RASA: RETENTION-AWARE STREAM ADAPTATION FOR ONLINE CONTINUAL LEARNING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning (CL) enables large language models (LLMs) to learn new tasks while preserving previously acquired knowledge. Expansion-based methods extend the model through task-specific modules, with low-rank adaptation (LoRA) enabling parameter-efficient expansion. However, in online continual learning (OCL) settings where data arrive in a stream and cannot be revisited, current-task information alone is insufficient to evaluate how adaptation affects performance on previous tasks. In this work, we propose Retention-Aware Stream Adaptation (RASA), a probabilistic framework that combines online statistics of parameter changes with historical feature information. RASA models parameter changes induced by the optimization process as a local Gaussian transition distribution, whose covariance is matched to a reference covariance estimated from the stream to determine adaptive time steps. To incorporate retention, it uses aggregated feature second moments from previous tasks to constrain cumulative changes in LoRA linear outputs on historical input features. A local generalized posterior then combines this constraint with gradient information from the current batch for parameter estimation. The resulting procedure can be incorporated into existing training procedures, reusing available gradients and stored feature statistics without replay or additional forward or backward passes over the data. Experimental results demonstrate that RASA consistently improves the average performance of existing expansion-based methods in OCL.

1 INTRODUCTION

Deploying large language models (LLMs) in changing environments requires continual acquisition of new knowledge and task capabilities (Shi et al., 2025; Song et al., 2023). Continual learning (CL) addresses this requirement by adapting to sequentially arriving tasks while retaining previously acquired knowledge (Shi et al., 2025; Liang & Li, 2024). However, catastrophic forgetting, where learning a new task degrades performance on earlier tasks, remains a central obstacle to continual adaptation (Kirkpatrick et al., 2017; Wang et al., 2023a).

Regularization-based methods mitigate catastrophic forgetting by constraining changes to parameters important for previous tasks (Kirkpatrick et al., 2017). Replay-based methods preserve past task information by training on stored examples or generated samples alongside current data (Sun et al., 2020; Wang et al., 2024; He et al., 2024). The introduction of low-rank adaptation (LoRA) has enabled parameter-efficient task-specific expansion, motivating growing research on expansion-based CL for LLMs (Hu et al., 2022; Wang et al., 2023a; Zhao et al., 2024; Liang et al., 2025). For example, O-LoRA and GainLoRA train a new LoRA branch while freezing pretrained weights and earlier branches, providing capacity for a new task without directly modifying previously learned parameters (Rusu et al., 2016; Wang et al., 2023a; Liang et al., 2025).

Within this branch-expansion framework, prior work has sought to reduce interference across tasks when learning and integrating new branches (Wang et al., 2023a; Zhao et al., 2024; Liang et al., 2025). O-LoRA and InfLoRA constrain low-rank subspaces to reduce interference from learning new tasks without replaying past examples (Wang et al., 2023a; Liang & Li, 2024). To use the learned branches for each input without task identities, Dynamic ConPET learns module se-

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

lection, while SAPT and GainLoRA learn integration through shared attention and gating, respectively (Song et al., 2023; Zhao et al., 2024; Liang et al., 2025). In particular, GainLoRA limits the influence of a new branch on old tasks, recognizing that integrating a new branch can alter earlier-task outputs even when previous branches remain frozen (Liang et al., 2025).

However, deployments with continuously arriving data cannot readily rely on collecting a complete dataset for each new task and training over it repeatedly (Mai et al., 2022). To reflect this environment, we consider online continual learning (OCL) under three constraints. First, data arrive sequentially in small batches, and each sample is used for training only once (Mai et al., 2022; Aljundi et al., 2019; Chaudhry et al., 2019). Second, previously consumed training data cannot be revisited, and neither stored-example nor generative replay is permitted. Third, task identities and task boundaries are unavailable at inference, although task boundaries are observed during training.

Under these online constraints, learning a new task and retaining past knowledge must proceed with limited information from the current batch. However, reducing the current batch loss in an evolving stream does not guarantee effective learning across the new task or preservation of performance on previous tasks. Even when earlier branches remain frozen, a new branch participating in prediction can change model outputs on previous tasks (Liang et al., 2025). Without access to past data, directly evaluating these effects during training is difficult. The central challenge is therefore to acquire new knowledge from limited observations while accounting for how learning affects previously acquired knowledge.

We develop Retention-Aware Stream Adaptation (RASA) for LoRA-based expansion in OCL. RASA uses statistics of parameter changes induced by the optimization process to characterize current learning, together with feature statistics accumulated from previous tasks to inform retention. It first represents these parameter changes through a local Gaussian transition distribution and determines the time step through online covariance matching. It then constrains cumulative changes in LoRA linear outputs on historical input features using aggregated feature second moments and selects the next parameters through a local generalized posterior incorporating the current-task loss. Both stages use current-batch gradients and statistics maintained during training, requiring neither replay nor additional forward or backward passes over the data.

Our main contributions are as follows:

- We introduce online covariance matching for LoRA-based expansion in OCL, deriving adaptive time steps from the variance of parameter changes observed in the stream.
- We develop retention-aware posterior inference that selects parameters while constraining cumulative LoRA output changes on historical input features through aggregated feature moments.
- Experiments across LLM backbones and benchmarks demonstrate consistent average-performance gains and lower forgetting for LoRA expansion methods.

2 Related Work and Background

2.1 Online Continual Learning for LLMs

OCL studies adaptation to sequential data under limited training opportunities (Aljundi et al., 2019). In single-pass language learning, episodic memory supports replay and local adaptation (de Masson d'Autume et al., 2019), with meta-learning reducing memory requirements (Wang et al., 2020). We consider single-pass learning without stored-example or generative replay, observing task boundaries only during training (Section 3.1). RASA therefore carries historical information forward through feature statistics to guide current adaptation.

2.2 LoRA Expansion and Retention Constraints

LoRA-based expansion adds capacity for a new task while preserving previously learned modules. LoRA enables this expansion by representing the change added to a frozen weight matrix through trainable low-rank factors (Hu et al., 2022). The output of a layer with a single LoRA branch is

$$\boldsymbol{e} = (\boldsymbol{W}_0 + \gamma \boldsymbol{B}\boldsymbol{A})\boldsymbol{h}. \tag{1}$$

Here, $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is frozen, and $\boldsymbol{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\boldsymbol{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are trainable down- and up-projection factors with $r \ll \min(d_{\text{in}}, d_{\text{out}})$. The vectors $\boldsymbol{h}$ and $\boldsymbol{e}$ denote input and output features, and γ is a fixed scaling factor.

Prior work addresses both update directions and branch integration. SD-LoRA separates the learning of LoRA magnitudes and directions for class-incremental learning (Wu et al., 2025), while SLAO uses orthogonal initialization and continually merges updates into a single LoRA adapter (Qiao & Mahdavi, 2026). Gradient Projection Memory (GPM) projects gradients onto the orthogonal complement of stored activation subspaces (Saha et al., 2021). O-LoRA and InfLoRA, the latter proposed for vision CL, impose subspace constraints on low-rank adaptation to mitigate interference between tasks (Wang et al., 2023a; Liang & Li, 2024).

When task identity is unavailable at inference, learned branches must also be combined appropriately for each input. SAPT connects adapter learning and selection through shared attention, using reflection with pseudo samples to preserve attention for previous tasks (Zhao et al., 2024). Even when previous branches remain frozen, integrating a new branch can change outputs on earlier tasks. GainLoRA addresses this issue by integrating branches through gating and restricting the new branch's influence on previous tasks (Liang et al., 2025).

2.3 Probabilistic Modeling of Parameter Updates

In stochastic optimization, parameter updates vary across batches, so the learning process exhibits both an average movement and fluctuations around it. Stochastic differential equations (SDEs) represent these components as drift and diffusion and have been used to analyze the mean and covariance of parameter distributions (Li et al., 2019). This perspective allows the effects of update scaling on parameter movement and covariance to be considered jointly. From an inference perspective, Bayesian continual learning carries information across tasks through approximate parameter posteriors (Nguyen et al., 2018). Generalized Bayesian inference provides a framework for updating a reference distribution using a given loss without requiring a complete likelihood model (Bissiri et al., 2016). RASA adopts these perspectives to describe local parameter transitions through their mean and covariance and to formulate parameter selection through a local generalized posterior.

3 Methodology

In this section, we present RASA for LoRA-based expansion in online continual learning. We begin with the problem formulation (Section 3.1), followed by online covariance matching for adaptive time steps (Section 3.2) and historical feature constraints on cumulative linear-output changes (Section 3.3). We then combine these components through retention-aware posterior inference (Section 3.4). Algorithm 1 in Appendix A.1 summarizes the procedure.

3.1 Problem Formulation

We consider online continual learning (OCL), where a language model $p_{\boldsymbol{\Theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ learns a sequence of K tasks with distributions $\mathcal{D}_1, \ldots, \mathcal{D}_K$. At step t of task k, the model receives a batch $\mathcal{B}_t \sim \mathcal{D}_k$ and uses each example once, without replaying stored or generated examples. Task boundaries are known during training; task identities and boundaries are unavailable at inference.

Following LoRA expansion, each task introduces a new branch while pretrained weights and earlier branches remain frozen. Let $\boldsymbol{\theta} \in \mathbb{R}^{d_k}$ collect the trainable parameters of the current task, comprising the new LoRA factors and any task-specific routing parameters, and let $\hat{\boldsymbol{\theta}}_t$ denote their values before step t. Our objective is to minimize average expected risk after the task sequence (Appendix A.2). Under the single-pass, replay-free protocol, RASA determines each update using the current-batch cross-entropy (CE) gradient $\boldsymbol{g}_t^{\text{CE}} = \nabla \ell_t(\hat{\boldsymbol{\theta}}_t)$ and retained statistics. The gradient guides fitting to the new task, while the statistics carry information from earlier learning to regulate this adaptation.

3.2 Online Covariance Matching

In the single-pass setting of Section 3.1, the scale of update fluctuations can vary as the task distribution changes. We therefore choose the time step by comparing the current transition covariance

with a reference accumulated across tasks. To make this comparison possible, we introduce a local Gaussian transition model that expresses the covariance as a function of the time step.

Local transition model. Let $\boldsymbol{\delta}_t = \mathcal{O}_t(\hat{\boldsymbol{\theta}}_t, \boldsymbol{g}_t)$ denote the parameter displacement computed by the base method, including its learning rate and projection; $\boldsymbol{g}_t$ includes CE and method-specific regularization gradients. To characterize local fluctuations, define $v_t = \|\boldsymbol{\delta}_t - \bar{\boldsymbol{\delta}}_t\|_2^2/d_k$, where $\bar{\boldsymbol{\delta}}_t$ is the task-local running mean including the current displacement. We model $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{\delta}_t, v_t\boldsymbol{I})$: the mean retains the proposed direction, while the isotropic covariance summarizes its residual fluctuations. For a normalized time step $\alpha_t = \Delta t_t > 0$, the scaled transition $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_t + \alpha_t \boldsymbol{z}_t$ is

$$p_t^-(\boldsymbol{\theta}) = \mathcal{N}\Big(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}_t + \alpha_t\boldsymbol{\delta}_t, \alpha_t^2 v_t \boldsymbol{I}\Big). \tag{2}$$

Here, $\boldsymbol{I} \in \mathbb{R}^{d_k \times d_k}$ is the identity matrix, and $\mathrm{Cov}(\alpha_t \boldsymbol{z}_t) = \alpha_t^2 \,\mathrm{Cov}(\boldsymbol{z}_t)$ gives the covariance scaling in Eq. (2). Appendix A.3 provides an SDE interpretation of this local transition model.

Covariance matching. To assess the current transition covariance relative to earlier learning, we maintain $\bar{v}_t$ as the running mean of the residual energies observed throughout training, including the current observation. This reference retains the scale of past update fluctuations across task boundaries, whereas $\bar{\boldsymbol{\delta}}_t$ is reset for each task. We use $\bar{v}_t\boldsymbol{I}$ as the reference covariance and determine the time step by minimizing $D_{\mathrm{KL}}(\mathcal{N}(0, \alpha^2 v_t \boldsymbol{I}) \,\|\, \mathcal{N}(0, \bar{v}_t \boldsymbol{I}))$. Comparing the centered distributions isolates the covariance discrepancy from the mean displacement. For $v_t, \bar{v}_t > 0$, the minimum satisfies $\alpha^2 v_t = \bar{v}_t$. Bounding this factor limits extreme ratios:

$$\alpha_t^{\mathrm{raw}} = \sqrt{\frac{\bar{v}_t}{v_t}}, \tag{3}$$

$$\alpha_t = \mathrm{clip}(\alpha_t^{\mathrm{raw}}, L_t, U_t). \tag{4}$$

Appendix A.3 gives the minimization derivation, and Appendix A.4 specifies the running statistics and bounds. If $v_t \bar{v}_t = 0$, we use $\alpha_t^{\mathrm{raw}} = 1$.

3.3 Historical Feature Constraints

Section 3.2 uses the history of parameter displacements to determine update scale, but this history does not specify how an update affects responses to past inputs. We therefore retain historical feature moments and use them to constrain cumulative linear-output changes from the task-start state.

Historical moments. Fix the task-start parameters $\hat{\boldsymbol{\theta}}_k^0$ and define the effective branch weight $\boldsymbol{W}_\ell(\boldsymbol{\theta}) = \gamma_\ell \boldsymbol{B}_\ell \boldsymbol{A}_\ell$ and its change $\Delta\boldsymbol{W}_\ell = \boldsymbol{W}_\ell(\boldsymbol{\theta}) - \boldsymbol{W}_\ell(\hat{\boldsymbol{\theta}}_k^0)$. On a fixed input $\boldsymbol{h}$, $\Delta\boldsymbol{W}_\ell\boldsymbol{h}$ is the cumulative linear-output change, equal to the new branch's contribution when $\boldsymbol{W}_\ell(\hat{\boldsymbol{\theta}}_k^0) = 0$. The empirical identity $\mathbb{E}\|\Delta\boldsymbol{W}\boldsymbol{h}\|_2^2 = \mathrm{tr}(\Delta\boldsymbol{W}\mathbb{E}[\boldsymbol{h}\boldsymbol{h}^\top]\Delta\boldsymbol{W}^\top)$ shows that second moments suffice to evaluate its mean squared magnitude. To reduce storage, we partition input features into J_ℓ blocks and retain uncentered second moments and output energy over N_ℓ valid tokens from previous tasks:

$$\boldsymbol{C}_{\ell,j} = \frac{1}{N_\ell}\sum_i \boldsymbol{h}_{\ell,i}^{(j)} \boldsymbol{h}_{\ell,i}^{(j)\top}, \tag{5}$$

$$E_\ell = \frac{1}{N_\ell}\sum_i \|\boldsymbol{e}_{\ell,i}\|_2^2. \tag{6}$$

Here, $\boldsymbol{h}_{\ell,i}$ and $\boldsymbol{e}_{\ell,i}$ are recorded input and output features; current-task statistics enter the history only at the task boundary. The moments weight changes along higher-energy historical directions more strongly, distinguishing equal-norm weight changes by their effects on past inputs.

Cumulative response budget. Using stored moments, we upper-bound the mean squared linear-output change by

$$F_\ell(\boldsymbol{\theta}) = J_\ell \sum_j \mathrm{tr}\big(\Delta\boldsymbol{W}_{\ell,j}\boldsymbol{C}_{\ell,j}\Delta\boldsymbol{W}_{\ell,j}^\top\big), \tag{7}$$

where $\Delta \boldsymbol{W}_{\ell,j}$ contains the columns of block j. Normalizing by layerwise output energy defines

$$D_k(\boldsymbol{\theta}) = \frac{1}{2|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \frac{F_\ell(\boldsymbol{\theta})}{\max(E_\ell, \varepsilon)}, \tag{8}$$

$$\mathcal{R}_k = \left\{ \boldsymbol{\theta} : D_k(\boldsymbol{\theta}) \le \rho^2/2 \right\}, \tag{9}$$

where $\mathcal{L}$ comprises layers with historical statistics and $\varepsilon > 0$. When $\mathcal{L} \neq \varnothing$, every $\boldsymbol{\theta} \in \mathcal{R}_k$ satisfies (Appendix A.5)

$$\frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \frac{\sum_i \|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_{\ell,i}\|_2^2}{N_\ell \max(E_\ell, \varepsilon)} \le 2D_k(\boldsymbol{\theta}) \le \rho^2. \tag{10}$$

Thus, the constraint limits the new branch's cumulative effect on historical input features: the normalized root-mean-square linear-output change from the task-start state remains at most $\rho > 0$, using only stored moments and treating the features as fixed at collection time. Section 4.6 evaluates the effect on retention. Without historical statistics, $D_k \equiv 0$ and $\mathcal{R}_k = \mathbb{R}^{d_k}$.

3.4 Retention-Aware Posterior Inference

Updates satisfying the retention constraint can differ in their effect on the current-task objective. We define a local posterior to score feasible updates, then use candidate selection to compare directions from the base method and the CE gradient, scaled by α_t from Section 3.2.

Local posterior. For $\boldsymbol{\Delta} = \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t$, we combine a first-order approximation of the current-batch CE change with quadratic regularization under a positive diagonal metric $\boldsymbol{H}_t$:

$$\widehat{q}_t(\boldsymbol{\Delta}) = \boldsymbol{g}_t^{\mathrm{CE}\top} \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta}^\top \boldsymbol{H}_t \boldsymbol{\Delta}, \tag{11}$$

$$\pi_t(\boldsymbol{\theta}) \propto \exp\Big[-\widehat{q}_t(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t) \Big] \mathbf{1}\{\boldsymbol{\theta} \in \mathcal{R}_k\}. \tag{12}$$

The local reference is $\mathcal{N}(\hat{\boldsymbol{\theta}}_t, \boldsymbol{H}_t^{-1})$ (Appendix A.6); p_t^- instead determines candidate scale. Appendix A.2 specifies the metric and the CE gradient retained before regularization and projection.

Candidate selection. We compare the base update $\boldsymbol{\delta}_t$ with the CE direction $-\boldsymbol{H}_t^{-1} \boldsymbol{g}_t^{\mathrm{CE}}$, whose metric norm is capped at $\|\boldsymbol{\delta}_t\|_{\boldsymbol{H}_t}$. The latter addresses cases where accumulated gradient statistics or regularization cause the base update to deviate from the current-batch CE descent direction. After scaling by α_t, we apply the base method's projection and retention correction to both candidates. Let $\mathcal{V}_t$ contain the feasible, numerically valid corrected candidates and the current parameters. Restricted maximum a posteriori (MAP) estimation then gives

$$\hat{\boldsymbol{\theta}}_{t+1} \in \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathcal{V}_t} \widehat{q}_t(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t). \tag{13}$$

With $\widehat{q}_t(0) = 0$ and ties favoring current parameters, accepted updates strictly decrease the surrogate. Appendices A.2 and A.7 detail candidate construction and correction. RASA uses no replay or additional data passes; statistics and candidate processing add computation and memory.

4 Experiments

4.1 Benchmarks and Evaluation Protocol

Benchmarks. We evaluated on Standard CL (4 tasks), Long-sequence CL (15 tasks), SuperNI (15 tasks), and TRACE-8 (8 tasks) (Razdaibiedina et al., 2023; Wang et al., 2022; 2023b). Task compositions and orders are provided in Appendix B.1.

Protocol. In the main comparisons, we trained Meta-Llama-3.1-8B-Instruct and Qwen3-4B for one pass on at most 500 examples per task, without replay. Task boundaries were known only during training, and task identities were unavailable at inference. Each backbone–task-order pair used three random seeds (Appendix B.2). Appendices B.3–B.4 give optimizer settings and method-specific configurations.

Table 1: **Main results.** Comparison of each baseline and RASA on four CL benchmarks. Each value is the arithmetic mean over Meta-Llama-3.1-8B-Instruct and Qwen3-4B, the task orders for the corresponding benchmark, and three random seeds. Higher AP (average performance) and lower FR (forgetting) are better. **Bold** indicates the better value within each baseline–RASA pair. Results for individual backbones are provided in Tables B.6–B.7, and the aggregation procedure is described in Appendix B.2.

	Standard CL		Long-sequence CL		SuperNI		TRACE-8	
Method	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
OSFT	39.40	51.10	69.15	14.78	50.17	6.89	50.88	3.35
+RASA (ours)	**53.44**	**32.55**	**77.40**	**7.42**	**53.19**	**4.95**	**54.17**	**2.43**
Gain-InfLoRA	48.17	0.97	50.71	0.36	50.32	0.71	45.34	0.61
+RASA (ours)	**52.70**	**0.77**	**61.16**	**0.22**	**52.04**	**0.19**	**46.39**	**0.33**
Gain-OLoRA	56.35	−0.19	62.18	**0.80**	51.19	0.75	49.67	0.64
+RASA (ours)	**69.38**	**−0.63**	**70.21**	0.85	**53.09**	**0.61**	**53.70**	**0.23**
InfLoRA	50.32	27.45	76.96	5.55	50.95	5.15	55.20	1.42
+RASA (ours)	**61.40**	**11.04**	**78.73**	**4.61**	**52.81**	**4.15**	**56.66**	**1.10**
O-LoRA	40.40	49.07	65.39	16.29	50.11	5.75	54.87	3.06
+RASA (ours)	**43.80**	**32.08**	**69.61**	**12.58**	**52.31**	**4.07**	**56.73**	**2.25**
IncLoRA	47.63	25.33	53.17	25.55	46.49	8.56	53.72	7.40
+RASA (ours)	**62.26**	**15.45**	**67.23**	**11.73**	**51.32**	**5.80**	**57.01**	**1.96**
SeqLoRA	38.48	51.42	58.02	24.40	41.38	14.21	53.47	6.93
+RASA (ours)	**44.50**	**38.72**	**70.72**	**11.90**	**49.86**	**7.65**	**56.48**	**2.46**

Evaluation metrics. We used accuracy for Standard CL and Long-sequence CL, ROUGE-L for SuperNI (Lin, 2004), and the official task-specific metrics for TRACE-8 (Wang et al., 2023b). We report final average performance (AP, ↑), forgetting from the best previous performance (FR, ↓), and backward transfer relative to performance immediately after learning each task (BWT, ↑). Appendix B.5 gives their definitions and the TRACE-8 metrics.

4.2 Baseline Methods

We compared five LoRA expansion methods with and without RASA: IncLoRA, O-LoRA, InfLoRA, Gain-OLoRA, and Gain-InfLoRA. To test applicability beyond expansion, we additionally evaluated sequential shared-adapter SeqLoRA and subspace-constrained full fine-tuning with OSFT (Nayak et al., 2026) under the same OCL protocol. Gain-OLoRA/InfLoRA abbreviate Gain-LoRA (O-LoRA/InfLoRA), respectively. *Baseline* denotes each method without RASA; Appendices A.7 and B.4 give configurations and application details.

4.3 Main Results

Table 1 shows AP gains for all five LoRA expansion methods across all four benchmarks (see Appendices B.6 and B.7 for backbone-specific and additional-model results). These gains complemented the subspace constraints of O-LoRA and InfLoRA and the gated integration of GainLoRA. FR decreased in all expansion comparisons except Gain-OLoRA on Long-sequence CL, where it rose by 0.05 points while AP improved by 8.03 points. Gain-OLoRA had the highest baseline AP on Standard CL, yet RASA improved AP from 56.35 to 69.38 and reduced FR from -0.19 to -0.63. Thus, RASA also improved methods with strong performance and little forgetting. RASA also improved AP and reduced FR for SeqLoRA and OSFT on all four benchmarks, supporting its applicability to the shared-adapter and subspace fine-tuning settings evaluated here.

Table 2: **Component analysis.** O-LoRA, Gain-OLoRA, and SeqLoRA on Long-sequence CL, averaged over Meta-Llama-3.1-8B-Instruct, Qwen3-4B, Orders 4–6, and three seeds. A: online covariance matching; R: feature-moment retention constraint and correction; P: posterior-based candidate selection. Higher AP and lower FR are better; **bold** marks column bests. Appendix B.8 gives backbone-specific results and variant settings.

	O-LoRA		Gain-OLoRA		SeqLoRA	
Variant	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
Baseline	65.39	16.29	62.18	0.80	58.02	24.40
Baseline + A	67.48	14.43	66.60	1.28	69.19	12.65
Baseline + R	62.98	**9.49**	62.87	**0.23**	64.34	13.56
Baseline + A + R	69.09	13.30	69.06	1.10	69.96	**11.85**
RASA (+ A + R + P)	**69.61**	12.58	**70.21**	0.85	**70.72**	11.90

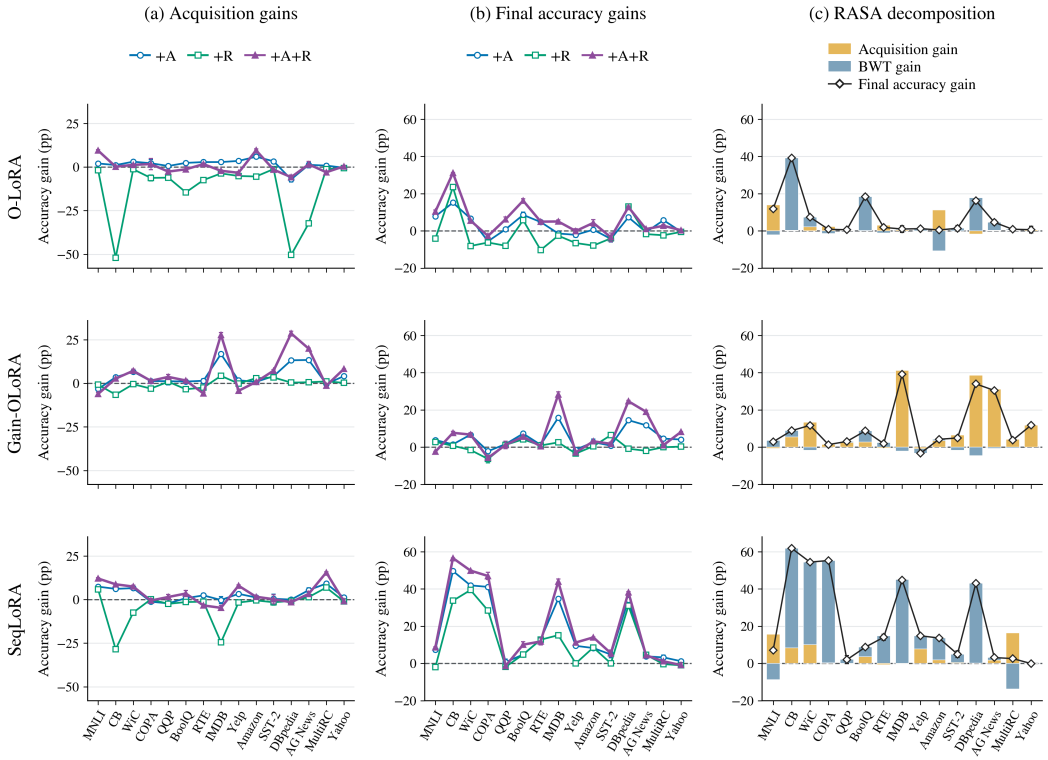


Figure 1: **Task-wise gains over the baseline.** O-LoRA, Gain-OLoRA, and SeqLoRA with Meta-Llama-3.1-8B-Instruct on Long-sequence CL (Order 4), averaged over three seeds. Gains are percentage-point (pp) differences from each baseline. Panels (a,b): acquisition and final accuracy gains for +A (blue circles), +R (green squares), and +A+R (purple triangles). A: online covariance matching; R: feature-moment retention constraint and correction. Panel (c): RASA final gain (black diamonds), decomposed into acquisition (yellow bars) and BWT (cyan bars) gains. BWT is final accuracy minus acquisition. Tasks follow learning order; dashed lines mark zero gain.

4.4 Component Analysis

We evaluated the individual and combined contributions of RASA components using O-LoRA and Gain-OLoRA, and additionally SeqLoRA under shared-adapter training (Table 2). RASA achieved the highest AP for all three methods. With O-LoRA, +A improved AP, supporting the utility of the adaptive time step, while +R lowered FR, supporting the retention effect of the feature-moment constraint. Although +R alone reduced AP, +A+R achieved higher AP and lower FR than +A, demonstrating the complementary roles of adaptation and retention. Adding P further improved AP

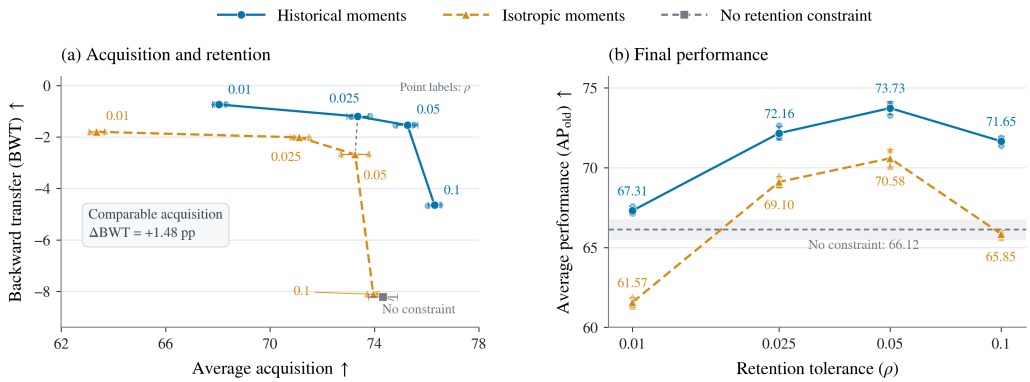


Figure 2: **Effects of historical feature moments.** Gain-OLoRA with Meta-Llama-3.1-8B-Instruct on Long-sequence CL (Order 4): Historical moments (blue circles), Isotropic moments (orange triangles), and No retention constraint (gray). (a) Acquisition and BWT; (b) final $\mathrm{AP}_{\mathrm{old}}$ versus ρ. Metrics exclude the last task; higher is better. Labels in (a) give ρ; the dashed connector and ΔBWT compare similar acquisition (pp: percentage points). Means, error bars, and shading summarize three seeds (mean ± 1 sample standard deviation) per condition and ρ.

for all three methods. Appendix B.8 compares RASA, Opt-only, and CE-only with A+R and the posterior acceptance rule held fixed.

4.5 Task-wise Analysis of Performance Gains

With Meta-Llama-3.1-8B-Instruct on Long-sequence CL (Order 4), we analyzed O-LoRA, Gain-OLoRA, and SeqLoRA. For task i, acquisition is its accuracy $a_{i,i}$ immediately after training, final accuracy is $a_{K,i}$ after the sequence, and BWT is $a_{K,i} - a_{i,i}$. Gains are paired differences from the baseline for each task and seed, so final gain equals acquisition gain plus BWT gain.

In Figure 1(a,b), A alone improved acquisition, but its gains on COPA and SST-2 with O-LoRA did not persist through subsequent learning. R alone reduced the decline after learning CB but lowered acquisition. On CB with all three methods, A+R achieved acquisition comparable to A and higher final performance than either component alone. Its final performance averaged over all methods and seeds also exceeded either component alone, supporting their complementarity in learning and retention.

In Figure 1(c), RASA achieved positive final gains on most tasks, showing that the improvement in average performance was accompanied by broad task-wise gains. For Gain-OLoRA, which already exhibited little decline after learning, the primary source of improvement was acquisition gain, which raised the baseline's low accuracy immediately after learning. For SeqLoRA, BWT gain contributed substantially by mitigating subsequent declines in previously acquired performance. By addressing each method's limitations in learning and retention, RASA achieved the highest mean final AP for each method (Table B.13).

4.6 Effects of Historical Feature Moments

For this analysis, we define average acquisition over the $K-1$ tasks excluding the last task as $\mathrm{Acq}_{\mathrm{old}} = (K-1)^{-1} \sum_{i=1}^{K-1} a_{i,i}$, and final average performance over these tasks as $\mathrm{AP}_{\mathrm{old}} = (K-1)^{-1} \sum_{i=1}^{K-1} a_{K,i}$. On this shared set of tasks, $\mathrm{AP}_{\mathrm{old}} = \mathrm{Acq}_{\mathrm{old}} + \mathrm{BWT}$.

We investigated how the directional information in historical feature moments affected retention using Gain-OLoRA with Meta-Llama-3.1-8B-Instruct on Long-sequence CL (Order 4). **Historical moments** used the second moments in Eq. (5). **Isotropic moments** replaced $\boldsymbol{C}_{\ell,j}$ with $\mathrm{tr}(\boldsymbol{C}_{\ell,j})\boldsymbol{I}/b_{\ell,j}$ in the upper bound F_ℓ on linear output changes in Eq. (7). Here, $b_{\ell,j}$ is the dimension of the corresponding block; this replacement preserves the trace while removing directional weighting. Normalization by the output energy E_ℓ in Eq. (8) and the retention tolerance ρ remained

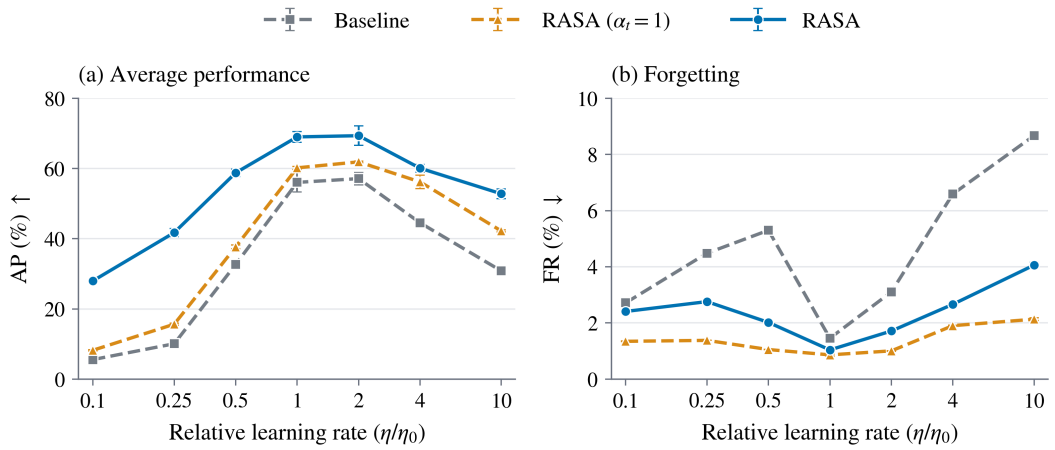


Figure 3: **Performance across learning rates.** (a) AP (↑) and (b) FR (↓) on Long-sequence CL (Order 4) with Meta-Llama-3.1-8B-Instruct and Gain-InfLoRA. Learning rates are relative to $\eta_0 = 5 \times 10^{-4}$. Baseline (gray squares) uses no RASA; RASA ($\alpha_t = 1$; orange triangles) omits online covariance matching; RASA (blue circles) uses all components. Means over three seeds; error bars show ± 1 standard deviation.

unchanged. **No retention constraint** removed both the constraint and correction. All three conditions retained online covariance matching and posterior-based candidate selection.

Historical moments achieved higher mean acquisition and BWT than Isotropic moments at the same ρ (Table B.14). In particular, Historical moments with $\rho = 0.025$ and Isotropic moments with $\rho = 0.05$ achieved similar acquisition, but the former better preserved performance during subsequent learning. This supports the contribution of directional information in historical feature moments to balancing acquisition and retention. For both moment conditions, acquisition increased and BWT decreased as ρ increased, with the highest $\mathrm{AP}_{\mathrm{old}}$ attained at $\rho = 0.05$ among the evaluated settings. Historical moments achieved higher final performance than both comparison conditions at every ρ, suggesting that directional weighting in the retention constraint helps account for both learning new tasks and retaining previous tasks.

4.7 Learning Rate Sensitivity

We analyzed how the effect of online covariance matching varied with the nominal learning rate. Using Meta-Llama-3.1-8B-Instruct and Gain-InfLoRA on Long-sequence CL with Order 4, we evaluated learning rates from 0.1 to 10 times the reference learning rate $\eta_0 = 5 \times 10^{-4}$ in Appendix B.3. We compared baseline Gain-InfLoRA without RASA, RASA with the time step fixed at $\alpha_t = 1$, and RASA. The retention constraint and candidate selection were identical in the two RASA conditions.

RASA achieved higher AP than both alternatives at every evaluated learning rate, especially over the unit-time-step variant at small rates. Fixing $\alpha_t = 1$ lowered FR but also final AP; RASA improved both metrics over the baseline. These results support covariance matching alongside retention-aware posterior inference across learning rates. Appendix B.10 reports time and memory increases over pure baselines on Standard CL (Order 1).

5 Conclusion

We presented RASA for LoRA-based expansion in LLM OCL, combining online covariance matching with retention-aware posterior inference guided by historical feature moments. Without replay or additional forward or backward passes, RASA consistently improves average performance and generally reduces forgetting across four benchmarks and multiple backbones. Component analyses support complementary adaptation and retention effects, while SeqLoRA and OSFT experiments demonstrate gains beyond expansion.

AI USE STATEMENT

We used generative AI tools to assist with manuscript writing and polishing and to identify relevant literature for the Related Work section. The authors take responsibility for the final manuscript, including its claims and references.

REPRODUCIBILITY STATEMENT

Appendix A provides the algorithm, derivations, and implementation details. Appendix B specifies the training settings, benchmarks, evaluation protocol, task orders, metrics, random seeds, score aggregation, and additional results; Section 4.1 summarizes the experimental protocol.

REFERENCES

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Pier Giovanni Bissiri, Chris C. Holmes, and Stephen G. Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 78 (5):1103–1130, 2016.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. SEEKR: Selective attention-guided knowledge retention for continual learning of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3266. Association for Computational Linguistics, 2024.

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

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

Qianxiao Li, Cheng Tai, and Weinan E. Stochastic modified equations and dynamics of stochastic gradient algorithms I: Mathematical foundations. *Journal of Machine Learning Research*, 20(40): 1–47, 2019.

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*, pp. 23638–23647, 2024.

Yan-Shuo Liang, Jia-Rui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 76577–76607. Curran Associates, Inc., 2025.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81. Association for Computational Linguistics, 2004.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Oleg Silkin, Mustafa Eyceoz, Hao Wang, Aldo Pareja, and Akash Srivastava. Sculpting subspaces: Constrained full fine-tuning in LLMs for continual learning. In *International Conference on Learning Representations*, 2026.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

Fuli Qiao and Mehrdad Mahdavi. Merge before forget: A single LoRA continual learning via continual merging. In *International Conference on Learning Representations*, 2026.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *International Conference on Learning Representations*, 2023.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025. Article 120.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. Association for Computational Linguistics, 2013.

Chenyang Song, Xu Han, Zheni Zeng, Kuai Li, Chen Chen, Zhiyuan Liu, Maosong Sun, and Tao Yang. ConPET: Continual parameter-efficient tuning for large language models. *arXiv preprint arXiv:2309.14763*, 2023.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. LAMOL: LAnguage MOdeling for lifelong language learning. In *International Conference on Learning Representations*, 2020.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671. Association for Computational Linguistics, 2023a.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023b.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. InsCL: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 663–677. Association for Computational Linguistics, 2024.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri,

Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109. Association for Computational Linguistics, 2022.

Zirui Wang, Sanket Vaibhav Mehta, Barnabás Póczos, and Jaime Carbonell. Efficient meta lifelong-learning with limited memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 535–548. Association for Computational Linguistics, 2020.

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. SAPT: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11641–11661. Association for Computational Linguistics, 2024.

A METHOD DETAILS AND DERIVATIONS

A.1 ONLINE PROCEDURE

Algorithm 1 Retention-Aware Stream Adaptation (RASA)

Require: Initial parameters $\hat{\boldsymbol{\theta}}_1$, task streams $\{\mathcal{T}_k\}_{k=1}^{K}$,
optimization process $\mathcal{O}_t$, retention tolerance ρ
  Initialization: $S^{\mathrm{past}} \leftarrow \varnothing$; $M, V \leftarrow 0$; $a, b, h \leftarrow 1$
  for each task $\mathcal{T}_k$ **do**
    Reset optimizer; initialize active parameters and task-start anchor $\hat{\boldsymbol{\theta}}_k^0$
    $\bar{\boldsymbol{\delta}} \leftarrow \mathbf{0}$, $n \leftarrow 0$, $S^k \leftarrow \varnothing$
    for each arriving batch $\mathcal{B}_t$ **do**
      Compute $\boldsymbol{g}_t^{\mathrm{CE}}$ and $\boldsymbol{g}_t$; accumulate feature statistics in S^k
      $\boldsymbol{\delta}_t \leftarrow \mathcal{O}_t(\hat{\boldsymbol{\theta}}_t, \boldsymbol{g}_t)$ ▷ One AdamW step
      Online Covariance Matching:
      Update $\bar{\boldsymbol{\delta}}$, v_t, and $\bar{v}_t$ from $\boldsymbol{\delta}_t$ ▷ Eqs. (A23)–(A26)
      $\alpha_t \leftarrow \mathrm{clip}\Big(\sqrt{\bar{v}_t / v_t}, L_t, U_t\Big)$ ▷ Eqs. (A27)–(A29)
      Retention-aware posterior inference:
      Form $\boldsymbol{H}_t$ from post-step Adam state; $\boldsymbol{u}_t \leftarrow -\boldsymbol{H}_t^{-1}\boldsymbol{g}_t^{\mathrm{CE}}$ ▷ Eq. (A9)
      $\boldsymbol{d}_t^{\mathrm{opt}} \leftarrow \boldsymbol{\delta}_t, \quad \boldsymbol{d}_t^{\mathrm{CE}} \leftarrow \min\Big\{1, \frac{\|\boldsymbol{\delta}_t\|_{\boldsymbol{H}_t}}{\|\boldsymbol{u}_t\|_{\boldsymbol{H}_t}}\Big\}\boldsymbol{u}_t$
      $\widetilde{\boldsymbol{\theta}}_t^{(j)} \leftarrow \mathcal{C}_k\Big(\hat{\boldsymbol{\theta}}_t, \mathcal{P}_t(\hat{\boldsymbol{\theta}}_t + \alpha_t \boldsymbol{d}_t^{(j)})\Big), \quad j \in \{\mathrm{opt}, \mathrm{CE}\}$
      $\mathcal{V}_t \leftarrow \{\hat{\boldsymbol{\theta}}_t\} \cup \{\widetilde{\boldsymbol{\theta}}_t^{(j)} : D_k(\widetilde{\boldsymbol{\theta}}_t^{(j)}) \le \rho^2/2\}$
      $\hat{\boldsymbol{\theta}}_{t+1} \leftarrow \arg\min_{\boldsymbol{\theta} \in \mathcal{V}_t} \widehat{q}_t(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t)$ ▷ Eqs. (A13)–(A14)
      Retain post-step Adam moments and counter for every selection outcome
      Update a, b, h only if $\hat{\boldsymbol{\theta}}_{t+1} \neq \hat{\boldsymbol{\theta}}_t$ ▷ Eqs. (A30)–(A31)
    end for
    $S^{\mathrm{past}} \leftarrow S^{\mathrm{past}} \oplus S^k$
  end for

Failed or nonfinite candidates are excluded. Ties favor the current parameters, then the optimizer candidate. Use $\alpha_t^{\mathrm{raw}} = 1$ for zero variance and $\boldsymbol{d}_t^{\mathrm{CE}} = 0$ for zero metric norm.

In Algorithm 1, S^{past} and S^k store the layerwise sums of input outer products, sums of squared output norms, and token counts accumulated over previous tasks and the current task, respectively. The operator $\oplus$ merges the corresponding sums and counts, from which the moments and output energies in Eqs. (5)–(6) are computed. The count $h = 1 + |\mathcal{A}_t|$ includes the initial unit observation in Eqs. (A30)–(A31) and the steps at which a nonzero update is accepted. The selection set $\mathcal{V}_t$ consists of the current parameters and corrected candidates that pass the retention constraint and numerical checks.

A.2 OBJECTIVE, METRIC, AND CANDIDATE SELECTION

Task notation and objective. We consider a language model $f : \mathcal{X} \to \mathcal{Y}$ with full parameters $\boldsymbol{\Theta}$, assigning a conditional probability $p_{\boldsymbol{\Theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ to each target sequence $\boldsymbol{y} \in \mathcal{Y}$ given an input sequence $\boldsymbol{x} \in \mathcal{X}$. Task k has distribution $\mathcal{D}_k$ and ends at step t_k, with $t_0 = 0$, so $\mathcal{B}_t$ arrives from $\mathcal{D}_k$ for $t_{k-1} < t \le t_k$. The generic active-parameter vector is $\boldsymbol{\theta} \in \mathbb{R}^{d_k}$; $\hat{\boldsymbol{\theta}}_t$ and $\hat{\boldsymbol{\theta}}_{t+1}$ denote the estimates adopted before and after step t, respectively. The task-start anchor is $\hat{\boldsymbol{\theta}}_k^0 = \hat{\boldsymbol{\theta}}_{t_{k-1}+1}$, saved after preparing the active parameters.

The objective after the full task sequence is to minimize the average expected risk:

$$R_k(\boldsymbol{\Theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_k}[\ell(\boldsymbol{\Theta};\boldsymbol{x},\boldsymbol{y})], \tag{A1}$$

$$R_{\text{all}}(\boldsymbol{\Theta}) = \frac{1}{K}\sum_{k=1}^{K} R_k(\boldsymbol{\Theta}). \tag{A2}$$

Here, ℓ is the autoregressive negative log-likelihood,

$$\ell(\boldsymbol{\Theta};\boldsymbol{x},\boldsymbol{y}) = -\log p_{\boldsymbol{\Theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = -\sum_{j=1}^{|\boldsymbol{y}|} \log p_{\boldsymbol{\Theta}}(y_j \mid \boldsymbol{x}, \boldsymbol{y}_{<j}), \tag{A3}$$

where y_j is the jth target token, $\boldsymbol{y}_{<j}$ is the preceding target prefix, and $|\boldsymbol{y}|$ is the target sequence length. For within-task updates, we omit the frozen parameters and write $\ell(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y})$. With past data unavailable, the current-batch objective and its CE gradient are

$$\ell_t(\boldsymbol{\theta}) = \frac{1}{m_t}\sum_{u=1}^{m_t} \frac{1}{Z_{t,u}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}_{t,u}} \ell(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y}), \tag{A4}$$

$$\boldsymbol{g}_t^{\text{CE}} = \nabla_{\boldsymbol{\theta}}\ell_t(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_t}. \tag{A5}$$

Here, m_t is the number of microbatches in the logical batch $\mathcal{B}_t$, $\mathcal{B}_{t,u}$ is its uth microbatch, and $Z_{t,u}$ counts valid target tokens, excluding prompt and padding tokens. Equation (A4) averages the microbatch token means; when microbatches contain different numbers of tokens, this differs from a single mean over all tokens.

Optimizer displacement and metric. We retain the CE gradient before regularization, projection, or clipping, and pass

$$\boldsymbol{g}_t = \boldsymbol{g}_t^{\text{CE}} + \boldsymbol{g}_t^{\Omega} \tag{A6}$$

to the optimizer. Here, $\boldsymbol{g}_t^{\Omega}$ is the regularization gradient, including the coefficient specified by the underlying method, and is zero when no regularization is used. Parameter-only regularization is computed without an additional data pass. The displacement produced by the optimization process $\mathcal{O}_t$ and the resulting parameters are

$$\boldsymbol{\delta}_t = \mathcal{O}_t(\hat{\boldsymbol{\theta}}_t, \boldsymbol{g}_t), \tag{A7}$$

$$\boldsymbol{\theta}_{t+1} = \hat{\boldsymbol{\theta}}_t + \boldsymbol{\delta}_t. \tag{A8}$$

The displacement incorporates the underlying method's update and projection; $\boldsymbol{\theta}_{t+1}$ denotes the result of that optimization process alone, before RASA selection.

The AdamW metric used in our experiments depends on the current step's bias-corrected gradient second-moment estimate $\widehat{s}_{t,i}$, the parameter-group learning rate $\eta_{t,i}$, and the optimizer constant $\epsilon_{\text{opt},i}$. Here, i indexes coordinates of the active parameter vector, and $\widehat{s}_{t,i}$ is the bias-corrected exponential moving average of the squared gradient coordinate supplied to the optimizer:

$$H_{t,ii} = \max\left(10^{-12}, \frac{\sqrt{\widehat{s}_{t,i}} + \epsilon_{\text{opt},i}}{\max(\eta_{t,i}, 10^{-30})}\right). \tag{A9}$$

The positive diagonal matrix $\boldsymbol{H}_t$ serves as a quadratic regularization metric. Optimizers without AdamW second moments require a separate metric definition.

At each logical batch, AdamW is advanced once; $\boldsymbol{H}_t$ uses the post-step second moment and step counter. Candidate selection changes only parameters: the updated Adam moments and counter are retained even when the CE candidate is selected or all candidates are rejected. Displacement statistics use the original projected AdamW proposal, including rejected proposals, whereas time-step moments are updated only for nonzero accepted updates using the nominal α_t. No second optimizer step is taken for the CE candidate.

Candidate construction and acceptance. The optimizer direction reflects accumulated gradient statistics and regularization, so it need not decrease CE on the current batch. We also form a direction from $\boldsymbol{u}_t = -\boldsymbol{H}_t^{-1}\boldsymbol{g}_t^{\mathrm{CE}}$, the displacement to the posterior mode before imposing the retention constraint. With $\|\boldsymbol{v}\|_{\boldsymbol{H}_t} = \sqrt{\boldsymbol{v}^\top \boldsymbol{H}_t \boldsymbol{v}}$, the two directions are

$$\boldsymbol{d}_t^{\mathrm{opt}} = \boldsymbol{\delta}_t, \tag{A10}$$

$$\boldsymbol{d}_t^{\mathrm{CE}} = \min\left\{1, \frac{\|\boldsymbol{\delta}_t\|_{\boldsymbol{H}_t}}{\|\boldsymbol{u}_t\|_{\boldsymbol{H}_t}}\right\}\boldsymbol{u}_t. \tag{A11}$$

Thus, the CE direction's metric norm is at most that of the optimizer displacement. If $\|\boldsymbol{u}_t\|_{\boldsymbol{H}_t} = 0$, we set $\boldsymbol{d}_t^{\mathrm{CE}} = \boldsymbol{0}$.

For each $j \in \{\mathrm{opt}, \mathrm{CE}\}$, we use the same time step $\Delta t_t = \alpha_t$, apply the underlying method's parameter projection $\mathcal{P}_t$, and perform retention correction $\mathcal{C}_k$:

$$\widetilde{\boldsymbol{\theta}}_t^{(j)} = \mathcal{C}_k\Big(\hat{\boldsymbol{\theta}}_t, \mathcal{P}_t(\hat{\boldsymbol{\theta}}_t + \alpha_t \boldsymbol{d}_t^{(j)})\Big). \tag{A12}$$

The map $\mathcal{P}_t$ projects onto the parameter space permitted by the underlying method and is the identity when no separate projection is required. Appendix A.7 specifies $\mathcal{C}_k$. For each corrected candidate, define $\boldsymbol{\Delta}_t^{(j)} = \widetilde{\boldsymbol{\theta}}_t^{(j)} - \hat{\boldsymbol{\theta}}_t$. We exclude candidates if correction fails or if the directional dot product $(\boldsymbol{g}_t^{\mathrm{CE}})^\top \boldsymbol{\Delta}_t^{(j)}$, metric norm $\|\boldsymbol{\Delta}_t^{(j)}\|_{\boldsymbol{H}_t}$, surrogate value $\widehat{q}_t(\boldsymbol{\Delta}_t^{(j)})$, or normalized upper bound D_k is nonfinite. Among candidates that pass these checks, the acceptance rule is

$$\mathcal{S}_t = \left\{ j \in \{\mathrm{opt}, \mathrm{CE}\} : \begin{array}{c} D_k(\widetilde{\boldsymbol{\theta}}_t^{(j)}) \le \rho^2/2, \\ \widehat{q}_t(\boldsymbol{\Delta}_t^{(j)}) < 0, \quad \|\boldsymbol{\Delta}_t^{(j)}\|_{\boldsymbol{H}_t}^2 > 0 \end{array} \right\}, \tag{A13}$$

$$\hat{\boldsymbol{\theta}}_{t+1} = \begin{cases} \widetilde{\boldsymbol{\theta}}_t^{(j^*)}, & \mathcal{S}_t \neq \varnothing, \quad j^* \in \operatorname{argmin}_{j \in \mathcal{S}_t} \widehat{q}_t(\boldsymbol{\Delta}_t^{(j)}), \\ \hat{\boldsymbol{\theta}}_t, & \mathcal{S}_t = \varnothing. \end{cases} \tag{A14}$$

The strict surrogate decrease favors the current parameters in a tie; ties between the two nonzero candidates favor the opt candidate. If no candidate is acceptable, the current parameters are retained. By Eq. (A39), this implements the restricted MAP rule in Eq. (13). Scoring uses only existing gradients and stored feature statistics, without an additional data pass.

A.3 Transition Distribution and Covariance Matching

Auxiliary transition model. The following auxiliary SDE realizes the Gaussian scaling model in Section 3.2. Let s denote continuous time within step t and let $\boldsymbol{\theta}_t(s)$ be a random parameter process initialized at $\hat{\boldsymbol{\theta}}_t$. With a reference interval of one, the normalized time step is $\Delta t_t = \alpha_t > 0$. This dimensionless factor scales the observed displacement $\boldsymbol{\delta}_t$, which already incorporates the optimizer learning rate. We choose the diffusion coefficient $\sqrt{\Delta t_t v_t}$ so that the displacement magnitude and its standard deviation scale with the same time step. Holding $\boldsymbol{\delta}_t$, v_t, and Δt_t fixed within a step gives

$$d\boldsymbol{\theta}_t(s) = \boldsymbol{\delta}_t\, ds + \sqrt{\Delta t_t v_t}\, d\mathbf{W}_s, \qquad 0 \le s \le \Delta t_t, \tag{A15}$$

$$\boldsymbol{\theta}_t(0) = \hat{\boldsymbol{\theta}}_t, \tag{A16}$$

where $\mathbf{W}_s$ is a d_k-dimensional standard Brownian motion. For a fixed time step $\alpha = \Delta t_t > 0$ and variance proxy $v_t \ge 0$, integrating Eq. (A15) with the initial condition in Eq. (A16) over $[0, \alpha]$ gives

$$\boldsymbol{\theta}_t(\alpha) = \hat{\boldsymbol{\theta}}_t + \alpha\boldsymbol{\delta}_t + \sqrt{\alpha v_t}(\mathbf{W}_\alpha - \mathbf{W}_0), \tag{A17}$$

$$\mathbf{W}_\alpha - \mathbf{W}_0 \sim \mathcal{N}(0, \alpha\boldsymbol{I}), \tag{A18}$$

$$\boldsymbol{\theta}_t(\alpha) \sim \mathcal{N}(\hat{\boldsymbol{\theta}}_t + \alpha\boldsymbol{\delta}_t, \alpha^2 v_t \boldsymbol{I}). \tag{A19}$$

This establishes Eq. (2), conditioned on the current parameters and update statistics. Its mean $\boldsymbol{m}_t^- = \hat{\boldsymbol{\theta}}_t + \alpha_t\boldsymbol{\delta}_t$ and covariance $\boldsymbol{Q}_t = \alpha_t^2 v_t \boldsymbol{I}$ describe the uncorrected local movement used to determine candidate scale, and $\boldsymbol{I}$ is the $d_k \times d_k$ identity matrix. At $\alpha = 1$, the mean equals the parameters produced by the underlying optimization process in Eq. (A8). The α^2 dependence of the covariance follows from the local modeling assumption that sets the diffusion coefficient to $\sqrt{\alpha v_t}$. We do not assume a continuous-time limit of the actual optimizer, and the implementation does not sample Brownian noise.

Table A.1: RASA statistics and numerical settings. Initialization and update rules for update statistics and numerical constants, distinguishing task-local statistics from statistics maintained across tasks.

Item	Initialization and update
Task-local update mean and count	Reset at each task boundary. The first observation sets the mean to that update and $v_{k,1} = 0$.
Reference count M and mean V	Initialized to zero and maintained across tasks. Equation (A25) includes all finite observations, including the first zero residual and steps whose final candidate is rejected.
Time-step moments a, b	$a_0 = b_0 = 1$. The nominal α_t before correction is recorded in Eqs. (A30)–(A31) only when a nonzero update is accepted. Maintained across tasks.
Zero variance	If $v_t = 0$ or $\bar{v}_t = 0$, then $\alpha_t^{\mathrm{raw}} = 1$. No separate variance floor is used.
Retention settings	$\rho = 0.05$ and output-energy denominator constant $\varepsilon = 10^{-12}$. Without past statistics, $D_k \equiv 0$.

Proposition A.1 (Covariance matching). If $v_t, \bar{v}_t > 0$ and $0 < L_t \le U_t < \infty$, the time step that minimizes the Kullback–Leibler (KL) divergence between the centered Gaussian distributions over $\alpha \in [L_t, U_t]$ is given by Eq. (4).

Proof. The closed-form expression for the Gaussian KL divergence yields

$$\begin{aligned}\mathcal{J}_t(\alpha) &= D_{\mathrm{KL}}\big(\mathcal{N}(0, \alpha^2 v_t \boldsymbol{I}) \,\|\, \mathcal{N}(0, \bar{v}_t \boldsymbol{I})\big) \\ &= \frac{d_k}{2}\left(\frac{\alpha^2 v_t}{\bar{v}_t} - \log\frac{\alpha^2 v_t}{\bar{v}_t} - 1\right),\end{aligned} \tag{A20}$$

$$\mathcal{J}_t'(\alpha) = d_k\left(\frac{\alpha v_t}{\bar{v}_t} - \frac{1}{\alpha}\right), \tag{A21}$$

$$\mathcal{J}_t''(\alpha) = d_k\left(\frac{v_t}{\bar{v}_t} + \frac{1}{\alpha^2}\right) > 0. \tag{A22}$$

Thus, the unique minimizer over $\alpha > 0$ is $\sqrt{\bar{v}_t / v_t}$. Since the objective decreases to the left of this value and increases to its right, the minimizer over $[L_t, U_t]$ is $\mathrm{clip}(\sqrt{\bar{v}_t / v_t}, L_t, U_t)$. □

A.4 Update Statistics and Numerical Settings

The variance proxy and reference variance are computed exclusively from the optimizer displacement $\boldsymbol{\delta}_t$ in Eq. (A7), which incorporates the learning rate, optimizer state, and weight decay. They are not recomputed from the CE direction or corrected update. For the nth observed update $\boldsymbol{\delta}_{k,n}$ in task k, we maintain its running mean and dimension-normalized residual energy:

$$\bar{\boldsymbol{\delta}}_{k,n} = \bar{\boldsymbol{\delta}}_{k,n-1} + \frac{\boldsymbol{\delta}_{k,n} - \bar{\boldsymbol{\delta}}_{k,n-1}}{n}, \tag{A23}$$

$$v_{k,n} = \frac{\|\boldsymbol{\delta}_{k,n} - \bar{\boldsymbol{\delta}}_{k,n}\|_2^2}{d_k}. \tag{A24}$$

Here, $\|\cdot\|_2$ is the Euclidean norm, and $v_t = v_{k,n}$ when global step t corresponds to observation n in task k. This is a residual-energy proxy rather than an unbiased sample variance. The mean and count are reset at task boundaries; the first observation sets $\bar{\boldsymbol{\delta}}_{k,1} = \boldsymbol{\delta}_{k,1}$ and $v_{k,1} = 0$.

Let M and V denote the count and mean of the variance proxies accumulated before the current observation. Including the current observation, the reference variance is

$$\bar{v}_t = \frac{MV + v_t}{M + 1}, \tag{A25}$$

$$(M, V) \leftarrow (M + 1, \bar{v}_t). \tag{A26}$$

Both statistics are initialized to zero and maintained across tasks. All finite observations contribute, including the first zero residual of each task and steps whose final candidate is rejected.

The reference covariance is $\boldsymbol{Q}_t^{\mathrm{ref}} = \bar{v}_t \boldsymbol{I}$. For positive variances, matching it to $\boldsymbol{Q}_t$ in Eq. (2) yields the raw time step in Eq. (3); if $v_t = 0$ or $\bar{v}_t = 0$, we instead set $\alpha_t^{\mathrm{raw}} = 1$, without a separate variance floor. To limit extreme variance ratios, we use bounds determined by the second moments a_{t-1} of accepted nominal time steps and b_{t-1} of their inverses:

$$L_t = \min\left(1, (2b_{t-1})^{-1/2}\right), \tag{A27}$$

$$U_t = \max\left(1, (2a_{t-1})^{1/2}\right), \tag{A28}$$

$$\Delta t_t = \alpha_t = \mathrm{clip}(\alpha_t^{\mathrm{raw}}, L_t, U_t). \tag{A29}$$

Here, $\mathrm{clip}(x, L, U) = \min\{U, \max\{L, x\}\}$. Let $\mathcal{A}_t$ be the set of step indices at which a nonzero update is accepted through step t. Then

$$a_t = \frac{1 + \sum_{i \in \mathcal{A}_t} \alpha_i^2}{1 + |\mathcal{A}_t|}, \tag{A30}$$

$$b_t = \frac{1 + \sum_{i \in \mathcal{A}_t} \alpha_i^{-2}}{1 + |\mathcal{A}_t|}. \tag{A31}$$

Only the nominal α_i before retention correction is recorded, and only if a nonzero update is accepted. The initialization $a_0 = b_0 = 1$ gives initial bounds $[1/\sqrt{2}, \sqrt{2}]$; both moments are maintained across tasks.

The factor of 2 in Eqs. (A27)–(A28) and the initial unit observation in Eqs. (A30)–(A31) are fixed constants. The bounds vary with the accepted time-step moments and are not absolute bounds shared across the entire training run. When $v_t = 0$, the transition covariance is zero, but candidate scoring remains well defined because the posterior in Eq. (12) uses the positive metric $\boldsymbol{H}_t$.

A.5 Historical Linear-Output Change Bound

Proposition A.2 (Stored-feature bound). Suppose that $\mathcal{L} \neq \varnothing$ and that the second moments in Eq. (5) are given. Let $\widetilde{E}_\ell = \max(E_\ell, \varepsilon)$. Then, for every $\boldsymbol{\theta} \in \mathcal{R}_k$,

$$\frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \frac{1}{N_\ell \widetilde{E}_\ell} \sum_i \|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_{\ell,i}\|_2^2 \leq 2D_k(\boldsymbol{\theta}) \leq \rho^2. \tag{A32}$$

Proof. Applying the blockwise Cauchy–Schwarz inequality to each layer and token and substituting Eq. (5) gives the following, where tr denotes the matrix trace:

$$\begin{aligned} \frac{1}{N_\ell} \sum_i \|\Delta \boldsymbol{W}_\ell \boldsymbol{h}_{\ell,i}\|_2^2 &= \frac{1}{N_\ell} \sum_i \left\| \sum_j \Delta \boldsymbol{W}_{\ell,j} \boldsymbol{h}_{\ell,i}^{(j)} \right\|_2^2 \\ &\leq \frac{J_\ell}{N_\ell} \sum_{i,j} \|\Delta \boldsymbol{W}_{\ell,j} \boldsymbol{h}_{\ell,i}^{(j)}\|_2^2 \\ &= J_\ell \sum_j \mathrm{tr}\big(\Delta \boldsymbol{W}_{\ell,j} \boldsymbol{C}_{\ell,j} \Delta \boldsymbol{W}_{\ell,j}^\top\big) = F_\ell(\boldsymbol{\theta}). \end{aligned} \tag{A33}$$

Dividing by $\widetilde{E}_\ell$ and averaging over layers gives the first inequality by Eq. (8). The second follows from $\boldsymbol{\theta} \in \mathcal{R}_k$. □

This result establishes the historical response budget in Eq. (10): feasibility controls cumulative change from the task-start anchor using only stored moments. The anchor is feasible, and candidate selection preserves feasibility at every accepted iterate (Appendix A.6). Here, $\boldsymbol{C}_{\ell,j}$ is an uncentered second moment; no independence assumption between blocks is required. In the LoRA-expansion setting, we interpret the stored-feature constraint under an assumption that layer-input features on historical examples remain close to their values at collection time during subsequent adaptation. The

frozen pretrained backbone and earlier branches motivate this assumption, but do not guarantee it. Proposition A.2 remains exact for the recorded features regardless of subsequent drift. The bound does not account for feature drift, nonlinear propagation, or changes in branch routing, and therefore supports a linear-response retention surrogate rather than an end-to-end loss or accuracy guarantee.

A.6 Generalized Posterior and Candidate Selection

Suppose that $\boldsymbol{H}_t \succ 0$ and $\rho > 0$, and fix the stored statistics. Completing the square in Eq. (11) gives

$$\widehat{q}_t(\boldsymbol{\Delta}) = \frac{1}{2}(\boldsymbol{\Delta} - \boldsymbol{u}_t)^\top \boldsymbol{H}_t(\boldsymbol{\Delta} - \boldsymbol{u}_t) - \frac{1}{2}\boldsymbol{g}_t^{\mathrm{CE}\top}\boldsymbol{H}_t^{-1}\boldsymbol{g}_t^{\mathrm{CE}}, \tag{A34}$$

$$\boldsymbol{u}_t = -\boldsymbol{H}_t^{-1}\boldsymbol{g}_t^{\mathrm{CE}}, \tag{A35}$$

$$\pi_t(\boldsymbol{\theta}) \propto \mathcal{N}(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}_t + \boldsymbol{u}_t, \boldsymbol{H}_t^{-1})\, \mathbf{1}\{\boldsymbol{\theta} \in \mathcal{R}_k\}. \tag{A36}$$

The Gaussian integral is finite. Moreover, D_k is continuous and $D_k(\hat{\boldsymbol{\theta}}_k^0) = 0$, so a feasible neighborhood exists around the anchor. The normalization constant is therefore positive and finite. When no historical statistics are available, $\mathcal{R}_k = \mathbb{R}^{d_k}$.

Generalized updating. Let $\nu_t = \mathcal{N}(\hat{\boldsymbol{\theta}}_t, \boldsymbol{H}_t^{-1})$ and

$$\widetilde{Z}_t = \int_{\mathcal{R}_k} e^{-\boldsymbol{g}_t^{\mathrm{CE}\top}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t)} \nu_t(\boldsymbol{\theta})\, d\boldsymbol{\theta}. \tag{A37}$$

For any density q such that $q(\mathcal{R}_k) = 1$ and the following terms are finite, substituting $\log \pi_t$ into the definition of KL divergence gives

$$\mathbb{E}_q[\boldsymbol{g}_t^{\mathrm{CE}\top}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t)] + D_{\mathrm{KL}}(q\|\nu_t) = D_{\mathrm{KL}}(q\|\pi_t) - \log \widetilde{Z}_t. \tag{A38}$$

By the nonnegativity of KL divergence, π_t minimizes the left-hand side. This is an instance of the generalized Bayesian inference described in Section 2.3. The distribution ν_t is a data-dependent local reference, distinct from the transition distribution p_t^- in Eq. (2). The two constructions meet in candidate selection: covariance matching supplies α_t, historical moments determine $\mathcal{R}_k$, and π_t ranks the corrected feasible candidates.

Proposition A.3 (Restricted MAP). Suppose that the current parameters are feasible and that $\mathcal{F}_t$ is the set of corrected candidates that satisfy the retention constraint and pass the finite-value checks. The acceptance rule in Eqs. (A13)–(A14) implements the restricted MAP update in Eq. (13), maximizing π_t over $\mathcal{F}_t \cup \{\hat{\boldsymbol{\theta}}_t\}$. Ties favor the current parameters; a tie between two nonzero candidates favors the opt candidate.

Proof. For any feasible $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_t + \boldsymbol{\Delta}$, canceling the normalization constant gives

$$\log \pi_t(\boldsymbol{\theta}) - \log \pi_t(\hat{\boldsymbol{\theta}}_t) = -\widehat{q}_t(\boldsymbol{\Delta}). \tag{A39}$$

Maximizing the density is therefore equivalent to minimizing the surrogate. Since the surrogate value at the current parameters is $\widehat{q}_t(0) = 0$, a nonzero candidate is selected only if $\widehat{q}_t < 0$. Under $\boldsymbol{H}_t \succ 0$, a zero metric norm is equivalent to $\boldsymbol{\Delta} = 0$, so the norm check in Eq. (A13) is consistent with this rule. The anchor is feasible at the start of each task. Thereafter, the procedure either accepts a feasible candidate or retains the current parameters, preserving feasibility by induction. □

A.7 Feature Statistics and Retention Correction

Feature collection. For each layer, we accumulate valid-token input outer products, squared output norms, and token counts from the initial training forward pass. The input second moments are uncentered: outer products are averaged without subtracting the input mean. Output energy is measured at the corresponding LoRA-adapted linear layer. Current-task sums and counts are maintained separately and merged into the past statistics at the end of the task. Thus, Eq. (5) gives a token-count-weighted moment rather than an unweighted average of taskwise means, and the training constraint uses only statistics from previous tasks.

For input block widths $b_{\ell,j}$, moment storage requires $O(\sum_{\ell,j} b_{\ell,j}^2)$ memory. With a common block width b, $J_\ell = \lceil d_{\mathrm{in},\ell}/b \rceil$, and the per-layer storage cost is $O(d_{\mathrm{in},\ell} b)$. The partition also determines the bound in Eq. (A33) and the feasible set. In our LoRA experiments, each target module uses four equal, contiguous input blocks, fixed throughout training; the input dimension must be divisible by four. Block widths are 1024 for Meta-Llama-3.1-8B-Instruct and 640 for Qwen3-4B.

Expandable LoRA. Following the notation in the main text, the effective weight of the active branch is $\boldsymbol{W}_\ell(\boldsymbol{\theta}) = \gamma_\ell \boldsymbol{B}_\ell \boldsymbol{A}_\ell$, where $\boldsymbol{A}_\ell \in \mathbb{R}^{r \times d_{\mathrm{in},\ell}}$, $\boldsymbol{B}_\ell \in \mathbb{R}^{d_{\mathrm{out},\ell} \times r}$, and γ_ℓ is a fixed multiplier. We save the task-start anchor after preparing the active parameters.

Consider the case where previous branches are frozen and the new branch is initialized with $\boldsymbol{B}_\ell^0 = 0$. Let $\boldsymbol{\theta}^{\mathrm{cand}}$ denote a projected candidate. Keeping its $\boldsymbol{A}_\ell$ factors unchanged and rescaling all new up factors as $\boldsymbol{B}_\ell \leftarrow c\boldsymbol{B}_\ell$ gives

$$\Delta \boldsymbol{W}_\ell' = c \Delta \boldsymbol{W}_\ell, \tag{A40}$$

$$F_\ell' = c^2 F_\ell, \tag{A41}$$

$$D_k' = c^2 D_k. \tag{A42}$$

The first equality follows because the branch weight at the anchor is zero; the remaining equalities follow from the quadratic form in Eq. (7). Therefore, the following choice ensures feasibility in exact arithmetic:

$$c = \begin{cases} 1, & D_k(\boldsymbol{\theta}^{\mathrm{cand}}) \le \rho^2/2, \\ \rho / \sqrt{2 D_k(\boldsymbol{\theta}^{\mathrm{cand}})}, & D_k(\boldsymbol{\theta}^{\mathrm{cand}}) > \rho^2/2. \end{cases} \tag{A43}$$

In the implementation, the scaling factor for an infeasible candidate is multiplied by $1 - 10^{-5}$, and feasibility is rechecked up to three times. The down factors and active gating parameters retain their projected candidate values. This correction is not a Euclidean projection in parameter space. As in Eq. (8), D_k normalizes and aggregates upper bounds on the linear-output changes induced by LoRA effective-weight changes on the input features observed when the statistics were collected.

Shared LoRA. When the same factors continue to be updated in subsequent tasks, we evaluate points between the current feasible parameters and the projected candidate:

$$\boldsymbol{\theta}(c) = \hat{\boldsymbol{\theta}}_t + c(\boldsymbol{\theta}^{\mathrm{cand}} - \hat{\boldsymbol{\theta}}_t). \tag{A44}$$

The same c is applied to all active parameters. Starting from $c = 1$, we perform at most 16 halvings. If no feasible candidate is found, the direction is excluded. Interpolating both LoRA factors makes their weight product quadratic in c and D_k generally quartic, so we do not assume that feasibility is monotone in c. Both correction procedures evaluate D_k using the cumulative weight change relative to the task-start anchor.

B Experimental Details and Additional Results

B.1 Task Orders

Orders 1–3 comprise the four Standard CL tasks, Orders 4–6 comprise the 15 Long-sequence CL tasks, and Orders 7–8 comprise the 15 SuperNI tasks. Order 9 follows the default task sequence of TRACE (Wang et al., 2023b). Amazon, Yahoo, and Yelp denote Amazon Reviews, Yahoo Answers, and Yelp Reviews, respectively. We abbreviate the long SuperNI task names using their task IDs.

B.2 Score Aggregation

Standard CL comprises four classification tasks: AG News, Amazon Reviews, DBpedia, and SST-2. Long-sequence CL uses the 15 classification tasks listed in Table B.1 (Zhang et al., 2015; Socher et al., 2013; Razdaibiedina et al., 2023).

The main comparison uses three, three, two, and one task orders for Standard CL, Long-sequence CL, SuperNI, and TRACE-8, respectively (Table B.1), with three random seeds: 1337, 555, and 128. We first compute AP and FR for each run and then average them with equal weight over all order–seed combinations within each benchmark.

Table B.1: Task sequences for Standard CL (Orders 1–3), Long-sequence CL (Orders 4–6), SuperNI (Orders 7–8), and TRACE-8 (Order 9). Arrows indicate the order in which tasks are learned. Table B.4 provides the full names corresponding to the SuperNI task IDs.

Order	Benchmark	Task sequence
1	Standard CL	DBpedia → Amazon → SST-2 → AG News
2	Standard CL	DBpedia → Amazon → AG News → SST-2
3	Standard CL	SST-2 → Amazon → AG News → DBpedia
4	Long-sequence CL	MNLI → CB → WiC → COPA → QQP → BoolQ → RTE → IMDB → Yelp → Amazon → SST-2 → DBpedia → AG News → MultiRC → Yahoo
5	Long-sequence CL	MultiRC → BoolQ → WiC → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yelp → Amazon → Yahoo
6	Long-sequence CL	Yelp → Amazon → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yahoo → MultiRC → BoolQ → WiC
7	SuperNI	`task1572` → `task363` → `task1290` → `task181` → `task002` → `task1510` → `task639` → `task1729` → `task073` → `task1590` → `task748` → `task511` → `task591` → `task1687` → `task875`
8	SuperNI	`task748` → `task073` → `task1590` → `task639` → `task1572` → `task1687` → `task591` → `task363` → `task1510` → `task1729` → `task181` → `task511` → `task002` → `task1290` → `task875`
9	TRACE-8	C-STANCE → FOMC → MeetingBank → Py150 → ScienceQA → NumGLUE-cm → NumGLUE-ds → 20Minuten

Table B.2: Training settings for LoRA methods and OSFT. Each baseline–RASA pair uses the same optimizer and nominal learning rate. In the learning-rate analysis (Section 4.7), only the LoRA learning rate is changed by the specified factors.

Setting	LoRA methods	OSFT
Optimizer	AdamW; $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$	AdamW; $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$
Nominal learning rate	5×10^{-4}, constant	10^{-5}, constant
Weight decay	0	0.01
Microbatch / accumulation	2 / 16	2 / 16
Effective batch size / epochs	32 / 1	32 / 1
Optimizer state	Reset for each task	Reset for each task
Trainable parameters	Active adapter/router; frozen backbone	Factors associated with lower singular values and non-target parameters

Each value in Tables B.6 and B.7 averages 9, 9, 6, and 3 runs, respectively, for a given backbone. Table 1 further averages the results of the two backbones with equal weight, so each value represents 18, 18, 12, and 6 runs, respectively. We do not average scores across different benchmarks.

The component analysis uses the same Long-sequence CL aggregation: all three task orders (Orders 4–6) and seeds 1337, 555, and 128. Each value in Tables B.10 and B.11 averages nine runs for one backbone, while Table 2 averages all 18 runs across the two backbones with equal weight.

Task scores are on a 0–100 scale, and AP, FR, and BWT are defined in Appendix B.5. FR is not clipped at zero and can therefore be negative. It is generally not equivalent to the negative of BWT, which uses a different reference point.

Table B.3: Method-specific configurations. Trainable parameters, projection rules, and regularization settings for each baseline. These settings are identical within each baseline–RASA pair.

Method	Configuration
SeqLoRA / IncLoRA	Reuse a single adapter / add a new branch for each task, respectively. Previous branches are frozen under expansion.
O-LoRA / Gain-OLoRA	Train the new $\boldsymbol{A}, \boldsymbol{B}$. The penalty is $0.5 \sum_{\text{old}} \|\boldsymbol{A}_{\text{old}} \boldsymbol{A}_{\text{new}}^{\top}\|_1$, where the norm is the sum of absolute values of all entries. No separate L2 penalty is used.
InfLoRA / Gain-InfLoRA	Train only the new $\boldsymbol{B}$ and keep $\boldsymbol{A}$ fixed. After the first task, project $\boldsymbol{A}$ onto the complement of the past Gradient Projection Memory (GPM) subspace (Saha et al., 2021) and normalize each row to norm $1/\sqrt{3}$.
Gain variants	Add training of the current router/key to the corresponding base update.
OSFT	Apply to `q/k/v/o_proj` and `gate/up/down_proj`. Freeze the top 75% of singular components and train the bottom 25% of factors. Update the SVD at each task boundary.

B.3 Experimental Training Settings

We use the following training settings for each base method and its RASA counterpart.

The main experiments use microbatches of two examples with 16 accumulation steps per full logical batch (effective batch size 32). We retain the final incomplete batch and normalize its objective by the actual number of microbatches, as in Eq. (A4).

The main experiments use at most 500 training examples per task; Appendix B.11 additionally reports results with full training data. For Standard CL, Long-sequence CL, and SuperNI, the subset consists of the first $\min(500, N)$ records of the provided training split, where N is its size; no random subset sampling is performed. LoRA methods preserve the order of the selected records, whereas OSFT shuffles them once per task. The OSFT shuffle seed is $1337 + (k - 1)$, where k is the task index starting from 1. For TRACE-8, we use fixed files with overlapping prompts between training and evaluation removed.

Although the nominal learning rate is constant, RASA applies adaptive scaling and retention correction to the optimizer displacement. Unlike the optimizer state, which is reset for each task, RASA's reference variance and time-step statistics are maintained across tasks. The retention tolerance is $\rho = 0.05$; the remaining initialization rules and numerical constants follow Appendix A.4.

B.4 Baseline Methods and Configurations

SeqLoRA. A single LoRA adapter is trained sequentially across tasks without an additional constraint on forgetting (Liang et al., 2025).

IncLoRA. A new LoRA branch is added for each task, while previous branches are frozen (Wang et al., 2023a).

O-LoRA. Task-specific branches are added, and orthogonality regularization between the new branch and previous branches reduces interference (Wang et al., 2023a).

InfLoRA. The training subspace of each new branch is constructed to be orthogonal to the gradient subspace of previous tasks, reducing interference (Liang & Li, 2024).

Gain-OLoRA / Gain-InfLoRA. These methods apply GainLoRA's input-dependent gating to O-LoRA and InfLoRA, respectively, to integrate their branches and limit the influence of the new branch on previous tasks (Liang et al., 2025).

OSFT. OSFT preserves the principal singular subspaces of weight matrices and trains in the orthogonal subspace, updating the SVD at each task boundary (Nayak et al., 2026). Because its

Table B.4: SuperNI task identifiers. The table lists the IDs and dataset task names of the 15 tasks in Orders 7–8. All tasks are evaluated using ROUGE-L (↑) (Wang et al., 2022; Lin, 2004).

Task ID	Task name
task002	quoref_answer_generation
task073	commonsenseqa_answer_generation
task181	outcome_extraction
task363	sst2_polarity_classification
task511	reddit_tifu_long_text_summarization
task591	sciq_answer_generation
task639	multi_woz_user_utterance_generation
task748	glucose_reverse_cause_event_detection
task875	emotion_classification
task1290	xsum_summarization
task1510	evalution_relation_extraction
task1572	samsum_summary
task1590	diplomacy_text_generation
task1687	sentiment140_classification
task1729	personachat_generate_next

Table B.5: TRACE-8 evaluation metrics. We use the official TRACE metric for each task in Order 9. Higher values are better for all metrics (Wang et al., 2023b).

Task	Metric
C-STANCE, FOMC, ScienceQA, NumGLUE-cm, NumGLUE-ds	Accuracy
MeetingBank	ROUGE-L
Py150	Edit similarity
20Minuten	SARI

activation-based rank selection reuses past-task samples, we use a fixed retention ratio of 0.75 across all targeted layers and tasks to satisfy our replay-free OCL protocol. The same setting is used for OSFT and OSFT+RASA.

Configurations. The model architecture and underlying training procedure of each baseline follow its official paper, with the protocol-specific adaptations specified here. We apply the common OCL protocol in Appendix B.3 and the settings specified below. Apart from the update procedure, statistics, and hyperparameters required to apply RASA, all settings are identical within each baseline–RASA pair.

LoRA is applied to `q_proj` and `v_proj` in all Transformer blocks, with rank $r = 4$, LoRA scaling coefficient $\alpha_{\mathrm{LoRA}} = 32$, and zero dropout. The multiplier is $\alpha_{\mathrm{LoRA}}/r = 8$. The base branch initializes $\boldsymbol{A}$ with Kaiming uniform initialization and $\boldsymbol{B}$ to $\mathbf{0}$. No quantization is used.

RASA with OSFT. OSFT applies gradient projection, gradient-norm clipping at 1, AdamW, and parameter projection, in that order. RASA uses the resulting projected displacement for covariance matching and applies OSFT projection and retention correction to both candidate directions. The active parameter vector includes the trainable lower-singular-value factors and non-target parameters. For retention, $\boldsymbol{W}_\ell(\boldsymbol{\theta})$ denotes the reconstructed effective weight of a target linear map. Its change from the task-start anchor is evaluated with the feature-moment bound and normalization in Eqs. (7)–(8), using four input blocks. The anchor is saved after the task-boundary SVD refresh. Candidate selection retains the post-step Adam state as described in Appendix A.2. We evaluate on held-out sets of seen tasks; test scores are not used to select candidates.

Table B.6: Main results with Meta-Llama-3.1-8B-Instruct. Each value is the arithmetic mean over the task orders and three random seeds for the corresponding benchmark (Appendix B.2). Higher AP and lower FR are better. Bold indicates the better value within each baseline–RASA pair.

Method	Standard CL		Long-sequence CL		SuperNI		TRACE-8	
	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
OSFT	38.32	53.03	70.88	13.00	49.18	6.89	52.41	3.72
+RASA (ours)	**57.25**	**28.47**	**77.92**	**8.52**	**52.60**	**4.33**	**55.73**	**2.41**
Gain-InfLoRA	48.25	1.60	51.27	0.53	50.91	0.77	49.50	0.85
+RASA (ours)	**56.20**	**1.33**	**67.93**	**0.37**	**52.30**	**0.50**	**50.54**	**0.58**
Gain-OLoRA	56.70	−0.60	60.98	0.85	51.21	0.88	53.59	0.60
+RASA (ours)	**72.83**	**−1.33**	**71.99**	**0.80**	**53.14**	**0.69**	**58.19**	**0.34**
InfLoRA	55.53	24.17	78.30	4.39	50.76	5.84	56.07	2.03
+RASA (ours)	**67.95**	**8.17**	**80.34**	**2.81**	**52.00**	**4.43**	**57.66**	**1.57**
O-LoRA	38.05	52.37	63.76	19.92	49.95	6.47	55.85	3.28
+RASA (ours)	**41.45**	**34.76**	**68.39**	**12.18**	**52.33**	**4.82**	**58.76**	**1.91**
IncLoRA	47.75	21.84	55.91	17.63	44.78	9.41	57.25	4.26
+RASA (ours)	**67.78**	**10.57**	**63.80**	**13.13**	**51.30**	**6.53**	**58.95**	**2.06**
SeqLoRA	39.05	52.97	51.65	29.20	35.98	19.36	52.57	9.17
+RASA (ours)	**42.15**	**44.13**	**70.64**	**11.96**	**49.66**	**7.87**	**57.71**	**2.85**

B.5 Task Identifiers and Evaluation Metrics

Let K denote the number of tasks and $a_{k,i}$ the score on task i after learning task k. Average performance (AP, ↑) measures final average performance:

$$\mathrm{AP} = \frac{1}{K}\sum_{i=1}^{K} a_{K,i}. \tag{B1}$$

Forgetting (FR, ↓) measures the decline from the best previous performance:

$$\mathrm{FR} = \frac{1}{K-1}\sum_{i=1}^{K-1}\left(\max_{i\le k<K} a_{k,i} - a_{K,i}\right). \tag{B2}$$

Backward transfer (BWT, ↑) measures the change induced by subsequent learning relative to performance immediately after learning each task:

$$\mathrm{BWT} = \frac{1}{K-1}\sum_{i=1}^{K-1}\left(a_{K,i} - a_{i,i}\right). \tag{B3}$$

Positive BWT indicates positive backward transfer (Lopez-Paz & Ranzato, 2017).

B.6 Main Results

Tables B.6 and B.7 report the results in Table 1 separately for each backbone. We use three, three, two, and one task orders for Standard CL, Long-sequence CL, SuperNI, and TRACE-8, respectively, with seeds 1337, 555, and 128. Scores are aggregated as described in Appendix B.2.

RASA improves AP in all 28 comparisons for each backbone, supporting Table 1 across models. FR decreases in all 28 comparisons for Meta-Llama-3.1-8B-Instruct and 26 for Qwen3-4B. The two exceptions are Gain-OLoRA and O-LoRA on Long-sequence CL with Qwen3-4B: FR increases by 0.16 and 0.33 points, while AP improves by 5.05 and 3.81 points, respectively.

B.7 Additional Backbones

We report additional results on Standard CL, Long-sequence CL, and SuperNI using six LoRA methods with Qwen3-8B and Llama-2-7B-Chat (Tables B.8 and B.9). The training and RASA

Table B.7: Main results with Qwen3-4B. Each value is the arithmetic mean over the task orders and three random seeds for the corresponding benchmark (Appendix B.2). Higher AP and lower FR are better. Bold indicates the better value within each baseline–RASA pair.

Method	Standard CL		Long-sequence CL		SuperNI		TRACE-8	
	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
OSFT	40.48	49.17	67.41	16.56	51.16	6.89	49.34	2.98
+RASA (ours)	**49.63**	**36.63**	**76.87**	**6.32**	**53.78**	**5.56**	**52.60**	**2.44**
Gain-InfLoRA	48.08	0.33	50.15	0.18	49.72	0.65	41.17	0.37
+RASA (ours)	**49.20**	**0.20**	**54.38**	**0.06**	**51.77**	**−0.13**	**42.24**	**0.07**
Gain-OLoRA	56.00	0.23	63.38	**0.74**	51.16	0.61	45.74	0.68
+RASA (ours)	**65.93**	**0.07**	**68.43**	0.90	**53.03**	**0.52**	**49.21**	**0.11**
InfLoRA	45.10	30.73	75.62	6.70	51.14	4.45	54.33	0.81
+RASA (ours)	**54.85**	**13.90**	**77.12**	**6.41**	**53.62**	**3.86**	**55.66**	**0.62**
O-LoRA	42.75	45.77	67.01	**12.65**	50.27	5.02	53.88	2.83
+RASA (ours)	**46.15**	**29.40**	**70.82**	12.98	**52.28**	**3.32**	**54.70**	**2.59**
IncLoRA	47.50	28.81	50.42	33.47	48.20	7.70	50.19	10.54
+RASA (ours)	**56.73**	**20.33**	**70.65**	**10.32**	**51.34**	**5.07**	**55.07**	**1.85**
SeqLoRA	37.90	49.87	64.39	19.59	46.77	9.05	54.37	4.69
+RASA (ours)	**46.85**	**33.30**	**70.79**	**11.84**	**50.05**	**7.42**	**55.24**	**2.07**

Table B.8: Additional results with Qwen3-8B. We compare six LoRA baselines with RASA on Standard CL, Long-sequence CL, and SuperNI. Each value is the arithmetic mean over the task orders and three random seeds for the corresponding benchmark (Appendix B.2). Higher AP and lower FR are better. Bold indicates the better value within each baseline–RASA pair.

Method	Standard CL		Long-sequence CL		SuperNI	
	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
Gain-InfLoRA	33.68	0.29	43.05	0.08	28.17	−0.10
+RASA (ours)	**37.81**	**0.13**	**55.46**	**−0.17**	**39.03**	**−0.28**
Gain-OLoRA	58.78	0.57	63.16	0.27	52.10	0.72
+RASA (ours)	**70.94**	**−0.21**	**68.27**	**−0.15**	**54.48**	**0.34**
InfLoRA	52.33	6.67	73.92	3.70	51.32	1.99
+RASA (ours)	**59.17**	**3.42**	**77.36**	**2.96**	**54.05**	**1.53**
O-LoRA	39.93	49.23	73.36	9.61	48.03	7.21
+RASA (ours)	**42.29**	**32.61**	**74.81**	**7.48**	**51.72**	**5.02**
IncLoRA	39.13	51.27	57.26	28.19	46.96	9.93
+RASA (ours)	**57.64**	**34.85**	**60.18**	**12.74**	**50.27**	**6.84**
SeqLoRA	41.25	48.57	73.13	11.56	49.89	7.08
+RASA (ours)	**42.37**	**31.92**	**75.91**	**8.13**	**52.48**	**5.41**

settings are identical to those in the main comparison (Appendices B.3 and B.4) and are held fixed within each baseline–RASA pair. Task orders, seeds (1337, 555, and 128), and score aggregation follow Appendix B.2.

On both additional backbones, RASA improves AP for every method–benchmark combination and reduces FR in most comparisons. For Gain-InfLoRA with Llama-2-7B-Chat, FR increases on Standard CL and Long-sequence CL, while final AP improves. These additional gains over existing CL methods are consistent with the trends observed for the two backbones in the main comparison.

B.8 Component Analysis

Tables B.10 and B.11 present the component analyses for individual backbones that are averaged in Table 2. Each value is the arithmetic mean over all three Long-sequence CL task orders (Or-

Table B.9: Additional results with Llama-2-7B-Chat. We compare six LoRA baselines with RASA on Standard CL, Long-sequence CL, and SuperNI. Each value is the arithmetic mean over the task orders and three random seeds for the corresponding benchmark (Appendix B.2). Higher AP and lower FR are better. Bold indicates the better value within each baseline–RASA pair.

Method	Standard CL		Long-sequence CL		SuperNI	
	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
Gain-InfLoRA	59.88	**0.90**	65.74	**0.39**	47.98	1.24
+RASA (ours)	**63.15**	1.78	**76.56**	0.92	**48.89**	**1.22**
Gain-OLoRA	66.63	3.77	69.46	1.63	49.71	0.75
+RASA (ours)	**74.54**	**2.91**	**71.91**	**1.22**	**52.83**	**0.61**
InfLoRA	48.88	35.77	72.77	3.12	47.32	5.78
+RASA (ours)	**59.31**	**15.43**	**74.56**	**2.24**	**47.95**	**4.51**
O-LoRA	25.53	67.97	35.16	53.82	41.68	11.40
+RASA (ours)	**27.51**	**46.28**	**43.40**	**37.64**	**46.49**	**8.37**
IncLoRA	24.08	67.90	28.71	59.46	36.90	16.91
+RASA (ours)	**32.80**	**39.73**	**42.08**	**38.17**	**42.51**	**11.62**
SeqLoRA	35.28	54.77	44.67	35.38	27.87	27.48
+RASA (ours)	**36.51**	**45.36**	**55.15**	**28.72**	**31.64**	**24.39**

Table B.10: Component analysis with Meta-Llama-3.1-8B-Instruct on Long-sequence CL. Each value is the arithmetic mean over all three task orders (Orders 4–6) and three random seeds (Appendix B.2). A denotes online covariance matching, R denotes the feature-moment retention constraint and correction, and P denotes posterior-based candidate selection. Higher AP and lower FR are better. Bold indicates the best value in each column.

Variant	O-LoRA		Gain-OLoRA		SeqLoRA	
	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
Baseline	63.76	19.92	60.98	0.85	51.65	29.20
Baseline + A	65.92	14.44	68.60	1.42	68.65	13.12
Baseline + R	60.72	**11.17**	65.59	**0.24**	60.34	15.34
Baseline + A + R	67.81	13.55	70.14	1.23	69.41	12.67
RASA (+ A + R + P)	**68.39**	12.18	**71.99**	0.80	**70.64**	**11.96**

Table B.11: Component analysis with Qwen3-4B on Long-sequence CL. Each value is the arithmetic mean over all three task orders (Orders 4–6) and three random seeds (Appendix B.2). A denotes online covariance matching, R denotes the feature-moment retention constraint and correction, and P denotes posterior-based candidate selection. Higher AP and lower FR are better. Bold indicates the best value in each column.

Variant	O-LoRA		Gain-OLoRA		SeqLoRA	
	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
Baseline	67.01	12.65	63.38	0.74	64.39	19.59
Baseline + A	69.04	14.42	64.59	1.14	69.72	12.17
Baseline + R	65.23	**7.81**	60.15	**0.22**	68.33	11.77
Baseline + A + R	70.37	13.05	67.97	0.96	70.50	**11.02**
RASA (+ A + R + P)	**70.82**	12.98	**68.43**	0.90	**70.79**	11.84

ders 4–6) and seeds 1337, 555, and 128, following the main comparison (Appendix B.2). A denotes online covariance matching; R denotes the feature-moment retention constraint and correction; and P denotes posterior-based candidate selection, including the construction of opt/CE candidates and surrogate-based acceptance and selection.

Variants without A use $\alpha_t = 1$, and variants without R omit the retention constraint and correction. Variants without P use only the optimizer direction and omit the opt/CE comparison and surrogate acceptance. All variants retain the projection and regularization rules of the underlying method.

Posterior candidate selection. We analyze P on top of fixed A+R with Meta-Llama-3.1-8B-Instruct on Standard CL. Opt-only and CE-only restrict P to the optimizer and CE candidates, respectively, whereas RASA uses both. All variants retain the same covariance matching, retention constraint and correction, candidate construction and scaling, and posterior acceptance rule (Eqs. (A13)–(A14)). Opt-only therefore differs from the A+R variant, which omits P and its acceptance rule. Table B.12 reports the candidate comparison.

Table B.12: **Candidate selection within P.** Meta-Llama-3.1-8B-Instruct on Standard CL. A+R and posterior acceptance are retained in all variants. (a) AP (↑) and FR (↓); bold marks column bests. (b) RASA selection and rejection rates, as percentages of all posterior decisions. RASA AP/FR are means from Table B.6. Other entries are mean ± sample standard deviation across three seeds, each averaged over Orders 1–3.

(a) Candidate comparison

Variant	O-LoRA		Gain-OLoRA		SeqLoRA	
	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
Opt-only	40.01 ± 0.23	35.12 ± 0.30	70.74 ± 0.62	$\mathbf{-1.41 \pm 0.01}$	41.45 ± 0.11	$\mathbf{43.57 \pm 0.39}$
CE-only	40.44 ± 0.15	35.52 ± 0.23	71.63 ± 0.48	-1.00 ± 0.01	40.41 ± 0.41	45.14 ± 0.06
RASA	**41.45**	**34.76**	**72.83**	-1.33	**42.15**	44.13

(b) RASA selection and rejection rates (%)

Method	Opt selected	CE selected	Rejected
O-LoRA	75.60 ± 0.60	23.10 ± 0.17	1.30 ± 0.46
Gain-OLoRA	68.43 ± 0.32	27.93 ± 0.45	3.63 ± 0.58
SeqLoRA	72.33 ± 1.07	25.30 ± 0.50	2.37 ± 1.56

RASA achieves higher mean AP than both single-candidate variants for all three methods. CE-only performs better than Opt-only for O-LoRA and Gain-OLoRA, whereas the reverse holds for SeqLoRA. The gains therefore cannot be explained solely by replacing the optimizer candidate with CE, supporting adaptive candidate selection under a common acceptance rule. RASA attains the lowest mean FR for O-LoRA; for Gain-OLoRA and SeqLoRA, Opt-only has lower FR but also lower AP, indicating a trade-off between final performance and forgetting. Both candidates are selected during training, with optimizer candidates used more frequently and rejection uncommon. These frequencies characterize how P operates; the performance comparison establishes the benefit of using both candidates and does not separately isolate rejection.

Table B.13: Final AP for the variants in Figure 1 with Meta-Llama-3.1-8B-Instruct on Long-sequence CL (Order 4). Each value is mean ± sample standard deviation over three seeds. AP averages all 15 tasks; higher is better. A, R, and P follow Table 2; RASA uses all three. Bold marks the best mean AP within each method.

Variant	O-LoRA	Gain-OLoRA	SeqLoRA
Baseline	63.82 ± 0.05	61.04 ± 0.05	51.69 ± 0.24
Baseline + A	66.70 ± 0.07	65.91 ± 0.26	68.91 ± 0.09
Baseline + R	62.49 ± 0.16	61.42 ± 0.22	63.26 ± 0.22
Baseline + A + R	70.12 ± 0.15	67.51 ± 0.08	71.72 ± 0.26
RASA	$\mathbf{70.93 \pm 0.19}$	$\mathbf{71.96 \pm 0.14}$	$\mathbf{73.74 \pm 0.22}$

Table B.14: Results for Figure 2 with Meta-Llama-3.1-8B-Instruct and Gain-OLoRA on Long-sequence CL (Order 4). Each value is mean ± sample standard deviation over three seeds. All metrics exclude the last task; higher is better. Historical and Isotropic denote the moment conditions in Section 4.6; No retention constraint omits both the constraint and correction.

Condition	ρ	$\mathrm{Acq}_{\mathrm{old}}$	BWT	$\mathrm{AP}_{\mathrm{old}}$
Historical	0.01	68.04 ± 0.25	-0.73 ± 0.01	67.31 ± 0.25
Isotropic	0.01	63.36 ± 0.29	-1.80 ± 0.01	61.57 ± 0.30
Historical	0.025	73.36 ± 0.41	-1.20 ± 0.02	72.16 ± 0.43
Isotropic	0.025	71.12 ± 0.35	-2.01 ± 0.01	69.10 ± 0.34
Historical	0.05	75.27 ± 0.41	-1.54 ± 0.02	73.73 ± 0.42
Isotropic	0.05	73.26 ± 0.54	-2.68 ± 0.03	70.58 ± 0.54
Historical	0.1	76.30 ± 0.25	-4.65 ± 0.03	71.65 ± 0.27
Isotropic	0.1	73.96 ± 0.22	-8.11 ± 0.01	65.85 ± 0.23
No retention constraint	–	74.33 ± 0.56	-8.22 ± 0.09	66.12 ± 0.64

B.9 Results for Figures 1 and 2

Both analyses use Meta-Llama-3.1-8B-Instruct on Long-sequence CL (Order 4), with seeds 1337, 555, and 128. Tables B.13 and B.14 report the arithmetic mean and sample standard deviation across these seeds after computing each metric within a run. These results concern one backbone and one task order; the multi-order component results in Appendix B.8 use the aggregation in Appendix B.2.

Table B.13 summarizes final AP over all 15 tasks for the five variants in Figure 1. RASA attains the highest mean AP for all three methods, and +A+R exceeds both +A and +R in mean AP for each method.

Table B.14 jointly reports acquisition, BWT, and final performance for the conditions in Figure 2. All three metrics exclude the last task, and $\mathrm{AP}_{\mathrm{old}} = \mathrm{Acq}_{\mathrm{old}} + \mathrm{BWT}$ is computed within each run before aggregation. The summaries use the recorded values rounded to two decimal places. Historical moments achieves higher mean acquisition, BWT, and final performance than Isotropic moments at every matched ρ, supporting the comparison in Section 4.6.

B.10 COMPUTATIONAL COST

Table B.15: Computational cost on Standard CL (Order 1). Baseline denotes the method without RASA. Time is in ms/step and memory in GiB; all entries average over Order 1. For each metric, Δ is the recorded RASA − Baseline difference in the same units. Values are rounded for display.

	Training time (ms/step)			Mean GPU peak (GiB)			Mean CPU peak (GiB)		
Method	Baseline	RASA	Δ	Baseline	RASA	Δ	Baseline	RASA	Δ
Meta-Llama-3.1-8B-Instruct									
SeqLoRA	1645.84	1976.39	+330.55	22.416	25.430	+3.014	1.393	1.497	+0.104
IncLoRA	1615.82	1997.45	+381.63	22.435	25.449	+3.014	1.397	1.474	+0.077
O-LoRA	1658.55	2025.75	+367.20	22.435	25.449	+3.014	1.413	1.474	+0.060
InfLoRA	1651.67	1942.36	+290.70	22.162	25.208	+3.046	1.502	1.547	+0.046
Gain-OLoRA	1694.81	2186.91	+492.10	23.186	26.229	+3.043	1.514	1.533	+0.019
Gain-InfLoRA	1709.14	2047.02	+337.88	22.914	25.986	+3.072	1.525	1.556	+0.032
Qwen3-4B									
SeqLoRA	1220.43	1439.24	+218.81	22.954	24.273	+1.319	1.410	1.475	+0.065
IncLoRA	1250.06	1499.10	+249.04	22.970	24.289	+1.319	1.408	1.504	+0.097
O-LoRA	1333.48	1529.91	+196.43	22.970	24.289	+1.319	1.410	1.495	+0.086
InfLoRA	1192.86	1416.28	+223.42	22.566	24.125	+1.559	1.518	1.520	+0.003
Gain-OLoRA	1282.94	1559.00	+276.06	24.582	25.924	+1.342	1.534	1.552	+0.018
Gain-InfLoRA	1346.36	1620.73	+274.36	24.185	25.760	+1.575	1.531	1.544	+0.012

Time is CUDA-synchronized wall time per training step; memory values average training-phase peaks. CPU peak RSS is recorded using `VmHWM`, reset before each phase, with `VmSwap`=0 verified at phase end.

RASA adds 196.43–492.10 ms/step and 0.003–0.104 GiB of CPU peak memory. Additional GPU peak memory is 3.014–3.072 GiB for Meta-Llama-3.1-8B-Instruct and 1.319–1.575 GiB for Qwen3-4B. Statistics maintenance and candidate evaluation and correction introduce computation and storage costs without additional data passes.

B.11 Training with Full Data

We examine whether RASA's benefits extend beyond the 500-example-per-task setting by comparing six LoRA baselines with RASA using the entire training set of each task. Table B.16 reports results for Meta-Llama-3.1-8B-Instruct on Standard CL, Long-sequence CL, and SuperNI. We use the same three random seeds (1337, 555, and 128) and task orders as in the main experiments: Orders 1–3 for Standard CL, Orders 4–6 for Long-sequence CL, and Orders 7–8 for SuperNI (Appendix B.2). The training schedule is also unchanged: one epoch per task (Appendix B.3).

Table B.16: Full-data training with Meta-Llama-3.1-8B-Instruct. Values are means over the task orders and three random seeds for each benchmark (Appendix B.2). Higher AP and lower FR are better; bold indicates the better value within each baseline–RASA pair.

	Standard CL		Long-sequence CL		SuperNI	
Method	AP ↑	FR ↓	AP ↑	FR ↓	AP ↑	FR ↓
Gain-InfLoRA	81.95	**0.39**	80.90	2.60	54.58	1.57
+RASA (ours)	**83.29**	0.50	**82.80**	**1.79**	**55.77**	**1.36**
Gain-OLoRA	80.09	2.90	72.20	13.06	54.89	2.34
+RASA (ours)	**82.24**	**2.25**	**76.36**	**10.27**	**55.21**	**2.01**
InfLoRA	80.18	1.31	74.12	12.27	53.77	**1.32**
+RASA (ours)	**82.30**	**0.59**	**77.35**	**9.40**	**55.01**	1.41
O-LoRA	49.38	43.37	67.31	24.02	50.60	8.50
+RASA (ours)	**58.65**	**30.29**	**71.28**	**19.62**	**53.09**	**6.53**
IncLoRA	49.30	43.56	63.25	23.05	48.28	8.31
+RASA (ours)	**69.33**	**27.75**	**71.50**	**16.79**	**52.69**	**6.13**
SeqLoRA	45.94	48.26	55.15	32.12	44.57	12.98
+RASA (ours)	**49.64**	**43.80**	**71.17**	**18.27**	**50.78**	**6.96**

RASA improves mean AP in all 18 method–benchmark comparisons, including Gain-InfLoRA, which already achieves high baseline AP on Standard CL and Long-sequence CL. FR decreases in 16 comparisons. The exceptions are Gain-InfLoRA on Standard CL and InfLoRA on SuperNI, where FR increases slightly alongside higher AP. These results indicate that RASA's performance benefits extend to full-data training, with improvements in forgetting depending on the method and benchmark.