When the Merge Coefficient Stops Mattering: Proximity Regularized Merging for Continual LoRA Adaptation

Yixuan Liu[1,2,3,*] **Yuhao Sun**[1,*] **Sen Song**[2,3,†] **Jin Li**[1,†]
[1]Sapient Intelligence
[2]School of Biomedical Engineering, Tsinghua Medicine, Tsinghua University
[3]Tsinghua Laboratory of Brain and Intelligence, Tsinghua University

Abstract

Rehearsal-free continual learning with parameter-efficient adapters can be cast as a sequence of task-vector write-in operations: for each new task, a low-rank adapter is learned and merged into a running model. We propose Proximity Regularized Merging (PRM), a minimal modification to sequential LoRA merging that adds a proximal penalty during task-vector training with no additional model passes or trainable parameters, while leaving the subsequent write-in rule unchanged. PRM acts as a robust task-vector regularizer: in the reported Base→+Prox diagnostics, it improves AAA across multiple write-in rules, backbones, and class-incremental settings, while its fixed-coefficient variant remains competitive with strong coefficient-based baselines. Mechanistically, matched-prefix norm controls and proximal-strength sweeps show that proximal training shrinks the task-vector radius, lowers Fisher-weighted interference, broadens the coefficient plateau, and exposes a stability–plasticity trade-off. Together, these results suggest that the effectiveness of sequential LoRA merging depends not only on how much of a task vector is written in, but also on whether the task vector itself has been trained to be mergeable.

1 Introduction

Modern pretrained models are increasingly expected to acquire new classes, domains, and tasks after deployment. Continual learning studies this problem under a sequential data regime, where a model must learn from the current task while retaining performance on previous ones. In the rehearsal-free setting, the constraint is sharper: previous task data cannot be stored or replayed because of privacy, storage, or deployment constraints. Existing rehearsal-free approaches address this challenge from different angles. Parameter-isolation and prompt-based methods reduce interference by routing computation through task-specific components or learned prompts [Wang et al., 2022b,a, Smith et al., 2023], while regularization-based methods constrain changes to parameters estimated to be important for previous tasks [Kirkpatrick et al., 2017, Zenke et al., 2017, Aljundi et al., 2018]. These approaches have substantially improved rehearsal-free continual learning, but they leave open a complementary consolidation question: when a new task is represented as a compact parameter update, under what conditions can that update be safely absorbed into the single running model that must serve all tasks seen so far?

Task vectors and model merging provide a natural language for this question. In model merging, fine-tuned parameter differences are treated as task vectors and combined to reuse capabilities across

*Equal contribution. †Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

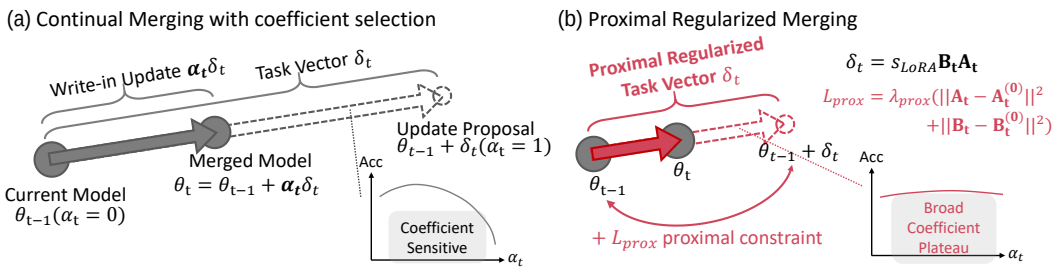


Figure 1: From coefficient selection to task-vector shaping. **(a)** Continual merging with coefficient selection. At task t, θ_{t-1} denotes the current model before learning the new task. LoRA training produces a task vector δ_t and an update proposal $\theta_{t-1} + \delta_t$. A merge coefficient α_t is then selected to determine how much of δ_t is written into the current model, yielding the merged model $\theta_t = \theta_{t-1} + \alpha_t\delta_t$. A coefficient-sensitive write-in path therefore makes careful selection of α_t central to continual merging. **(b)** PRM constrains the task-specific LoRA factors with a proximal penalty during training, producing a task vector δ_t with a shorter Euclidean distance than unconstrained training. The resulting write-in path has a broader high-accuracy coefficient plateau, reducing sensitivity to α_t and making the update more mergeable.

models [Ilharco et al., 2023, Yadav et al., 2023, Yu et al., 2024]. With LoRA-style parameter-efficient adaptation, each new task produces a compact low-rank update that can be stored, scaled, and merged into the current model [Hu et al., 2022]. In sequential LoRA merging, it is important to distinguish two objects that are often coupled in practice: the task vector and the merge coefficient. The task vector, denoted informally as δ_t, is the update learned from the current task and therefore determines the direction and scale of the candidate model change. The merge coefficient, denoted α_t, is a scalar used only when consolidating that learned vector into the running model, e.g., by forming $\theta_{t-1} + \alpha_t\delta_t$. Recent rehearsal-free continual-merging methods instantiate this idea with different vector training procedures and coefficient rules, including MagMax, BECAME, and P&M [Marczak et al., 2024, Li et al., 2025, Qiu et al., 2025]. In particular, P&M highlights the coefficient as a central design object (Figure 1(a)): after perturbation-aware LoRA training, it computes a Fisher-based coefficient to decide how much of the learned vector should be written into the running model.

This coefficient-centric view is useful, but incomplete. A merge coefficient can scale a learned vector, but it cannot change the path induced by that vector. If useful performance occurs only in a narrow interval of α_t, then consolidation remains fragile: approximation noise in the coefficient estimate, task-order variation, or a mismatch between validation and test behavior can move the model away from the safe part of the path. Conversely, when the learned task vector already induces a stable write-in path, many coefficients lead to similar seen-task performance, and the exact coefficient rule becomes less important. Thus, the central question is not whether a task vector is "mergeable" in the abstract, nor only how to tune α_t for a fixed vector. We instead ask: **under what training conditions can LoRA task vectors be merged robustly into the running model, with reduced dependence on the merge coefficient?**

Our method is motivated by a simple observation shared by several lines of prior work: forgetting is closely related to where and how far the model moves in parameter space. Classical continual-learning regularizers penalize changes to parameters estimated to matter for past tasks [Kirkpatrick et al., 2017, Zenke et al., 2017, Aljundi et al., 2018]; Fisher-based merging uses curvature information to avoid destructive combinations [Matena and Raffel, 2022]; and flatness, weight averaging, and mode-connectivity studies show that endpoint accuracy alone is not enough—the path between models also matters [Wortsman et al., 2022, Foret et al., 2021, Izmailov et al., 2018, Mirzadeh et al., 2021]. These observations suggest that a LoRA task vector should not merely fit the current task; it should also remain close enough to the current model so that scaling it by different coefficients does not quickly enter regions that interfere with previous tasks.

In this work, we propose PRM (*Proximity Regularized Merging*), which adds a proximal penalty to the current task's LoRA factors during task-vector training while leaving the downstream write-in rule unchanged (Figure 1(b)). This lets us test whether robust continual merging is driven less by finding a delicate coefficient for a fixed vector, and more by learning a vector whose write-in path is already stable. Across fixed-α, CoFiMA, CoMA, Model-Avg, BECAME, and MagMax [Marczak

et al., 2024, Li et al., 2025, Marouf et al., 2024], proximal shaping consistently improves the corresponding base rule. With one fixed coefficient, PRM attains the best reported mean AAA across our tested settings, indicating that the gain is not merely better coefficient selection. Mechanism analyses show that PRM mainly shrinks the task-vector radius, lowering a norm-based bound on Fisher-weighted interference and broadening the coefficient plateau, until excessive regularization harms current-task plasticity. These findings support a simple conclusion: training LoRA task vectors to be easier to merge can improve continual-learning performance and make precise merge-coefficient selection less critical.

2 Related Work

Rehearsal-free continual learning. Rehearsal-free continual learning studies how to learn a sequence of tasks without storing previous-task data. Traditional methods reduce forgetting through parameter regularization and importance estimation, such as EWC, SI, and MAS [Kirkpatrick et al., 2017, Zenke et al., 2017, Aljundi et al., 2018]; output distillation, as in LwF [Li and Hoiem, 2017]; or gradient and subspace projection, as in GPM [Saha et al., 2021]. For pretrained models, prompt-based approaches learn task-adaptive prompts or prompt components, including L2P, DualPrompt, and CODA-Prompt [Wang et al., 2022b,a, Smith et al., 2023]. Parameter-efficient methods instead update compact modules or low-rank parameters: LoRA provides a widely used low-rank adaptation parameterization [Hu et al., 2022], and recent LoRA-based continual learners such as InfLoRA, SD-LoRA, and CL-LoRA further adapt low-rank modules for rehearsal-free class-incremental learning [Liang and Li, 2024, Wu et al., 2025, He et al., 2025]. In this work, we use LoRA task vectors as the update representation for sequential write-in.

Task vectors and model merging. Model merging combines the parameters or task vectors of separately trained models, for example through weight averaging and model soups [Wortsman et al., 2022], Fisher-weighted averaging [Matena and Raffel, 2022], task arithmetic [Ilharco et al., 2023], or interference-aware merging rules such as TIES and DARE [Yadav et al., 2023, Yu et al., 2024]. These methods are typically studied in a static setting where all source models or task vectors are available at merge time. Continual learning requires a sequential version of this problem: each new update must be incorporated before the next task arrives. Continual model-averaging methods such as CoMA and CoFiMA combine the previous and current task models, with CoFiMA using Fisher information to weight parameters during averaging [Marouf et al., 2024]. MagMax performs sequential task-vector merging through maximum-magnitude update selection [Marczak et al., 2024], while BECAME derives an adaptive merging coefficient from a Bayesian continual-learning formulation [Li et al., 2025]. P&M further formulates a two-stage sequential merging procedure, training a perturbation-aware task-vector proposal and then computing a Fisher-derived write-in coefficient for consolidating it into the running model [Qiu et al., 2025]. PRM follows this sequential LoRA write-in template but modifies only the task-vector training objective, adding a proximal term before any downstream write-in rule is applied.

3 Method: Shaping LoRA Task Vectors for Mergeability

We study rehearsal-free class-incremental learning with a frozen pretrained backbone and sequentially learned LoRA task vectors. Our goal is to learn task vectors that are not only accurate on the current task, but also easy to write into the running model. We first formulate sequential LoRA write-in, then motivate PRM as a proximal task-vector shaping objective, and finally define the mergeability metrics used in the analysis.

3.1 Sequential LoRA Write-in

A task sequence $\mathcal{T}_1, \ldots, \mathcal{T}_T$ arrives one task at a time. When training task t, examples from previous tasks $\mathcal{T}_{<t}$ are unavailable. Let θ_{t-1} denote the running model before task t. For each adapted projection, fresh LoRA factors A_t, B_t induce an effective task vector

$$\delta_t = s_{\mathrm{LoRA}} B_t A_t, \tag{1}$$

where s_{LoRA} is the LoRA scaling factor. In practice, δ_t denotes the collection of such low-rank updates across all adapted projections. After task-vector training on the current task, this task vector

is written into the running model by

$$\theta_t = \theta_{t-1} + \alpha_t \delta_t, \qquad \alpha_t \in [0, 1]. \tag{2}$$

This one-dimensional write-in operation separates two design questions: which task vector should be learned, and which coefficient should be used to write it into the running model? Existing coefficient-centric write-in rules mainly address the second question: given a learned task vector, how much of it should be written into the running model? Fisher-derived rules such as P&M [Qiu et al., 2025] are one instance of this view. In contrast, PRM focuses on the first question: can task-vector training itself make the induced write-in path more mergeable, so that performance becomes less sensitive to the precise choice of α_t?

3.2 Mergeability and Write-in Curvature

We call δ_t mergeable if this path is stable across a broad range of α coefficients while retaining current-task plasticity. A mergeable task vector should achieve high peak performance at some coefficient, but should not require a narrowly tuned coefficient to be usable.

To motivate the objective, consider the old-task loss $\mathcal{L}_{<t}$ along the write-in path. A local second-order expansion around θ_{t-1} gives

$$\mathcal{L}_{<t}(\theta_{t-1} + \alpha\delta_t) \approx \mathcal{L}_{<t}(\theta_{t-1}) + \alpha g_{<t}^\top \delta_t + \frac{1}{2}\alpha^2 \delta_t^\top H_{<t} \delta_t. \tag{3}$$

The linear term can shift the preferred coefficient, while the quadratic term controls how sharply the old-task loss changes along the write-in path. Thus, the curvature-sensitive component of coefficient sensitivity is governed by $\delta_t^\top H_{<t}\delta_t$.

As in Fisher-based merge methods, we approximate this curvature with a stored diagonal Fisher summary from previous tasks. Let $\bar{F}_{<t}$ denote the aggregate diagonal Fisher information computed from previous tasks before their data are discarded. We define the Fisher-weighted interference proxy

$$q_t = \langle \delta_t, \bar{F}_{<t}\delta_t \rangle. \tag{4}$$

A smaller q_t indicates that the new task vector moves less in directions estimated to be important for previous tasks, and should therefore induce a flatter, less coefficient-sensitive write-in path.

Crucially, q_t is quadratic in δ_t, so the task-vector radius directly controls a norm-based upper bound on the Fisher-weighted curvature term. For positive semidefinite $\bar{F}_{<t}$,

$$q_t \le \|\bar{F}_{<t}\|_{\mathrm{op}} \|\delta_t\|_2^2. \tag{5}$$

Thus, reducing the task-vector radius lowers an upper bound on Fisher-weighted interference. This provides a merge-rule-agnostic way to shape mergeability: lowering this bound reduces the curvature-sensitive variation along the write-in path $\theta_{t-1} + \alpha\delta_t$, and is expected to broaden the near-optimal coefficient plateau $W_{t,\varepsilon}$, provided that current-task plasticity is not over-regularized.

This observation motivates a Fisher-free norm surrogate rather than directly minimizing q_t during task-vector training. Directly optimizing q_t would require additional curvature estimation and would tie the training objective to a particular Fisher machinery, whereas controlling the radius of δ_t is geometry-agnostic. For each LoRA product,

$$\|\delta_t\|_F = s_{\mathrm{LoRA}} \|B_t A_t\|_F \le s_{\mathrm{LoRA}} \|B_t\|_F \|A_t\|_F. \tag{6}$$

The proximal penalty controls the distance of A_t, B_t from their initial values, and therefore controls the factor norms up to constants set by initialization:

$$\|A_t\|_F \le \|A_t^{(0)}\|_F + \|A_t - A_t^{(0)}\|_F, \qquad \|B_t\|_F \le \|B_t^{(0)}\|_F + \|B_t - B_t^{(0)}\|_F.$$

Thus, constraining the LoRA factors provides a practical norm-based surrogate for limiting write-in curvature. This surrogate biases task-vector training toward smaller-radius updates, thereby lowering a norm-based upper bound on Fisher-weighted interference while leaving the write-in coefficient rule unchanged. The full PRM objective and implementation specifics are deferred to Appendix A.

3.3 PRM: Proximal Shaping for Mergeable Task Vectors

As illustrated in Figure 1(b), PRM shapes the task vector during training by adding a proximal penalty to the task-specific LoRA factors. In our main instantiation, this loss uses the same perturbation-aware forward pass as the corresponding perturbation baseline. For current-task data $(x, y) \sim \mathcal{D}_t$ and perturbation $\tilde{\epsilon} \sim \mathcal{P}_\epsilon$, we optimize

$$\begin{aligned}\mathcal{L}_{\text{PRM}} =& \mathbb{E}_{(x,y)\sim\mathcal{D}_t,\, \tilde{\epsilon}\sim\mathcal{P}_\epsilon}\, \mathcal{L}_{\text{CE}}\big(f_{\theta_{t-1}+(1+\tilde{\epsilon})\delta_t(A_t,B_t)}(x), y\big) \\ &+ \lambda_{\text{prox}} \left(\|A_t - A_t^{(0)}\|_F^2 + \|B_t - B_t^{(0)}\|_F^2 \right),\end{aligned} \tag{7}$$

where $(A_t^{(0)}, B_t^{(0)})$ are the initialization values of the task-specific LoRA factors, and $\mathcal{P}_\epsilon$ is the perturbation distribution used by the corresponding perturbation-aware baseline. The proximal penalty acts only on the current task's LoRA factors; it does not regularize or move the running shared model θ_{t-1} during task-vector training.

The cross-entropy term preserves current-task plasticity, while the proximal term limits the radius of the LoRA factor update and therefore lowers a norm-based upper bound on q_t. By reducing the curvature-sensitive variation along $\theta_{t-1} + \alpha\delta_t$, this biases the resulting product $\delta_t(A_t, B_t)$ toward a broader and less coefficient-sensitive write-in path. This is the key distinction from coefficient-only methods: PRM changes the task vector that will be written in, not the definition of the write-in operation.

Note that PRM differs from EWC-LoRA, since it anchors each fresh LoRA proposal to its own initialization rather than regularizing a shared adapter toward previous-task solutions with past-task Fisher weights.

Because PRM is coefficient-rule agnostic, we evaluate two instantiations:

- **PRM-fixed**: uses a constant coefficient $\alpha_t = \alpha_{\text{fix}}$ for all tasks. This is our default variant, because it removes per-task Fisher coefficient estimation and directly tests whether proximal task-vector shaping can make a single coefficient competitive.
- **PRM-Fisher**: uses the Fisher-derived write-in coefficient rule. This variant keeps the downstream write-in rule fixed and isolates the effect of changing the learned task vector.

After task-vector training, both variants perform the same write-in operation in Eq. (2); they differ only in the coefficient rule used for write-in.

3.4 Metrics for Mergeability

We quantify mergeability with two complementary metrics: a path-level plateau-width metric and a vector-level interference metric. We evaluate the write-in path $\theta_t(\alpha)$ on a fixed coefficient grid $\mathcal{A} = \{0.00, 0.05, \ldots, 0.95\}$. Let $\overline{\text{Acc}}_{\le t}(\alpha)$ denote validation accuracy averaged over all tasks seen so far after writing in δ_t with coefficient α. These sweeps are used only for analysis and are never used to select the deployed coefficient in benchmark comparisons.

Stable plateau width. For tolerance ε, the near-optimal plateau width is

$$W_{t,\varepsilon} = \frac{1}{|\mathcal{A}|} \left| \left\{ \alpha \in \mathcal{A} : \overline{\text{Acc}}_{\le t}(\alpha) \ge \max_{\alpha' \in \mathcal{A}} \overline{\text{Acc}}_{\le t}(\alpha') - \varepsilon \right\} \right|. \tag{8}$$

Larger $W_{t,\varepsilon}$ means more coefficients achieve near-optimal seen-task accuracy. Compared with reporting only the best point on the sweep, $W_{t,\varepsilon}$ directly captures the property that motivates mergeability: the write-in path is broadly usable rather than narrowly peaked.

Vector-level interference. At the vector level, we report the Fisher-weighted interference q_t from Eq. (4) together with its direction-normalized counterpart

$$q_t^{\text{dir}} = \frac{q_t}{\|\delta_t\|_F^2 + \varepsilon_{\text{num}}}. \tag{9}$$

Equivalently,

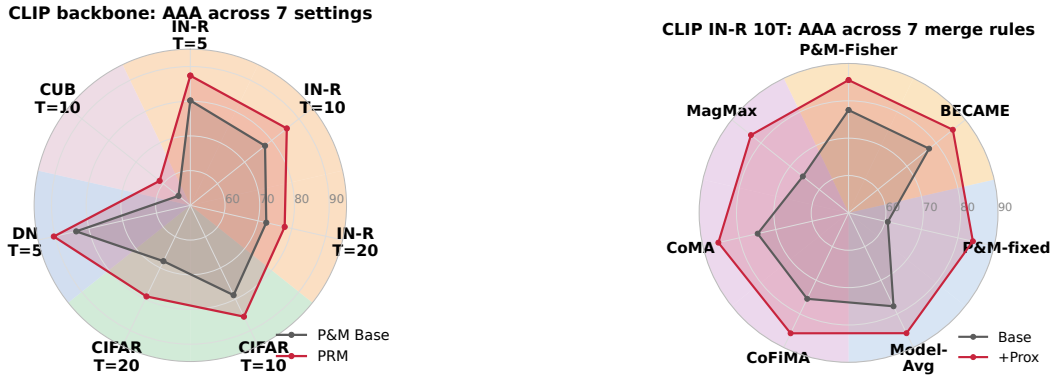
$$q_t = \big(\|\delta_t\|_F^2 + \varepsilon_{\text{num}}\big)\, q_t^{\text{dir}},$$

and for non-negligible $\|\delta_t\|_F$, this is approximately $q_t \approx \|\delta_t\|_F^2 q_t^{\text{dir}}$. This decomposition separates two effects: whether proximal shaping reduces interference by shrinking the update magnitude ($\|\delta_t\|_F^2$ drops), or by reorienting the update away from Fisher-important subspaces (q_t^{dir} drops). The mechanism analyses in §4.4 use this decomposition to attribute the gain.

All mergeability metrics are diagnostic. P&M and PRM-Fisher use the Fisher-derived write-in rule, while PRM-fixed uses a fixed coefficient chosen in advance.

4 Results

To answer the mergeability question posed in §1, we evaluate PRM at three levels: benchmark performance, transfer across settings and merge rules, and path-level attribution. The benchmarks test whether learning more mergeable task vectors makes a fixed write-in coefficient competitive across datasets and task splits. The paired Base→+Prox comparisons test whether the same task-vector change improves several write-in rules and settings. Path-level analyses then explain the effect: coefficient sweeps inspect variation along $\theta_{t-1} + \alpha\delta_t$, norm-matched controls separate direction from scale, the λ_{prox} sweep studies the stability–plasticity trade-off, and ablations attribute the gain across task-vector shaping, coefficient selection, and write-in.



(a) Datasets (CLIP backbone)

(b) Merge rules (CLIP IN-R 10T)

Figure 2: Paired transfer checks for proximal task-vector shaping. **(a)** Dataset-level comparison on the CLIP backbone: PRM improves over the corresponding P&M Base across the plotted settings. **(b)** Merge-rule comparison on CLIP IN-R 10T: the same proximal task-vector-training term improves each reported Base→+Prox pair while leaving the write-in rule unchanged. Full values and setup checks are reported in Appendix D and Appendix C.

4.1 Experimental setup

We evaluate rehearsal-free class-incremental learning on ImageNet-R [Hendrycks et al., 2021] with 5/10/20 tasks, CIFAR-100 [Krizhevsky, 2009] with 10/20 tasks, DomainNet [Peng et al., 2019] with 5 tasks, and CUB-200 [Wah et al., 2011] with 10 tasks. The primary metric is average anytime accuracy (AAA), the mean seen-class accuracy averaged over task boundaries. We report final accuracy and use final forgetting as a complementary stability metric. All numbers are mean±std over three seeds unless otherwise noted.

All benchmark experiments use ViT-B/16 [Dosovitskiy et al., 2021] with rank-10 LoRA adapters on attention key/value projections. The main comparison table uses AugReg checkpoint [Steiner et al., 2021], where baseline coverage is broadest; the paired transfer and path diagnostics use CLIP [Radford et al., 2021] as a diagnostic setting, where the separation between unshaped and proximally shaped updates is clearest. Additional backbone checks and optimization details are reported in Appendix C and Appendix A. PRM uses a single $\lambda_{\text{prox}} = 10^{-2}$ in all benchmark tables and does not tune it per dataset or split.

We compare with sequential parameter-efficient fine-tuning methods (LoRA, EWC-LoRA, Inf-LoRA, SD-LoRA), prompt-based class-incremental methods (L2P and CODA-Prompt where avail-

able), and task-vector merging baselines including Model-Avg, MagMax, CoFiMA, CoMA, BECAME, and P&M. We report two variants: PRM-fixed uses a constant coefficient α=0.8, while PRM-Fisher keeps P&M's Fisher-derived rule to isolate the effect of proximal task-vector shaping. Unless specified, PRM denotes PRM-fixed.

Table 1: Results on ImageNet-R (ViT-B/16 AugReg). **PRM-fixed** and **PRM-Fisher** are ours.

	ImageNet-R					
	T=5		T=10		T=20	
Method	Acc ↑	AAA ↑	Acc ↑	AAA ↑	Acc ↑	AAA ↑
Full-FT	$64.92 \pm .87^{\dagger}$	$75.57 \pm .50^{\dagger}$	$60.57 \pm 1.06^{\dagger}$	$72.31 \pm 1.09^{\dagger}$	$49.95 \pm 1.31^{\dagger}$	$65.32 \pm .69^{\dagger}$
L2P	$73.04 \pm .71^{\dagger}$	$76.94 \pm .61^{\dagger}$	$71.26 \pm .44^{\dagger}$	$76.13 \pm .46^{\dagger}$	$68.97 \pm .51^{\dagger}$	$74.16 \pm .32^{\dagger}$
BECAME	$81.69 \pm .25$	$86.25 \pm .37$	$80.05 \pm .22$	$85.65 \pm .80$	$76.60 \pm .45$	$83.32 \pm .52$
CoFiMA	$81.99 \pm .25$	$\underline{86.37 \pm .47}$	$80.13 \pm .37$	$85.80 \pm .60$	$76.28 \pm .41$	$\underline{83.58 \pm .40}$
CoMA	$81.69 \pm .17$	$86.21 \pm .51$	$79.88 \pm .61$	$85.80 \pm .51$	$75.77 \pm .36$	$83.37 \pm .53$
MagMax	$79.39 \pm .13$	$82.93 \pm .56$	$79.11 \pm .25$	$83.43 \pm .65$	$77.07 \pm .40$	$82.69 \pm .63$
Model-Avg	$81.90 \pm .34$	$86.32 \pm .53$	$80.22 \pm .21$	$85.81 \pm .84$	$76.66 \pm .41$	$83.43 \pm .49$
EWC-LoRA	$81.19 \pm .39$	$84.85 \pm .44$	$78.08 \pm .27$	82.81 ± 1.00	$72.05 \pm .87$	77.45 ± 1.19
Inf-LoRA	$74.77 \pm .25^{\dagger}$	$78.18 \pm .24^{\dagger}$	$74.65 \pm .14^{\dagger}$	$78.15 \pm .24^{\dagger}$	$73.59 \pm .19^{\dagger}$	$77.93 \pm .19^{\dagger}$
SD-LoRA	$79.15 \pm .20^{\dagger}$	$83.01 \pm .42^{\dagger}$	$77.34 \pm .35^{\dagger}$	$82.04 \pm .24^{\dagger}$	$75.26 \pm .37^{\dagger}$	$80.22 \pm .72^{\dagger}$
LoRA	$72.96 \pm .91$	$79.39 \pm .57$	66.54 ± 1.19	$76.72 \pm .81$	56.92 ± 1.17	$72.24 \pm .58$
P&M	$\underline{82.00 \pm .30}$	$\mathbf{86.54 \pm .43}$	$79.96 \pm .51$	$85.55 \pm .90$	$76.59 \pm .47$	$83.17 \pm .40$
PRM-fixed	$\mathbf{82.05 \pm .39}$	$86.03 \pm .52$	$\mathbf{81.32 \pm .25}$	$\mathbf{86.39 \pm .53}$	$\mathbf{78.12 \pm .57}$	$\mathbf{83.70 \pm .81}$
PRM-Fisher	$81.64 \pm .43$	$85.78 \pm .42$	$\underline{80.87 \pm .44}$	$\underline{85.96 \pm .48}$	$\underline{77.93 \pm .30}$	$83.14 \pm .92$

Each cell is mean±std over 3 seeds; †=lifted from [Qiu et al., 2025] Tabs. 2/4/5/9/11. **Best per column bold**; second best underlined.

4.2 Benchmark consequence: a fixed coefficient becomes competitive

Table 1 reports the ImageNet-R comparison on the AugReg backbone. If the learned task vector is stable over a range of write-in coefficients, a single fixed coefficient should not be heavily penalized relative to a Fisher-derived coefficient chosen at each task.

On ImageNet-R, PRM-fixed achieves the best final accuracy in all three task partitions and the best AAA for T=10 and T=20; P&M leads the T=5 AAA column by 0.51 points. These results show that a single fixed coefficient remains competitive with the Fisher-derived rule and performs particularly well on the harder ImageNet-R partitions.

Full AugReg results on CIFAR-100, DomainNet, and CUB-200 are reported in Appendix B. Across the seven reported AugReg settings, PRM-fixed obtains the highest mean AAA (86.85), compared with 86.41 for P&M, while using the same $\alpha = 0.8$ on every task. PRM-Fisher reaches a mean AAA of 86.73 while keeping P&M's Fisher-derived write-in rule. The two variants therefore show that, once task vectors are proximally shaped, a fixed rule can be competitive in practice, and that the gain is not explained solely by changing the coefficient rule.

4.3 Proximal shaping transfers across settings and write-in rules

To test whether proximal shaping transfers beyond P&M, we hold each write-in rule fixed and change only task-vector training: Base uses the original recipe, and +Prox adds the same $\lambda_{\text{prox}} = 10^{-2}$ penalty. We run this diagnostic on CLIP; per-cell values and backbone checks are in Appendix D and Appendix C.

Figure 2(a) shows positive P&M Base→PRM gains across the plotted CLIP settings. Figure 2(b) changes the write-in rule while keeping the proximal modification fixed; every reported Base→+Prox comparison improves on CLIP IN-R 10T, and the +Prox variants occupy a narrower AAA range than their Base counterparts. These paired comparisons locate the gain at task-vector shaping. We next inspect how the resulting task vectors behave along the write-in path $\theta_{t-1} + \alpha\delta_t$.

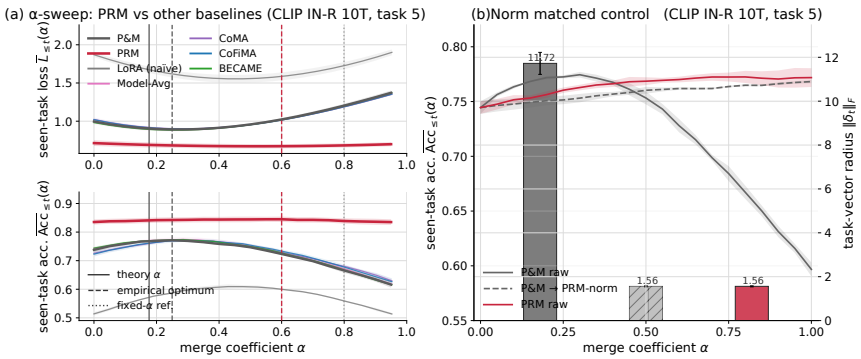


Figure 3: Seen-task loss and accuracy along the write-in coefficient on CLIP ViT-B/16 IN-R 10T at task 5. **(a)** PRM has the lowest loss, highest accuracy, and smallest variation; the fixed $\alpha = 0.8$ lies near its empirical best. **(b)** Rescaling the P&M vector to the PRM norm greatly reduces accuracy variation, indicating that scale explains much of the coefficient sensitivity. Bars show Frobenius norms; curves show seen-task accuracy.

4.4 Write-in paths under coefficient sweeps

For a trained task vector δ_t, we evaluate $\theta_t(\alpha) = \theta_{t-1} + \alpha\delta_t$ over a fixed coefficient grid and measure seen-task loss and accuracy. This checks the mergeability property defined in §3.4: whether many coefficients give similar seen-task performance.

Figure 3(a) reports a representative sweep at task 5 on CLIP ViT-B/16 IN-R 10T. Across the plotted coefficient grid, LoRA gives the largest loss and lowest accuracy, the write-in baselines occupy an intermediate regime, and PRM gives the lowest loss and highest accuracy with the smallest variation. The fixed reference α=0.8 achieves accuracy close to the empirical maximum of the PRM path without searching over α for that task. In this representative sweep, proximal shaping changes the path induced by the task vector.

Norm-matched control. The reduced variation in Figure 3(a) could come from a changed direction, or from traversing a shorter distance because $\|\delta_t\|_F$ is smaller. We use P&M because it is the closest unproximal counterpart to PRM: it shares the perturbation-aware task-vector training but lacks the proximal penalty. To separate direction from scale, we preserve the P&M direction and replace only its Frobenius norm with the PRM norm. Figure 3(b) shows that rescaling the P&M vector from norm 11.72 to 1.56 substantially reduces accuracy variation at large α. The rescaled curve does not coincide with raw PRM, so directional or other task-vector-training effects may still contribute, but scale explains a large part of the coefficient sensitivity in this case. Appendix E reports the corresponding continuous scaling control.

4.5 Proximal strength and the stability–plasticity trade-off

The norm-matched and scaling controls identify task-vector scale as an important source of coefficient sensitivity, but they do not determine how strongly task-vector training should be regularized. We therefore sweep λ_{prox} on CLIP IN-R 10T and measure both diagnostic path metrics and full benchmark performance; the exact grid is reported in Appendix A.

Figure 4(a) reports the diagnostic metrics from §3.4. $W_{t,\varepsilon}$ measures the fraction of coefficients within ε of the best sweep point; q_t is Fisher-weighted interference; and $q_t^{\text{dir}} = q_t/(\|\delta_t\|_F^2 + \varepsilon_{\text{num}})$ separates update scale from direction. As λ_{prox} increases, $W_{t,\varepsilon}$ increases and q_t drops by orders of magnitude, whereas q_t^{dir} is not monotonic. Since $q_t \approx \|\delta_t\|_F^2 q_t^{\text{dir}}$, the reduction in interference is mainly explained by a smaller update norm rather than a monotonic rotation away from Fisher-important subspaces.

Figure 4(b) shows the corresponding benchmark trade-off. As λ_{prox} increases through 10^{-1}, forgetting decreases, while AAA and Acc peak at intermediate values. At $\lambda_{\text{prox}} = 1$, forgetting rises and both accuracy metrics decline, indicating that the proximal penalty has become too strong. We use λ_{prox}=10^{-2} because it lies in the high-performing range before the large-λ_{prox} degra-

dation. Across four AugReg settings and three independently permuted task orders per setting, $\lambda_{\text{prox}} \in [10^{-3}, 3{\times}10^{-2}]$ yields within-setting AAA spreads of only 0.64–1.40 percentage points, while the default 10^{-2} remains within 0.12–0.70 points of the best swept value (Appendix F.1).

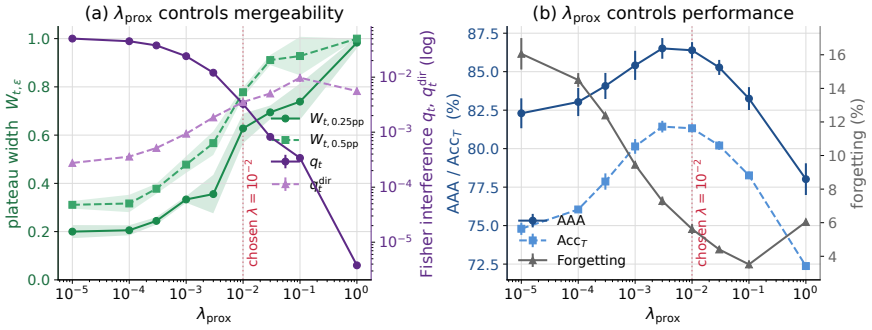


Figure 4: Effect of proximal strength on path diagnostics and benchmark performance. The x-axis is logarithmic and $\lambda_{\text{prox}} = 0$ is mapped to 10^{-5} for visualization. **(a)** As λ_{prox} increases, the fraction of coefficients within 0.25 and 0.5 percentage points of the sweep optimum increases, while q_t decreases by orders of magnitude. The direction-normalized interference q_t^{dir} is not monotonic. **(b)** Forgetting decreases through $\lambda_{\text{prox}} = 10^{-1}$, while AAA and final accuracy peak at an intermediate strength; at $\lambda_{\text{prox}} = 1$, all three metrics degrade.

4.6 Attributing the gain across task-vector and write-in components

Table 2 separates perturbation-aware training, proximal shaping, the coefficient rule, and the final write-in. The lower NoWrite block sets $\alpha = 0$ for $t \geq 1$ and is used only for attribution.

Table 2: Ablation and NoWrite attribution on ImageNet-R 10T (3 seeds, mean±std). $\overline{W}_{t,0.01}$ is near-optimal plateau width and $\overline{q}_t$ is Fisher-weighted interference in 10^{-3} units, averaged over sweep tasks $\{2, 5, 10\}$.

Method	pert	prox	α-rule	AAA	Acc$_T$	Forg	$\overline{W}_{t,0.01}$	$\overline{q}_t$ [10^{-3}]
Upper block: deployable variants								
LoRA baseline	–	–	α=1	76.72±0.81	66.54±1.19	26.66±1.73	0.38	63.94
P&M-fixed	✓	–	α=0.8	82.30±0.97	74.80±0.40	16.06±0.94	0.44	45.12
P&M	✓	–	Fisher	85.55±0.90	79.96±0.51	3.50±0.41	0.44	45.12
LoRA-Prox	–	✓	α=0.8	85.61±0.48	80.27±0.31	7.72±0.62	0.84	4.93
PRM-fixed	✓	✓	α=0.8	86.39±0.53	81.32±0.25	5.62±0.29	0.91	3.26
PRM-Fisher	✓	✓	Fisher	85.96±0.48	80.87±0.44	4.28±0.28	0.91	3.56
Lower block: NoWrite (attribution-only; α=0 for $t \geq 1$)								
LoRA-NoWrite	–	–	α=0	72.88±1.42	65.73±1.12	3.05±0.36	0.38	63.94
P&M-NoWrite	✓	–	α=0	76.64±0.74	69.97±0.98	3.10±0.44	0.44	45.12
LoRA-Prox-NoWrite	–	✓	α=0	82.78±0.33	76.95±0.20	5.00±0.31	0.84	5.62
PRM-NoWrite	✓	✓	α=0	84.45±0.33	78.77±0.07	4.98±0.12	0.91	3.26

Merely merging the original P&M vector with α=0.8 gives AAA= 82.30±0.97; the Fisher-derived P&M rule recovers to 85.55 ± 0.90, while PRM-fixed reaches 86.39 ± 0.53 with the same fixed coefficient. The path diagnostics move in the same direction: PRM-fixed increases $\overline{W}_{t,0.01}$ from 0.44 to 0.91 and reduces $\overline{q}_t$ from 45.12 to 3.26 in 10^{-3} units.

The next comparisons localize the gain within the pipeline. LoRA-Prox, which adds the proximal term without P&M's perturbation-aware objective, already reaches AAA= 85.61±0.48 and reduces $\overline{q}_t$ to 4.93×10^{-3}, suggesting that proximal shaping is the component most directly associated with the mergeability change in this ablation; adding perturbation-aware training improves further to 86.39 ± 0.53. Once the proximal term is present, the coefficient rule has limited remaining effect in

this setting: PRM-fixed and PRM-Fisher differ by only 0.43 AAA points and have nearly identical diagnostics ($\overline{W}_{t,0.01} = 0.91$, $\overline{q}_t = 3.26$ versus 3.56 in 10^{-3} units).

The NoWrite rows isolate performance prior to write-in. The ordering is LoRA-NoWrite (72.88) $<$ P&M-NoWrite (76.64) $<$ LoRA-Prox-NoWrite (82.78) $<$ PRM-NoWrite (84.45) $<$ full PRM-fixed (86.39). The final write-in step still adds 1.94 AAA points over PRM-NoWrite, but PRM-NoWrite is only 1.10 points below full P&M. Thus, much of the improvement is already present in the shaped task vector before it is written into the running model. Overall, the benchmark gains trace primarily to proximal task-vector shaping rather than to a new write-in coefficient rule or to a trivial collapse of the update.

4.7 Computational Cost Analysis

To quantify the additional cost introduced by PRM, we analyze its theoretical operation count and empirical resource usage. Theoretically, the proximal term adds no trainable parameters or additional model passes. With the ViT-B/16 backbone and rank-10 K/V LoRA used in our experiments, it operates on $P_{\mathrm{LoRA}} = 368{,}640$ factor scalars and requires only $\mathcal{O}(P_{\mathrm{LoRA}})$ element-wise operations per optimization step. In comparison, a backbone forward pass requires approximately 17.6 GFLOPs per image, or 4.50 TFLOPs for a batch of 256. The proximal computation therefore accounts for less than 0.0001% of the backbone forward computation.

Empirically, Table 3 profiles the end-to-end training time, peak allocated GPU memory, and persistent auxiliary storage of P&M and PRM under fixed and Fisher-based coefficient rules. Across both coefficient rules, PRM changes the end-to-end training time by at most 0.7% and increases peak allocated memory by at most 1.9 MiB (less than 0.01%). The 81 MiB/task stored by both Fisher variants is required by their shared Fisher-based coefficient rule. Overall, PRM adds negligible training time and peak-memory usage and leaves inference unchanged.

Table 3: Measured computational and storage costs on ImageNet-R 10T using AugReg ViT-B/16 with rank-10 K/V LoRA, batch size 256, 10 training epochs per task, and one NVIDIA A100 GPU. Time is measured end-to-end over all tasks, and memory denotes peak allocated GPU memory.

Method	Time (s)	Peak memory (MiB)	Storage (MiB/task)
P&M-fixed	4397	21131.9	0
PRM-fixed	4379	21133.3	0
P&M-Fisher	4753	25853.5	81
PRM-Fisher	4722	25855.4	81

5 Conclusion and Limitations

We propose PRM, a proximal regularization method for rehearsal-free continual LoRA merging. It shapes task-vector training by penalizing current LoRA factors' distance from initialization, avoiding new merge-coefficient rules. Across multiple experiment settings, PRM consistently improves its base variants and matches or outperforms strong coefficient-based baselines. Diagnostics show that proximal shaping broadens the usable range of merge coefficients, reduces Fisher-weighted interference, and moves task vectors into a smaller-radius write-in regime. These results suggest that effective sequential LoRA merging depends not only on how a task vector is scaled at merge time, but also on whether it has been trained to induce a stable write-in path.

Limitations. Our empirical scope remains limited to rehearsal-free class-incremental vision benchmarks, leaving broader continual-learning regimes, modalities, and longer streams for future study. PRM is also studied mainly under additive LoRA write-in, and its extension to other adapters, full-model updates, or non-additive merging rules remains open. Future work should evaluate PRM in broader settings, develop richer magnitude- and direction-aware task-vector constraints, and further clarify its connection to the local loss landscape.

Acknowledgments and Disclosures

This work was supported by the National Key Research and Development Program of China under Grant 2025ZD0215701; the National Natural Science Foundation of China under Grants T2341003 and U2336214; and the Beijing Natural Science Foundation under Grant L257019. Computational resources were provided by Sapient Intelligence.

We thank Tianshu Sun of the University of Cambridge for her continued involvement in this project and for valuable discussions.

The authors declare no additional competing interests.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Jiangpeng He, Zhihao Duan, and Fengqing Zhu. Cl-lora: Continual low-rank adaptation for rehearsal-free class-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 30534–30544, 2025.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8340–8349, 2021.

Edward J Hu, Yuanzhi Li, Yelong Shen, Shean Wang, Phillip Wallis, Liang Wang, Zeyuan Allen-Zhu, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, pages 875–884, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Mei Li, Yuxiang Lu, Qinyan Dai, Suizhi Huang, Yue Ding, and Hongtao Lu. Became: Bayesian continual learning with adaptive model merging. *arXiv preprint arXiv:2504.02666*, 2025.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. MagMax: Leveraging model merging for seamless continual learning. In *European Conference on Computer Vision*, pages 379–395. Springer, 2024.

Imad Eddine Marouf, Subhankar Roy, Enzo Tartaglione, and Stéphane Lathuilière. Weighted ensemble models are strong continual learners. In *European Conference on Computer Vision*, pages 306–324. Springer, 2024.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*, 2021.

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/CVF international conference on computer vision*, pages 1406–1415, 2019.

Haomiao Qiu, Miao Zhang, Ziyue Qiao, and Liqiang Nie. Train with perturbation, infer after merging: A two-stage framework for continual learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11909–11919, 2023.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European conference on computer vision*, pages 631–648. Springer, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 139–149, 2022b.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-LoRA: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations*, 2025.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

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

Appendix

A Experimental details and reproducibility

This appendix collects the experimental protocol details referenced in the main text. We keep the description at the level needed for reproducibility, and avoid implementation-specific names.

A.0 Code availability. Code and configuration files are available at `https://github.com/Edward61xuan/PRM-Continual-LoRA`.

A.1 Task construction and evaluation. We evaluate rehearsal-free class-incremental learning on four image datasets and seven task splits: ImageNet-R with 5, 10, and 20 tasks; CIFAR-100 with 10 and 20 tasks; DomainNet with 5 tasks; and CUB-200 with 10 tasks. For each (dataset, T) pair we use a fixed class permutation that is shared across all in-house methods so that comparisons are paired at the seed level; previous-task data are not available during task-vector training for any subsequent task. Lifted baselines marked with † in the result tables are reused under the matching protocol of Qiu et al. [2025].

Let $a_{t,j} \in [0,1]$ denote the validation accuracy on task j measured after training through task t (with $a_{t,j}=0$ by convention for $t<j$). We report average anytime accuracy (AAA), final accuracy (ACC_T), and final forgetting:

$$\mathrm{AAA} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{t}\sum_{j=1}^{t} a_{t,j}, \qquad \mathrm{Acc}_T = \frac{1}{T}\sum_{j=1}^{T} a_{T,j},$$

$$\mathrm{Forg} = \frac{1}{T-1}\sum_{j=1}^{T-1}\Big(\max_{t\in\{j,\dots,T\}} a_{t,j} - a_{T,j}\Big). \tag{10}$$

The maximum in the forgetting expression is taken over all training times that have observed task j, including the final state $t{=}T$, which makes the per-task forgetting non-negative. This matches the non-negative variant of Chaudhry et al. [2018] and is the metric used in our codebase. All tables in the main text and appendix report mean±std over three seeds unless explicitly marked otherwise.

A.2 Backbone, LoRA, head, and optimizer. The vision backbone is a ViT-B/16 [Dosovitskiy et al., 2021] frozen during continual training. We use the AugReg checkpoint [Steiner et al., 2021] `vit_base_patch16_224.augreg2_in21k_ft_in1k` for the main benchmark study and the CLIP-OpenAI checkpoint [Radford et al., 2021] `vit_base_patch16_clip_224.openai` for the CLIP transfer study; the two backbones share the same training pipeline and differ only in the checkpoint and the cosine-head temperature (30 for AugReg, 28 for CLIP-OpenAI). Each adapted attention layer carries rank-10 LoRA factors (A_t, B_t) on the key and value projections, with A_t initialised by Kaiming-uniform (gain $\sqrt{5}$) and B_t initialised to zero so that the new task contributes the zero write-in at $t{=}0$ epochs. We apply no additional LoRA scaling, i.e. the effective task vector is $\delta_t = B_t A_t$; expressed in the notation of Eq. (1) this corresponds to $s_{\mathrm{LoRA}}{=}1$. Only the per-task LoRA factors and the per-task linear classifier head are updated during task-vector training. Task-vector training uses AdamW [Loshchilov and Hutter, 2019] with base learning rate 10^{-3}, classifier-head learning-rate multiplier 10 (so the head trains at 10^{-2}), batch size 256, and a multi-step learning-rate schedule with no warmup; the default schedule runs for 10 epochs per task on every dataset except DomainNet, which uses 5 epochs per task. When the P&M perturbation objective is used, the perturbation magnitude is fixed at $\epsilon{=}0.5$ with sampling probability $p_0{=}0.33$, following Qiu et al. [2025]; the perturbed forward pass is then $f_{\theta_{t-1}+(1+\tilde{\epsilon})\delta_t}(x)$.

A.3 PRM objective and algorithm. For current-task data $(x,y)\sim\mathcal{D}_t$ and perturbation $\tilde{\epsilon}\sim\mathcal{P}_\epsilon$, PRM optimizes the following task-vector-training objective:

$$\mathcal{L}_{\mathrm{PRM}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t,\,\tilde{\epsilon}\sim\mathcal{P}_\epsilon}\mathcal{L}_{\mathrm{CE}}\big(f_{\theta_{t-1}+(1+\tilde{\epsilon})\delta_t(A_t,B_t)}(x),y\big) + \lambda_{\mathrm{prox}}\left(\|A_t - A_t^{(0)}\|_F^2 + \|B_t - B_t^{(0)}\|_F^2\right),$$

where $(A_t^{(0)}, B_t^{(0)})$ are the LoRA-factor initializations. The proximal term acts only on the current task's LoRA factors; θ_{t-1} is not regularized during task-vector training. We use $\lambda_{\mathrm{prox}}{=}10^{-2}$ in all main-table experiments; the proximal-strength sweep grid for the diagnostic in §4.5 is listed in paragraph A.5. Algorithm 1 summarizes the per-task pipeline.

Algorithm 1 PRM for rehearsal-free sequential LoRA merging.

Require: Pretrained backbone θ_0; task stream $\{\mathcal{D}_t\}_{t=1}^{T}$; proximal weight λ_{prox}; coefficient rule (PRM-fixed: α_t=0.8; PRM-Fisher: the Fisher rule below).
Initialise running model with θ_0 (and an empty Fisher summary $\bar{F}_0 \leftarrow 0$ if the coefficient rule uses Fisher information).
for $t = 1, \ldots, T$ **do**

Step 1 (initialise task t). Freeze θ_{t-1}. For every adapted attention layer ℓ, allocate a fresh LoRA pair $(A_t^{(0)}, B_t^{(0)})$ with $A_t^{(0)} \sim \text{Kaiming-Uniform}(\sqrt{5})$ and $B_t^{(0)} = 0$ so that the initial task vector $\delta_t^{(0)}{=}B_t^{(0)}A_t^{(0)}{=}0$ and the model is unchanged before training. Allocate a fresh per-task linear classifier head.

Step 2 (train the task-vector proposal). Optimise (A_t, B_t) and the head on $\mathcal{D}_t$ with AdamW for the configured number of epochs:

(2a) At each step, sample a perturbation $\tilde{\epsilon} \sim \mathcal{P}_\epsilon$ (zero with probability $1{-}p_0$); compute logits at $\theta_{t-1} + (1{+}\tilde{\epsilon})B_tA_t$ and the cross-entropy loss $\mathcal{L}_{\text{CE}}$.

(2b) Add the proximal regulariser $\lambda_{\text{prox}}\big(\|A_t{-}A_t^{(0)}\|_F^2{+}\|B_t{-}B_t^{(0)}\|_F^2\big)$ on the current-task LoRA factors; the running shared model θ_{t-1} is not regularised.

(2c) Backpropagate and apply the AdamW update.

Form the effective task vector $\delta_t \leftarrow B_tA_t$ across all adapted projections.

Step 3 (choose the write-in coefficient).

• PRM-fixed: set α_t=0.8.

• PRM-Fisher: estimate the per-task diagonal Fisher $\hat{F}_t$ on $\mathcal{D}_t$ over the fused QKV projection weights by averaging $\big(\nabla_{W_{\text{qkv}}^{(\ell)}} \mathcal{L}_{\text{CE}}(x,y)\big)^2$ over the task data, and form the closed-form Fisher rule from Qiu et al. [2025]:

$$\alpha_t = -\frac{\displaystyle\sum_{\ell}\sum_{i=0}^{t}\big\langle F_i^{(\ell)} \odot \delta_t^{(\ell)},\ W_{<t}^{(\ell)} - \widetilde{W}_i^{(\ell)} - \delta_t^{(\ell)}\big\rangle}{\displaystyle\sum_{\ell}\sum_{i=0}^{t}\big\langle F_i^{(\ell)} \odot \delta_t^{(\ell)},\ \delta_t^{(\ell)}\big\rangle},$$

where the sum runs over adapted layers ℓ and over previously observed tasks $i \in \{0, \ldots, t\}$; $W_{<t}^{(\ell)}{=}\sum_{j<t}\alpha_j\, B_jA_j$ is the accumulated LoRA delta on layer ℓ before the current task; $\widetilde{W}_i^{(\ell)}{=}\sum_{j<i}\alpha_j\, B_jA_j$ is the accumulated delta evaluated at the snapshot when task i was trained; and $\odot$ is the elementwise product on $\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. The Fisher diagonal is sliced from the K and V row blocks of the QKV weight matrix.

Step 4 (write back & evaluate). Update the running model $\theta_t \leftarrow \theta_{t-1} + \alpha_t\, \delta_t$, and (when needed by the coefficient rule) accumulate the diagonal Fisher summary by storing $\hat{F}_t$ alongside $\bar{F}_{t-1}$. Evaluate θ_t on the validation splits of all tasks $j \le t$ to populate the row $a_{t,\cdot}$ of the accuracy matrix used in Eq. (10).

end for

A.4 Compared methods and implementation sources. Table 4 groups the methods compared in §4.2 and §4.3 by their task-vector-training objective and write-in rule. All in-house runs use the same task splits, backbone, LoRA rank, optimizer, and evaluation protocol described in paragraphs A.1–A.2. Cells marked with † in result tables are lifted under the matching protocol of Qiu et al. [2025]; cells without ± aggregate a single available seed. +Prox rows in Appendix D keep the listed write-in rule and only modify task-vector training by adding the proximal penalty with the same $\lambda_{\text{prox}}{=}10^{-2}$.

A.5 Diagnostic protocols and path metrics. Three diagnostic axes are reused throughout the paper. The *coefficient (α) sweep* freezes a (θ_{t-1}, δ_t) pair and evaluates the merged model at every $\alpha \in \{0.00, 0.05, \ldots, 0.95\}$ on the validation splits of all tasks seen so far. The λ_{prox} *sweep* used in Figure 4 of the main text uses the grid $\{0,\ 10^{-4},\ 3{\cdot}10^{-4},\ 10^{-3},\ 3{\cdot}10^{-3},\ 10^{-2},\ 3{\cdot}10^{-2},\ 10^{-1},\ 1\}$; path metrics from these sweeps are averaged over tasks $\{2, 5, 10\}$ on ImageNet-R 10T when reported. The *NoWrite probe* is a diagnostic-only write-in rule that sets $\alpha_t{=}0$ at the write-in step:

Table 4: Compared methods and implementation sources. All in-house runs share the protocol of paragraphs A.1–A.2. Dagger entries in result tables are lifted under the matching protocol of Qiu et al. [2025]; cells without $\pm$ aggregate a single available seed. "Same task-vector training" is shorthand for the perturbation-aware LoRA recipe used by P&M; "+ proximal" adds the penalty in Eq. (7).

Method	**Task-vector training**	**Write-in rule**	**Source**
Full-FT	joint update of all parameters	no merge (single model)	in-house
LoRA	per-task LoRA, no perturbation	write-in with α=1	in-house
EWC-LoRA	shared LoRA + EWC penalty	write-in with α=1	in-house
L2P, DualPrompt, CODA-Prompt	prompt-pool training (no LoRA merge)	no merge (prompt routing)	lifted from Qiu et al. [2025] where dagger-marked
Inf-LoRA	subspace-projected LoRA training	subspace consolidation	lifted from Qiu et al. [2025] where dagger-marked
SD-LoRA	soft-router LoRA training	router-weighted consolidation	lifted from Qiu et al. [2025] where dagger-marked
P&M	perturbation-aware LoRA	Fisher-derived α_t [Qiu et al., 2025]	in-house
P&M-fixed	same task-vector training as P&M	fixed α_t=0.8	in-house
Model-Avg	same task-vector training as P&M	uniform model averaging	in-house
CoMA	same task-vector training as P&M	continual model averaging [Marouf et al., 2024]	in-house
CoFiMA	same task-vector training as P&M	Fisher-weighted CoMA [Marouf et al., 2024]	in-house
BECAME	same task-vector training as P&M	adaptive-coefficient merging [Li et al., 2025]	in-house
MagMax	same task-vector training as P&M	maximum-magnitude selection [Marczak et al., 2024]	in-house
PRM-fixed	same task-vector training + proximal penalty	fixed α_t=0.8	in-house (this work)
PRM-Fisher	same task-vector training + proximal penalty	Fisher-derived α_t [Qiu et al., 2025]	in-house (this work)
+Prox (rule)	task-vector training of (rule) + proximal penalty	write-in rule of (rule), unchanged	in-house (this work)

task-vector training still produces a per-task LoRA proposal, but the proposal is not permanently accumulated into the running model. We use NoWrite only to attribute gains to task-vector shaping rather than to write-in coefficient choices (Appendix F.2); it is not a deployable continual method.

For an α grid $\mathcal{A}$ and seen-task accuracy curve $\overline{\mathrm{Acc}}_{\le t}(\alpha)$, the path-level metrics are

$$
\begin{aligned}
R_t &= \max_{\alpha\in\mathcal{A}} \overline{\mathrm{Acc}}_{\le t}(\alpha) - \min_{\alpha\in\mathcal{A}} \overline{\mathrm{Acc}}_{\le t}(\alpha),\\
W_{t,\varepsilon} &= \tfrac{1}{|\mathcal{A}|}\big|\{\alpha : \overline{\mathrm{Acc}}_{\le t}(\alpha) \ge \max_{\alpha'} \overline{\mathrm{Acc}}_{\le t}(\alpha') - \varepsilon\}\big|,\\
G_{t,\mathrm{fix}}(\alpha_r) &= \max_{\alpha\in\mathcal{A}} \overline{\mathrm{Acc}}_{\le t}(\alpha) - \overline{\mathrm{Acc}}_{\le t}(\alpha_r).
\end{aligned}
$$

The tolerance ε in $W_{t,\varepsilon}$ is in fractional accuracy: $W_{t,0.0025}$ is the fraction of the grid within 0.25 percentage points of the optimum. Cross-(seed, task) aggregates always average over tasks within a seed first and then report mean and standard deviation across seeds, so that task spread does not leak into seed error bars.

A.6 Diagonal Fisher summary and task-vector statistics. For methods requiring Fisher information, we maintain a diagonal Fisher summary over the adapted parameters following Qiu et al.

[2025]. The same stored summary is used for the Fisher-derived coefficient in P&M and PRM-Fisher and for the diagnostic interference score $q_t = \langle \delta_t, \bar{F}_{<t}\delta_t \rangle$ in Eq. (4); it is updated at task boundaries and is not recomputed during the α-sweep diagnostics. Our implementation stores the Fisher diagonal for the full fused-QKV matrix (81 MiB per task); the coefficient computation uses its K/V row blocks. Each task-vector proposal additionally yields the Frobenius norm $\|\delta_t\|_F$ and the direction-normalized interference $q_t^{\mathrm{dir}}{=}q_t/(\|\delta_t\|_F^2+\varepsilon_{\mathrm{num}})$. For non-negligible $\|\delta_t\|_F$, this gives the approximate decomposition $q_t \approx \|\delta_t\|_F^2\, q_t^{\mathrm{dir}}$ used in §3.4. The matched-prefix interventions in Appendix E freeze a single shared prefix and train two twin task-vector proposals (P&M and PRM) from identical initialization, data order, and randomness, differing only in the proximal weight; the resulting twin task vectors are then post-hoc rescaled or norm-swapped and evaluated on the same α grid, so the observed differences cannot be attributed to a different history of previous merges.

A.7 Compute. All in-house experiments were run on NVIDIA A100-SXM4 GPUs with 80 GB memory, using one visible GPU per seed. A full PRM run takes approximately 2.5 GPU-hours for ImageNet-R 10T, 2.4 GPU-hours for CIFAR-100 10T, 2.6 GPU-hours for DomainNet 5T, and 0.4 GPU-hours for CUB-200 10T. The total compute for the reported in-house experiments was approximately 625 GPU-hours. Preliminary sweeps and failed or aborted runs used approximately 26 additional GPU-hours.

A.8 Baseline implementation details. Each baseline is described below with its origin, the principle of its update rule, and whether the result was produced by our own implementation under the protocol of paragraphs A.1–A.2 or by running the authors' released codebase. Lifted numbers (marked † in result tables) reuse the protocol of Qiu et al. [2025].

Full-FT. *In-house.* Standard joint fine-tuning of the entire backbone on each task; no merging is performed and no parameter is shared across tasks beyond the running model itself. We used Full-FT only as a non-merging reference; the dagger-marked Full-FT cells are lifted from Qiu et al. [2025].

LoRA. *In-house.* A per-task LoRA adapter is trained with the optimizer settings of paragraph A.2 and written into the running model with $\alpha_t{=}1$. No perturbation, no proximal term, no Fisher. This is the simplest naive sequential-LoRA baseline.

EWC-LoRA. *In-house, following Kirkpatrick et al. [2017] adapted to LoRA.* A *single* shared LoRA adapter is reused across tasks (the per-task allocation step in Algorithm 1 is skipped), and at task t the loss is $\mathcal{L}_{\mathrm{CE}}+\frac{\lambda_{\mathrm{ewc}}}{2}\sum_{s<t}\sum_{\theta} F_s[\theta](\theta-\theta_s^*)^2$, where θ_s^* is the snapshot of the shared LoRA parameters after task s and F_s is the per-task diagonal Fisher computed at θ_s^*. No additional consolidation is applied; the model at task t is the EWC-regularised solution itself.

Inf-LoRA [Liang and Li, 2024]. *Lifted only.* All Inf-LoRA cells in Tables 1, 5 are taken from Qiu et al. [2025]'s comparison tables, which in turn cite the numbers reported in the original Inf-LoRA paper. Those numbers were obtained under the Inf-LoRA paper's own protocol, namely the `vit_base_patch16_224_in21k` backbone (no AugReg ImageNet-1k fine-tune), the upstream class ordering (alphabetical class folder order, `shuffle=false`), and the upstream epoch budget (50 epochs/task on ImageNet-R, 20 epochs/task on CIFAR-100). We did not re-run Inf-LoRA in our protocol. Aligning its DualGPM subspace projection to a common AugReg backbone with 10 epochs/task without changing the algorithm is non-trivial: the SVD-based projector initialization in Liang and Li [2024] depends on the specific backbone's feature statistics, and the upstream learning-rate schedule was tuned to the upstream epoch budget. We therefore follow Qiu et al. [2025]'s convention of reporting Inf-LoRA's published numbers as a reference rather than as a strictly protocol-matched cell, and we mark every such cell with †.

SD-LoRA [Wu et al., 2025]. *Lifted only.* All SD-LoRA cells in Tables 1, 5 are likewise taken from Qiu et al. [2025]'s tables, which in turn cite the original SD-LoRA paper. Those numbers use SD-LoRA's published protocol with the `vit_base_patch16_224` (ImageNet-1k pretrain) backbone and the upstream optimizer / epoch budget, neither of which matches our AugReg + 10-epoch setup. We did not re-run SD-LoRA; cells are marked with †.

L2P, DualPrompt, CODA-Prompt. *Lifted only*; we did not run these methods ourselves. Cells appear in the main results tables only when a †-marked cell is available from Qiu et al. [2025] under the matching protocol.

P&M [Qiu et al., 2025]. *In-house re-implementation.* Per-task LoRA training with the perturbation objective of Qiu et al. [2025]: at each step the forward pass is taken at $\theta_{t-1} + (1+\tilde{\epsilon})B_tA_t$ with $\tilde{\epsilon} \sim \mathcal{P}_\epsilon$ (zero with probability $1-p_0$), and the cross-entropy loss is back-propagated through the perturbed forward. At the end of task-vector training, the Fisher-derived coefficient α_t from Step 3 of Algorithm 1 is used for write-in. Our re-implementation matches the published P&M protocol to within seed noise on ImageNet-R.

P&M-fixed. *In-house.* Same task-vector training as P&M; the write-in rule replaces the Fisher coefficient by a constant $\alpha_t{=}0.8$ across all tasks.

Model-Avg. *In-house re-implementation.* Sequential model averaging: treating the per-task LoRA proposal as $\theta_t^*{=}\theta_{t-1} + \delta_t$, the running mean update $\theta_t = \frac{t}{t+1}\theta_{t-1} + \frac{1}{t+1}\theta_t^*$ collapses to the scalar schedule $\alpha_t{=}1/(t{+}1)$ in our scalar write-in interface. We use the in-house P&M task-vector training for δ_t and only change the write-in schedule.

CoMA [Marouf et al., 2024]. *In-house re-implementation.* Continual model averaging: the running mean form above with a constant scalar gate. Task-vector training is the same as P&M; the write-in rule uses the CoMA scalar schedule.

CoFiMA [Marouf et al., 2024]. *In-house re-implementation.* Fisher-weighted continual model averaging. In our LoRA setting the per-element CoFiMA gate $g[j] = (aF_t[j])/((1-a)F_{<t}[j] + aF_t[j])$ (with a a hyper-parameter) replaces the scalar schedule of CoMA, applied elementwise to δ_t before write-in. Task-vector training is the same as P&M.

BECAME [Li et al., 2025]. *In-house re-implementation.* Closed-form scalar coefficient $\alpha^*{=}\sum_j F_t[j]\,\delta[j]^2 / \sum_j (F_t[j]{+}F_{<t}[j])\,\delta[j]^2$, where $F_t, F_{<t}$ are the diagonal Fishers on the K and V row blocks, and $\delta = \delta_t$. Task-vector training is the same as P&M. We followed the closed-form merge derivation in the BECAME public release (`https://github.com/limei0818/BECAME`).

MagMax [Marczak et al., 2024]. *In-house re-implementation.* Per-coordinate maximum-magnitude task-vector merging: at each layer and each coordinate j, the merged delta picks $\delta_t[j]$ from the task with the largest $|\delta_t[j]|$. We realise this in the scalar write-in interface by setting $\alpha_t{=}1$ together with a per-coordinate gate that is one-hot in the winning task and zero elsewhere; the gate of every past task is recomputed at every write-in because a coordinate may change owner. Task-vector training is the same as P&M.

PRM (this work). *In-house.* Algorithm 1 with $\lambda_{\text{prox}}{=}10^{-2}$. PRM-fixed and PRM-Fisher share task-vector training and differ only in Step 3.

+Prox (rule). *In-house.* It uses P&M-style task-vector training with the proximal penalty added; the write-in step uses the merge rule of the named baseline (CoFiMA, CoMA, BECAME, MagMax, Model-Avg, or P&M-fixed/P&M). Used in the universality study (Appendix D).

A.9 Datasets and class-incremental splits. We use four image datasets, all under the standard class-incremental protocol: each class appears in exactly one task and the class-to-task assignment is fixed before training.

ImageNet-R [Hendrycks et al., 2021]. 200 classes drawn from artistic renditions, sketches, paintings, and other domain shifts of ImageNet categories. We use the publicly released ImageNet-R split as redistributed by Qiu et al. [2025] for direct protocol compatibility, with the 200 classes re-grouped into $T \in \{5, 10, 20\}$ tasks of $\{40, 20, 10\}$ classes each.

CIFAR-100 [Krizhevsky, 2009]. 100 image classes at $32{\times}32$ resolution. Images are upsampled to the backbone's $224{\times}224$ input size with bilinear interpolation. We follow the class-incremental split protocol used in Qiu et al. [2025], with $T \in \{10, 20\}$ tasks containing $\{10, 5\}$ classes each.

S-DomainNet [Peng et al., 2019]. We use the 200-class S-DomainNet subset specified by the train/test manifests distributed with the P&M codebase. It contains 47,532 training images and 20,450 evaluation images spanning all six domains: clipart, infograph, painting, quickdraw, real, and sketch. We partition the 200 classes into five class-incremental tasks of 40 classes each, pooling the available domains within each task.

CUB-200 [Wah et al., 2011]. The Caltech-UCSD Birds-200 dataset with 200 fine-grained bird species. We use the standard train/test split of the public release, re-grouped into $T{=}10$ tasks of 20 classes each.

Class permutation per seed. For each (dataset, T, seed) triple, the 200 (or 100) classes are permuted by a PCG64 random number generator instantiated from the seed. The first C/T classes are assigned to task 1, the next C/T to task 2, and so on, where C is the number of classes. The same permutation is reused across every method compared in-house, so that paired (seed, task) cells are directly comparable. Lifted-from-Qiu-et-al. baselines are run under the matching protocol of Qiu et al. [2025], which uses the same family of permutations on the same class lists.

Train/validation handling. Within each task we use the dataset's standard training set for task-vector training and the dataset's standard validation/test set restricted to the classes seen so far for evaluation. No samples from previous tasks' training sets are revisited (rehearsal-free). Preprocessing follows the timm defaults associated with the AugReg or CLIP-OpenAI ViT-B/16 checkpoint used by the run; only the input resize and normalisation constants differ between the two backbones.

B Extended benchmark results on AugReg

Table 5 reports the per-dataset breakdown referred to in §4.2: CIFAR-100 with 10/20 tasks, DomainNet 5T, and CUB-200 10T. Conventions match Table 1.

Table 5: Main results on CIFAR-100, DomainNet, and CUB-200 (ViT-B/16 AugReg).

	CIFAR-100				DomainNet		CUB-200	
	T=10		T=20		T=5		T=10	
Method	Acc ↑	AAA ↑	Acc ↑	AAA ↑	Acc ↑	AAA ↑	Acc ↑	AAA ↑
Full-FT	$69.49 \pm .50^{\dagger}$	$80.35 \pm .87^{\dagger}$	$51.21 \pm 1.26^{\dagger}$	$61.83 \pm 1.20^{\dagger}$	$51.46 \pm .47^{\dagger}$	$67.08 \pm 1.13^{\dagger}$	$51.43 \pm 1.41^{\dagger}$	$69.74 \pm .93^{\dagger}$
L2P	$83.18 \pm 1.20^{\dagger}$	$87.69 \pm 1.05^{\dagger}$	$\underline{85.66 \pm .16}^{\dagger}$	$\mathbf{90.60 \pm .23}^{\dagger}$	$70.26 \pm .25^{\dagger}$	$75.83 \pm .98^{\dagger}$	$65.18 \pm 2.49^{\dagger}$	$76.12 \pm 1.27^{\dagger}$
BECAME	$87.25 \pm .49$	91.22 ± 1.76	$84.82 \pm .48$	89.89 ± 1.35	83.47 ± 1.21	87.54 ± 1.52	69.36 ± 1.03	$76.95 \pm .81$
CoFiMA	$86.90 \pm .19$	90.99 ± 1.68	$84.91 \pm .10$	$\underline{90.23 \pm 1.22}$	$83.71 \pm .95$	87.62 ± 1.17	69.43 ± 1.15	77.10 ± 1.55
CoMA	$85.92 \pm .26$	90.38 ± 2.06	$83.46 \pm .30$	89.52 ± 1.21	$83.59 \pm .67$	87.63 ± 1.12	70.65 ± 1.85	$79.05 \pm .50$
MagMax	$84.97 \pm .72$	87.59 ± 2.27	$84.74 \pm .43$	88.02 ± 1.99	$80.20 \pm .76$	82.34 ± 1.24	70.95 ± 2.14	$78.46 \pm .75$
Model-Avg	87.16 ± 1.02	90.91 ± 2.43	$84.74 \pm .89$	89.82 ± 1.63	$83.45 \pm .93$	87.51 ± 1.18	70.58 ± 1.54	$78.81 \pm .93$
EWC-LoRA	$85.95 \pm .23$	89.54 ± 1.33	$81.79 \pm .83$	85.11 ± 2.39	$83.88 \pm .36$	$85.27 \pm .83$	46.76 ± 3.19	63.95 ± 1.14
Inf-LoRA	$86.75 \pm .35^{\dagger}$	$91.72 \pm .15^{\dagger}$	$82.18 \pm .88^{\dagger}$	$88.35 \pm .77^{\dagger}$	$71.59 \pm .23^{\dagger}$	$78.29 \pm .50^{\dagger}$	$\underline{76.68 \pm .57}^{\dagger}$	$\underline{81.39 \pm .14}^{\dagger}$
SD-LoRA	$\underline{88.01 \pm .31}^{\dagger}$	$\mathbf{92.54 \pm .18}^{\dagger}$	–	–	$72.82 \pm .37^{\dagger}$	$78.44 \pm .66^{\dagger}$	$\mathbf{77.48 \pm .20}^{\dagger}$	$\mathbf{85.59 \pm .44}^{\dagger}$
LoRA	$73.46 \pm .41$	81.61 ± 2.31	63.99 ± 1.30	78.55 ± 1.69	$72.46 \pm .51$	$77.70 \pm .74$	64.43 ± 2.13	$73.82 \pm .40$
P&M	$87.46 \pm .56$	91.23 ± 1.85	$84.85 \pm .58$	89.97 ± 1.39	$83.87 \pm .84$	87.90 ± 1.02	72.73 ± 1.87	$80.54 \pm .42$
PRM-fixed	$88.10 \pm .52$	92.03 ± 1.03	$85.43 \pm .68$	89.65 ± 1.72	$\underline{84.74 \pm .46}$	$\mathbf{89.33 \pm .59}$	73.65 ± 1.89	$80.81 \pm .43$
PRM-Fisher	$\mathbf{88.25 \pm .40}$	$\underline{92.08 \pm .93}$	$\mathbf{85.87 \pm .54}$	89.87 ± 1.64	$\mathbf{84.76 \pm .40}$	$\underline{89.26 \pm .52}$	74.07 ± 1.66	$81.04 \pm .74$

Each cell is mean±std over 3 in-house seeds; †=lifted from Qiu et al. 2025 NeurIPS Tabs. 2/4/5/9/11 (matching protocol); cells lacking "±"=single available seed; "–"=data unavailable. **Best** per column bold; <u>second best</u> underlined.

Across these AugReg settings, PRM remains competitive with the strongest sequential-merging baselines and improves over P&M on most settings, with the clearest gains on DomainNet 5T and CUB-200 10T. Among sequential-merging baselines, PRM-fixed or PRM-Fisher achieves the strongest or near-strongest AAA in most cells. The lifted Inf-LoRA and SD-LoRA numbers are included for reference under the matching protocol; since they use a different parameter- isolation style, we avoid treating them as part of the same merging-rule comparison.

C Backbone transfer: AugReg and CLIP

The main paper reports AugReg in Table 1 and uses CLIP for the transfer diagnostics in §4.3. Here we collect paired ImageNet-R 10T comparisons where both backbones have Base and +Prox runs (Table 6). The five-backbone radar further includes supervised ImageNet-21K, DINO [Caron et al., 2021], and MAE [He et al., 2022] ViT checkpoints.

AugReg starts from a stronger Base than CLIP. The gain from +Prox is therefore smaller on AugReg but remains positive. On the fixed α=0.8 rule, the AAA gain is $+4.09$ on AugReg and $+23.36$ on CLIP; on the Fisher rule, the gain is $+0.41$ on AugReg and $+8.02$ on CLIP. The five-backbone radar in Figure 5 extends this view beyond AugReg and CLIP to Sup-21K, DINO, and MAE, showing that PRM improves over P&M on every reported backbone, with the largest lifts on the backbones

Table 6: Backbone-paired ImageNet-R 10T comparison. Forgetting reduction is Forg(Base) − Forg(+Prox), so positive means improvement.

Write-in rule	Backbone	AAA		ΔAAA	FORG		ΔFORG
		Base	+Prox		Base	+Prox	(reduction)
P&M (α=0.8)	AugReg	82.30 ± 0.97	86.39 ± 0.53	+4.09	16.06 ± 0.94	5.62 ± 0.29	+10.44
P&M (α=0.8)	CLIP	60.77 ± 0.02	84.13 ± 0.90	+23.36	52.41 ± 1.19	12.77 ± 0.69	+39.64
P&M (Fisher α)	AugReg	85.55 ± 0.90	85.96 ± 0.48	+0.41	3.50 ± 0.41	4.28 ± 0.28	-0.78
P&M (Fisher α)	CLIP	77.53 ± 0.73	85.55 ± 0.60	+8.02	7.72 ± 1.09	5.93 ± 0.81	+1.79

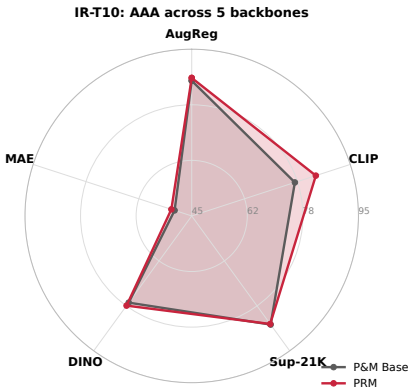


Figure 5: Five-backbone radar on ImageNet-R 10T. Each axis is one backbone (AugReg, CLIP, Sup-21K, DINO, MAE) and the radial value is AAA. The two polylines compare P&M Base against PRM; PRM improves over P&M on every reported backbone, with the largest lift on the backbones whose unshaped Base is the lowest.

whose unshaped Base AAA is the lowest. This supports the view that proximal shaping is most useful when the unregularized write-in path is brittle.

Full CLIP main results. Tables 7 and 8 give the full CLIP-backbone counterparts of Tables 1 and 5, restricted to the merge-baseline rows we ran in-house on CLIP (BECAME, CoFiMA, CoMA, MagMax, Model-Avg, P&M). We did not run Inf-LoRA, SD-LoRA, L2P, EWC-LoRA, LoRA, or Full-FT on CLIP, so those rows are omitted; cells where a particular merge baseline did not finish are rendered as "–". Both PRM-fixed and PRM-Fisher achieve the best AAA among the in-house CLIP merge baselines in every reported CLIP cell: on the seven (dataset, T) settings, PRM-Fisher beats the strongest in-house CLIP merge baseline by between 5.3 and 11.0 pp, and PRM-fixed by between 2.0 and 8.4 pp. The largest gains occur on the harder splits (CIFAR-100 T=20 and IN-R T=20).

D Transfer beyond P&M

Appendix D tests whether the same proximal task-vector-training modification transfers beyond the P&M write-in coefficient rule. For each merge rule, Base denotes the original training recipe and +Prox adds $\lambda_{\text{prox}}=10^{-2}$ while leaving the write-in rule unchanged.

The full per-cell tables are reported in Tables 9–13. Conventions match Table 9. Cells marked with † aggregate a single available seed and should be interpreted as directional checks rather than fully powered estimates.

Table 7: Main results on ImageNet-R with the CLIP-OpenAI ViT-B/16 backbone. PRM-fixed and PRM-Fisher (bold rows) are our methods.

	ImageNet-R					
	T=5		T=10		T=20	
Method	Acc ↑	AAA ↑	Acc ↑	AAA ↑	Acc ↑	AAA ↑
BECAME	75.20 ± 1.51	80.18 ± 1.80	69.68 ± 0.76	77.58 ± 0.34	61.35 ± 1.79	72.24 ± 2.83
CoFiMA	72.92 ± 0.85	78.92 ± 1.62	65.00 ± 1.04	75.53 ± 0.39	51.82 ± 1.35	68.76 ± 1.14
CoMA	72.47 ± 0.71	78.78 ± 1.63	63.54 ± 1.39	74.96 ± 0.28	48.43 ± 0.74	66.63 ± 1.25
MagMax	62.01 ± 0.52	67.81 ± 1.05	58.86 ± 0.66	65.66 ± 0.26	55.29 ± 0.93	64.67 ± 1.24
Model-Avg	75.41 ± 1.08	79.94 ± 1.67	70.04 ± 0.87	77.75 ± 0.58	61.05 ± 1.69	72.08 ± 2.64
P&M	75.35 ± 1.29	80.20 ± 1.76	69.95 ± 0.77	77.53 ± 0.73	61.61 ± 1.88	72.45 ± 2.96
PRM-fixed	$\underline{81.86 \pm 1.10}$	$\underline{86.61 \pm 1.26}$	$\underline{77.19 \pm 0.43}$	$\underline{84.13 \pm 0.90}$	$\underline{66.49 \pm 0.05}$	$\underline{75.51 \pm 0.99}$
PRM-Fisher	$\mathbf{82.41 \pm 0.74}$	$\mathbf{87.36 \pm 0.63}$	$\mathbf{78.71 \pm 0.75}$	$\mathbf{85.55 \pm 0.60}$	$\mathbf{70.67 \pm 1.85}$	$\mathbf{77.88 \pm 1.27}$

Each cell is mean±std over 3 in-house seeds on the CLIP-OpenAI ViT-B/16 backbone. **Best** per column bold; second best underlined. Bold/underline are applied independently to ACC and AAA.

Table 8: Main results on CIFAR-100, DomainNet, and CUB-200 with the CLIP-OpenAI ViT-B/16 backbone.

	CIFAR-100				DomainNet		CUB-200	
	T=10		T=20		T=5		T=10	
Method	Acc ↑	AAA ↑	Acc ↑	AAA ↑	Acc ↑	AAA ↑	Acc ↑	AAA ↑
BECAME	69.86 ± 1.41	78.57 ± 1.60	58.56 ± 3.17	68.13 ± 3.88	80.89 ± 1.10	83.74 ± 1.22	43.97 ± 1.69	54.36 ± 3.01
CoFiMA	61.99 ± 1.41	73.91 ± 1.58	46.26 ± 1.53	62.65 ± 1.48	78.98 ± 0.26	82.99 ± 0.81	44.10 ± 1.00	55.51 ± 2.61
CoMA	58.15 ± 2.39	72.12 ± 1.77	36.08 ± 2.57	56.31 ± 1.16	78.16 ± 0.23	82.74 ± 0.81	39.49 ± 0.24	53.80 ± 2.51
MagMax	52.06 ± 1.73	60.97 ± 1.05	43.11 ± 1.56	52.37 ± 1.06	68.49 ± 1.53	72.54 ± 0.60	40.92 ± 0.45	52.73 ± 1.57
Model-Avg	68.15 ± 0.46	76.12 ± 2.02	57.35 ± 4.67	67.22 ± 4.37	81.26 ± 0.55	83.90 ± 1.13	45.79 ± 1.49	56.04 ± 2.52
P&M	70.04 ± 1.26	78.72 ± 1.56	58.33 ± 3.16	67.85 ± 3.93	80.91 ± 1.21	83.72 ± 1.26	44.23 ± 1.59	54.38 ± 2.93
PRM-fixed	$\underline{77.54 \pm 0.53}$	$\underline{84.77 \pm 1.00}$	$\underline{69.22 \pm 1.17}$	$\underline{76.51 \pm 0.62}$	$\underline{85.29 \pm 0.77}$	$\underline{88.69 \pm 1.21}$	$\underline{47.68 \pm 1.75}$	$\underline{58.05 \pm 2.14}$
PRM-Fisher	$\mathbf{78.66 \pm 0.74}$	$\mathbf{85.59 \pm 1.43}$	$\mathbf{71.60 \pm 0.39}$	$\mathbf{79.13 \pm 0.68}$	$\mathbf{86.16 \pm 0.08}$	$\mathbf{90.34 \pm 0.13}$	$\mathbf{53.39 \pm 0.74}$	$\mathbf{61.31 \pm 2.52}$

Each cell is mean±std over 3 in-house seeds on the CLIP-OpenAI ViT-B/16 backbone. **Best** per column bold; second best underlined. Bold/underline are applied independently to ACC and AAA.

Table 9: Beyond P&M generality on CLIP IN-R 10T (3 seeds, mean±std). Each write-in rule paired with no-prox `Base` and +Prox ($\lambda_{\text{prox}} = 10^{-2}$). ΔAAA and ΔFORG are computed as +Prox minus Base; positive ΔAAA and negative ΔFORG both indicate improvement. In this setting, every reported rule receives positive ΔAAA and negative ΔFORG. Bold marks improvements.

Merge rule	AAA ↑			Forgetting ↓		
	Base	+Prox	Δ	Base	+Prox	Δ
P&M (Fisher α)	77.53 ± 0.73	85.55 ± 0.60	**+8.02**	7.72 ± 1.09	5.93 ± 0.81	**-1.79**
P&M (α=0.8 fixed)	60.77 ± 0.02	84.13 ± 0.90	**+23.36**	52.41 ± 1.19	12.77 ± 0.69	**-39.64**
CoFiMA	75.53 ± 0.39	85.77 ± 0.55	**+10.24**	26.13 ± 1.64	9.14 ± 0.77	**-16.99**
CoMA	74.96 ± 0.28	85.75 ± 0.55	**+10.79**	28.70 ± 2.15	9.23 ± 0.95	**-19.47**
Model-Avg	77.75 ± 0.58	85.74 ± 0.58	**+7.99**	8.75 ± 0.64	5.69 ± 0.77	**-3.06**
BECAME	77.58 ± 0.34	85.70 ± 0.56	**+8.12**	8.35 ± 0.96	5.75 ± 0.71	**-2.60**
MagMax	65.66 ± 0.31	83.42 ± 1.28	**+17.76**	34.68 ± 1.72	9.55 ± 0.57	**-25.12**

Table 10: Beyond P&M generality on CLIP IN-R 20T (3 seeds, mean±std). Same protocol as Table 9; harder split with twice the merge events. Δ columns are +Prox minus Base; positive ΔAAA and negative ΔFORG indicate improvement. Positive ΔAAA persists across all 7 rules; forgetting changes are rule-dependent, with the largest reductions on rules whose Base over-writes (e.g. α=0.8 fixed).

Merge rule	AAA ↑			Forgetting ↓		
	Base	+Prox	Δ	Base	+Prox	Δ
P&M (Fisher α)	72.45 ± 2.96	77.88 ± 1.27	**+5.44**	6.26 ± 0.68	7.42 ± 1.12	+1.16
P&M (α=0.8 fixed)	49.49 ± 0.80	75.51 ± 0.99	**+26.02**	67.48 ± 1.52	20.71 ± 1.36	**-46.77**
CoFiMA	68.76 ± 1.14	78.14 ± 1.24	**+9.38**	33.41 ± 1.19	14.70 ± 0.96	**-18.71**
CoMA	66.63 ± 1.25	78.04 ± 1.31	**+11.41**	40.16 ± 1.21	14.52 ± 1.03	**-25.64**
Model-Avg	72.08 ± 2.64	77.82 ± 1.52	**+5.74**	6.59 ± 0.84	7.45 ± 1.12	+0.85
BECAME	72.24 ± 2.83	77.90 ± 1.28	**+5.66**	6.47 ± 0.99	7.41 ± 1.06	+0.94
MagMax	64.67 ± 1.52	77.26 ± 1.12	**+12.59**	32.69 ± 1.81	12.28 ± 1.49	**-20.41**

Table 11: Beyond P&M generality on CLIP CIFAR-100 20T (CLIP backbone). †: cell aggregates a single available seed; standard deviation is omitted.

Merge rule	AAA ↑			Forgetting ↓		
	Base	+Prox	Δ	Base	+Prox	Δ
P&M (Fisher α)	67.85 ± 3.93	79.13 ± 0.68	**+11.28**	8.66 ± 2.72	9.33 ± 0.74	+0.67
P&M (α=0.8 fixed)	34.81 ± 2.05	76.51 ± 0.62	**+41.70**	82.84 ± 1.24	19.80 ± 1.17	**-63.04**
CoFiMA	62.65 ± 1.48	78.94 ± 0.59	**+16.29**	39.94 ± 3.20	15.13 ± 0.66	**-24.81**
CoMA	56.31 ± 1.16	$79.53^{\dagger}$	**+23.22**	58.26 ± 5.10	$15.60^{\dagger}$	**-42.66**
Model-Avg	67.22 ± 4.37	79.13 ± 0.93	**+11.91**	8.91 ± 2.75	9.26 ± 0.57	+0.35
BECAME	68.13 ± 3.88	79.12 ± 0.74	**+11.00**	8.72 ± 2.82	9.40 ± 0.64	+0.67
MagMax	52.37 ± 1.06	77.92 ± 1.19	**+25.54**	50.49 ± 2.68	14.43 ± 0.52	**-36.07**

Table 12: Beyond P&M generality on CLIP DomainNet 5T (CLIP backbone).

Merge rule	AAA ↑			Forgetting ↓		
	Base	+Prox	Δ	Base	+Prox	Δ
P&M (Fisher α)	83.72 ± 1.26	90.34 ± 0.13	**+6.62**	11.44 ± 0.82	6.87 ± 0.53	**-4.58**
P&M (α=0.8 fixed)	72.30 ± 0.21	88.69 ± 1.21	**+16.39**	37.74 ± 0.05	9.27 ± 1.11	**-28.46**
CoFiMA	82.99 ± 0.81	90.34 ± 0.39	**+7.35**	18.48 ± 0.54	7.55 ± 0.67	**-10.93**
CoMA	82.74 ± 0.81	90.09 ± 0.21	**+7.35**	19.97 ± 0.53	7.75 ± 0.73	**-12.22**
Model-Avg	83.90 ± 1.13	90.34 ± 0.38	**+6.45**	12.02 ± 0.54	7.08 ± 0.66	**-4.94**
BECAME	83.74 ± 1.22	90.26 ± 0.08	**+6.52**	11.39 ± 0.80	6.94 ± 0.65	**-4.45**
MagMax	72.54 ± 0.60	88.42 ± 1.10	**+15.88**	33.20 ± 1.88	9.15 ± 0.94	**-24.05**

Table 13: Beyond P&M generality on CLIP CUB-200 10T (CLIP backbone).

Merge rule	AAA ↑			Forgetting ↓		
	Base	+Prox	Δ	Base	+Prox	Δ
P&M (Fisher α)	54.38 ± 2.93	61.31 ± 2.52	**+6.94**	14.70 ± 1.50	14.40 ± 2.26	**-0.30**
P&M (α=0.8 fixed)	46.11 ± 1.69	58.05 ± 2.14	**+11.94**	66.75 ± 1.13	35.84 ± 3.39	**-30.91**
CoFiMA	55.51 ± 2.61	61.36 ± 3.10	**+5.85**	27.14 ± 1.77	25.27 ± 2.64	**-1.87**
CoMA	53.80 ± 2.51	61.64 ± 2.61	**+7.84**	44.65 ± 0.42	25.58 ± 2.20	**-19.07**
Model-Avg	56.04 ± 2.52	61.82 ± 2.39	**+5.78**	16.95 ± 1.53	15.15 ± 2.42	**-1.80**
BECAME	54.36 ± 3.01	61.32 ± 2.48	**+6.96**	14.79 ± 1.17	14.45 ± 2.40	**-0.34**
MagMax	52.73 ± 1.57	59.48 ± 1.85	**+6.75**	44.26 ± 0.68	23.39 ± 0.77	**-20.87**

Cross-setting summary. Table 14 aggregates the per-cell results by setting. Across the available CLIP settings, +Prox improves AAA for every reported merge rule. The largest mean gains occur in the settings where the Base models are most sensitive to the write-in coefficient.

Table 14: Per-setting transfer summary on CLIP. "Rules improved/total" counts the available merge rules with positive ΔAAA from Base to +Prox. Forgetting reduction is Forg(Base) $-$ Forg(+Prox), so positive means improvement.

Setting	Rules *improved/total* (AAA)	Mean ΔAAA	Median ΔAAA	Best ΔAAA	Mean Forg. reduction
CLIP IN-R 10T	7/7	+12.33	+10.24	+23.36	+15.52
CLIP IN-R 20T	7/7	+10.89	+9.38	+26.02	+15.51
CLIP CIFAR-100 20T	7/7	+20.13	+16.29	+41.70	+23.55
CLIP DomainNet 5T	7/7	+9.51	+7.35	+16.39	+12.80
CLIP CUB-200 10T	7/7	+7.44	+6.94	+11.94	+10.74

E Matched-prefix scale interventions

Appendix E isolates task-vector radius from direction using matched-prefix interventions. Both interventions start from the same prefix model; the only difference is the task vector trained from that prefix.

E.1 Common-prefix protocol

We first run the unregularized P&M pipeline on CLIP ImageNet-R 10T up to the prefix before task 5. From this common prefix, we train two task-vector proposals with identical initialization, data order, and randomness: one P&M proposal with λ_{prox}=0 and one PRM proposal with λ_{prox}=10^{-2}. The two proposals are then reloaded onto the same prefix for post-hoc coefficient sweeps. At every $\alpha \in \{0, 0.05, \ldots, 1.0\}$, we evaluate seen-task CE loss and accuracy on the held-out validation splits and compute R_t, $W_{t,\varepsilon}$, and $G_{t,\text{fix}}$ as defined in Appendix A.

E.2 Norm-matched task-vector intervention

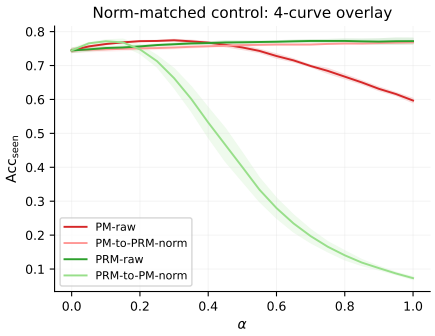
Table 15: Norm-matched task-vector intervention on CLIP IN-R 10T (task 5). Each row fixes the direction source and norm source before evaluating the same common-prefix α sweep.

Vector	Direction	Norm	$\|\delta_t\|_F$	q_t [10^{-3}]	q_t^{dir} [10^{-3}]	R_t [pp]	$G_{t,\text{fix}}$ [pp]	$\text{ACC}_{\le t}$@α=0.8
P&M raw	P&M	P&M	11.72 ± 0.50	162.82 ± 12.22	1.18 ± 0.01	17.77 ± 0.53	10.73 ± 0.46	66.72 ± 0.61
P&M$\to$PRM-norm	P&M	PRM	1.56 ± 0.02	2.88 ± 0.11	1.18 ± 0.01	2.37 ± 0.44	0.33 ± 0.11	76.49 ± 0.09
PRM raw	PRM	PRM	1.56 ± 0.02	29.91 ± 0.51	12.29 ± 0.30	3.07 ± 0.82	0.27 ± 0.19	77.25 ± 0.55
PRM$\to$P&M-norm	PRM	P&M	11.72 ± 0.50	1694.62 ± 177.89	12.29 ± 0.30	70.01 ± 0.49	63.28 ± 0.74	14.05 ± 1.31

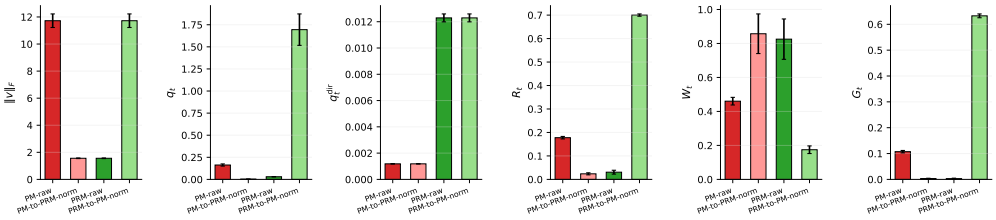
The four-vector table reports the source of direction and the source of norm for each vector, and Figure 6 visualizes the corresponding α-paths and path metrics. Rescaling the P&M vector to the PRM norm preserves its direction but reduces its radius by approximately $7.5\times$; its R_t drops from 17.77 pp to 2.37 pp, and $G_{t,\text{fix}}$ drops from 10.73 pp to 0.33 pp. Conversely, inflating the PRM direction to the P&M norm reduces $\overline{\text{Acc}}_{\le t}$@$\alpha$=0.8 from 77.25 to 14.05. The direction-normalized interference is actually larger under PRM, so the lower total interference is explained mainly by the smaller radius entering $q_t \approx \|\delta_t\|_F^2 q_t^{\text{dir}}$ for non-negligible $\|\delta_t\|_F$.

E.3 Continuous task-vector rescaling intervention

We use the name "continuous task-vector rescaling intervention" to make clear that the intervention rescales the task vector, not the model. Figure 7 reports six panels: the Frobenius norm and Fisher-weighted interference confirm the expected $\|\delta(c)\|_F = c\|\delta(1)\|_F$ and $q_t(c) = c^2 q_t(1)$ scaling, while R_t, $W_{t,\varepsilon}$, $G_{t,\text{fix}}$, and $\overline{\text{Acc}}_{\le t}$@$\alpha$=0.8 all vary continuously with c. Together with the norm-matched swap, this analysis supports the interpretation that task-vector radius is a major axis behind the observed coefficient sensitivity.

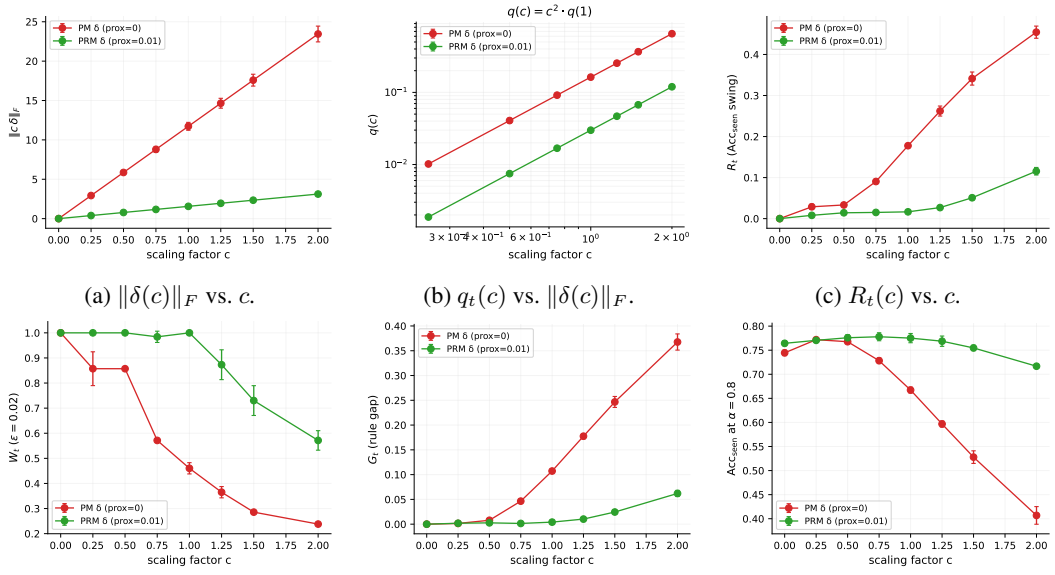


(a) $\overline{\mathrm{Acc}}_{\leq t}(\alpha)$, four-vector overlay.



(b) Path metrics (R_t, q_t, $G_{t,\mathrm{fix}}$).

Figure 6: Norm-matched intervention on CLIP IN-R 10T (task 5). Rescaling $\delta_{\mathrm{P\&M}}$ to PRM's norm without altering its direction substantially reduces coefficient sensitivity; inflating δ_{PRM} to P&M's norm substantially increases coefficient sensitivity.



(a) $\|\delta(c)\|_F$ vs. c. (b) $q_t(c)$ vs. $\|\delta(c)\|_F$. (c) $R_t(c)$ vs. c.

(d) $W_{t,0.005}(c)$ vs. c. (e) $G_{t,\mathrm{fix}}(c)$ vs. c. (f) $\overline{\mathrm{Acc}}_{\leq t}@\alpha{=}0.8$ vs. c.

Figure 7: Continuous task-vector rescaling intervention on CLIP IN-R 10T (task 5). We sweep $c \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, \frac{5}{4}, \frac{3}{2}, 2\}$ for both P&M and PRM vectors. The sweep verifies the expected scaling relations $\|\delta(c)\|_F = c\|\delta(1)\|_F$ and $q_t(c) = c^2 q_t(1)$, while coefficient sensitivity and fixed-α accuracy vary continuously with c.

Table 16: Robustness to the proximal weight across datasets and task orders with the ViT-B/16 AugReg backbone. Each cell reports mean AAA / forgetting (%, lower forgetting is better) over three independently permuted task orders.

λ_{prox}	CIFAR-100 20T	CUB-200 10T	DomainNet 5T	ImageNet-R 10T
0	84.04 / 17.04	77.02 / 23.51	82.89 / 15.16	82.30 / 16.06
10^{-3}	89.79 / 7.61	78.35 / 15.86	88.33 / 7.22	85.41 / 9.45
3×10^{-3}	89.71 / 6.45	78.82 / 12.05	89.73 / 4.93	86.51 / 7.29
10^{-2} **(default)**	89.62 / 5.37	79.02 / 8.77	89.42 / 3.85	86.39 / 5.62
3×10^{-2}	90.26 / 4.25	79.72 / 5.94	88.99 / 3.31	85.28 / 4.41
10^{-1}	89.21 / 4.22	79.16 / 3.44	88.42 / 4.35	83.25 / 3.52

F Additional ablation notes

The main text reports the full proximal-strength sweep on ImageNet-R 10T (Figure 4) and the component ablation (Table 2). Here we report the robustness of λ_{prox} across settings and task orders, followed by the component interpretation and the diagnostic caveat for NoWrite.

F.1 Proximal-strength robustness across settings and task orders

We extend the proximal-strength sweep to four dataset/task-granularity settings and three independently permuted task orders per setting. The same value $\lambda_{\text{prox}}{=}10^{-2}$ is used as the default across all settings. Table 16 reports the mean over the three orders.

Across 10^{-3}–3×10^{-2}, the within-setting AAA spread is 0.64–1.40 percentage points, and every value in this interval improves both AAA and forgetting over $\lambda_{\text{prox}}{=}0$. The default 10^{-2} is within 0.12–0.70 percentage points of the best swept AAA in every setting and reduces forgetting by 10.44–15.74 percentage points relative to $\lambda_{\text{prox}}{=}0$. For new settings under the same training protocol, we use 10^{-2} as the default; stronger regularization can be reduced when current-task optimization underfits and the task-vector norm is overly compressed.

F.2 Component-ablation interpretation

Table 2 separates perturbation-aware task-vector training, proximal task-vector shaping, the write-in coefficient rule, and the final write-in. The key comparisons are: P&M-fixed vs. PRM-fixed isolates the effect of adding proximity at the same fixed α; P&M vs. PRM-Fisher keeps the Fisher write-in coefficient rule; and LoRA-Prox tests whether the proximal term already changes mergeability without perturbation. The NoWrite rows are attribution probes rather than deployable methods.

F.3 NoWrite caveat

NoWrite is purely diagnostic. Setting $\alpha{=}0$ at the write-in step means the per-task LoRA factors are still trained under the task-vector-training objective, but no task vector is permanently accumulated into the running model. This isolates task-vector-training capacity before write-in, but it is not a deployable continual-learning method: it cannot retain new post-task capacity through the shared running model. The NoWrite rows therefore support attribution, not a baseline comparison.

G Broader impacts

PRM is a general method for rehearsal-free continual adaptation. Potential positive impacts include reducing the need to store previous-task data, improving continual adaptation in storage- or privacy-constrained settings, and making sequential LoRA merging less sensitive to per-task coefficient tuning. Potential negative impacts may arise if continual adaptation is deployed in safety-sensitive or socially consequential applications without adequate validation: model updates could preserve or amplify biases in pretrained backbones, and failures or forgetting could affect downstream users. This work does not release high-risk generative models or scraped datasets, and deployment in sensitive domains should require application-specific safety, fairness, and robustness evaluation.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction match the method and experimental scope. The paper states that PRM modifies task-vector training rather than the write-in rule, and the results in Sections 4 and Appendix B support the claims across the reported class-incremental vision settings.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses limitations in Section 5, including the restricted empirical scope to rehearsal-free class-incremental vision benchmarks, additive LoRA write-in, and the need to study other modalities, adapters, longer streams, and non-additive merge rules.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not present formal theorems or proof-based theoretical results. The analytical expressions in Section 3 are used as motivating approximations and diagnostic quantities rather than as formal theory claims.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Sections 3 and 4, together with Appendix A, describe the sequential LoRA write-in procedure, PRM objective, datasets, task splits, metrics, backbones, hyperparameters, coefficient rules, diagnostic sweeps, and baseline implementation sources needed to reproduce the main experimental conclusions.

Guidelines:

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

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

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

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

   Answer: [Yes]

   Justification: Code, configurations, data-preparation utilities, and reproduction instructions are publicly available at `https://github.com/Edward61xuan/PRM-Continual-LoRA`; see Appendix A.0.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4.1 and Appendix A specify the datasets and task splits, class-incremental evaluation protocol, ViT-B/16 backbones, rank-10 LoRA placement, optimizer, learning rates, batch size, epoch counts, perturbation settings, proximal weight, coefficient rules, and metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

Justification: The main tables report mean and standard deviation over three seeds, and Appendix A states how cross-seed and cross-task aggregates are computed. Cells with fewer available runs or lifted results are marked separately where applicable.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix A.7 reports the GPU type, GPU memory, number of visible GPUs per seed, approximate per-run GPU-hours, total compute for the reported in-house experiments, and additional compute used for preliminary or failed runs.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics, and the work uses standard public vision benchmarks and pretrained models without collecting private data, conducting human-subject experiments, or releasing high-risk models or datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The paper discusses potential positive impacts such as reducing the need to retain past data and improving continual adaptation, as well as potential negative impacts from unreliable or biased sequential updates in sensitive deployment settings.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The work does not release high-risk models, image generators, pretrained language models, or newly scraped datasets that would require special misuse safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [No]

Justification: The paper cites the datasets, pretrained checkpoints, and baseline methods used in the experiments, but the current submission does not explicitly list the license or terms of use for every existing asset. We use only publicly available research datasets, pretrained checkpoints, and software packages, and do not redistribute or repackage the datasets or checkpoints.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: The released PRM implementation is documented by the repository README, environment and requirements files, configuration files, and reproduction guide, and is distributed under the MIT License.

Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, human-subject experiments, participant instructions, or participant compensation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval or equivalent institutional review is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as an important, original, or non-standard component of the core method, experiments, or scientific results. Any writing, editing, or formatting assistance, if used, does not affect the methodology, rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.