-style with a Llama-3 70B judge model. We run 3 seeds per pretrained checkpoint.

In Figure 3, hard-switch performance declines between the final two checkpoints on both datasets. Source-supervised methods also decline over this interval on Anthropic-HH. On Tulu-v1, performance with source supervision generally improves with longer pretraining, whereas hard-switch performance declines after peaking at the 2.515T checkpoint (see Appendix C.3 for the complete schedule comparison). At the final checkpoint, smoothing improves on hard switching by 2.3 percentage points on Anthropic-HH and 1.9 points on Tulu-v1.

Offline-to-Online RL Offline-to-online RL is a natural testbed for loss smoothing because adaptation can change both the data distribution and the optimization objective. An agent is first pretrained from a fixed offline dataset, and is then fine-tuned with direct environment interaction. Several prior studies have shown that fine-tuning from offline trained initializations can result in worse performance than training directly from scratch (Nakamoto et al., 2023; Shin et al., 2025). Policies learned from a fixed dataset can overfit to unsupported actions and suffer from biased value estimates, yet overly conservative offline training can limit improvement once interaction begins. The key challenge is to properly leverage imperfect offline knowledge while enabling efficient online exploration and adaptation without destabilizing performance.

We evaluate our loss smoothing approach on a variety of standard tasks from D4RL (Fu et al., 2020). We instantiate loss smoothing on two existing offline-to-online methods. For the MuJoCo locomotion tasks, the offline phase uses TD3+BC (Fujimoto & Gu, 2021) pretraining followed by online TD3 (Fujimoto et al., 2018) fine-tuning. For the AntMaze and Adroit manipulation tasks, we use SPOT (Wu et al., 2022) offline training followed by SPOT online training. We apply smoothing to both the actor and the critic. For the TD3/TD3-BC actor objective, since we are smoothing between two different objective functions that can have drastically different scale, we apply smoothing to both the objective and its scale. We provide further details in Appendix A.5.

Each agent is trained for 1M offline gradient steps followed by 500k online environment steps, using five seeds per environment. We compare against the offline pretrained policy (Offline), online RL from scratch (Online), abrupt fine-tuning from the pretrained agent (Hard-Switch), and offline-to-online baselines SPOT (Wu et al., 2022), AWAC (Nair et al., 2021), Cal-QL (Nakamoto et al., 2023), and OPT (Shin et al., 2025). We standardize the online interaction budget, evaluation protocol, offline datasets, and implementation pipeline across methods; stochastic-policy baselines use stochastic versions of the same TD3-style MLP profiles where the algorithm requires action log-probabilities.

We report D4RL normalized scores for individual tasks in Table 7, and aggregate performance with rliable (Agarwal et al., 2021) IQMs with 95% confidence intervals in Figure 4. Loss smoothing performs well across the benchmark rather than specializing to a single regime. It achieves the highest suite IQM on MuJoCo and AntMaze, and remains competitive on the Adroit tasks.

Takeaway

Even if we do not care about source task performance, loss smoothing using source supervision can improve downstream target performance across a variety of domains.

5 Analysis

5.1 How does loss smoothing change what the model learns?

Loss smoothing is intended to upweight the components shared between the source and target tasks and to slow down the conflicting ones. We make this concrete in a toy example in which the two can

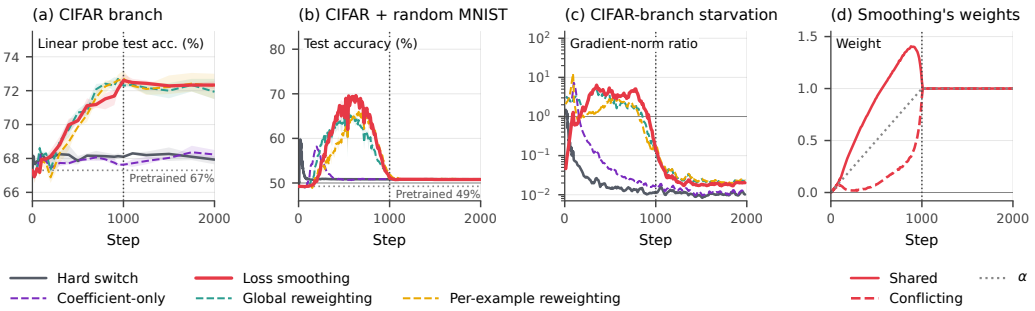


Figure 5: Toy example: CIFAR (shared) + MNIST (conflicting). (a) How well the model has learned the shared feature: linear-probe accuracy on the last hidden layer of the CIFAR branch. (b) Reliance on it: accuracy with the correct CIFAR image and a random MNIST image. (c) Starvation of the CIFAR branch: norm of the gradient each method applies to it, relative to the one it would get with the same weights but its own logits alone (below one: starved). (d) Smoothing's batch-averaged shared and conflicting weights, with α for reference. Dotted vertical lines mark the end of smoothing.

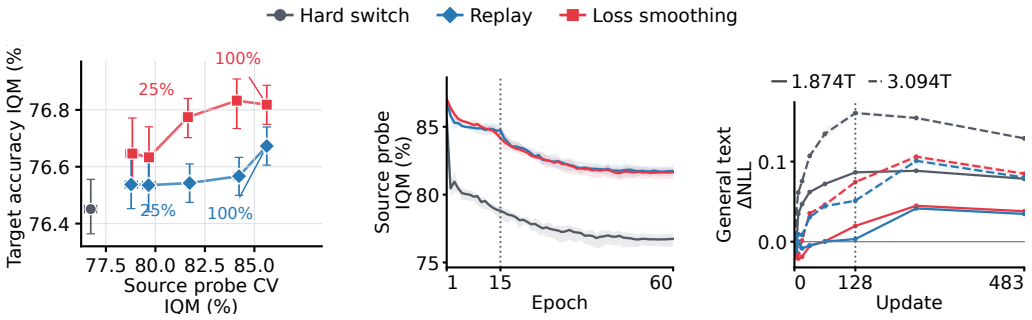
be cleanly separated, and show that the effect of smoothing on what the model learns comes from this reweighting. We consider a pretrain–fine-tune setting for binary image classification. Each input is a pair of a CIFAR image and an MNIST image, and the network has two branches: a separate MLP computes logits for each image, and the prediction is made from the sum of the two logits. The CIFAR image is *shared* between the tasks, keeping its mapping to the labels, while the MNIST image is *conflicting*, with its mapping swapped: (automobile, 0) and (truck, 1) are classes 0 and 1 during pretraining, and (automobile, 1) and (truck, 0) during fine-tuning, with MNIST images drawn on the fly during training. Due to simplicity bias Shah et al. (2020), training relies more on the simpler MNIST image. However, the CIFAR feature is the more stable cue: when both features can solve the target task equally well, a model that learns and uses the shared feature more is expected to generalize better. We therefore track how well the model has learned the shared feature, with a linear probe on the last hidden layer of the CIFAR branch, and how much it relies on it, as its test accuracy with the correct CIFAR image and a random MNIST image. Details are given in Appendix B.

The two branches let us compare how the hard switch and loss smoothing weight the shared and conflicting parts of the gradient on each example. Consider an example with CIFAR image x, paired with an MNIST image c under the target mapping, or with o under the source mapping. With two classes, the cross-entropy depends only on the logit margin towards the label, $S + C$, the sum of the two branches' margins, and its negative gradient is the error $e = 1 - p(y)$ times the margin gradients. With e_t and e_s the errors on the target pair (x, c) and the source pair (x, o), the two methods update the branches as

$$\begin{aligned} \text{hard switch} &= e_\mathrm{t}\,\nabla S(x) + e_\mathrm{t}\,\nabla C(c), \\ \text{loss smoothing} &\approx \lambda\, e_\mathrm{t}\,\nabla S(x) + \kappa\, e_\mathrm{t}\,\nabla C(c), \end{aligned} \qquad \text{with} \quad \begin{aligned} \lambda &= \alpha + (1-\alpha)\, e_\mathrm{s}/e_\mathrm{t}, \\ \kappa &= \alpha - (1-\alpha)\, e_\mathrm{s}/e_\mathrm{t}. \end{aligned} \tag{1}$$

The smoothed row is approximate in its conflicting part: the source term acts on a different MNIST image o, so the update reduces to this form only if $\nabla C(o) \approx -\nabla C(c)$. This holds almost exactly here, since the MNIST branch carries only the digit identity, whose mapping is swapped between the tasks, and it is what makes the branch fully conflicting. Otherwise, the branch would also carry structure other than the swapped label signal, which would be shared between the tasks: smoothing would then weight it as shared, whereas the approximation treats the whole branch as conflicting.

The hard switch and smoothing thus apply the same shared and conflicting parts of the gradient, only with different weights: 1:1 for the hard switch, and λ:κ for smoothing, which depend on α and on each example's progress on both tasks. We first compare the two in Figure 5. After pretraining, the model has learned the CIFAR feature to some extent, but relies almost entirely on the MNIST image. Under the hard switch, the MNIST branch learns the target mapping within the first 50 steps, and the quickly falling target error leads to gradient starvation (Pezeshki et al., 2021): the CIFAR branch receives only a small fraction of the gradient it would get on its own, and the shared feature stays almost at its pretrained level. Smoothing's weights combine a general schedule, α, which scales the whole gradient, with a reweighting of the two parts around it. The reweighting keeps the conflicting



(a) DomainNet adaptation (b) DomainNet retention (c) OLMo-1B retention

Figure 6: Target adaptation and source retention. **(a)** DomainNet target-test IQM versus refitted ImageNet source-probe CV IQM, with 95% bootstrap intervals. Points vary the source-supervision window over 5, 10, 25, 50, and 100% of training. **(b,c)** Source retention during and after source supervision; dotted lines mark the cutoffs. Bands show 95% IQM bootstrap intervals for DomainNet and seed SD for OLMo.

weight close to zero for most of the smoothing period, which slows down the MNIST branch. As a result, the CIFAR branch is not starved: the model keeps learning the shared feature throughout smoothing and starts to rely on it. Towards the end of smoothing, the conflicting weight grows, the MNIST branch learns the target mapping, and the reliance on the CIFAR image goes down again, but what the CIFAR branch has learned persists after smoothing ends, consistent with Section 5.2.

To check which part of smoothing is responsible, we split the change from 1:1 to λ:κ into stages: (i) *coefficient-only*, with weights α:α, keeps only smoothing's general schedule; (ii) *global reweighting* uses one pair of weights $\bar{\lambda}$:$\bar{\kappa}$ for all examples, obtained by averaging over the batch at each step; (iii) *per-example reweighting* uses the example-specific λ:κ, as in the approximate smoothed update in Eq. (1). Smoothing's general schedule alone does not help: coefficient-only follows the hard switch with a delay. In contrast, both global and per-example reweighting keep smoothing's benefits for the shared feature. Global reweighting is already enough here, since all examples share the same conflict and progress similarly; when examples differ more in how much they conflict and how fast they are learned, the example-specific weights may also play a role.

Overall, this toy example shows that the effect of smoothing on what the model learns is driven by the reweighting of the shared and conflicting parts. Here, we could reproduce it directly, since the two parts are separated by construction. In practice, they cannot be disentangled, but smoothing performs the reweighting automatically, based on the model's behavior on the source and target data.

5.2 Shared feature retention during and after adaptation

The toy study suggests that smoothing preserves useful source features that would otherwise be lost during adaptation. We now examine whether a similar pattern appears in pretrained models. Does source-feature retention accompany improved target performance, and does that retention persist after source supervision ends?

The retention gap emerges during the smoothing period. In Figures 6b and 6c, hard switching causes a sharp decrease in source performance early in adaptation, visible as lower source-probe accuracy on DomainNet and higher general-text loss on OLMo. Smoothing limits this initial degradation, allowing more of the source model's learned features and behavior to survive adaptation. This retention accompanies successful target learning. Figure 6a shows that smoothing improves both target accuracy and source retention over hard switching, and achieves higher target accuracy than replay at comparable source retention.

We then follow what survives after smoothing ends, using a 25% smoothing window for DomainNet and 128 steps for OLMo-1B. DomainNet source-probe accuracy declines only slightly after source supervision is removed. OLMo general-text loss initially rises and then partially recovers, ending slightly above its value at the cutoff. Both retain an advantage over hard switching through the end of target-only training. For DomainNet, the 25% window recovers more than half the retention benefit

of smoothing throughout training. As in the toy example, the advantage established during smoothing persists after the model returns to the target objective alone.

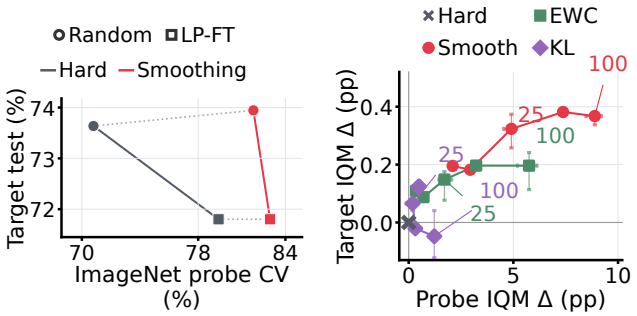
Takeaway

Early source supervision changes the subsequent course of adaptation, leaving a retention advantage that persists through target-only training.

5.3 Can preserving the initialization recover the gains?

The toy study illustrates how source supervision can guide feature learning during adaptation. We next ask whether standard methods for preserving the pretrained model can give comparable benefits: regularization toward the initialization and preparation of the target head.

The first approach we examine is to simply regularize adaptation towards the pretrained model. We replace the source loss in loss smoothing with either KL regularization toward the pretrained model's predictions or EWC, while preserving the target-loss coefficient and intervention schedule. Both controls incorporate source knowledge: KL uses the pretrained model's ImageNet predictions on target examples, and EWC uses a Fisher estimate from source data. Neither reproduces smoothing's source-target trade-off (Figure 7b), indicating that directly optimizing the source objective provides benefits beyond these standard ways of preserving the initialization. OLMo continuations from the same model and optimizer state further show that source supervision improves retention beyond simply reducing the target-loss coefficient (Appendix D.2).



(a) LP-FT (b) Source penalties

Figure 7: Source retention and target adaptation with (a) random versus LP-FT target heads, (b) smoothing versus KL and EWC penalties. Panel b shows changes from hard switch; curves vary intervention duration.

The second approach we test is whether we can achieve improvements similar to loss smoothing by aligning the model to the target distribution and preventing feature distortion, as Kumar et al. (2022) do by replacing the random head at initialization with a linear probe head (i.e. training the head on the frozen backbone before full fine-tuning). We replace the random head with the linear probe head at initialization for both hard switch and loss smoothing. In Figure 7a, we see that for both methods, using the linear probe head decreases the target performance. For hard switch, using a linear probe head could be worthwhile as it results in a big improvement in source retention, but for loss smoothing, the improvement in source retention is negligible and results in a significant degradation in target accuracy. Preparing the target head improves retention under hard switching, but does not reproduce smoothing's target gains. This suggests that smoothing does more than protect against the initial mismatch introduced by a random head.

Takeaway

Direct source supervision improves the adaptation-retention trade-off beyond the initialization-based penalties. Its performance also outpaces both head preparation and target-loss attenuation.

5.4 Model Sensitivity

Figure 3 demonstrates that loss smoothing may be better at fine-tuning "overtrained" checkpoints than the standard recipes. Springer et al. (2025) claim that the brittleness of these late checkpoints is due to them being more sensitive to updates than earlier checkpoints. For the sensitivity and retention analyses, we use a controlled Tulu-v1 experiment with a common learning rate of 5×10^{-5} across methods, rather than selecting a separate rate for each method as in the benchmark. We look at how smoothing affects the sensitivity of an early checkpoint (1.874 trillion pretraining tokens) and a late checkpoint (3.094 trillion pretraining tokens) in Figure 8. First, we measure how much the source

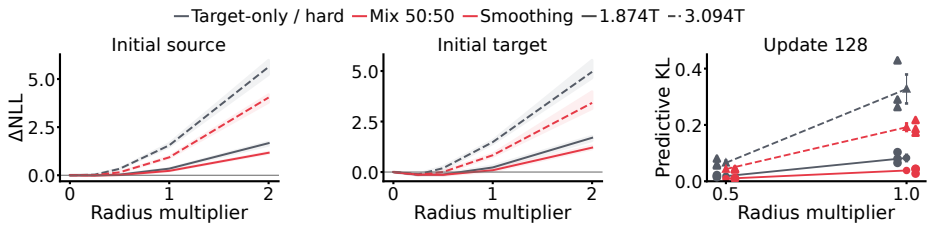


Figure 8: OLMo sensitivity at initialization (left and middle) and at the end of source mixing (right). We compare loss changes along target-only and mixed update directions, and prediction changes under a target-only perturbation. We compare equally sized parameter perturbations along target-only and mixed source-target AdamW update directions constructed with fresh optimizer moments. The left and middle panels measure loss changes at initialization; the right measures prediction changes after 128 adaptation updates. Further details in Appendix G.

loss (left) and target loss (middle) change when moving in the target loss and mixed source-target loss AdamW update directions (at equalized norms). We see that the late checkpoint is indeed more sensitive than the early checkpoint, with its loss increasing faster as we perturb it further. However, the mixed gradient direction is significantly less sensitive than the target only direction, especially on the late checkpoint. We also compare the state of the model at the end of smoothing (step 128) to the hard switch model at the equivalent step. We then compare the effect of taking a (optimizer adjusted) step based on only the target loss. We find a similar trend as before, where the late checkpoint perturbs results more than the early checkpoint, but loss smoothing results in a model that is less sensitive compared to a hard switch. Together, the adaptation gains at late pretrained checkpoints (Figure 3) and the reduced sensitivity in the controlled fixed-learning-rate study (Figure 8) suggest that smoothing can help mitigate overtraining-related brittleness.

Takeaway

Loss smoothing can significantly mitigate the sensitivity that emerges from "overtraining" models.

6 Discussion

Across a broad variety of distribution shifts including continual supervised learning, pretrained vision adaptation, offline-to-online RL, and LLM fine-tuning, we find that target task adaptation can be improved by softening the transition between the source task and target task. Specifically, we use temporary supervision from the source distribution, in the form of loss smoothing, to gradually shift the objective function from the source to the target. The toy setting identifies one way source supervision can improve adaptation: sustaining learning of shared features that target-only training otherwise neglects. In larger models, smoothing improves target performance while retaining more source information, consistent with the reuse of useful pretrained features. The retention advantage persists after source supervision ends. Comparisons with KL and EWC penalties, target-head preparation, and target-loss attenuation show that continued source supervision provides benefits these alternatives do not reproduce. Finally, we find that loss smoothing can also help mitigate the sensitivity that is encountered when fine-tuning "overtrained" models, resulting in more stable adaptation.

Limitations and Future Work The method requires access to, or an estimate of, the source objective, which may involve replay data, stored targets, or continued access to previous training data. It also introduces schedule choices and hyperparameters. Future work should explore finding automatic smoothing schedules based on the geometry of the loss landscape. Compared to normal training, loss smoothing also requires more compute at the start of adaptation through an additional forward and backward pass on source data. The source and target gradients contribute to one optimizer update, approximately doubling the compute needed in the smoothing window.

AI USE STATEMENT

We used generative AI tools to propose and refine hypotheses, design and assess research methods and experiments, implement methods, and interpret results. We also used these tools for brainstorming research topics and identifying gaps, searching for information and relevant literature, summarizing and analyzing existing literature, suggesting experimental parameters, creating and editing code and research artifacts, preparing scientific figures, and drafting and editing manuscript text for readability.

We did not use generative AI to help develop theoretical models and conceptual frameworks, formulate mathematical claims, generate synthetic datasets, provide critical ingredients for mathematical proofs, write proofs, translate text, clean or reformat datasets, or conduct qualitative or thematic data analysis. We also did not use these tools to formulate survey or interview questions, transcribe research recordings, format references, propose the paper's title or keywords, or suggest its structure.

The authors take responsibility for the final content of this work, including all text, claims, code, figures, and other artifacts produced with AI assistance.

REPRODUCIBILITY STATEMENT

Section 3 defines the loss-smoothing objective and transition schedule. Appendix A documents the datasets and preprocessing, model initializations, training settings, and evaluation procedures used in our experiments. Further results, control specifications, and duration ablations are provided in Appendices C to E to support reproduction and comparison with the reported findings.

REFERENCES

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical Learning Periods in Deep Networks. In *International Conference on Learning Representations*, September 2018.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021.

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.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, April 2022.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 1577–1594. PMLR, July 2023.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and SIMONE CALDERARA. Dark Experience for General Continual Learning: A Strong, Simple Baseline. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930. Curran Associates, Inc., 2020.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2089–2095, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1213.

Lawrence Feng, Gaurav Rohit Ghosal, Jacob Mitchell Springer, Ziqian Zhong, and Aditi Raghunathan. Early Data Exposure Improves Robustness to Subsequent Fine-Tuning. In *Third Conference on Language Modeling*, August 2026.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

Scott Fujimoto and Shixiang Shane Gu. A Minimalist Approach to Offline Reinforcement Learning, December 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.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. OLMo: Accelerating the Science of Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15789–15809, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.841.

Md Yousuf Harun and Christopher Kanan. Overcoming the Stability Gap in Continual Learning, September 2024.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. Analyzing the Forgetting Problem in Pretrain-Finetuning of Open-domain Dialogue Response Models. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1121–1133, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.95.

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.

Suhas Kotha and Percy Liang. Replaying pre-training data improves fine-tuning, March 2026.

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

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding Self-Training for Gradual Domain Adaptation. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5468–5479. PMLR, November 2020.

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.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical Fine-Tuning Improves Adaptation to Distribution Shifts, June 2023.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models, May 2023.

Zhizhong Li and Derek Hoiem. Learning without Forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, December 2018. ISSN 1939-3539. doi: 10.1109/tpami.2017.2773081.

Tianhui Liu and Lili Mou. Do Neural Networks Lose Plasticity in a Gradually Changing World?, February 2026.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets, April 2021.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated Offline RL Pre-Training for Efficient Online Fine-Tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, December 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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1406–1415, 2019.

Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, pp. 1256–1272, Red Hook, NY, USA, December 2021. Curran Associates Inc. ISBN 978-1-7138-4539-3.

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.

Harshay Shah, View Profile, Kaustav Tamuly, View Profile, Aditi Raghunathan, View Profile, Prateek Jain, View Profile, Praneeth Netrapalli, and View Profile. The Pitfalls of Simplicity Bias in Neural Networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Guide Proceedings, pp. 9573–9585. December 2020. ISBN 978-1-7138-2954-6. doi: 10.5555/3495724.3496527.

Yongjae Shin, Jeonghye Kim, Whiyoung Jung, Sunghoon Hong, Deunsol Yoon, Youngsoo Jang, Geon-Hyeong Kim, Jongseong Chae, Youngchul Sung, Kanghoon Lee, and Woohyung Lim. Online Pre-Training for Offline-to-Online Reinforcement Learning. In *Forty-Second International Conference on Machine Learning*, June 2025.

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

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained Language Models Are Harder to Fine-Tune. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 56719–56789. PMLR, October 2025.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the Minimalist Approach to Offline Reinforcement Learning, October 2023.

Akiyoshi Tomihari and Issei Sato. Understanding Linear Probing then Fine-tuning Language Models from NTK Perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

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*, 2023.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah Smith, Iz Beltagy, and Hannaneh Hajishirzi. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74764–74786. Curran Associates, Inc., 2023.

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported Policy Optimization for Offline Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35: 31278–31291, December 2022.

A Experimental Details and Reproducibility

Our experiments use source supervision in several forms: a stored batch from the previous classification task, replayed pretraining examples, or a separate offline reinforcement-learning loss. This appendix describes those choices and the evaluation protocols. The broad adaptation benchmarks and the controlled diagnostic studies answer different questions, so we keep their training budgets and selection procedures separate.

A.1 Schedules and reporting conventions

For a target coefficient α_k, the mixed objective is $(1 - \alpha_k)J_s + \alpha_k J_t$. Unless stated otherwise, linear smoothing ramps α_k from zero to one over an initial window and then uses only the target loss. Temporary replay holds both loss coefficients constant during a window and removes the source term afterward. Persistent replay keeps the mixture throughout adaptation. A coefficient-only control follows the target coefficient of smoothing while setting the source coefficient to zero. We distinguish this control from changing the optimizer's learning rate.

We report target outcomes separately from source retention. In DomainNet, the interquartile mean (IQM) trims 25% from each tail of the pooled domain–seed scores. Confidence intervals resample seeds jointly across methods, keeping the six domains fixed; the main paired comparisons use 50,000 percentile-bootstrap draws. They describe uncertainty over these seeds, not over an independently sampled population of domains. Other figures use seed standard errors or standard deviations, as specified in their captions. Repeated checkpoints, probe domains, and perturbations of one trained model are not additional independent training seeds.

A.2 Continual supervised adaptation

We train a multilayer perceptron with two width-100 ReLU hidden layers and a ten-class output on flattened CIFAR-10 images. Inputs are cast to floating point and divided by 255. Each sequence contains 100 tasks. A task applies either a new permutation of the input coordinates or a new mapping from the ten classes to output labels. The first task is trained normally; subsequent tasks introduce the distribution shift.

Learning-rate and EWC settings are selected by validation accuracy. All methods use SGD, batch size 256, learning rate 0.01, an L2 coefficient of 0.001, and 20,000 updates per task. We run ten model/data-order replicates per condition. Smoothing stores a batch of 256 examples from the preceding task and combines its loss with the current-task loss. Its source coefficient decays to zero over 5,000 updates for shuffled labels and 500 for permuted inputs. The learning-rate warmup comparator uses the corresponding transition length without the old-task loss. We evaluate the current task at the end of each training phase, using the current input transformation and label map.

The EWC comparator uses coefficient 10 for shuffled labels and 1 for permuted inputs, with penalty windows of 100 and 500 updates, respectively.

We evaluate smoothing windows of 100, 500, 1,000, 5,000, 10,000, and 20,000 updates in Appendix E.

A.3 Pretrained vision adaptation

We adapt a ViT-B/16 with 224×224 inputs separately to Clipart, Infograph, Painting, Quickdraw, Real, and Sketch from DomainNet. Each target head has 345 outputs. The source classifier is retained for source supervision and source-head evaluation, while a separate target head is trained for DomainNet. We use an ImageNet-pretrained initialization.

Each domain uses a class-stratified 10% validation split of the official training data; the official test split is reserved for target evaluation. The ImageNet initialization is timm's `vit_base_patch16_224.augreg2_in21k_ft_in1k` checkpoint. Training applies a random resized crop to 224×224 with bicubic interpolation and random horizontal flipping. Evaluation resizes the shorter side to 256 pixels and takes a central 224×224 crop. Inputs are normalized with mean and standard deviation $(0.5, 0.5, 0.5)$. Training lasts 60 epochs with batch size 256, AdamW, learning rate 2×10^{-5}, weight decay 0.1, a 5% learning-rate warmup, and cosine decay. Runs

Setting	Source → target	Shift instantiated
Continual Supervised Adaptation	Previous task → current task	Permuted inputs change the input distribution; shuffled labels change the label mapping, with cross-entropy loss fixed.
Offline-to-Online RL	Offline → online	Transitions, bootstrapped targets, and sometimes the actor objective all change, changing the data distribution, learning signals, and loss function.
LLM Fine-tuning	Pretraining → instruction tuning	Prompts, sequences, and target tokens narrow, changing inputs and learning signals while retaining a next-token loss form.
DomainNet ViT	ImageNet → DomainNet	The visual domain and labels change, while the ImageNet source objective is retained through replay or smoothing.

Table 1: How each experiment instantiates the source-to-target objective transition from Section 3.

use seeds 0, 1, and 2. We report the terminal epoch-60 model rather than selecting the best test checkpoint.

Smoothing and temporary replay use windows covering 5, 10, 25, 50, or 100% of training. Replay assigns weight 0.5 to each loss during its window. Both methods evaluate source and target losses on their respective examples before combining them. Each update draws a source batch of 256 examples with replacement, independently of the target batch. For LP-FT, the target head is precomputed on frozen source features and loaded before adaptation. All methods then fine-tune the backbone and target head for the full 60 epochs; head preparation is additional preprocessing, not part of that budget. The retained source head stays frozen. Head preparation fits an L2-regularized linear classifier to frozen 768-dimensional features. A class-stratified 20% subset of the adaptation training split selects the inverse penalty $C \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$ by accuracy; the head is then refitted on that training split. Fine-tuning starts with a fresh AdamW optimizer. The smoothing schedule starts at the beginning of these 60 adaptation epochs. The head preparation and KL/EWC comparisons are discussed in Appendix D.4.

Source-head validation accuracy evaluates the original classifier directly. Source-probe accuracy instead refits a linear classifier on frozen final-layer features from a fixed, class-stratified pool of 5,000 source-training images, using five-fold cross-validation. These measure different kinds of retention. Off-domain probes use the corresponding 5,000-image pools from each DomainNet style. Their construction and interpretation are described in Appendix F; they are not official-test transfer estimates.

A.4 OLMo INSTRUCTION TUNING

Models and data. The benchmark fine-tunes all parameters of OLMo-1B (Groeneveld et al., 2024), using its original tokenizer. We use four late checkpoints along one pretraining trajectory (Table 2). Anthropic-HH uses the chosen transcript from each preference pair (Bai et al., 2022); Tulu-v1 uses a deterministic 200,000-example subset selected by shuffling with seed zero (Wang et al., 2023). Messages are serialized as `role: content`, with blank lines between turns. Sequences are packed to length 1,024. The training sets contain 160,800 HH examples and 200,000 Tulu examples. The causal language-model loss scores tokens throughout each serialized conversation, including user messages, rather than masking to assistant responses alone.

Source replay uses a capped 500-million-token mixture of the Common Crawl, Falcon, StarCoder, C4, Reddit, and peS2o components, in proportion to their OLMo pretraining weights. This is an estimate of the source objective from a subset of pretraining data, not a replay of the complete pretraining corpus.

Optimization and benchmark selection. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, zero weight decay, and global gradient clipping at norm 1.0. The learning rate warms up for 20 updates and then follows cosine decay. Training uses bfloat16 computation, float32 parameters, and a global batch of 256 packed sequences per target batch. While source supervision is active,

Table 2: OLMo-1B pretrained checkpoints. Token counts identify pretraining progress; all checkpoints have the same model size.

Checkpoint	Pretraining tokens
`step561250-tokens2353B`	2.353T
`step600000-tokens2515B`	2.515T
`step650000-tokens2725B`	2.725T
`step738000-tokens3094B`	3.094T

each update also uses a separate source batch of 256 packed sequences; source and target losses are averaged separately before applying their mixing coefficients. HH training lasts 141 updates and Tulu training 483 updates. The smoothing windows are 16, 64, and 128 updates; the benchmark replay baseline uses a persistent 50:50 mixture. Each configuration has three data-order seeds.

The full benchmark compares learning rates $\{10^{-6}, 5\times10^{-6}, 10^{-5}, 5\times10^{-5}, 8\times10^{-5}, 10^{-4}, 2\times10^{-4}\}$ for each method and checkpoint. We partition the 805 AlpacaEval prompts into 322 validation prompts and 483 evaluation prompts, using one fixed split shared across all datasets, checkpoints, methods, learning rates, and training seeds. Prompt IDs order was randomized and then the first 322 form the validation set. For each dataset, checkpoint, and method, we select the rate with the highest mean validation win rate across the three seeds, breaking ties in favor of the smaller rate. Reported benchmark scores use only the 483 evaluation prompts. Error bars show seed variation conditional on the fixed split and selected rate, rather than uncertainty over prompt sampling or learning-rate selection. Appendix C.3 reports the results for all smoothing durations.

Behavioral evaluation. We generate responses to the 805 AlpacaEval prompts (Li et al., 2023), using greedy decoding, at most 1,024 new tokens, and the prompt format `user: <instruction>` followed by `assistant:`. A Llama-3-70B-Instruct judge compares each answer against a fixed reference within a target dataset. The benchmark reference is the hard-switch model fine-tuned from the 1.874T checkpoint with learning rate 10^{-5} and seed zero. Candidate/reference order is deterministically randomized. Win rate counts a tie as half a win. All benchmark comparisons use the same judging protocol.

Controlled adaptation study. The source-feedback and sensitivity experiments use Tulu at the 1.874T and 3.094T checkpoints, a fixed peak learning rate of 5×10^{-5}, and three paired seeds. Unlike the benchmark sweep, this comparison fixes the rate across methods. The four schedules in Table 3 share a 128-update intervention window and run for 483 updates. The temporary constant mixture matches the *discrete sum* of the linear source coefficients, which is 64.5. It therefore differs slightly from exactly 50:50 mixing. Matching this sum does not match learning-rate-weighted exposure or the optimizer trajectory.

Table 3: Loss coefficients in the controlled OLMo study, for internal update index $k = 0, \ldots, 127$. After this window every arm uses source coefficient zero and target coefficient one.

Method	Source coefficient	Target coefficient
Hard switch	0	1
Linear smoothing	$1 - k/128$	$k/128$
Temporary replay	$129/256$	$127/256$
Coefficient-only	0	$k/128$

The controlled runs start with fresh optimizer moments and retain them when source supervision ends. NLL is measured on fixed held-out panels: 99 general-text documents with 8,192 scored tokens from each of the six source components (49,152 total), and 219 Anthropic-HH instruction examples with 32,768 scored tokens. Thus the instruction panel measures held-out instruction prediction rather than the Tulu training loss. A loss change is relative to the same run's measured initial state. We report final instruction-judging outcomes separately from these loss diagnostics. The controlled endpoint evaluation also uses the 805 AlpacaEval prompts, greedy generation with at most 1,024 new tokens,

and a Llama-3-70B-Instruct judge with randomized answer order. Its fixed reference is the 1.874T Tulu hard-switch model trained at learning rate 10^{-5} with seed zero, rather than one of the controlled 5×10^{-5} runs. These behavioral scores are reported separately from the benchmark sweep.

A.5 Offline-to-online reinforcement learning

Tasks and budgets. We use D4RL medium and random datasets for HalfCheetah, Hopper, and Walker2d; UMaze, UMaze-Diverse, Medium-Play, and Medium-Diverse for AntMaze; and the cloned datasets for Pen, Hammer, Door, and Relocate in Adroit. The locomotion and AntMaze datasets use version 2, and Adroit uses version 1. Offline-initialized agents receive one million gradient updates before 500,000 online environment steps. The online-from-scratch control receives the same interaction budget without offline pretraining. Each condition uses five seeds.

The MuJoCo transition is TD3+BC to TD3. AntMaze and Adroit use SPOT-style offline and online losses. Actor and critic networks use two width-256 ReLU layers. MuJoCo and Adroit use layer normalization; AntMaze does not. Online TD3-family learning rates are 10^{-4} for the actor and 3×10^{-4} for the critic. The discount is 0.99 and target-network update coefficient is 0.005. Batch size is 256, with one gradient update per environment step; actor and target updates occur every two critic updates. Exploration and target-policy noise standard deviations are 0.1 and 0.2, with target noise clipped at 0.5. TD3+BC pretraining uses a behavior-cloning coefficient of 2.5.

Replay and support regularization. Smoothing draws separate batches from the fixed offline dataset and the online replay buffer, then mixes the losses. The online branch uses only online data. The SPOT baseline initializes its replay buffer with offline transitions and appends online experience. Replay buffers have capacity two million transitions. SPOT uses a width-750 VAE trained for 100,000 updates, with learning rate 10^{-3} and $\beta = 0.5$.

The SPOT support coefficient follows $\lambda_t = \lambda_0 \max(\lambda_{\text{floor}}, 1 - t/10^6)$, with $(\lambda_0, \lambda_{\text{floor}}) = (0.05, 0.2)$ for AntMaze and $(1, 0.5)$ for Adroit. The SPOT baseline uses this decay schedule; the hard-switch baseline keeps the support coefficient constant.

Objective scale. For TD3+BC-to-TD3, we also anneal the scale of the online actor objective. This is to account for the fact that the two objectives have very different scales. The online actor branch is normalized by a stop-gradient estimate of $\mathbb{E}|Q|$ and multiplied by

$$c_k = \exp\big((1 - \alpha_k) \log c_0 + \alpha_k \log c_1\big),$$

where c_0 is the offline behavior-cloning coefficient and c_1 is the current online $|Q|$ scale. At the end of the transition, normalization and scaling cancel to recover the raw TD3 actor objective. The main horizon is 50,000 environment steps for MuJoCo and Adroit and 250,000 for AntMaze. Appendix E compares horizons of 10,000, 50,000, 100,000, and 250,000 steps. These are comparisons on the reported tasks rather than selection on a separate validation suite. The other algorithm parameters are fixed at the values described here.

Other baselines. AWAC uses a replay buffer initialized with offline data and appended online data, actor/critic learning rates 3×10^{-4}, advantage temperature 0.1 on AntMaze and 1 elsewhere, and advantage weights clipped at 100. Cal-QL samples half its batch from each of separate offline and online buffers; it uses CQL coefficient 5 and ten sampled actions. On AntMaze it uses Lagrange CQL with target action gap 0.8, max target backup, and CQL-difference clipping below at -200. OPT first collects 25,000 steps with the frozen policy, then performs 50,000 critic-only updates before joint fine-tuning. Our implementation uses uniform rather than prioritized replay. Stochastic-policy baselines use Gaussian policy heads on the same domain-specific MLP profiles.

Evaluation. Training uses the algorithms' reward transformations, but evaluation uses raw environment rewards before D4RL normalization. On AntMaze, SPOT-family and AWAC runs use $r - 1$ during training; Cal-QL and the raw TD3 online control use $10r - 5$. We evaluate every 10,000 online steps, with 128 episodes for MuJoCo and ten for AntMaze and Adroit. Per-task endpoint scores average the final three evaluations within each seed; tables then report the mean and standard error across seeds. The main suite plots report IQMs with 95% task-stratified bootstrap intervals. These aggregates need not have the same ordering as arithmetic suite averages.

B Details of the toy example

Setup and training. Each input is a pair of a CIFAR-10 image (automobile or truck) and an MNIST image (digit 0 or 1). Pixel values are scaled to $[0, 1]$, without normalization or augmentation. Each branch is an MLP with three hidden layers of 128 ReLU units and two output logits, and the prediction is made from the sum of the two branches' logits. Pairs are formed on the fly: each CIFAR image is paired with an MNIST image drawn at random from the digit class given by the target mapping or, for the source term of smoothing, by the source mapping. We pretrain on the source mapping with SGD (learning rate 0.01, batch size 256, 2000 steps). Fine-tuning uses SGD with the same learning rate and batch size for 2000 steps. Loss smoothing uses the linear schedule of Section 3, with α growing from 0 to 1 over the first $\tau = 1000$ steps, followed by 1000 steps of target-only training. All runs start from the same pretrained model; the three seeds differ in the fine-tuning data order and in the MNIST images drawn.

Measures. The linear probe (Figure 5a) is a logistic regression on the standardized last hidden layer of the CIFAR branch, fit on 4000 fixed CIFAR training images and evaluated on all CIFAR test images of the two classes. Reliance (Figure 5b) is the model's test accuracy when each CIFAR test image is paired with an MNIST test image of a random class, so that only the CIFAR branch carries information about the label. Starvation (Figure 5c) is measured on a fixed set of 4096 training pairs as the norm of the gradient that each method applies to the CIFAR branch, divided by the norm of the gradient that the same method would apply if the prediction used the CIFAR logits alone, i.e. with the same weights but the CIFAR-alone error in place of the error of the full model. For the CIFAR-alone loss, the CIFAR logits are centred over the batch, since the split of a constant offset between the two branches is arbitrary and would otherwise dominate this gradient. We show the geometric mean over seeds, smoothed over 50 steps. The weights in Figure 5d are the batch averages of λ and κ, weighted by the target error e_{t}, so that they give the same total push on the logits as the per-example weights.

Baselines. All baselines differ from the hard switch only during the smoothing period and use target-only training afterwards. Coefficient-only multiplies the target loss by α; with SGD, this is the same as scaling the learning rate by smoothing's schedule. The two reweighting baselines compute λ and κ as in Eq. (1), from the current model's errors on the target pair and on a source pair with the same CIFAR image, and multiply the gradients of the target loss with respect to the CIFAR and MNIST logits by these weights. The source pair is used only to compute the weights, not in the gradient. Per-example reweighting applies each example's own λ and κ; global reweighting applies their batch averages, weighted by e_{t} as in Figure 5d, to every example.

Derivation of Eq. (1). With two classes, the cross-entropy of an example is a function of the logit margin towards its label, $\ell(m) = \log(1 + e^{-m})$, and $-\ell'(m) = 1 - p(y) = e$. For the target pair, $m = S(x) + C(c)$, so the hard switch updates the branches with $e_{\mathrm{t}} \big(\nabla S(x) + \nabla C(c)\big)$. The smoothed loss $\alpha\, \ell\big(S(x) + C(c)\big) + (1 - \alpha)\, \ell\big(S(x) + C(o)\big)$ instead gives

$$\big(\alpha e_{\mathrm{t}} + (1 - \alpha)\, e_{\mathrm{s}}\big)\, \nabla S(x) + \alpha e_{\mathrm{t}}\, \nabla C(c) + (1 - \alpha)\, e_{\mathrm{s}}\, \nabla C(o).$$

The CIFAR term equals $\lambda e_{\mathrm{t}}\, \nabla S(x)$. For the MNIST branch, we split its two gradients into their average $s = \big(\nabla C(c) + \nabla C(o)\big)/2$, the direction in which both pairs push the branch the same way, and their half-difference $d = \big(\nabla C(c) - \nabla C(o)\big)/2$, in which they push it in opposite ways. Substituting $\nabla C(c) = s + d$ and $\nabla C(o) = s - d$ gives the exact update

$$\lambda e_{\mathrm{t}} \big(\nabla S(x) + s\big) + \kappa e_{\mathrm{t}}\, d,$$

compared to $e_{\mathrm{t}} \big(\nabla S(x) + s + d\big)$ for the hard switch. The shared part s thus receives the shared weight λ, and the conflicting part d the conflicting weight κ. When the two gradients mirror each other, $\nabla C(o) = -\nabla C(c)$, we have $s = 0$ and $d = \nabla C(c)$, which gives Eq. (1).

C Complete Adaptation Results

The overview combines settings with different objectives and evaluation metrics. Here we unpack those aggregates, keeping each benchmark's natural unit of comparison. This also makes visible the settings in which smoothing ties a simpler control or does not improve the target outcome.

C.1 Continual CIFAR adaptation

Table 4 reports current-task test accuracy after the final task of the 100-task sequence. Under shuffled labels, smoothing improves adaptation substantially relative to hard switching, while learning-rate warmup does not reproduce that improvement. Under input permutations, warmup is competitive with smoothing. The result therefore depends on the structure of the shift, rather than simply on whether the distribution changes. In particular, the permuted-input result leaves room for an optimization-speed explanation in that setting; the shuffled-label comparison is the stronger evidence for a contribution from the old-task loss.

Table 4: CIFAR-10 current-task test accuracy (%) at task 100. Values are means and standard deviations over ten runs.

Shift	Method	Test accuracy (%)	SD (%)
Shuffled	Hard switch	44.86	1.40
Shuffled	Loss smoothing	49.54	1.84
Shuffled	LR warmup	44.74	1.30
Shuffled	EWC	47.75	0.68
Permuted	Hard switch	52.18	0.60
Permuted	Loss smoothing	53.23	0.96
Permuted	LR warmup	53.62	0.49
Permuted	EWC	52.60	0.52

C.2 DomainNet across domains and intervention windows

The aggregate source–target frontier can conceal variation between target domains. Table 5 reports the final target-test scores for each domain under the same ImageNet initialization. The gains from smoothing are modest compared with the variation in absolute difficulty across domains. Preparing the target head improves source compatibility but reduces target accuracy under this protocol.

Table 6 gives the complete window sweep behind the ImageNet frontier. Both smoothing and replay improve source-probe accuracy as source supervision lasts longer. Their target outcomes are less similar: smoothing reaches its highest observed target IQM at the 50% window, whereas the replay curve continues improving through the full-duration setting. We present all tested windows rather than using the best observed target-test point as a separately validated schedule choice.

Table 5: Final DomainNet target-test accuracy (%) by training domain, with ImageNet initialization. Smoothing and temporary replay use a 25% source window. Values are mean $\pm$ standard error across three seeds; LP-FT loads a precomputed head before the same 60-epoch adaptation budget.

Domain	Hard switch	Smoothing	Replay	LP-FT
Clipart	81.87 ± 0.21	82.03 ± 0.09	81.81 ± 0.09	80.52 ± 0.05
Infograph	48.09 ± 0.01	48.63 ± 0.07	48.49 ± 0.17	44.90 ± 0.14
Painting	77.44 ± 0.06	77.69 ± 0.09	77.58 ± 0.05	75.66 ± 0.09
Quickdraw	71.27 ± 0.02	71.83 ± 0.07	71.31 ± 0.08	69.27 ± 0.04
Real	87.76 ± 0.04	87.84 ± 0.02	87.79 ± 0.04	87.37 ± 0.04
Sketch	75.39 ± 0.09	75.65 ± 0.07	75.49 ± 0.09	73.09 ± 0.09

Table 6: The full ImageNet-initialized DomainNet window sweep. Source values are refitted-probe CV IQMs; target values are epoch-60 test IQMs. Brackets give 95% seed-bootstrap intervals with domains held fixed. A 100% replay window is persistent replay.

Condition	ImageNet probe CV IQM (95% CI)	DomainNet test IQM (95% CI)
Hard switch	76.736 [76.422, 77.034]	76.451 [76.364, 76.556]
Loss smoothing (5%)	78.836 [78.508, 79.122]	76.646 [76.540, 76.772]
Loss smoothing (10%)	79.676 [79.502, 79.902]	76.633 [76.531, 76.740]
Loss smoothing (25%)	81.642 [81.366, 81.906]	76.774 [76.702, 76.840]
Loss smoothing (50%)	84.110 [83.900, 84.310]	76.833 [76.734, 76.909]
Loss smoothing (100%)	85.624 [85.388, 85.870]	76.818 [76.749, 76.887]
Replay (5%)	78.776 [78.478, 79.094]	76.537 [76.453, 76.641]
Replay (10%)	79.656 [79.350, 79.988]	76.535 [76.442, 76.619]
Replay (25%)	81.730 [81.552, 81.910]	76.543 [76.475, 76.610]
Replay (50%)	84.224 [83.962, 84.446]	76.567 [76.499, 76.633]
Replay (100%)	85.626 [85.398, 85.838]	76.673 [76.605, 76.740]
LP-FT hard switch	82.064 [81.866, 82.204]	74.580 [74.516, 74.652]

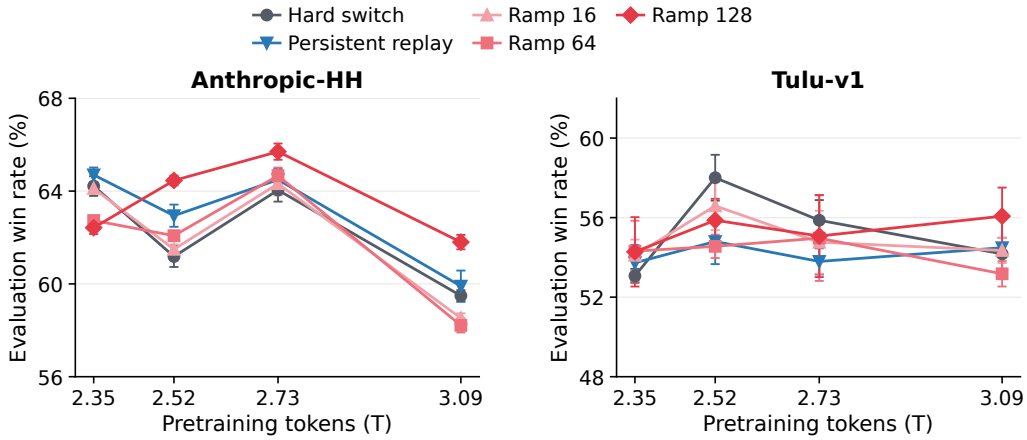


Figure 9: OLMo-1B AlpacaEval results across pretraining checkpoints and source-mixing schedules, evaluated on 483 prompts after learning-rate selection on 322 separate validation prompts. Error bars show seed SEM.

C.3 OLMo results across pretraining checkpoints

For each dataset, checkpoint, and method, we select the learning rate from seven candidates using mean win rate on 322 validation prompts across three seeds. The figure reports win rate on the other 483 prompts, with standard errors across seeds. The shared split and selection procedure are described in Appendix A.4. Replay uses a persistent 50% source fraction.

With a 128-update window, smoothing outperforms hard switching at three of four checkpoints on Anthropic-HH and two of four on Tulu-v1. Persistent replay remains competitive, and changing the smoothing duration changes which checkpoints benefit. Source supervision can therefore improve adaptation of late pretrained models, while the most effective schedule depends on the dataset and checkpoint.

A 128-update window leaves only 13 target-only updates for HH, compared with 355 for Tulu. Dataset and post-intervention training duration therefore change together in this comparison.

C.4 Retention and behavioral performance in the controlled OLMo study

The controlled Tulu experiment fixes the learning rate to isolate the adaptation procedure. It should not be interpreted as another set of seeds for the tuned benchmark. Its improved general-text NLL retention does not translate into a consistent improvement in judged response quality. Counting ties as half a win, linear smoothing and hard switching score approximately 52.6% each at the earlier checkpoint; at the later checkpoint they score 53.6% and 55.6%, respectively.

This distinction matters for the interpretation of Appendices D and E: source supervision can change what is retained after adaptation without improving every evaluation of the final model. The controlled experiment supports the trajectory and retention conclusions, while the benchmark sweep tests their practical value under learning-rate selection. A null behavioral gain in this fixed-rate study therefore limits the claim that retention predicts response quality; it does not erase the target gains measured in the separately tuned benchmark. Conversely, those benchmark gains do not establish that better retention caused them.

C.5 Offline-to-online reinforcement learning

Table 7 reports per-task normalized returns for the comparisons in the overview. The MuJoCo learning curves in Figure 10 show how those endpoints are reached. Offline initialization is particularly useful when exploration is difficult, while on some locomotion tasks the main challenge is moving beyond an imperfect offline policy. Smoothing is useful across these regimes, but it is not the best method on every individual task.

Baseline definitions are given in Appendix A.5. The table reports per-task means over seeds; the main overview reports suite IQMs.

Table 7: D4RL normalized returns after 500,000 online environment steps (Offline reports the pretrained policy before online interaction). Each seed's score averages its final three evaluations; entries give mean ± standard error over five seeds. Smoothing uses 50k-step windows on MuJoCo and Adroit and a 250k-step window on AntMaze. MuJoCo suffixes (m) and (r) denote medium and random datasets; dashes denote an inapplicable SPOT comparison.

Task	Offline	Online	Hard	SPOT	AWAC	Cal-QL	OPT	Smooth.
MuJoCo								
HalfCheetah (m)	47.3 ± 0.0	29.5 ± 7.5	48.2 ± 0.3	–	52.2 ± 0.3	54.5 ± 0.6	48.5 ± 0.2	48.2 ± 0.5
Hopper (m)	47.2 ± 1.9	69.6 ± 18.6	101.2 ± 5.8	–	81.9 ± 4.1	74.4 ± 5.6	105.5 ± 0.4	99.9 ± 5.5
Walker2d (m)	83.2 ± 0.4	61.5 ± 13.9	99.2 ± 1.9	–	86.4 ± 0.2	87.4 ± 0.4	94.6 ± 1.1	103.1 ± 0.7
HalfCheetah (r)	8.3 ± 0.3	18.4 ± 0.6	43.0 ± 2.6	–	32.9 ± 0.6	13.1 ± 1.8	37.5 ± 1.0	45.2 ± 1.7
Hopper (r)	13.5 ± 7.1	67.2 ± 17.5	55.0 ± 9.8	–	18.1 ± 1.3	11.9 ± 2.0	14.9 ± 2.1	77.9 ± 16.7
Walker2d (r)	7.2 ± 3.4	73.3 ± 7.5	74.7 ± 3.3	–	4.2 ± 0.3	11.0 ± 3.8	9.0 ± 1.7	68.0 ± 13.1
AntMaze								
Medium-Div.	38.0 ± 11.1	0.0 ± 0.0	8.0 ± 4.3	35.3 ± 10.6	0.0 ± 0.0	28.0 ± 16.7	68.0 ± 10.1	94.0 ± 3.1
Medium-Play	58.0 ± 7.3	0.0 ± 0.0	38.0 ± 5.4	73.3 ± 7.8	0.0 ± 0.0	0.0 ± 0.0	84.0 ± 4.5	83.3 ± 8.0
Umaze	26.0 ± 16.0	0.0 ± 0.0	47.3 ± 20.9	34.0 ± 21.1	6.7 ± 1.5	98.7 ± 1.3	44.0 ± 21.9	80.0 ± 20.0
Umaze-Div.	18.0 ± 9.7	0.0 ± 0.0	68.0 ± 14.2	67.3 ± 17.3	14.7 ± 6.2	47.3 ± 3.9	81.3 ± 7.2	76.0 ± 19.1
Adroit								
Door	0.3 ± 0.2	58.3 ± 14.2	0.5 ± 0.4	0.2 ± 0.1	-0.1 ± 0.0	19.9 ± 14.6	72.1 ± 3.8	63.4 ± 10.8
Hammer	2.2 ± 0.4	100.5 ± 11.6	116.2 ± 1.5	118.3 ± 2.4	0.3 ± 0.0	120.9 ± 3.4	112.9 ± 5.8	122.1 ± 2.5
Pen	47.8 ± 9.0	101.6 ± 8.8	114.0 ± 7.6	115.8 ± 3.6	55.3 ± 4.7	103.0 ± 7.1	121.8 ± 4.7	118.2 ± 3.5
Relocate	-0.0 ± 0.0	-0.1 ± 0.0	-0.1 ± 0.0	-0.1 ± 0.0	-0.2 ± 0.0	-0.2 ± 0.0	0.3 ± 0.1	-0.1 ± 0.0

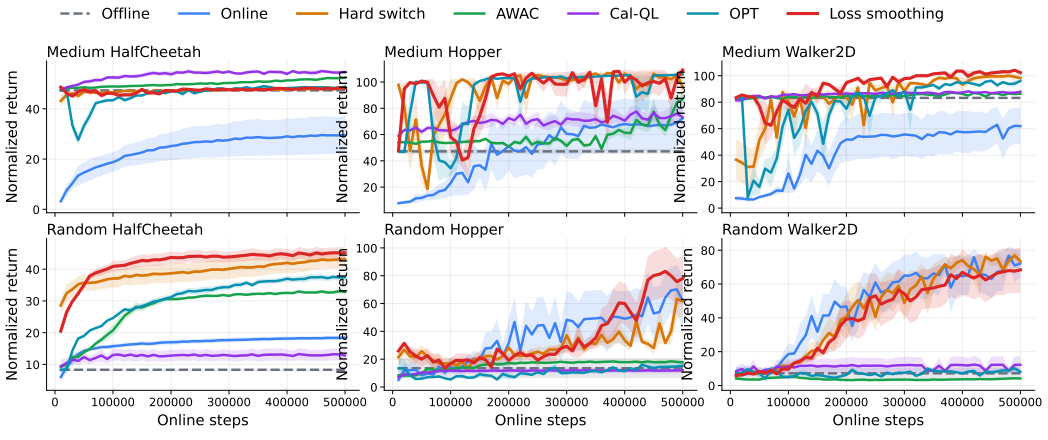


Figure 10: Offline-to-online learning curves for the six MuJoCo tasks. Bands show standard errors across seeds. Curves complement the endpoint comparison by distinguishing early disruption, recovery, and continued improvement during online interaction.

D What Does Source Feedback Add Beyond Slowing Target Learning?

Mixing the source and target losses changes two things at once: it adds source feedback and reduces the coefficient on the target loss. We separate these effects with controls that preserve the target coefficient, and then ask whether other ways of staying close to the initialization reproduce the benefit. The controls address different alternatives; none is a general test against every possible regularizer or learning-rate schedule.

D.1 A controlled comparison in OLMo

The controlled Tulu study uses the same two pretrained checkpoints, data-order seeds, and learning-rate schedule across four methods. In addition to hard switching and linear smoothing, we include temporary constant replay and a *coefficient-only* control. The latter follows smoothing's target-loss coefficient but omits the source term. All four methods become ordinary target-only training after update 128. Their precise schedules are given in Appendix A.4.

Figure 11 shows the resulting general-text and held-out instruction-loss trajectories. The instruction panel uses Anthropic-HH examples, while adaptation trains on Tulu. Source mixing reduces the disruption to general-text prediction beyond what is achieved by reducing the target coefficient alone. This protection comes with a change in how quickly held-out instruction prediction improves. Temporary constant replay is competitive with linear smoothing, so these results support the value of source feedback more clearly than a unique advantage of a linear ramp. This comparison tests the broader role of source supervision: a competitive constant mixture narrows the schedule claim without removing the evidence for the source term.

The coefficient-only control should not be read as an equivalent learning-rate warmup. With AdamW, changing a loss coefficient also changes the gradients used to update the optimizer's moments, and can interact with clipping. The comparison isolates removal of the source term while retaining the target-loss schedule; it does not hold every applied parameter update fixed.

D.2 Adding source feedback at the same model state

Comparing full trajectories leaves open whether a difference reflects source feedback at the current state or the history that produced that state. To make a more direct comparison, we branch from the same hard-switch model at update 16. For each checkpoint and seed, the continuations receive either source–target mixing, target-coefficient attenuation without source data, or ordinary target training. The intervention ends at update 48, after which all branches receive target-only training through update 80. The mixed branch uses source and target coefficients $(0.5, 0.5)$; the coefficient-only branch uses $(0, 0.5)$, and ordinary target training uses $(0, 1)$. All branches copy the parent's parameters, AdamW moments, learning-rate clock, and random state, and share subsequent target batches. Neither branching nor the return to target-only training resets the optimizer or restarts warmup.

Source mixing improves general-text NLL relative to the coefficient-only branch in all six checkpoint–seed pairs, and the difference survives the common target-only continuation (Figure 12). Thus the source term can change the subsequent trajectory even when the starting model is held fixed.

D.3 Learning-rate warmup in continual CIFAR

The continual-CIFAR comparison in Table 4 provides a complementary test under SGD. The warmup control follows the coefficient on the current-task loss without replaying the previous task. Warmup is competitive with smoothing under input permutations, but does not reproduce its benefit under shuffled labels.

These shifts therefore give different answers. The shuffled-label result supports a contribution from source supervision beyond slowing target learning. The permuted-input result limits that interpretation: in this setting, the simpler warmup control achieves a similar target outcome. Nevertheless loss smoothing achieves the best or ties for the best performance across both shifts, but these results reestablish that the improvement of loss smoothing is dependent on the type of distribution shift.

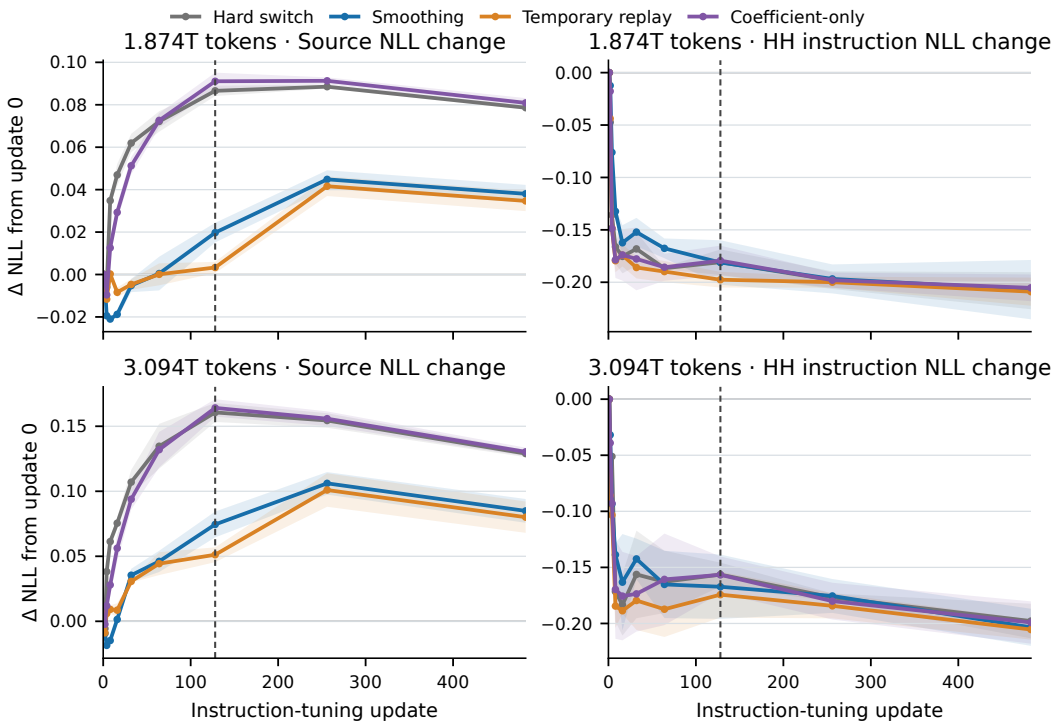


Figure 11: Controlled OLMo adaptation with hard switching, linear smoothing, temporary constant replay, and target-coefficient attenuation alone. General-text and held-out instruction NLL changes are measured relative to each run's initialization. Instruction NLL is evaluated on Anthropic-HH; the models are trained on Tulu. Each method uses three paired seeds at each pretrained checkpoint; bands are 95% Student-t confidence intervals. The vertical marker identifies the end of the 128-update intervention; subsequent training is target-only.

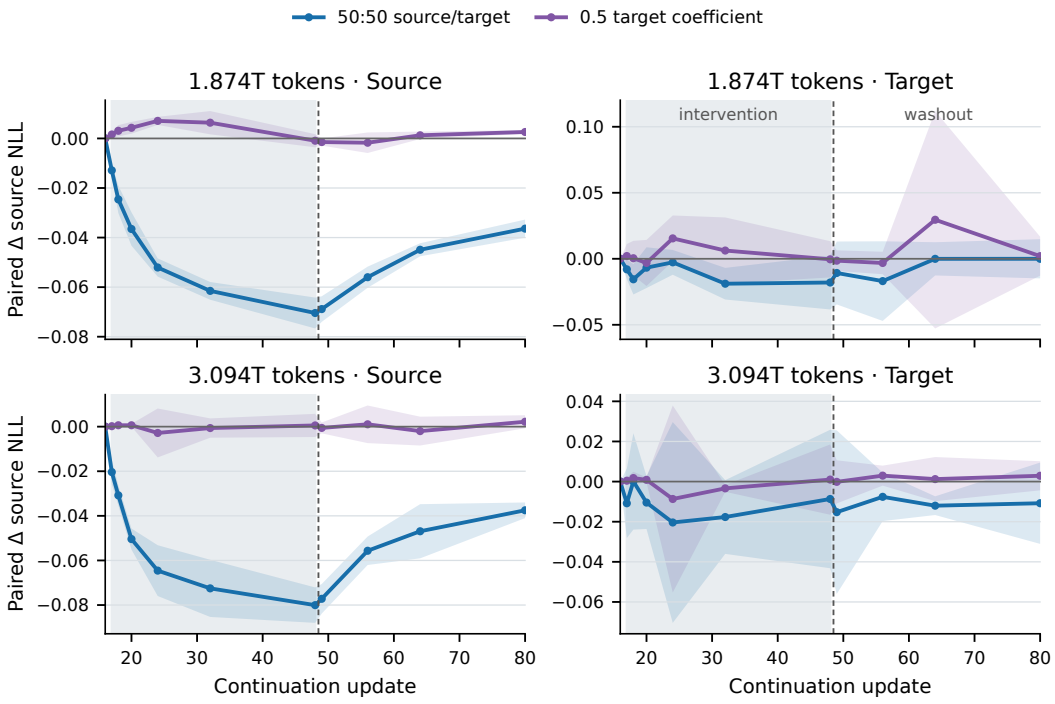


Figure 12: OLMo continuations from identical update-16 hard-switch states. The source-containing and coefficient-only interventions end at update 48; all branches then train on the target alone through update 80. Comparisons show branch-minus-target-only differences, paired within pretrained checkpoint and seed, with 95% Student-t confidence intervals. The source-loss difference remaining after this common continuation demonstrates an effect beyond target-coefficient attenuation.

D.4 INITIALIZATION-BASED PENALTIES AND HEAD PREPARATION

The DomainNet controls replace the source loss with either a KL penalty toward the pretrained model or an EWC penalty toward its parameters. They retain smoothing's target coefficient and the duration of the auxiliary term. KL compares predictions from the retained ImageNet head on target examples, using $T^2\mathrm{KL}(p^T_{\theta_0}\|p^T_\theta)$ with temperature $T = 2$, averaged over the batch. The teacher is the frozen pretrained model.

EWC penalizes backbone parameters with $\frac{1}{2}\sum_i F_i(\theta_i - \theta_{0,i})^2$. We estimate the diagonal predictive Fisher on 8,192 ImageNet training images sampled without replacement (seed 20260923), using the same input normalization as adaptation. Labels are sampled from the pretrained model's predictive distribution. For each batch of 32 images, we square the gradient of the mean negative log-likelihood and multiply by 32, then average these estimates across batches. Neither method replays source examples during adaptation.

Penalty strengths were calibrated on Clipart and Quickdraw calibration updates to give a median ratio of weighted penalty-gradient norm to target-gradient norm of 0.5. This yields an EWC coefficient of approximately 204.48 and a KL coefficient of 0.0174. We evaluate intervention windows of 5, 10, 25, 50, and 100% of training. The main penalty frontier reports paired differences from the same random-head hard-switch baseline, using source-probe CV and target-test IQM. Its 95% intervals resample the three seeds jointly while keeping the six domains fixed.

Head preparation supplies a different control. LP-FT loads a target head fitted in advance on frozen source features, then fine-tunes the head and backbone for 60 epochs, matching the random-head adaptation budget. Preparing the head improves source compatibility, but does not improve target performance in this experiment. Moreover, the target benefit of adding smoothing is visible with a random head and is negligible with the prepared head. Preservation and target adaptation are therefore distinct outcomes; reducing initial head mismatch does not by itself guarantee a better target solution. The lack of a smoothing gain after head preparation limits its usefulness under that initialization procedure; the main random-head result is a different, explicitly measured comparison. For the main head-comparison figure, both axes are averaged across domains within each seed, with standard errors across the three seed means. The source axis uses a refitted source probe, rather than the retained source classifier.

E Duration and Persistence of Source Supervision

We examine how the duration of source supervision affects adaptation and which effects remain after it ends. Duration sweeps test how long to retain the source objective, while target-only continuations show whether the resulting differences persist after source supervision is removed.

E.1 Duration and Replay Amount in Continual Learning

The repeated-task CIFAR ablations vary the transition length and the size of the stored source batch (Figures 13 and 14). The useful duration differs between input and label shifts. Short windows can provide too little guidance, whereas long windows can keep training tied to the previous task. Increasing replay storage beyond a batch of 256 examples gives little additional benefit in these experiments. These sweeps motivate treating duration and replay amount as practical hyperparameters, rather than assuming more source supervision is always better.

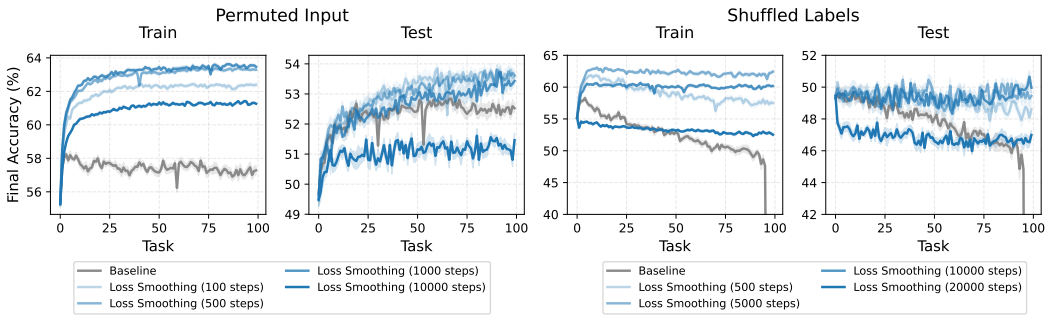


Figure 13: Continual-CIFAR duration sweep. The preferred transition length differs between shuffled labels and permuted inputs; both insufficient and excessive source exposure can reduce performance.

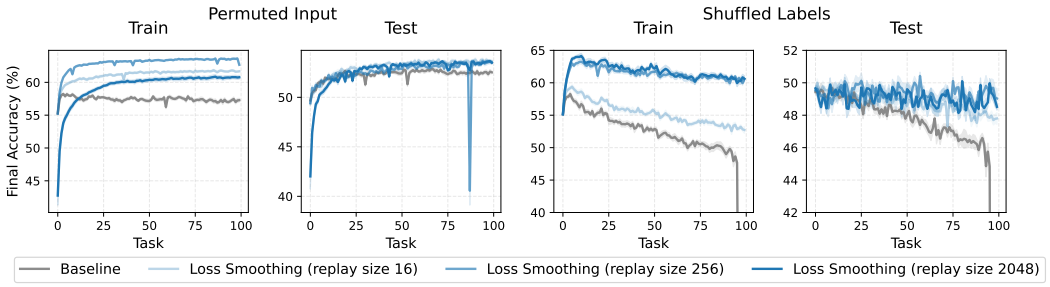


Figure 14: Continual-CIFAR replay-size sweep. A stored batch of 256 examples is generally sufficient in this setting; larger buffers do not consistently improve adaptation.

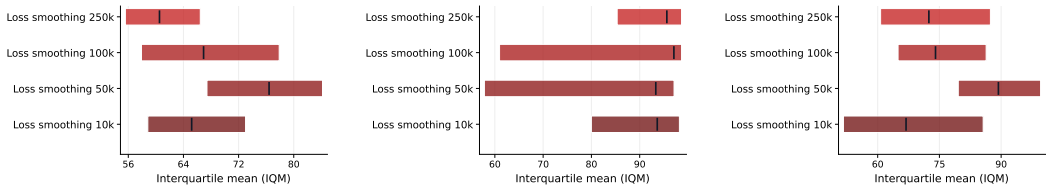
E.2 What Persists in Pretrained Vision and Language Models?

The main DomainNet persistence plot follows a 25% source window: source supervision ends at epoch 15, followed by 45 target-only epochs. Its source outcome is five-fold CV accuracy of refitted ImageNet linear probes. The curves summarize all six domains and three seeds using epoch-wise IQMs; bands are pointwise 95% seed-cluster bootstrap intervals.

The OLMo comparison has a longer target-only tail relative to its source window: mixing ends at update 128, followed by 355 target-only updates. The full four-arm trajectories in Figure 11 show that the source-loss advantage of mixing remains at the end, although source loss itself rises after mixing stops. Temporary replay also preserves this advantage.

E.3 OFFLINE-TO-ONLINE SMOOTHING HORIZONS

Figure 15 varies the source window over 10k, 50k, 100k, and 250k environment steps. The main results use a 50k window for MuJoCo and Adroit and a 250k window for AntMaze, selected from these comparisons. The useful horizon depends on the environment suite. In particular, retaining offline guidance for longer can be useful when online exploration is difficult, whereas a shorter transition can suffice when online data quickly supports improvement. The sweep demonstrates this dependence, but it does not provide a universally optimal schedule, although 50k steps of smoothing is optimal on 2 of the 3 suites.



(a) MuJoCo (b) AntMaze (c) Adroit

Figure 15: Offline-to-online horizon ablations. Each panel reports suite IQM with 95% bootstrap intervals. All settings have the same 500k-step online interaction budget; only the duration of source mixing changes within each smoothing procedure.

F REPRESENTATIONS: PRESERVATION, ACCESSIBILITY, AND USE

A representation can support a useful classifier without the model's existing head making use of that information. Conversely, similar feature geometry does not guarantee better classification. We therefore report readout performance, refitted probes, and feature similarity as separate measurements, rather than treating any one of them as a complete measure of representation quality.

F.1 READOUT AND PROBE PROTOCOLS

DomainNet readout diagnostics apply the trained 345-class target head to each style without refitting. Probe diagnostics instead freeze the final representation and fit a separate linear classifier for each evaluation style. Each uses a fixed, class-stratified pool of 5,000 labeled training images. Probe accuracy is measured by five-fold cross-validation with regularization selected by CV. It describes linear accessibility under this fitting procedure; it is not zero-shot transfer or an independent official-test estimate.

Probes use L2-regularized logistic regression with at most 1,000 iterations. Within each fold, features are standardized using the training-fold mean and standard deviation. We search inverse regularization strengths $C \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ and select the highest mean CV accuracy, breaking ties by lower CV log loss and then higher C. The same folds select C and provide the reported score; there is no outer test fold. This shared selection procedure supports method comparisons but can make absolute CV accuracy optimistic.

For off-domain comparisons, the five evaluated styles share one trained model. We form paired differences within training domain, evaluation style, and seed, and show means and standard errors across the three seeds. The image IDs and labels in these DomainNet pools match across hard switching and smoothing for every pair. The five styles evaluated from one run are not treated as five independent training replicates.

The source-probe analysis uses a separate source-training pool. In the source-retention trajectories, some ImageNet probe image IDs differ across arms despite matched labels: three smoothing Sketch cells and six replay Real/Sketch cells differ from hard switching. The main persistence figure uses these refitted-probe scores, so its cross-method comparisons include this variation in the source-image pools.

F.2 OFF-DOMAIN INFORMATION AND THE TRAINED CLASSIFIER

Figure 16 compares the change in the unchanged trained head's accuracy with the change in refitted-probe accuracy. The probe gains are more consistent across domain pairs than the gains of the existing head. This means the representation can make information more accessible to a new readout without producing an equally large improvement in the classifier already trained on the target domain. The comparison does not establish whether that information was newly learned or retained from pretraining.

F.3 INITIAL, BEST OBSERVED, AND FINAL ACCESSIBILITY

Figure 17 reports best observed and final off-domain probe accuracy. All scores are changes from the corresponding initialization. For each run and evaluation style, we retain both the epoch-60 score and the best observed post-initialization probe score. This separates the highest measured accessibility during adaptation from the amount remaining at its end.

The best score is chosen independently for each run and evaluation style, using every logged probe checkpoint, including early fractional epochs. It is a descriptive maximum, not the performance of a model selected on independent validation data. Different points may attain their maxima at different times; the line between the best and final scores does not represent a shared trajectory. The full comparison also shows why a single pooled transfer score would hide important domain differences, particularly for Quickdraw.

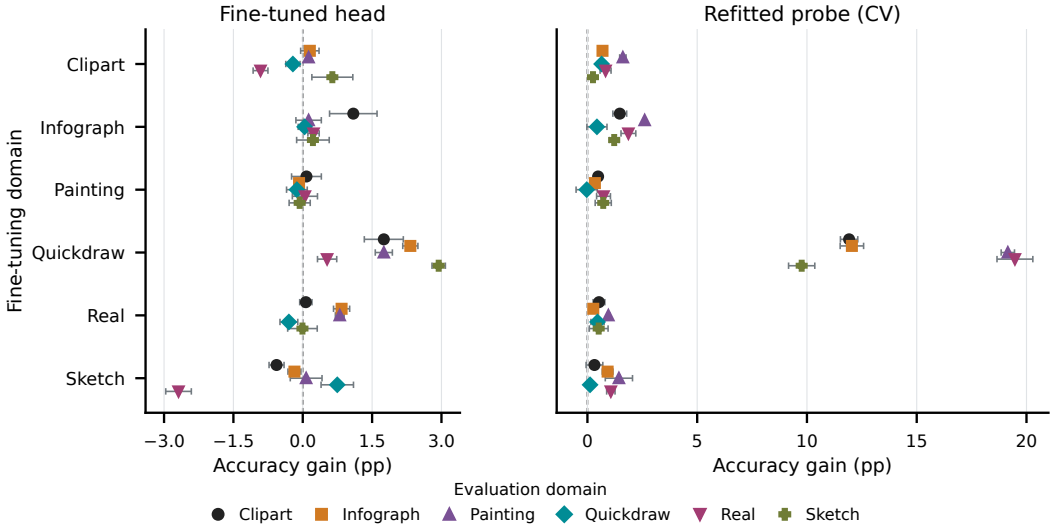


Figure 16: Off-domain DomainNet accuracy changes for the trained target head and independently refitted probes, comparing smoothing (25%) with hard switching. Results cover all six training domains and their five off-domain evaluation styles. Each comparison uses three paired seeds. Readout and probe scores use the same labeled training-pool diagnostics, but only the probes fit a new classifier for the evaluation style.

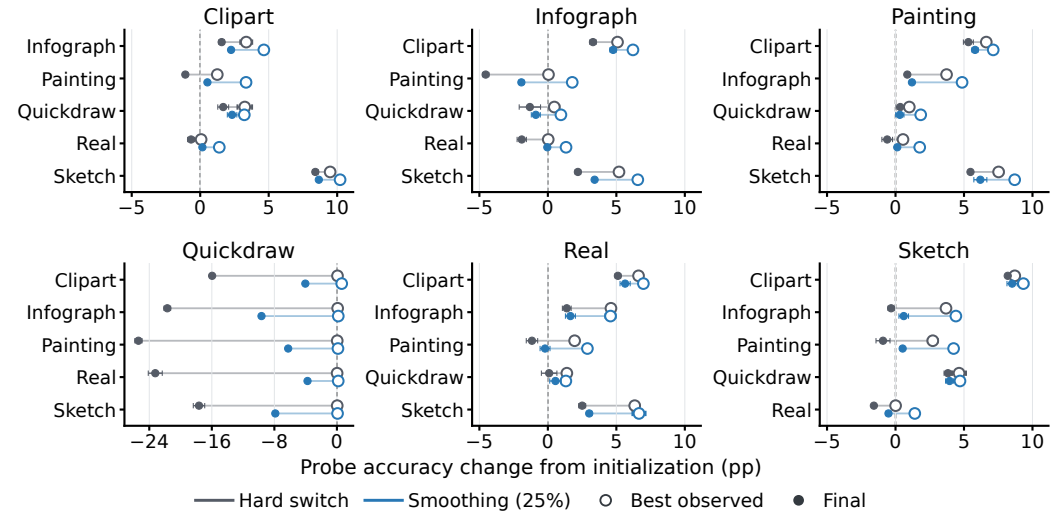


Figure 17: Complete off-domain probe results. Panels identify the fine-tuning domain; rows identify the evaluation domain. Hollow and filled points show best observed and final probe accuracy changes from initialization. Values are means over three seeds, with standard errors. Maxima are chosen per run and evaluation style before averaging. Quickdraw uses a wider horizontal scale.

F.4 Feature geometry across depth

We measure centered linear CKA between each model's initial and adapted representations on the same ordered ImageNet validation examples. The held-out feature loader selects 512 images deterministically from the run's seed. We evaluate every transformer block and the final representation. The endpoint comparison in Figure 18 uses all six domains and three seeds per method.

Smoothing preserves more of the initial geometry in the later layers, where hard switching produces the largest change. The earliest layer changes little under either method. CKA is invariant to orthogonal basis changes and uniform scaling, so this is a statement about feature geometry, not about identical activations or unchanged predictions.

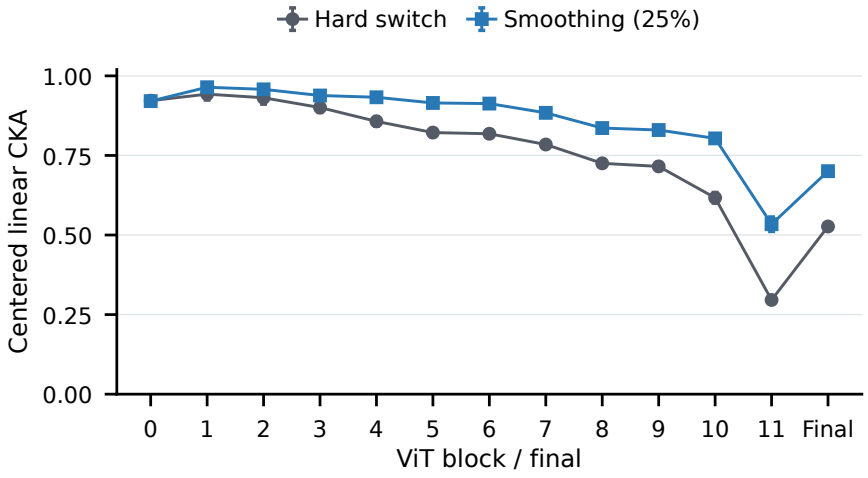


Figure 18: Initial-to-final feature CKA across all twelve ViT blocks and the final representation, for hard switching and smoothing (25%). The comparison uses the held-out ImageNet feature cohort and all six target domains. Error bars summarize variation across the 18 domain–seed runs; they are descriptive standard errors.

G Finite-Step Sensitivity and Optimizer Dynamics

The sensitivity experiments measure how specified parameter perturbations change losses or predictions. Their purpose is to characterize the response to those updates, rather than to estimate global flatness or prove a mechanism for the final adaptation results. We distinguish the diagnostic at initialization from the diagnostic at states reached during training.

G.1 Directions at initialization

The left and middle panels of Figure 8 perturb the 1.874T and 3.094T pretrained checkpoints along either a target-only or a fixed 50:50 source–target AdamW update direction. Both directions are constructed with fresh optimizer moments and rescaled to a common parameter displacement. One displacement unit is the norm of the target-only update at the earlier hard-switch initialization.

We measure source and target NLL changes from the actual unperturbed parent. The ranges in these panels span three diagnostic panels, not three training seeds. Comparing equally sized displacements separates the direction of movement from its norm, but the result remains specific to the measured directions, radii, and evaluation data.

G.2 Sensitivity of states reached during adaptation

The second assay starts from hard-switch and smoothed models at updates 16 and 128. For each pretrained checkpoint and seed, we construct a target-only AdamW direction either with fresh moments or with the optimizer moments saved at that parent. We report perturbations along the update direction at two radii, 0.5 and 1.

Within each seed and measured update, one radius unit is the norm of the fresh target-only update at the earlier hard-switch parent, using nominal learning rate 5×10^{-5}. That absolute radius is shared across checkpoints, methods, and moment conditions. It is not necessarily the norm of one ordinary training step at every parent. Predictions are evaluated at fixed positions using full-vocabulary $\mathrm{KL}(p_{\mathrm{parent}} \| p_{\mathrm{perturbed}})$.

Figure 19 compares predictive-KL responses along the update direction across these conditions. Each point subtracts the hard-switch response from the smoothed model's response after matching seed, checkpoint, training update, moment condition, and radius. Negative values therefore mean that smoothing's parent undergoes less predictive change under that perturbation. Comparing moment conditions and radii shows that the answer depends on how the direction is constructed and where it is measured; the main figure highlights the fresh-moment positive-direction comparison at the source cutoff.

G.3 Scope of the measurement

Update 128 is the end of source mixing, not the end of training: the models subsequently receive 355 target-only updates. We do not have a corresponding final-update sensitivity measurement and therefore do not infer persistence of the sensitivity difference from persistence of source NLL alone.

Finally, a gradient norm, an applied AdamW update norm, and distance from initialization are different quantities. Moment accumulation and clipping can change their relationship. The perturbation protocol controls the applied displacement directly and reports its optimizer-state convention; it should not be interpreted as a statement that source mixing always makes ordinary training updates smaller.

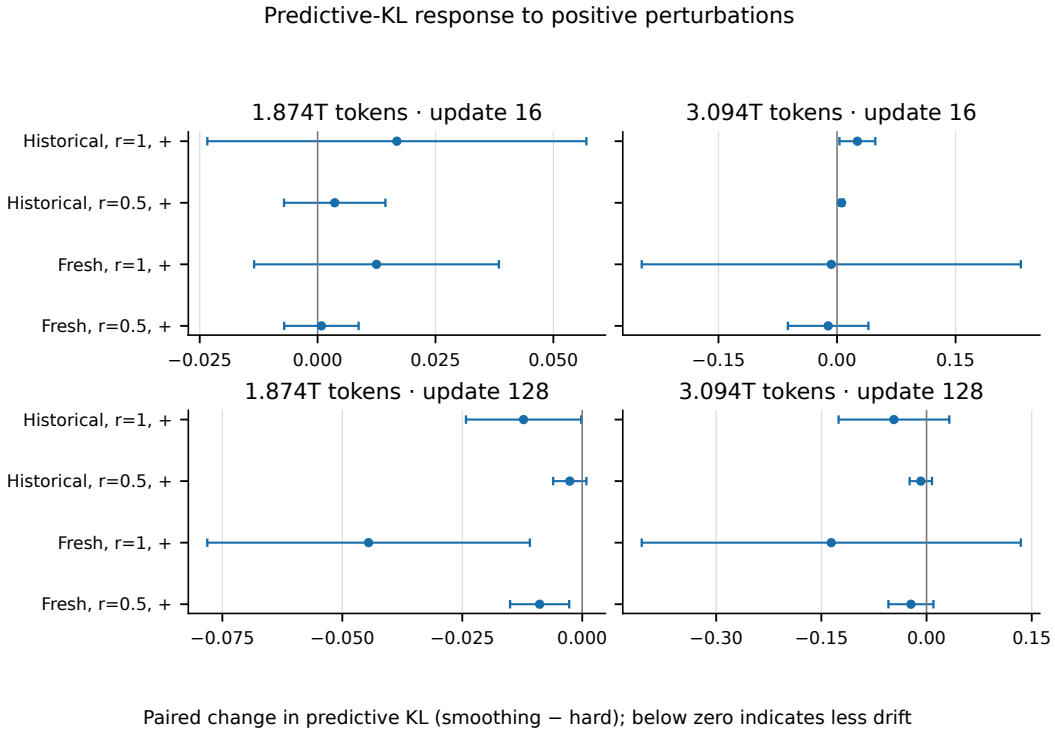


Figure 19: Paired difference in predictive-KL response, smoothing minus hard switching, along the update direction, with fresh or saved optimizer moments and radius multipliers 0.5 and 1. Negative values indicate less predictive drift for the smoothed parent. Points are means across three paired seeds, with 95% Student-t intervals. Each panel identifies the pretrained checkpoint and the update at which its parent states were saved.

H Computational Cost and Source-Access Requirements

Temporary source supervision adds work while its source branch is active. How much work depends on the size of the source batch, the duration of the window, and how the source loss is obtained. Matching optimizer updates therefore does not imply matching examples processed, elapsed time, or total preparation cost. We report the additional source exposure below.

H.1 Source exposure

In the ImageNet-initialized DomainNet sweep, smoothing for 25% of training processes a median of about 668,000 additional source images per run, approximately 25% of the corresponding target-image count. Temporary replay at the same duration processes almost the same number. The benefit is therefore not obtained at the same example budget as hard switching, although both use the same number of target-training epochs.

The controlled OLMo Tulu runs show a similar distinction. Linear smoothing and temporary replay each process about 160.0 million valid tokens, versus 126.5 million for hard switching, an increase of approximately 26.5%. These counts include the additional source branch and exclude padding. They measure token exposure, not FLOPs or a wall-time multiplier.

H.2 Data and storage requirements

The continual-CIFAR implementation retains one small batch from the preceding task. DomainNet and OLMo instead require access to source training examples, and the RL implementation retains the offline dataset alongside online experience. These are different access assumptions. A pretrained checkpoint alone is not sufficient to reconstruct the original source objective in the replay-based experiments.

Source data preparation, pretrained-model training, hyperparameter search, standalone probing, and model judging add costs beyond an individual adaptation run. The exposure counts above do not include these activities.