WHAT SHOULD CONTINUAL LORA REMEMBER? HISTORICAL ACTIVATION GEOMETRY FOR FIXED-CAPACITY ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual LoRA updates a shared low-rank adapter across tasks, but preserving previous behavior under fixed capacity depends on what information is retained from past data. We first show that adapter updates alone cannot identify their historical layer-response drift uniformly over past input distributions: the same update can induce zero or maximal drift on different histories. Guided by this result, we compare two uses of few-shot history, replaying labeled examples in the training loss and encoding prompt-only examples as activation geometry. We propose HARD AG-LORA, which stores a capped reservoir of historical layer activations, freezes the LoRA output factor, and projects subsequent input-factor displacements away from the leading historical activation subspace. This preserves direct adapter responses on the protected coordinates while maintaining a single fixed-rank adapter; a controlled variant further shows that storing activations when tasks are learned is more effective than recomputing them under the current model. Across three orders of four text-classification tasks on Qwen2.5-7B, HARD AG-LORA achieves 80.96% final average accuracy with 0.95 points of forgetting, compared with 77.82%/5.36 for Replay-64 and 80.72%/1.20 for growing-rank O-LoRA. On Meta-Llama-3.1-8B, it achieves 78.51%/4.58 versus 75.14%/8.20 for rank-32 O-LoRA. HARD AG-LORA keeps one rank-8 adapter and a task-count-independent activation sketch, providing an actionable fixed-capacity representation of historical information for continual adaptation.

1 INTRODUCTION

Large language models must continue adapting after deployment as domains, user preferences, and data distributions change. LoRA makes each adaptation efficient by learning low-rank updates to a frozen backbone (Hu et al., 2022), but parameter efficiency does not provide memory. Reusing one adapter can overwrite earlier behavior, whereas allocating one adapter per task makes capacity grow with the sequence. This raises a central challenge: how can one fixed-rank adapter keep learning without either forgetting or expanding indefinitely?

One line of work restores historical training signals. EWC protects important parameters (Kirkpatrick et al., 2017), LwF distills previous predictions (Li & Hoiem, 2018), and GEM constrains gradients with episodic memory (Lopez-Paz & Ranzato, 2017). For language models, LAMOL generates pseudo-examples (Sun et al., 2020), LFPT5 combines prompt tuning with pseudo-replay and consistency regularization (Qin & Joty, 2022), and Continual-T0 rehearses instruction-tuned tasks (Scialom et al., 2022). These methods can retain old behavior, but require stored or generated examples and repeated historical-loss optimization.

Another line isolates tasks or constrains update directions. Progressive Prompts adds task-specific prompts (Razdaibiedina et al., 2023); ConPET and SAPT learn and select parameter-efficient modules (Song et al., 2023; Zhao et al., 2024). O-LoRA assigns orthogonal low-rank subspaces (Wang et al., 2023), InfLoRA constructs interference-reducing spaces (Liang & Li, 2024), N-LoRA mitigates parameter collision (Yang et al., 2025), and CLoRA regularizes LoRA null-space directions (Lu et al., 2025). OWM and GPM similarly protect historical input or representation subspaces (Zeng et al.,

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.

2019; Saha et al., 2021). These approaches reduce interference, but history is still revisited through replay, stored in task-indexed modules, or encoded through a chosen subspace.

This raises a more basic question: what historical information makes an update safe? Even compatible tasks can interfere, and the effect of an adapter displacement D depends on the historical layer-input distribution P, not on the update alone. We formalize this dependence at a LoRA-equipped layer and identify the historical second moment C_P as a sufficient summary for the local quadratic drift we study. This layer-local result does not equate to end-to-end forgetting, but shows that distribution information—not necessarily the full historical dataset—must be retained.

Guided by this result, we compare labeled replay with prompt-only activation geometry and introduce the fixed-capacity AG-LoRA family. HARD AG-LoRA protects a capped subspace of stored historical activations, whereas REFRESH AG-LoRA recomputes the geometry from historical prompts under the current model.

Our contributions are threefold:

- We establish an information requirement for fixed-capacity continual LoRA: the same update can induce zero or maximal historical layer-response drift under different unit-norm histories, so updates alone cannot identify their effect uniformly across past input distributions.
- We compare replay with activation geometry and introduce HARD AG-LoRA, which retains a capped activation reservoir and constrains one fixed-rank adapter. The controlled REFRESH AG-LoRA variant isolates the historical reference used to define the protected subspace, showing that activations stored when a task is learned are more effective than activations recomputed under the current model.
- We evaluate Sequential LoRA, Replay-64, CLoRA, Freeze-B, O-LoRA, and InfLoRA, with refreshed geometry and random projection as controls. Across four Qwen tasks and three orders, rank-8 HARD AG-LoRA achieves 80.96% final average accuracy with 0.95 points of forgetting, versus $77.82\%/5.36$ for Replay-64 and $80.72\%/1.20$ for rank-32 O-LoRA. On Llama, it achieves $78.51\%/4.58$, versus $75.14\%/8.20$ for rank-32 O-LoRA, while retaining one rank-8 adapter and a task-count-independent activation sketch.

2 IS HISTORICAL INFORMATION NECESSARY?

We consider tasks $\{\mathcal{D}_t\}_{t=1}^{T}$ learned sequentially. At stage t, the learner receives the current dataset $\mathcal{D}_t$ and carries forward the previous model parameters together with a retained historical state H_{t-1}; earlier datasets are unavailable except through information encoded in these two states. We focus on a single fixed-rank LoRA adapter and a historical-state budget that does not grow with the number of tasks, with the goal of learning $\mathcal{D}_t$ while preserving performance on previously learned tasks.

Continual learning can fail for two distinct reasons: the tasks may be incompatible, or the learner may not know which changes damage previous behavior. We isolate the second case. Figure 1 summarizes the argument: even jointly solvable tasks can be forgotten by current-task-only training, and the resulting parameter update does not reveal its effect without information about historical inputs.

2.1 COMPATIBLE TASKS CAN STILL BE FORGOTTEN

Consider the bias-free classifier $f_w(x) = \mathrm{sign}(w^\top x)$ with $w = (a, b)^\top$. Task 1 contains $\pm e_1$ with labels ± 1, and Task 2 contains $\pm(-1, 1)^\top$ with labels ± 1. Suppose each stage selects the minimum-norm classifier satisfying only the current task's hard-margin constraint. The tasks impose $a \geq 1$ and $-a + b \geq 1$, yielding

$$w^{(1)} = (1, 0)^\top, \qquad w^{(2)} = (-1/2, 1/2)^\top. \tag{1}$$

The second solution reverses both Task-1 predictions, although $w^{\mathrm{joint}} = (1, 2)^\top$ satisfies both tasks. Forgetting therefore arises here from discarding the old constraint, not from insufficient model capacity. Figure 1(a) illustrates this gap; Appendix A gives the derivation.

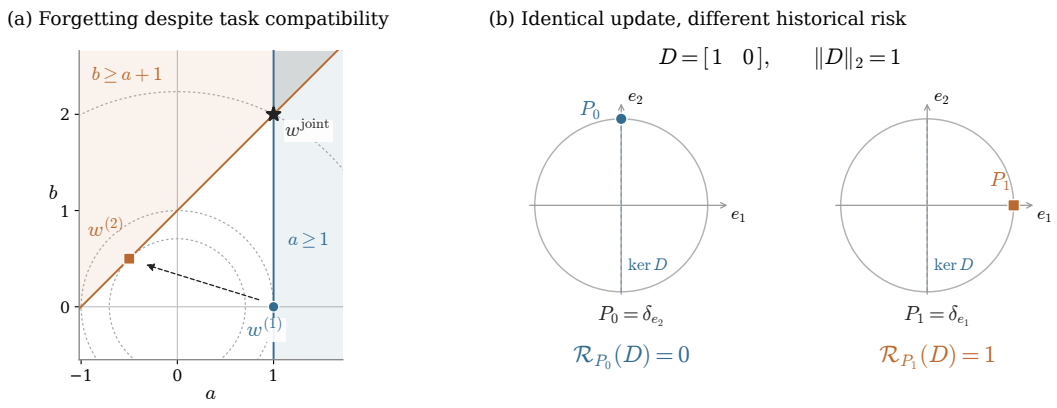


Figure 1: **Compatible tasks may still be forgotten, and the update alone cannot reveal the damage.** (a) The old and new constraints admit the joint solution w^{joint}, but optimizing each task separately selects $w^{(1)}$ and $w^{(2)}$, which give opposite predictions on Task 1. Dotted circles are norm contours; the arrow connects the two solutions rather than showing an optimization trajectory. (b) For the same update $D = [1, 0]$, historical inputs concentrated on e_2 incur zero local drift, whereas inputs concentrated on e_1 incur unit drift. Historical input information is therefore necessary to distinguish the two cases.

2.2 Updates alone do not identify historical drift

We now ask what information is missing. At a LoRA-equipped layer, let $U = sBA$ and let $D = U - U^{\text{ref}}$ be the effective adapter displacement from a task-boundary reference. For historical layer inputs $x \sim P$ evaluated at this reference, the direct response drift is

$$\begin{aligned} \mathcal{R}_P(D) &= \mathbb{E}_{x\sim P}\|Dx\|_2^2 = \operatorname{Tr}(DC_PD^\top), \\ C_P &= \mathbb{E}_{x\sim P}[xx^\top]. \end{aligned} \tag{2}$$

This quantity holds the layer input fixed. It measures local adapter-response drift rather than end-to-end forgetting, but it makes the information requirement explicit: the same displacement can be harmless or destructive depending on P.

Proposition 1 (Updates alone do not determine historical response drift). For any nonzero D with $\operatorname{rank}(D) < d_{\text{in}}$, there exist unit-norm input distributions P_0 and P_1 such that

$$\mathcal{R}_{P_0}(D) = 0, \qquad \mathcal{R}_{P_1}(D) = \|D\|_2^2. \tag{3}$$

Consequently, every distribution-blind scalar $g(D)$ satisfies

$$\sup_{P:\,\|x\|_2=1 \text{ a.s.}} |g(D) - \mathcal{R}_P(D)| \geq \frac{1}{2}\|D\|_2^2. \tag{4}$$

The two distributions place their mass in the null space and along the leading right singular vector of D, respectively; Figure 1(b) shows the two-dimensional case. For the difference of two rank-r adapters, $\operatorname{rank}(D) \leq 2r$, so the condition holds whenever $2r < d_{\text{in}}$. The proof and the tighter fixed-B case are in Appendix A.

This proposition is a worst-case identification result, not a claim that raw replay is necessary or that every learner without external storage must forget. It says that an update, viewed in isolation, is insufficient: some distribution-dependent historical information must be available to determine its effect.

2.3 From historical data to activation geometry

Equation 2 also identifies a compact sufficient statistic for the local quantity: the historical second moment C_P. It can be estimated from a few historical inputs and compressed into a low-rank

activation summary; the resulting drift depends on estimation and compression error rather than on access to the full historical dataset. Appendix A.3 gives the corresponding sketch bounds.

The information requirement therefore does not prescribe a unique memory mechanism. Few-shot history may be retained as labeled examples and replayed in the loss, or processed once and stored as activation geometry that constrains later updates. Section 3 compares these two uses under fixed capacity and then asks whether the activation geometry should remain fixed or be refreshed as the model changes.

3 Using Few-Shot History as Activation Geometry

Section 2 shows that preserving previous tasks requires information about their input distributions. We now ask how a fixed amount of few-shot history should enter continual LoRA. The same historical access supports two qualitatively different routes: Replay-64 keeps labeled examples and repeatedly optimizes their supervised loss, whereas activation-geometry protection processes prompt-only examples at task boundaries and constrains future adapter updates. We first compare these two uses of history, then determine whether the resulting activation geometry should be stored when it is observed or refreshed under the latest model.

3.1 Fixed-capacity setting

We keep a frozen backbone and one rank-r LoRA adapter throughout the task sequence. At layer l, the effective adapter is

$$U_l = sB_lA_l, \tag{5}$$

where s is the LoRA scale. Let P_l denote the historical distribution of inputs to this layer and let D_l be a change in its effective adapter. Holding the layer input fixed, the induced response drift is

$$\mathcal{R}_{P,l}(D_l) = \mathbb{E}_{x\sim P_l}\|D_lx\|_2^2 = \mathrm{Tr}(D_lC_{P,l}D_l^\top), C_{P,l} = \mathbb{E}_{x\sim P_l}[xx^\top]. \tag{6}$$

The second moment $C_{P,l}$ therefore identifies which input directions make an adapter displacement costly. Few-shot history can either approximate the old supervised objective directly or estimate this layerwise geometry and prevent displacement along its dominant directions. All methods considered below retain task-count-independent historical state and update the same single adapter; Replay-64's example buffer and the per-layer activation reservoirs are fixed-capacity(Appendix B).

3.2 How should few-shot history be used?

Replay-64. Replay-64 stores a total of 64 labeled historical examples in a buffer $\mathcal{M}$. Every current-task update is paired with an old-data mini-batch and optimizes

$$\mathcal{L}_{\mathrm{replay}}(\theta) = \mathcal{L}_t(\theta) + \beta\,\mathbb{E}_{z\sim\mathcal{M}}\ell(f_\theta; z). \tag{7}$$

Few-shot history is thus used in example space: labels are retained, old examples are repeatedly forwarded and backpropagated, and retention is encouraged through a stochastic estimate of the old supervised loss.

Hard AG-LoRA. Hard AG-LoRA instead converts temporary few-shot access into a fixed activation summary. After task $t-1$, it evaluates a prompt-only probe set, collects valid-token inputs to every targeted LoRA layer, and merges sampled rows into a capped reservoir $X_{l,t-1} \in \mathbb{R}^{M_l\times d_l}$. The labels and raw probes can then be discarded. Before task t, it forms

$$\widehat{C}_{l,t} = \frac{1}{M_l}X_{l,t-1}^\top X_{l,t-1}, \widehat{C}_{l,t}Q_{l,t} = Q_{l,t}\Lambda_{l,t}, \tag{8}$$

where $Q_{l,t}$ contains the leading eigenvectors selected by the activation-energy threshold and rank cap. The evaluated configuration keeps at most 64 FP16 rows per targeted layer and retains the smallest leading subspace reaching 95% energy, capped at rank 64.

After Task 1, Hard AG-LoRA freezes B_l and updates only A_l. At the start of task t, it snapshots $A_l^{\mathrm{ref}} = A_{l,t,0}$. For every optimizer proposal $\widetilde{A}_{l,t,j+1}$, it accepts

$$A_{l,t,j+1} = A_l^{\mathrm{ref}} + (\widetilde{A}_{l,t,j+1} - A_l^{\mathrm{ref}})\cdot(I - Q_{l,t}Q_{l,t}^\top). \tag{9}$$

Table 1: **Stored activation geometry retains previous tasks better than supervised replay or refreshed geometry on DBP–AMZ–YAH–AGN.** All rows use Qwen2.5-7B and one fixed rank-8 adapter. Relative to Replay-64, HARD AG-LORA raises final average accuracy from 77.65% to 80.84% and reduces forgetting from 5.57 to 1.53 points. REFRESH AG-LORA reaches 77.71% and 4.82 points, showing that activation geometry is most effective when the historical activations themselves are retained. BWT and forgetting are reported in percentage points.

Method	Final AA ↑	Forgetting ↓	BWT ↑
Replay-64	77.65	5.57	−5.57
HARD AG-LORA	**80.84**	**1.53**	−1.50
REFRESH AG-LORA	77.71	4.82	−4.82

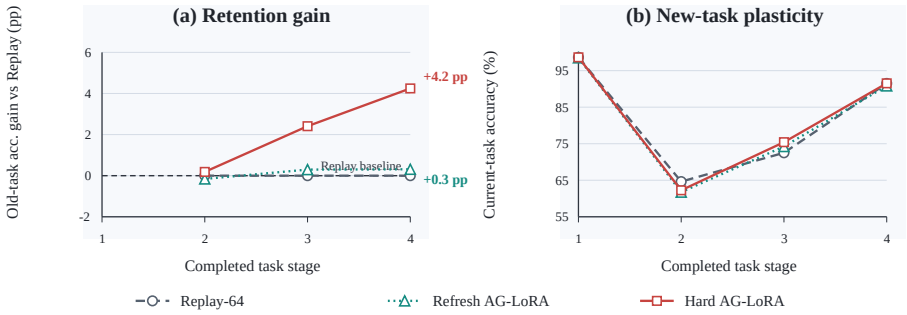


Figure 2: **Stored activation geometry improves retention without a systematic loss of new-task plasticity.** Left: mean old-task accuracy relative to Replay-64 after each completed stage. The advantage of HARD AG-LORA grows from +0.2 points after Amazon to +2.4 after Yahoo and +4.2 after AG News, whereas REFRESH AG-LORA ends at only +0.3. Right: accuracy on the newly learned task at each stage; all three methods follow a similar trajectory, so the retention gain is not explained by refusing to learn the current task.

Because B_l is fixed, the resulting effective displacement $D_{l,t,j} = U_{l,t,j} - U_{l,t,0}$ satisfies

$$D_{l,t,j} Q_{l,t} = 0. \tag{10}$$

The adapter therefore has zero direct response drift on the retained activation directions. Projection is applied to the finite displacement after every optimizer step, rather than only to the gradient, so optimizer state cannot accumulate a protected component. The full procedure is: train Task 1 with ordinary LoRA; collect its prompt-only activations; before every later task construct $Q_{l,t}$, freeze B_l, and train A_l with Equation 9; after the task, merge its activation rows into the fixed reservoirs and continue. The guarantee is layer-local: upstream activation changes and unprotected directions can still affect end-to-end predictions.

Table 1 gives the endpoint comparison. Under these fixed-capacity protocols, replacing repeated replay loss with stored activation geometry improves final average accuracy by 3.19 points, reduces forgetting by 4.04 points, and improves BWT by 4.07 points. The comparison does not merely favor a different regularization coefficient: Replay-64 and HARD AG-LORA expose the historical information to optimization in different forms. Replay estimates old empirical risk from 64 labeled examples, while HARD AG-LORA uses prompt-only examples once to estimate where later LoRA displacement would change historical layer responses.

The stagewise evidence explains the endpoint gain. Figure 2 shows that HARD AG-LORA separates from Replay-64 only after multiple tasks have accumulated, while immediate new-task accuracy remains comparable. Figure 3 localizes this difference: DBpedia remains near saturation for all methods, and the final AG News accuracy is also matched; the gain comes from preserving Amazon and Yahoo after subsequent updates. Together with Table 1, these results support using few-shot history to define a protected activation geometry rather than repeatedly inserting the same small buffer into the supervised loss.

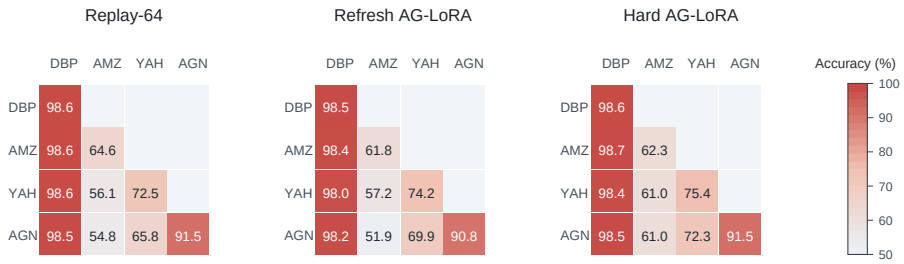


Figure 3: **The final retention gain of HARD AG-LORA is concentrated on the vulnerable Amazon and Yahoo tasks.** Rows are completed training stages and columns are evaluation tasks; blank cells denote tasks not yet learned. At the final stage, HARD AG-LORA exceeds Replay-64 by 6.2 points on Amazon and 6.5 on Yahoo, while DBpedia and AG News differ by less than 0.1 point. The shared 50–100% color scale makes the three methods directly comparable.

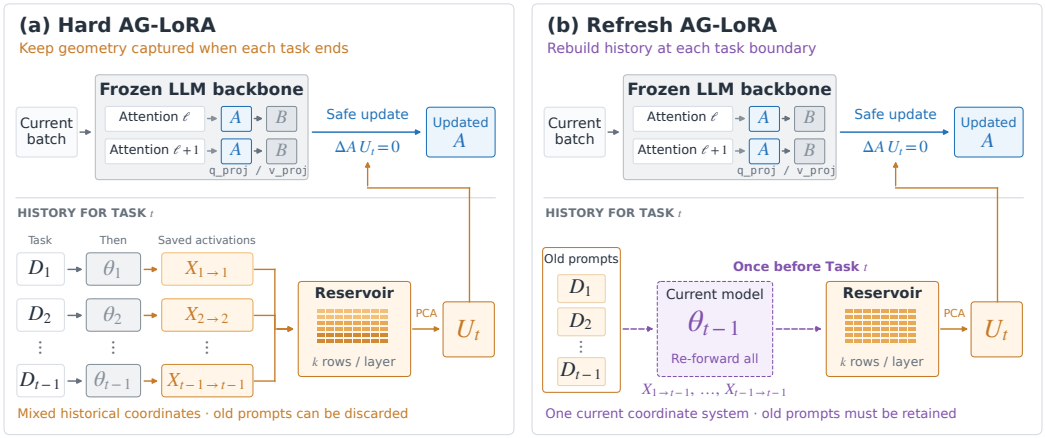


Figure 4: **Stored and refreshed geometry differ only in the historical coordinates they protect.** (a) HARD AG-LORA retains capped activation rows recorded when previous tasks end, so the basis $Q_{l,t}$ summarizes the historical geometry actually observed at those boundaries and the original prompts can be discarded. (b) REFRESH AG-LORA retains old prompts and re-forwards them under M_{t-1} before Task t, so its basis describes the same prompts after the model has already changed. Both variants then apply the same task-start anchored projection; the comparison isolates the geometry reference rather than the optimizer or constraint.

3.3 WHEN SHOULD ACTIVATION GEOMETRY BE REPRESENTED?

Once few-shot history is represented by activation geometry, the remaining question is which model should produce those activations. HARD AG-LORA stores activation rows when each task ends and keeps these rows fixed as later models evolve. REFRESH AG-LORA instead stores the historical prompts and, before task t, re-forwards them once through the boundary model M_{t-1} to rebuild the reservoirs in current-model coordinates. Both variants use the same leading-subspace extraction, task-start anchor, frozen B_l, and projection in Equation 9; only the source of $Q_{l,t}$ changes.

Figure 4 makes this controlled difference explicit. Refreshing may appear attractive because it expresses old prompts in the latest model coordinates, but it also lets the historical reference move with the model: directions that have already drifted can be weakened or replaced when the prompts are re-encoded. Stored activation rows instead provide a fixed record of the geometry present when each task was learned.

The three empirical views consistently favor the fixed reference. Table 1 shows that HARD AG-LORA improves final average accuracy over REFRESH AG-LORA by 3.13 points and reduces forgetting by 3.29 points. In Figure 2, the refreshed variant returns to the Replay-64 retention level by the

final stage, whereas the stored variant reaches a $+4.2$-point old-task gain. Figure 3 shows where this gap occurs: at the final stage, HARD AG-LORA exceeds REFRESH AG-LORA by 9.1 points on Amazon and 2.4 on Yahoo, while their current-task AG News accuracies remain within 0.7 points. Refreshing therefore does not buy additional plasticity sufficient to explain its weaker retention; it mainly replaces a stable historical summary with one that changes as the model changes.

These comparisons determine the final method. HARD AG-LORA uses temporary prompt-only access at each task boundary, stores a fixed-capacity reservoir of the resulting activation rows, extracts its leading geometry before the next task, and constrains every later LoRA displacement with the anchored projection in Equation 9. Raw historical prompts and labels are unnecessary after the rows are collected. Section 4 evaluates this choice across task orders and backbones; Appendix B gives the complete algorithm and implementation details.

4 EXPERIMENTS

We evaluate whether historical information helps, how replay compares with subspace protection, and which components account for the latter's gains. Main results precede a brief ablation summary; detailed ablations and scalar drift diagnostics are reported in the appendix.

4.1 EXPERIMENTAL SETUP

Tasks and training. We use Qwen2.5-7B-Instruct (Yang et al., 2024) and Meta-Llama-3.1-8B-Instruct (Grattafiori et al., 2024) on the DBpedia, Amazon Reviews, Yahoo Answers, and AG News text-classification datasets (Zhang et al., 2015) under three task orders. The prepared training sets contain 14,000, 5,000, 10,000, and 4,000 examples, respectively. The core comparison uses all of them for one BF16 epoch per task and uses AdamW at 10^{-3} with zero weight decay, global gradient-norm clipping at 1, and effective new-task batch 64. A fresh optimizer is created for every task. Rank-8 LoRA targets `q_proj` and `v_proj`, except in the additional InfLoRA-Qwen adaptation, which targets `k_proj` and `v_proj`. Exact prompts, data roles, and randomization are in Appendix C and Table 5.

Baselines and capacity. The core comparison includes four methods. Sequential LoRA maintains one rank-8 adapter without explicit protection. O-LoRA adds a rank-8 block per task, reaching rank capacity 32. Replay-64 retains a total of 64 labeled examples and adds 64 old examples per new-task update. HARD AG-LORA keeps one rank-8 adapter and at most 64 reservoir rows per target layer, with a 95% activation-energy threshold and risk rank at most 64. O-LoRA uses overlap coefficient 0.5, matching the retained official reproduction scripts.

Evaluation. Each task is evaluated on all 7,600 prepared test examples after every learned stage. Greedy decoding generates at most 12 tokens. Accuracy requires exact case-folded equality after taking the first generated line, removing an optional `Answer:`/`Label:` prefix, and stripping surrounding spaces and punctuation. Final average accuracy (AA) averages the four final task accuracies. Forgetting compares each historical task's best post-learning accuracy with its final accuracy; BWT compares immediate post-learning and final accuracies. We report order-level variation and persistent method-state size, excluding the shared frozen backbone; measured state sizes are available for Qwen.

4.2 MAIN RESULTS

For Qwen, Table 2 organizes the three-order results by method role and capacity regime: fixed-capacity baselines, growing-adapter baselines, an ablation/control, and our method. Sequential LoRA forgets heavily, while Replay-64 substantially improves retention with a single rank-8 adapter. Among the growing-adapter methods, O-LoRA largely prevents forgetting at effective rank 32, and the Qwen adaptation of InfLoRA records 81.80% AA and 0.54 points of forgetting while adding one rank-8 block per task. InfLoRA also targets `k_proj`/`v_proj`, rather than `q_proj`/`v_proj`. The fixed-rank Freeze-B control reaches 76.43%/6.62, and the evaluated CLoRA configuration reaches 59.93%/28.23. HARD AG-LORA retains one rank-8 adapter and records 80.96% AA with 0.95 points of forgetting. Among the reported fixed-capacity Qwen methods, HARD AG-LORA has the

Table 2: Three-order Qwen2.5-7B summary, grouped by method role and capacity regime. Values are mean ± sample standard deviation across task orders. AA is in percent and forgetting/BWT are percentage points. Bold marks the best observed mean in the table. InfLoRA targets `k_proj`/`v_proj`, unlike the other rows. Growing-adapter methods add a rank-8 block per task; fixed-capacity methods retain a single rank-8 adapter.

Method	Final AA ↑	Forgetting ↓	BWT ↑	Rank
Fixed-capacity baselines				
Sequential	36.73±11.10	52.62±10.29	-52.62 ± 10.29	8
Replay-64	77.82±0.97	5.36±2.00	-5.36 ± 2.00	8
CLoRA	59.93±12.26	28.23±14.88	-28.23 ± 14.88	8
Growing-adapter baselines				
O-LoRA	80.72±0.65	1.20±0.79	-0.93 ± 1.00	32
InfLoRA	81.80±0.34	0.54±0.33	-0.48 ± 0.41	32
Ablation				
Freeze-B	76.43±2.58	6.62±2.62	-6.60 ± 2.64	8
Ours				
REFRESH AG-LoRA	79.60±1.93	2.87±1.86	-2.87 ± 1.86	8
HARD AG-LoRA	**80.96±0.12**	**0.95±0.80**	-0.36 ± 1.77	8

Table 3: Persistent method-state overhead after four Qwen tasks. Our method retains a fixed rank-8 adapter and requires 35.82 MB of persistent state, compared with 40.39 MB for O-LoRA and 50.98 MB for InfLoRA, whose effective ranks grow to 32. "Minimum" denotes the minimal self-contained state required to continue learning, while "cache" denotes optional recomputable state. The shared frozen backbone is excluded.

Method	Rank	Adapter MB	Required aux. MB	Cache MB	Minimum MB	With cache MB
Sequential	8	10.11	0.00	0.00	10.11	10.11
Freeze-B	8	10.11	0.00	0.00	10.11	10.11
Replay-64	8	10.11	0.03	0.00	10.14	10.14
Refresh AG-LoRA	8	10.11	0.11	25.72	10.22	35.93
CLoRA	8	10.11	0.00	0.00	10.11	10.11
HARD AG-LoRA	8	10.11	25.71	0.00	35.82	35.82
O-LoRA	32	40.39	0.00	0.00	40.39	40.39
InfLoRA	32	29.38	21.60	0.00	50.98	50.98

highest three-order AA and lowest forgetting, while remaining close to the stronger growing-adapter baselines. See Appendix E.1 for per-order results and capacity details;

Replay-64 improves markedly over Sequential LoRA but remains below HARD AG-LoRA on AA and retention in every order. It retains labeled text and repeatedly computes historical losses; HARD AG-LoRA retains activation rows instead. Appendix E.1 reports the order-level results and the interpretation of Replay's forgetting/BWT relation.

Fixed-capacity state versus growing adapters. Table 3 reports values measured from the formal checkpoints. The rank-8 Qwen adapter has 2.52M parameters and occupies 10.11 MB in the saved safetensors format. O-LoRA's four rank-8 blocks occupy 40.39 MB. HARD AG-LoRA keeps the 10.11 MB rank-8 adapter and a capped 25.71 MB FP16 activation reservoir, for 35.82 MB total persistent method state after four tasks. Thus, at the evaluated four-task point, HARD AG-LoRA attains essentially the same Qwen accuracy and retention as O-LoRA while using less persistent state (35.82 MB versus 40.39 MB) and one quarter of the adapter rank. More importantly, O-LoRA adds a new rank-8 block at each task and therefore grows linearly with task count, whereas HARD AG-LoRA keeps one rank-8 adapter and a capped activation sketch. Our method maintains a fixed rank-8 adapter while keeping the persistent state bounded at 35.82 MB after four tasks, which is lower than O-LoRA and InfLoRA, whose effective ranks grow to 32. This suggests that strong continual learning performance does not necessarily require the adapter capacity to grow with the number of tasks.

Backbone replication on Llama 3.1. We additionally replicate the core comparison on Llama 3.1-8B-Instruct. In the matched fixed-10^{-3} comparison, HARD AG-LORA records 78.51% final AA, 4.58 points of forgetting, and -3.67 points BWT at rank 8, while O-LoRA reaches 75.14% AA and 8.20 points of forgetting at rank 32. The scheduled-LR Sequential reference reaches 76.73% AA and 6.90 points of forgetting, but its optimizer schedule differs from the matched pair. These results provide a second-backbone replication of the central fixed-capacity comparison; full per-order Llama results are reported in Appendix Table 14.

4.3 MORE ABLATION STUDIES

We isolate two possible explanations for subspace protection: freezing the LoRA output factor and restricting the update space, versus choosing the restricted directions from historical activations. On Qwen Order 1, we compare Sequential LoRA, Freeze-B, a matched-per-layer-rank random projector, and HARD AG-LORA under the same rank-8 protocol.

Freeze-B is already strong, but activation-informed projection further raises final AA from 79.37% to 80.84% and reduces forgetting from 3.62 to 1.53 points. Random projection is worse at 77.11% AA and 6.33 points of forgetting, showing that the gain is not explained only by freezing B or reducing the available update dimension. The full aggregate table, task-level results, matched-rank checks, activation-energy coverage, and constraint diagnostics are reported in Appendix E.3; three-order Freeze-B results appear in Appendix E.1.

4.4 SUPPLEMENTARY ANALYSIS: DIAGNOSING HISTORICAL DRIFT

To separate prediction from intervention, we evaluate checkpoint-level diagnostics along ordinary Sequential LoRA trajectories over four tasks and three task orders. For each old task, 256 calibration examples estimate internal statistics and 1,000 disjoint validation examples define fixed-anchor forgetting. Metric mappings use grouped out-of-fold evaluation, with run-cluster bootstrap uncertainty (Appendix C).

Activation geometry is the strongest tested internal diagnostic for accuracy forgetting. For individual old tasks, it achieves the lowest OOF MAE under all four held-out protocols (0.0829–0.0932); after averaging over old tasks, it again has the lowest internal-metric MAE under all applicable protocols (0.0638–0.0889). Conditional gains are split-dependent, with the clearest improvement for individual-task NLL under old-task holdout. These results support activation geometry as an informative signal for diagnosing task-specific forgetting. Score definitions and full results are in Appendices B and D.

5 CONCLUSION

The central question is what to retain from past data and how to use it during continual learning. The theory establishes a precise local requirement: identifying historical response drift across unrestricted input distributions needs distribution-dependent information. Replay supplies that information through historical losses; ACTIVATION GEOMETRY represents it as activation geometry that constrains updates through the AG-LORA family. Among its variants, HARD AG-LORA uses stored coordinates and a task-start anchor, retains one rank-8 adapter, and achieves competitive retention across Qwen and Llama task orders, and benefits from activation-informed directions beyond Freeze-B alone in the Qwen mechanism study. In the four-task diagnostic analysis, activation geometry is also the strongest internal predictor of accuracy forgetting for both aggregate and individual old-task targets. Together, the analysis and experiments identify activation geometry as a fixed-capacity, actionable representation of historical information for continual LoRA.

REPRODUCIBILITY STATEMENT

The appendix provides the theoretical proofs, complete algorithms and implementation details, experimental protocols, and full diagnostic, benchmark, ablation, replication, and trajectory results. The anonymous supplementary artifacts contain the code and configuration snapshot, exact run identifiers, unrounded results, retained trajectories, dataset hashes, and file checksums. The full implementation and experiment scripts will also be made available through the anonymous GitHub repository https://anonymous.4open.science/r/codesubmission-0F04/ during the review period.

AI USE STATEMENT

Generative AI tools were used to assist with code review, analysis scripts, manuscript editing, literature discovery, citation verification, and internal consistency checks. All theoretical, experimental, and scientific claims were independently verified by the authors against the implementation, recorded artifacts, and original sources. The authors reviewed all AI-assisted content and take full responsibility for the final manuscript.

REFERENCES

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115. Curran Associates, Inc., 2023. doi: 10.52202/075280-0441. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf.

Mehrdad Farajtabar, Navid Azizan, Alexander Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics*, pp. 3762–3773, 2020.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Jiangpeng He, Zhihao Duan, and Fengqing Zhu. Cl-lora: Continual low-rank adaptation for rehearsal-free class-incremental learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30534–30544. IEEE, 2025.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, pp. 4582–4597, 2021.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

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

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

Yuheng Lu, Bingshuo Qian, Caixia Yuan, Huixing Jiang, and Xiaojie Wang. Controlled low-rank adaptation with subspace regularization for continued training on large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19165–19181, 2025.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul A Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7452–7467, 2021.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773. IEEE, 2018.

Chengwei Qin and Shafiq Joty. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=HCRVf71PMF.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UJTgQBc91_.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=3AOj0RCNC2.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 6107–6122, 2022.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-loRA. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=TZdEgwZ6f3.

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. URL https://openreview.net/forum?id=Skgxcn4YDS.

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, 2023. doi: 10.18653/v1/2023.findings-emnlp.715.

Xiwen Wei, Guihong Li, and Radu Marculescu. Online-lora: Task-free online continual learning via low rank adaptation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6634–6645. IEEE, 2025.

Martin Wistuba, Prabhu Teja Sivaprasad, Lukas Balles, and Giovanni Zappella. Continual learning with low rank adaptation. In *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023. URL https://openreview.net/forum?id=DbsCOyoPRl.

ShiLin Xiao, Tianxiang Xu, Canran Xiao, Weihao Luo, Liwei Hou, and Chuangxin Zhao. Meta-ucf: Unified task-conditioned lora generation for continual learning in large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Lina Yang, Yusheng Liao, Yanfeng Wang, and Yu Wang. Slora: Balancing plasticity and forgetting in large language models for continual learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5437–5454, 2026.

Shuo Yang, Kun-Peng Ning, Yu-Yang Liu, Jia-Yu Yao, Yong-Hong Tian, Yi-Bing Song, and Li Yuan. Is parameter collision hindering continual learning in llms? In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 4243–4259, 2025.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, 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, 2024.

Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang. Continual prompt tuning for dialog state tracking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1124–1137, 2022.

A Theory and Historical-Information Guarantees

We separate two questions: can current-task optimization forget an earlier task even when the tasks are jointly learnable, and what information is needed to determine the historical effect of an update? A constructive example answers the first; a distributional ambiguity result answers the second. Together, they motivate retaining historical functional information.

A.1 Compatible tasks can still be forgotten

Consider the bias-free classifier

$$f_w(x) = \operatorname{sign}(w^\top x), \qquad w \in \mathbb{R}^2,$$

with hard-margin learning rule

$$\mathcal{A}(\mathcal{D}) = \arg\min_w \frac{1}{2}\|w\|_2^2 \quad \text{s.t.} \quad yw^\top x \geq 1 \quad \forall (x,y) \in \mathcal{D}. \tag{11}$$

At each stage only the current dataset is optimized; previous parameters may initialize the solver, but previous losses and constraints are not retained.

Let

$$e_1 = (1,0)^\top, \qquad v = (-1,1)^\top,$$

and

$$\begin{aligned} \mathcal{D}_1 &= \{(e_1,+1),(-e_1,-1)\}, \\ \mathcal{D}_2 &= \{(v,+1),(-v,-1)\}. \end{aligned} \tag{12}$$

Writing $w = (a,b)^\top$, the first task imposes $a \geq 1$, while the second imposes $-a + b \geq 1$. Their unique minimum-norm solutions are

$$w^{(1)} = (1,0)^\top, \qquad w^{(2)} = \left(-\frac{1}{2}, \frac{1}{2}\right)^\top. \tag{13}$$

Since

$$a^2 + b^2 \geq \frac{(-a+b)^2}{2} \geq \frac{1}{2},$$

the second optimum is attained at $w^{(2)}$.

Both examples from $\mathcal{D}_1$ have margin 1 under $w^{(1)}$ and margin $-1/2$ under $w^{(2)}$, so

$$\mathrm{Acc}_{\mathcal{D}_1}(w^{(1)}) = 100\%, \qquad \mathrm{Acc}_{\mathcal{D}_1}(w^{(2)}) = 0\%. \tag{14}$$

Yet

$$w^{\mathrm{joint}} = (1, 2)^\top$$

satisfies both tasks with margin 1 and is the minimum-norm solution under

$$a \geq 1, \qquad b \geq a + 1.$$

Thus this current-task-only rule forgets despite compatible labels and a joint solution. Previous parameters can contain historical information, but initialization alone does not preserve the associated behavior in this construction. Figure 1(a) visualizes this failure: although the two feasible regions intersect, optimizing the second task alone selects a minimum-norm solution that violates the first task's constraints.

A.2 Updates alone do not identify historical drift

For a frozen backbone weight

$$W_l \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}},$$

LoRA uses

$$\widetilde{W}_l = W_l + U_l, \qquad U_l = sB_lA_l, \tag{15}$$

where

$$A_l \in \mathbb{R}^{r \times d_{\mathrm{in}}}, \qquad B_l \in \mathbb{R}^{d_{\mathrm{out}} \times r}.$$

Let U_l^{ref} denote the adapter at a task boundary and define

$$D_l = U_l - U_l^{\mathrm{ref}}.$$

For historical layer inputs evaluated at the reference checkpoint, the direct adapter-response change is $D_l x_l$. Suppressing the layer index,

$$\begin{aligned} \mathcal{R}_P(D) &= \mathbb{E}_{x \sim P} \|Dx\|_2^2 \\ &= \mathrm{Tr}(DC_PD^\top), \qquad C_P = \mathbb{E}_{x \sim P}[xx^\top]. \end{aligned} \tag{16}$$

Here C_P is an uncentered second moment.

This is a layer-local functional drift, not behavioral forgetting: it holds the input activation fixed and does not account for upstream changes, nonlinear propagation, or decision-boundary crossings.

Proposition 1 (restatement and proof). Fix $D \neq 0$ with

$$\mathrm{rank}(D) < d_{\mathrm{in}}.$$

There exist unit-norm input distributions P_0, P_1 such that

$$\mathcal{R}_{P_0}(D) = 0, \qquad \mathcal{R}_{P_1}(D) = \|D\|_2^2. \tag{17}$$

Hence every scalar predictor $g(D)$ without access to the historical distribution satisfies

$$\sup_{P:\, \|x\|_2 = 1 \text{ a.s.}} |g(D) - \mathcal{R}_P(D)| \geq \frac{1}{2}\|D\|_2^2. \tag{18}$$

Proof. Choose a unit vector $v_0 \in \ker(D)$ and a leading right singular vector v_1 of D. Let P_0 and P_1 place all mass on v_0 and v_1, respectively. Their risks are 0 and $\|D\|_2^2$. For any $z \in \mathbb{R}$,

$$\max\{|z|, |z - \|D\|_2^2|\} \geq \frac{1}{2}\|D\|_2^2.$$

Taking $z = g(D)$ proves the claim. □

Figure 1(b) illustrates the ambiguity with $D = [1, 0]$: historical inputs concentrated in its null space incur zero drift, whereas inputs aligned with its leading right singular vector incur unit drift, despite an identical update.

For two rank-r adapters,

$$\operatorname{rank}(D_l) \le 2r,$$

and if B_l is fixed, the bound improves to r. Proposition 1 is therefore applicable whenever the corresponding rank bound is below d_{in}.

The proposition is a worst-case statement over unrestricted input distributions. Parameter-only predictors may work under additional distributional restrictions, and the required historical information may reside in the model parameters themselves.

A.3 Second-moment estimation and sketch bounds

For the quadratic risk in Equation 16, the historical second moment C_P is sufficient to evaluate drift for every D. Raw examples are one way to estimate it, but not the only representation.

Given m historical activations at a fixed reference checkpoint,

$$\widehat{C} = \frac{1}{m}\sum_{i=1}^{m} x_i x_i^\top, \qquad \widehat{\mathcal{R}}(D) = \operatorname{Tr}(D\widehat{C}D^\top). \tag{19}$$

Writing $C = C_P$,

$$\left|\widehat{\mathcal{R}}(D) - \mathcal{R}_P(D)\right| \le \|D\|_F^2 \|\widehat{C} - C\|_2. \tag{20}$$

Thus estimation quality depends on how well the sampled activations recover the stored geometry.

A fixed-capacity representation can further compress $\widehat{C}$. Let

$$\widetilde{C} = Q\Lambda Q^\top, \qquad Q^\top Q = I, \qquad \operatorname{rank}(\widetilde{C}) \le k,$$

and define

$$\epsilon_{\text{samp}} = \|\widehat{C} - C\|_2, \qquad \epsilon_{\text{sk}} = \|\widetilde{C} - \widehat{C}\|_2.$$

Then

$$\left|\mathcal{R}_P(D) - \operatorname{Tr}(D\widetilde{C}D^\top)\right| \le \|D\|_F^2 \left(\epsilon_{\text{samp}} + \epsilon_{\text{sk}}\right). \tag{21}$$

If $DQ = 0$, then

$$\mathcal{R}_P(D) \le \|D\|_F^2 \left(\epsilon_{\text{samp}} + \epsilon_{\text{sk}}\right). \tag{22}$$

Protection is exact on retained directions, while full-distribution protection depends on sampling coverage, discarded directions, and update magnitude. These deterministic bounds separate sampling error from compression error; finite-sample rates additionally require assumptions on the activation distribution.

The two-task example makes the geometry concrete. For $\mathcal{D}_1$,

$$C_1 = e_1 e_1^\top.$$

With

$$d = (w - w^{(1)})^\top,$$

the local drift is

$$\mathcal{R}_1(d) = (a-1)^2.$$

Protecting e_1 imposes $a = 1$; combined with $-a + b \ge 1$, the minimum-norm solution becomes

$$(1, 2)^\top,$$

which solves both tasks. Here a reference parameter and one protected direction suffice without repeated raw-example replay.

The next section asks how such limited historical access should be used: to optimize historical losses directly, or to constrain updates through historical activation geometry.

B METHOD AND IMPLEMENTATION DETAILS

Section 2 establishes that update information alone does not identify historical response drift. This appendix specifies how historical examples are converted into geometry, how that geometry constrains a shared LoRA adapter, and how the comparison methods use the same historical access.

B.1 HISTORICAL ACCESS AND NOTATION

At a task boundary, the learner receives a fixed probe set from the completed task's training split. Under *temporary access*, the probes are processed once and then discarded; under *persistent access*, prompt-only inputs remain available at later task boundaries. HARD AG-LORA uses temporary access, whereas REFRESH AG-LORA retains the per-task prompt sets needed to recompute its geometry. Replay-64 instead retains a fixed total of 64 labeled examples.

For LoRA layer l, let

$$A_{l,t,j} \in \mathbb{R}^{r \times d_{\mathrm{in},l}}, \qquad B_{l,t,j} \in \mathbb{R}^{d_{\mathrm{out},l} \times r}, \qquad U_{l,t,j} = sB_{l,t,j}A_{l,t,j},$$

where $s = \alpha/r$, t indexes tasks, and j indexes optimizer steps within a task. Let $\mathcal{R}_{l,t}$ denote the capped activation reservoir available before task t, $X_{l,t}$ its row-stacked matrix, and $Q_{l,t} \in \mathbb{R}^{d_{\mathrm{in},l} \times k_{l,t}}$ the corresponding orthonormal protected basis. We use

$$D_{l,t,j} = U_{l,t,j} - U_{l,t,0}$$

for the effective LoRA displacement from the start of task t.

A reservoir of M width-d rows requires $O(Md)$ stored numbers, whereas a rank-k spectral summary requires $O(dk + k)$. These quantities are not byte-matched to tokenized replay: a single prompt can produce many correlated activation rows, while Replay-64 repeatedly uses labels and raw text in the training loss.

B.2 ACTIVATION GEOMETRY AND LEADING DIRECTIONS

For a prompt-only historical input z, let $x_l(z; \bar{\theta}) \in \mathbb{R}^{d_{\mathrm{in},l}}$ be the input activation of a LoRA-equipped layer under geometry model $\bar{\theta}$. The empirical uncentered second moment is

$$\widehat{C}_l(\bar{\theta}) = \frac{1}{n_l} \sum_{i=1}^{n_l} x_{l,i}(\bar{\theta}) x_{l,i}(\bar{\theta})^\top. \tag{23}$$

If the effective displacement obeys

$$D_l Q_l = 0, \tag{24}$$

then $D_l x = 0$ for every $x \in \mathrm{span}(Q_l)$. The constraint therefore preserves the direct adapter response on protected coordinates; it does not freeze upstream activations or guarantee unchanged end-to-end predictions.

Proposition (Minimax optimality of leading activation directions). Let $C \succeq 0$ be in $\mathbb{R}^{d \times d}$, with eigenvalues $\lambda_1(C) \geq \cdots \geq \lambda_d(C)$. For $p \geq 1$, $0 \leq k < d$, and an effective-update norm budget $\gamma > 0$,

$$\min_{Q^\top Q = I_k} \sup_{\substack{DQ=0 \\ \|D\|_F \leq \gamma}} \mathrm{Tr}(DCD^\top) = \gamma^2 \lambda_{k+1}(C). \tag{25}$$

A leading k-dimensional eigenspace attains the minimum.

Proof. For fixed Q, every row of D lies in $\mathrm{span}(Q)^\perp$. The supremum allocates its Frobenius-norm budget to a unit vector maximizing $v^\top C v$ in that complement, giving

$$\gamma^2 \max_{\substack{v \perp Q \\ \|v\|_2 = 1}} v^\top C v.$$

The Courant–Fischer theorem minimizes this quantity at $\gamma^2 \lambda_{k+1}(C)$, attained when Q spans leading eigenvectors. □

In the evaluated implementation, the protected rank is the smallest k whose cumulative eigenvalue fraction reaches the energy threshold $\eta = 0.95$, followed by $k \leftarrow \min(k, 64)$. The symbol η is reserved for this energy threshold; γ above denotes only the norm budget in the minimax proposition above.

Algorithm 1: HARD AG-LORA over a task sequence.
Input: tasks $\{\mathcal{D}_t\}_{t=1}^{T}$; one shared rank-8 LoRA adapter $\{A_l, B_l\}$; per-layer reservoir capacity $C = 64$; energy threshold $\eta = 0.95$; rank cap $k_{\max} = 64$; base seed 0. Initialize an empty reservoir $\mathcal{R}_l$ and observed-row counter $n_l = 0$ for every targeted `q_proj`/`v_proj` layer; seed reservoir sampling with 97.

1. At the start of task t, snapshot $A_{l,t,0}, B_{l,t,0}$. If $t > 1$, convert $\mathcal{R}_l$ to FP32, eigendecompose $\mathcal{R}_l\mathcal{R}_l^\top$, discard eigenvalues no larger than $\max(10^{-6}\lambda_1, 10^{-12})$, and choose the smallest k reaching energy η, capped by $k_{\max}$. Reconstruct
$$Q_{l,t} = \mathcal{R}_l^\top V_{l,1:k}\Lambda_{l,1:k}^{-1/2},$$
apply reduced FP32 QR, and freeze B_l.
2. Train on the current-task loss with a fresh AdamW optimizer. After every optimizer step for $t > 1$, replace A_l by the task-anchored projection in Equation 26 and restore $B_l = B_{l,t,0}$. Adam moments are left unchanged; only the accepted parameter state is projected.
3. Evaluate all tasks learned through t. Select 256 prompt-only examples from the completed task's training split, remove answer tokens, and collect activations in evaluation/inference mode. The shared sampling protocol yields exactly 64 FP16 candidate rows S_l per target layer.
4. Merge each row of S_l into $\mathcal{R}_l$: increment n_l; append while $|\mathcal{R}_l| < C$; otherwise draw $j \sim \mathrm{Uniform}\{0, \ldots, n_l - 1\}$ and replace row j iff $j < C$. Save reservoir values, counts, and generator state before continuing.

Figure 5: Complete evaluated HARD AG-LORA loop. Task 1 is unconstrained; its post-training activations construct the geometry used for Task 2.

B.3 SHARED TASK-ANCHORED PROJECTION

Stored and refreshed geometry use the same within-task constraint. Before task $t > 1$, snapshot $A_{l,t,0}$ and $B_{l,t,0}$, freeze B_l, and keep $Q_{l,t}$ fixed throughout the task. If AdamW proposes $\widetilde{A}_{l,t,j+1}$, the accepted factor is

$$A_{l,t,j+1} = A_{l,t,0} + \left(\widetilde{A}_{l,t,j+1} - A_{l,t,0}\right)\left(I - Q_{l,t}Q_{l,t}^\top\right). \tag{26}$$

Consequently,

$$D_{l,t,j+1}Q_{l,t} = sB_{l,t,0}\left(A_{l,t,j+1} - A_{l,t,0}\right)Q_{l,t} = 0. \tag{27}$$

Projection is applied after every optimizer step to the finite displacement from the task-start anchor. Adam's moment tensors are not projected and may contain protected-direction components; however, every parameter proposal is projected before it becomes an accepted adapter state. The guarantee is therefore about the accepted effective LoRA displacement, not the optimizer state, upstream representations, or complete network output.

B.4 STORED AND REFRESHED ACTIVATION GEOMETRY

B.4.1 STORED GEOMETRY: HARD AG-LORA

After completing task t, HARD AG-LORA selects 256 prompt-only probes from that task's training split and collects inputs to every targeted LoRA layer. The sampling protocol yields 64 candidate rows per layer, which are merged row-by-row into the fixed-capacity reservoir $\mathcal{R}_l$ with uniform Algorithm-R replacement. The raw prompts are then discarded. Before the next task, $X_{l,t}$ is converted to FP32 and the small Gram matrix $X_{l,t}X_{l,t}^\top$ is eigendecomposed. Right singular directions are reconstructed in input space and orthonormalized to produce $Q_{l,t}$.

Algorithm 2: REFRESH AG-LORA over a task sequence.
Input: tasks $\{\mathcal{D}_t\}_{t=1}^{T}$; one shared rank-8 LoRA adapter $\{A_l, B_l\}$; prompt capacity $K = 256$ per completed task; per-layer reservoir capacity $C = 64$; energy threshold $\eta = 0.95$. Initialize an empty prompt history $\mathcal{H}$.

1. At the start of task t, snapshot $A_{l,t,0}, B_{l,t,0}$. If $t > 1$, forward every prompt in $\mathcal{H}$ once through the boundary model M_{t-1}, merge the resulting rows into a fresh reservoir of capacity C, and construct $Q_{l,t}$ with the same FP32 Gram eigendecomposition and energy rule as HARD AG-LORA. Freeze B_l for the task.
2. Train on $\mathcal{D}_t$ with a fresh AdamW optimizer. After every optimizer step for $t > 1$, apply the task-anchored projection in Equation 26 and restore $B_l = B_{l,t,0}$. Do not re-forward historical prompts during the task.
3. Evaluate all tasks learned through t. Select K prompt-only examples from the completed task's training split and append them to $\mathcal{H}$. Retain the prompt sets for all completed tasks so that the next boundary can rebuild the geometry under its current model.

Figure 6: Complete evaluated REFRESH AG-LORA loop. Unlike HARD AG-LORA, the historical prompts are retained and re-encoded once at each task boundary.

B.4.2 REFRESHED GEOMETRY: REFRESH AG-LORA

REFRESH AG-LORA retains 256 fixed prompt-only probes from each completed task's training split. Once before task t, it forwards all retained historical prompts through the boundary model M_{t-1}, rebuilds a fixed-capacity activation reservoir, and extracts $Q_{l,t}$ using the same Gram eigendecomposition and energy rule as HARD AG-LORA. The basis is then held fixed throughout task t, and Equation 26 is applied after every optimizer step. Historical prompts are not re-forwarded at individual steps. Both variants update the same rank-8 adapter; they differ only in whether the protected coordinates were stored when each task ended or recomputed under the latest boundary model.

B.5 REPLAY AND SOFT-PENALTY CONTROLS

B.5.1 REPLAY-64

Replay-64 retains a fixed total of 64 labeled historical examples. At later tasks, each micro-batch pads and concatenates eight current examples with eight replay examples and performs one forward pass:

$$\mathcal{L}_{\text{replay}}(\theta) = \text{LM-Loss}(f_\theta; \text{PadConcat}(\mathcal{B}_{\text{new}}, \mathcal{B}_{\text{old}})) \,. \tag{28}$$

Eight micro-batches are accumulated per optimizer update, giving 64 new and 64 historical examples. There is no separate replay coefficient. Because the loss is reduced over valid answer tokens in the concatenated batch, the effective new/old weighting depends on their respective valid-token counts. The fixed memory is divided as evenly as possible among old tasks, with approximately label-balanced selection within each task.

B.5.2 SOFT AG-LORA

Soft AG-LoRA uses the same stored activation reservoirs but replaces the hard constraint with an activation-drift penalty. At the start of task t, snapshot both factors as $A_{l,t,0}$ and $B_{l,t,0}$, but keep both A_l and B_l trainable. For $X_{l,t}$ containing $N_{l,t}$ historical activation rows, define

$$E_{l,t} = sX_{l,t}\left(A_{l,t}^\top B_{l,t}^\top - A_{l,t,0}^\top B_{l,t,0}^\top\right), \tag{29}$$

and

$$\mathcal{L}_{\text{AG}} = \frac{1}{L}\sum_{l=1}^{L} \frac{\|E_{l,t}\|_F^2}{N_{l,t} d_{\text{out},l}}. \tag{30}$$

The evaluated objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + 1.0\,\mathcal{L}_{\text{AG}}.$$

Algorithm 3: Replay-64 over a task sequence.
Input: tasks $\{\mathcal{D}_t\}_{t=1}^{T}$; one shared rank-8 LoRA adapter; memory capacity $M = 64$; new/old micro-batch sizes 8 and 8; gradient-accumulation count $G = 8$. Initialize an empty labeled memory $\mathcal{M}$.

1. For each task t, initialize a fresh AdamW optimizer. For every optimizer update, repeat G times: draw eight current-task examples and eight examples from $\mathcal{M}$ (cycling and reshuffling as needed), pad-and-concatenate the two batches, and accumulate the ordinary token-level LM loss on the combined batch. If $\mathcal{M}$ is empty, train Task 1 on current-task examples only.
2. Apply one optimizer step after the G accumulated forwards. Do not introduce a separate replay coefficient; the combined batch loss determines the effective new/old weighting through valid target-token counts.
3. Evaluate all tasks learned through t. Select labeled examples from $\mathcal{D}_t$, then rebalance $\mathcal{M}$ across completed tasks with near-equal label counts while keeping its total size at most M.

Figure 7: Complete evaluated Replay-64 loop. The persistent state is a fixed total of 64 labeled examples.

Algorithm 4: SOFT AG-LORA over a task sequence.
Input: tasks $\{\mathcal{D}_t\}_{t=1}^{T}$; one shared rank-8 LoRA adapter; per-layer reservoir capacity $C = 64$; regularization strength $\lambda = 1.0$. Initialize an empty activation reservoir $\mathcal{R}_l$.

1. At the start of task t, snapshot both factors as $A_{l,t,0}, B_{l,t,0}$. If $t > 1$, use the stored reservoir rows $X_{l,t}$ in the geometry penalty. Keep both A_l and B_l trainable.
2. For each current-task mini-batch, compute the task loss and
$$\mathcal{L}_{\mathrm{AG}} = \frac{1}{L}\sum_{l} \frac{\left\| sX_{l,t}\left(A_l^\top B_l^\top - A_{l,t,0}^\top B_{l,t,0}^\top\right)\right\|_F^2}{N_{l,t} d_{\mathrm{out},l}}.$$
Set $\mathcal{L} = \mathcal{L}_{\mathrm{task}} + \lambda \mathcal{L}_{\mathrm{AG}}$, backpropagate, and take one AdamW step. Apply no projection and freeze nothing.
3. Evaluate all tasks learned through t. Select 256 prompt-only examples from the completed task's training split, collect their layer inputs in inference mode, and merge the resulting rows into each fixed-capacity reservoir with the same Algorithm-R update used by HARD AG-LORA.

Figure 8: Complete evaluated SOFT AG-LORA loop. The activation penalty is soft: it discourages historical drift but does not impose a zero-drift constraint.

After each task, prompt activations from its training split are merged into the same fixed-capacity reservoirs. This penalty makes historically disruptive updates more expensive but does not enforce $D_{l,t,j}Q_{l,t} = 0$; both LoRA factors may change. Its Qwen Order-1 result is reported in Appendix E.3.

B.6 DIAGNOSTIC SCORE AND LOW-RANK COMPUTATION

For the diagnostic study, the normalized activation score is

$$G_{\mathrm{act}}(D) = \frac{1}{L}\sum_{l=1}^{L} \frac{1}{d_{\mathrm{out},l}} \mathrm{Tr}\left(D_l \frac{C_l}{\mathrm{Tr}(C_l) + \epsilon} D_l^\top\right). \tag{31}$$

It is a rescaled layer-local proxy, not a bound on end-to-end forgetting.

The difference between two rank-r effective LoRA weights has rank at most $2r$ and can be written $D_l = L_l H_l$. For activation sketch X_l, let $Z_l = X_l H_l^\top$. Then

$$\|X_l D_l^\top\|_F^2 = \mathrm{Tr}\left[(Z_l^\top Z_l)(L_l^\top L_l)\right].$$

Only projections into the small factor space and small Gram matrices are needed, avoiding explicit dense $d_{\mathrm{out},l} \times d_{\mathrm{in},l}$ updates.

C Common Experimental Protocols

We organize the evaluation around three complementary studies. The diagnostic study asks whether a distribution-conditioned diagnostic can characterize forgetting more effectively than training progress, update magnitude, and alternative internal metrics. The baseline comparison examines continual-learning strategies that preserve historical knowledge in different ways: no explicit protection, growing parameter subspaces, fixed-rank activation-guided protection, and direct replay. The mechanism study ablates the components of Hard AG-LoRA to determine whether its gains come from freezing the LoRA output factor, merely reducing the update space, or preserving activation-informed directions specifically.

C.1 Language data, prompts, and evaluation

Table 4 records the exact prepared-corpus sizes used by the formal Qwen and Llama runs.

Table 4: Prepared language datasets.

Dataset	Train	Test	Labels
DBpedia	14,000	7,600	14
Amazon Reviews	5,000	7,600	5
Yahoo Answers	10,000	7,600	10
AG News	4,000	7,600	4

Every source sequence has the following literal structure, with `Task` set to `TC` or `SC` and all task labels inserted in their file order:

```
Task:{type}
Dataset:{name}
{topic-or-sentiment instruction}
Option:  {label1}, ..., {labelK}
{input text}
Answer:
```

The tokenizer adds special tokens to the prefix; only the input-text body is truncated so the complete source fits 512 tokens. Supervised training appends the ground-truth label (at most 50 tokens) and EOS, masks the entire source in the loss, and trains by next-token prediction. Geometry collection stops at `Answer:` and therefore contains candidate-label strings but not the example's answer target.

Generation evaluation uses left padding, cache-enabled evaluation mode, greedy decoding (`do_sample=False`), the tokenizer's EOS and padding IDs, and at most 12 new tokens. The decoded continuation is correct iff its first nonempty line, after an optional case-insensitive `Answer:`/`Label:` prefix and surrounding spaces, periods, commas, colons, semicolons, exclamation/question marks, and single or double quotes are removed, exactly matches the case-folded label string. All learned tasks are evaluated after every stage.

C.2 Optimization, randomization, and method state

The formal runs use PyTorch AdamW with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, AMSGrad disabled, and weight decay 0. Ordinary updates use mini-batch 16 with four-step accumulation; gradients are clipped to global norm 1 immediately before each step. Each task constructs a fresh optimizer, so no Adam moments carry across task boundaries. For constrained variants, the affine projection in Equation 26 is applied immediately after every optimizer step and before gradients are cleared for the next update.

After a task is trained and evaluated, activation hooks run under `torch.inference_mode()` with `model.eval()`. The 256 prompt-only examples form 16 unshuffled batches of 16. For

each target layer, the attention mask removes padding; valid tokens from a batch are flattened, and four evenly spaced midpoint indices are selected. The resulting 64 rows are stored as FP16 CPU tensors. Reservoir updates operate on individual activation rows using standard uniform Algorithm-R replacement. Since every task contributes exactly 64 incoming rows, no explicit task weighting is applied, and every row observed so far has equal inclusion probability.

To form a protection basis, the implementation converts the retained matrix X_l to FP32 on CPU and eigendecomposes $X_l X_l^\top$. Eigenvalues are sorted descending, clipped at zero, and values no larger than $\max(10^{-6}\lambda_{l,1}, 10^{-12})$ are discarded. The selected rank is the smallest k satisfying $\sum_{i=1}^{k} \lambda_{l,i} / \sum_i \lambda_{l,i} \geq 0.95$, followed by the rank-64 cap. The right basis is reconstructed as $X_l^\top V_{l,1:k} \Lambda_{l,1:k}^{-1/2}$ and re-orthonormalized by reduced FP32 QR. Thus the implementation is algebraically equivalent to a thin SVD but avoids forming a hidden-width squared covariance. The 95% threshold, 64-row capacity, and rank-64 cap are fixed across the formal orders rather than selected per order.

C.3 Formal Qwen configuration

Table 5 collects the shared configuration of the formal four-task Qwen comparison. "Full" means the complete prepared split in the local O-LoRA-format data, rather than a fixed 2,000-example pilot subset. Task-aware prompts include the dataset name and candidate labels; the four datasets do not share a global label space.

Table 5: Shared Qwen2.5-7B continual-learning configuration.

Component	Setting
Precision / quantization	BF16 / none
Training data / epochs	full prepared split / 1 per task
Optimizer	AdamW, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$
Learning rate / weight decay	10^{-3} / 0
Mini-batch / accumulation / clip	16 / 4 / global norm 1
Optimizer state between tasks	reset
Source / target length	512 / 50 tokens
LoRA targets	`q_proj`, `v_proj`
Rank / α / dropout	8 / 32 / 0.1
Activation collection	eval mode; valid tokens; 64 rows/layer
Activation reservoir	64 rows per target layer
Protected subspace	FP32 Gram eigensolve; 95% energy; rank ≤ 64
Evaluation	7,600 examples/task after each stage; greedy 12 tokens

C.4 Diagnostic study: forgetting and update geometry

We study diagnosis independently of any intervention. Using ordinary Sequential LoRA trajectories on Qwen2.5-7B-Instruct ensures that none of the candidate diagnostics affects training. We run through all four tasks and record every old task at 0%, 25%, 50%, and 100% progress through each later task. The completed subset spans the same three task orders used in the baseline comparison. For each old task, 256 calibration examples construct activation/Fisher statistics, 1,000 disjoint validation examples define the forgetting trajectory, and official test data are reserved for final endpoints. The analysis treats either an individual old task or the mean over all old tasks as the prediction unit, and holds out the current task, old task, or task order as applicable. The diagnostic analysis is evaluated independently and is not used to select the method or tune the metric.

For diagnostics, we measure cumulative validation accuracy and NLL forgetting. Candidate internal metrics are effective-update norm, spectral/SVD energy, LoRA-subspace interference, directional Fisher, and the activation-geometry score in Equation 31; calibration-set output KL is a more expensive functional comparator. All metrics use the effective update $\Delta W = (\alpha/r)(B_t A_t - B_0 A_0)$ rather than raw factor distances.

We fit metric-to-forgetting mappings only on training folds and use pooled out-of-fold (OOF) MAE as the primary diagnostic criterion. Outer splits hold out the current task, old task, or task order as

applicable. Deconfounded models control for training progress, parameter norm, and current-task validation NLL and accuracy. Confidence intervals use 5,000 bootstrap samples clustered by the complete training run, preserving dependence among checkpoints. Supplementary R^2 values are reported in Appendix D.5.

C.5 Baseline comparison: continual-learning strategies

We next compare four ways of handling historical information on the same four classification tasks. The three task orders are DBP→AMZ→YAH→AGN, DBP→AMZ→AGN→YAH, and YAH→AMZ→AGN→DBP. Qwen2.5-7B-Instruct is trained in BF16 for one epoch per task using the full prepared training split, learning rate 10^{-3}, effective batch 64, and LoRA on `q_proj` and `v_proj`. Prompts contain dataset identity and candidate labels, so this is a task-aware rather than task-ID-free setting.

The comparison includes ordinary Sequential LoRA, which keeps one rank-8 adapter without explicit historical protection; O-LoRA, which adds a rank-8 block for every new task; HARD AG-LORA, which keeps one rank-8 adapter and uses the historical activation sketch to constrain later updates; and Replay-64, which keeps a fixed buffer of labeled historical examples and directly optimizes them during later tasks. O-LoRA freezes historical factor blocks, initializes one new rank-8 block per task, and adds $0.5 \sum_l \|A_l^{\mathrm{old}} A_l^{\mathrm{new}\top}\|_1$; 0.5 is the value in the retained official reproduction scripts and is fixed for all orders and both backbones.

Replay-64 draws labeled examples from the same 256-row training split for geometry. The fixed total capacity is divided across historical tasks as 64; 32+32; and 22+21+21, and selection within each task cycles through sorted labels to obtain near-equal label counts. Each later optimizer update receives 64 new and 64 old examples through eight accumulated forwards of 8+8; the replay loader reshuffles and cycles the 64-row history as needed. HARD AG-LORA instead forwards all 256 prompt-only source examples once at each task boundary, stores 64 sampled activation rows per target layer, and never replays their loss during later optimization.

For T tasks, final average accuracy is $\mathrm{AA} = T^{-1} \sum_i a_{T,i}$. Average forgetting compares each historical task's best accuracy after it was learned with its final accuracy; backward transfer (BWT) compares accuracy immediately after learning with final accuracy. Forgetting and BWT are reported in percentage points. To separate predictive performance from method-state cost, we also measure persistent adapter and auxiliary-state sizes directly from the formal checkpoints, excluding the shared frozen backbone.

C.6 Mechanism study: ablating protection

Finally, we isolate the mechanism of HARD AG-LORA on DBP→AMZ→YAH→AGN. We compare the full method with (i) Freeze-B without a projector, (ii) a random projector whose per-layer rank matches the activation-informed projector, and (iii) ordinary Sequential LoRA as the no-projection control. All variants retain one rank-8 adapter and use the same full-data BF16 training protocol. The random basis is independent of historical activation directions; only its rank is matched. The random basis uses the same selected per-layer rank and a deterministic task-and-layer-specific seed.

The ablation uses the same final AA, forgetting, and BWT metrics as the baseline comparison. We additionally verify the intended intervention through implementation diagnostics: B-factor drift, projection-constraint residuals, matched per-layer projector ranks, and the fraction of historical activation energy covered by the random directions.

D Four-Task Forgetting-Diagnostic Validation

This appendix evaluates whether relatively lightweight checkpoint-level signals can predict behavioral forgetting in the four-task Qwen study. The goal is diagnostic validation rather than continual-learning method comparison: none of the candidate metrics is used to train, select, or modify the sequential learner. The study uses the same task definitions and data roles as the four-task Qwen benchmark, but all diagnostic mappings are fitted only from non-test data.

The primary estimand is *fixed-anchor forgetting*: performance on an old task is compared with that task's validation performance immediately after it was learned. This differs from a best-so-far definition of forgetting and keeps the diagnostic target aligned with the amount of change accumulated since the task boundary.

D.1 Study design

Model, tasks, and runs. We use Qwen2.5-7B-Instruct with BF16 backbone weights and a frozen base model. A rank-8 LoRA adapter is trained on the `q_proj` and `v_proj` modules. The four tasks are DBpedia (DBP), Amazon Reviews (AMZ), Yahoo Answers (YAH), and AG News (AGN). We evaluate three pre-specified task orders inherited from the four-task benchmark:

$$\text{DBP} \to \text{AMZ} \to \text{YAH} \to \text{AGN},$$
$$\text{DBP} \to \text{AMZ} \to \text{AGN} \to \text{YAH},$$
$$\text{YAH} \to \text{AMZ} \to \text{AGN} \to \text{DBP}.$$

Each order is run with seeds 0 and 1, giving six complete sequential runs. The diagnostic study therefore reports results only for these six observed runs.

The prepared training-set sizes are 14,000 examples for DBpedia, 5,000 for Amazon Reviews, 10,000 for Yahoo Answers, and 4,000 for AG News. Each task is trained for one epoch with learning rate 10^{-3}, AdamW, a fresh optimizer at each task boundary, mini-batch size 16, gradient accumulation over four micro-batches, and global gradient-norm clipping at 1. The corresponding optimizer-step counts are 219, 79, 157, and 63, respectively. No replay, projection, or diagnostic penalty is used.

Checkpoints and behavioral evaluation. After each task is learned, its validation result becomes that task's fixed anchor. During each later task, the model is evaluated at 25%, 50%, and 100% training progress. A zero-progress state is retained as a transition reference but is excluded from the level-prediction analyses reported below. Consequently, a parameter state that is simultaneously the previous task's 100% checkpoint and the next task's 0% checkpoint is not counted twice in the main level fits.

For each task, 1,000 validation examples are used to measure accuracy and negative log-likelihood (NLL). These examples are disjoint from both the training data and the diagnostic calibration pool. The complete official test split of each task is reserved for final endpoint reporting and is never used to fit a diagnostic mapping.

D.2 Fixed-anchor forgetting targets

Let b_i denote the checkpoint immediately after task i is learned. For a later checkpoint t, the primary accuracy-forgetting target is

$$F_{\mathrm{acc},i}(t) = a_i(b_i) - a_i(t), \tag{32}$$

where $a_i(\cdot)$ is held-out validation accuracy. The corresponding NLL target is

$$F_{\mathrm{nll},i}(t) = \ell_i(t) - \ell_i(b_i), \tag{33}$$

where $\ell_i(\cdot)$ is validation NLL. We do not apply a non-negative clipping operation, so negative values are retained when an old task improves relative to its post-training anchor.

We report two prediction scopes. The *individual-old-task* scope keeps one target for each old task at each checkpoint. The *mean-old-tasks* scope first averages the forgetting values of all old tasks at the same run, stage, and checkpoint and predicts this aggregate. The positive-progress level data contain 108 individual-old-task observations and 54 mean-old-task observations.

A best-so-far, non-negative definition of forgetting is treated only as a separate sensitivity estimand and is not mixed with the primary results in this appendix.

D.3 Calibration pool and diagnostic signals

Each task has a fixed 256-example calibration pool drawn from its development split using split seed 20260921. After deterministic shuffling, the first 256 examples form the calibration pool and the next

1,000 form the validation set. The same task-level calibration subset is reused across seeds and task orders. Calibration examples are disjoint from training, validation, and official test data.

The candidate metrics differ in the information they require.

Parameter and update metrics. All update-based metrics use the effective LoRA change relative to the stored old-task reference,

$$\Delta W = \frac{\alpha}{r} \left(B_t A_t - B_0 A_0 \right), \tag{34}$$

rather than raw distances between LoRA factors. Parameter norm is the root-mean-square magnitude of ΔW. SVD spectral change uses the largest singular value squared, normalized by matrix size. LoRA interference measures the energy of the current effective update after projection onto the historical adapter active subspace.

Activation geometry. For each target `q_proj` or `v_proj` layer, the effective update is represented as $\Delta W = LR$. If X contains stored old-task input activations, then the induced representation change is

$$\Delta Y = (XR^\top)L^\top. \tag{35}$$

The layer-level activation-geometry score is

$$G_{\text{layer}} = \frac{\|\Delta Y\|_F^2}{n\, d_{\text{out}} \left(\frac{1}{n} \|X\|_F^2 \right)}, \tag{36}$$

where n is the number of stored activation rows and d_{out} is the layer output dimension. The checkpoint-level score is the arithmetic mean across finite layer scores.

The activation sketch contains 64 token-level rows. These rows are not randomly resampled at each checkpoint. With the calibration DataLoader unshuffled, valid token positions are selected deterministically by a uniform midpoint rule. The resulting sketch is created once for an old task at its task boundary and is reused at all later checkpoints.

Directional Fisher. Directional Fisher evaluates the mean squared directional interaction $(g^\top \Delta y)^2$ on a fixed 64-row old-task sketch. Unlike activation geometry, this metric uses calibration labels and output gradients. Its sketch is constructed deterministically from the first 64 calibration examples with batch size one and uses the same midpoint token-selection rule.

Calibration KL. Calibration KL compares stored old-task predictive logits with current logits on the calibration pool. It is therefore a functional comparator with a richer information budget than parameter-only diagnostics: it requires historical inputs and stored reference predictions. Activation geometry also uses historical inputs, but does not require stored reference logits.

This distinction is important when interpreting the results below: calibration KL is included as an accuracy-oriented functional comparator, rather than as an information-matched internal baseline.

D.4 OOF FITTING AND HELD-OUT PROTOCOLS

For each candidate metric, the metric-to-forgetting mapping is a univariate Huber robust linear regression with an intercept. Within each outer training fold, the metric is transformed as

$$x' = \operatorname{sign}(x) \log \left(1 + \frac{|x|}{m_{\text{train}}} \right), \tag{37}$$

where m_{train} is the median non-zero magnitude computed from the training portion only. Standardization is also fitted only on the training portion. If the Huber fit produces non-finite predictions, the implementation falls back to ridge regression with $\alpha = 1$.

We evaluate four held-out protocols:

1. **Current-task holdout**: all observations whose current task has a particular identity are held out. Other observations from the same complete sequential run may remain in the training portion. This protocol therefore tests generalization to an unseen current-task identity, not to an entirely unseen run.

Table 6: Individual-old-task pooled OOF MAE for fixed-anchor forgetting. Lower is better. Each cell contains $n = 108$ pooled OOF observations. The double holdout is the stricter order-plus-seed stress test.

	Accuracy forgetting				NLL forgetting			
Metric	Current	Old	Order	Double	Current	Old	Order	Double
Activation geometry	0.082939	0.090259	0.085345	0.093180	0.379460	0.440866	0.410141	0.457683
Parameter norm	0.092102	0.104651	0.096182	0.116629	0.404846	0.482109	0.433851	0.486099
SVD spectral	0.095353	0.105678	0.097251	0.118916	0.401383	0.476841	0.425661	0.485803
LoRA interference	0.092630	0.105899	0.096067	0.116415	0.412894	0.500171	0.440917	0.490371
Directional Fisher	0.117409	0.129686	0.114143	0.156572	0.507648	0.584602	0.517578	0.767702
Calibration KL	**0.055501**	**0.059431**	**0.055752**	**0.055451**	**0.188413**	**0.214840**	**0.185481**	**0.212303**

2. **Old-task holdout**: all observations for a particular old-task identity are held out. As above, other observations from the same run may remain in training. This tests generalization to an unseen historical-task identity.
3. **Task-order holdout**: one of the three complete task orders is held out. This split is run-disjoint by order. Because two of the benchmark orders share the DBP→AMZ prefix, however, this protocol should not be interpreted as eliminating all shared trajectory structure.
4. **Order-plus-seed double holdout**: for a held-out order o and seed s, the fitting set is restricted to
$$\{\text{order} \neq o\} \cap \{\text{seed} \neq s\}.$$
This removes both the held-out order and the held-out seed from the fitting set and serves as a stricter stress test. With only three orders and two seeds, each double-holdout fold is necessarily data-poor and is interpreted descriptively rather than as a high-powered inferential test.

Current-task, old-task, task-order, and double holdouts use 4, 4, 3, and 6 folds, respectively. Predictions from all folds of a given protocol are pooled before computing MAE and R^2; results are not averaged across protocols. For pooled OOF predictions,

$$R^2 = 1 - \frac{\sum_j (y_j - \hat{y}_j)^2}{\sum_j (y_j - \bar{y}_{\text{OOF}})^2}. \tag{38}$$

Thus a negative value is worse, in squared-error terms, than the constant predictor given by the pooled OOF target mean.

D.5 Standalone prediction of individual-task forgetting

Table 6 reports OOF MAE for the individual-old-task target. Activation geometry has the lowest point-estimate MAE among the tested internal diagnostics under every held-out protocol for both accuracy and NLL forgetting. Calibration KL is substantially more accurate overall, consistent with its access to stored historical predictive outputs.

The corresponding pooled OOF R^2 values are shown in Table 7. Activation geometry remains positive under all four protocols, whereas several simpler update statistics approach or fall below the pooled-mean baseline under the stricter splits. Directional Fisher is particularly unstable in this setting.

These results support a limited but consistent standalone conclusion: activation geometry is the most predictive of the tested internal diagnostics for *individual-task* forgetting in this six-run study. They do not imply that it is the strongest diagnostic under all information budgets, because calibration KL is uniformly more accurate.

D.6 Mean-old-task forgetting

We also aggregate the old-task targets within each run, stage, and checkpoint before fitting. After this aggregation the old-task identity is the single synthetic value "mean," so an old-task holdout is not defined. We therefore report current-task, task-order, and double holdouts only.

Table 7: Individual-old-task pooled OOF R^2 for fixed-anchor forgetting. Higher is better. Negative values are worse than the pooled OOF global-mean baseline under the definition above.

	Accuracy forgetting				NLL forgetting			
Metric	Current	Old	Order	Double	Current	Old	Order	Double
Activation geometry	0.374839	0.291263	0.351362	0.318087	0.261464	0.110036	0.187872	0.109273
Parameter norm	0.274933	0.125180	0.247522	-0.081177	0.161551	-0.055294	0.094755	-0.114599
SVD spectral	0.233568	0.093372	0.213625	-0.111824	0.147834	-0.051853	0.090073	-0.113772
LoRA interference	0.273253	0.093358	0.242349	-0.074897	0.145895	-0.115437	0.078034	-0.120461
Directional Fisher	-0.152442	-0.323381	-0.085369	-0.915947	-0.109250	-0.373356	-0.129181	-1.316782
Calibration KL	**0.685939**	**0.649016**	**0.682997**	**0.712283**	**0.813376**	**0.779656**	**0.822691**	**0.740238**

Table 8: Mean-old-task pooled OOF results for fixed-anchor forgetting. Each cell contains $n = 54$ pooled OOF observations and reports MAE / R^2. Old-task holdout is undefined after averaging over old-task identity.

	Accuracy forgetting			NLL forgetting		
Metric	Current	Order	Double	Current	Order	Double
Activation geometry	0.063816 / 0.530362	0.074032 / 0.412479	0.088940 / 0.232081	0.257983 / 0.343977	0.291706 / 0.247132	0.380293 / 0.040097
Parameter norm	0.071183 / 0.467540	0.077586 / 0.390037	0.096447 / 0.038105	0.296306 / 0.309748	0.298972 / 0.264459	0.378609 / -0.077227
SVD spectral	0.068957 / 0.462802	0.076060 / 0.389988	0.094687 / 0.081118	0.271771 / 0.343996	0.285006 / 0.296321	0.359532 / 0.003931
LoRA interference	0.067450 / 0.489216	0.075892 / 0.400298	0.093905 / 0.077499	0.282758 / 0.315264	0.293422 / 0.262205	0.366010 / -0.038034
Directional Fisher	0.094343 / 0.082242	0.086263 / 0.145444	0.123785 / -0.654926	0.356786 / 0.059207	0.381121 / -0.115504	0.529814 / -0.997562
Calibration KL	**0.035625 / 0.842877**	**0.037726 / 0.835121**	**0.034964 / 0.856468**	**0.108327 / 0.874185**	**0.113386 / 0.872860**	**0.134970 / 0.773109**

For mean accuracy forgetting, activation geometry again has the lowest MAE among the internal diagnostics under all three available protocols. The aggregated NLL result is less uniform: activation geometry has the lowest internal-metric MAE under current-task holdout, while SVD spectral change has the lower MAE under task-order and double holdouts. Thus the most consistent standalone advantage of activation geometry occurs at the individual-task level and for mean accuracy forgetting, rather than for every possible aggregation of NLL.

D.7 Conditional contribution beyond observed controls

Standalone prediction does not establish that activation geometry contains information beyond other observable properties of the training trajectory. We therefore compare a controls-only Huber fit with the same fit augmented by activation geometry.

For level prediction, the controls are

$$\{\text{stage fraction}, \text{task age}, \text{progress fraction}, \text{parameter norm}, \text{current-task validation NLL}, \text{current-task validation accuracy}\}.$$

No task-identity one-hot variables are included. Because the final two controls use labeled validation behavior, this analysis is a deconfounding test rather than a claim that the predictor is purely internal or label-free.

For each OOF observation, the controls-only and controls-plus-geometry predictions are paired. We define

$$\text{MAE gain} = \text{MAE}_{\text{controls}} - \text{MAE}_{\text{controls+geometry}}, \tag{39}$$

so positive values favor adding activation geometry. Uncertainty intervals use 5,000 paired bootstrap resamples clustered by complete `run_id`. Because there are only six independent run clusters, these intervals should be interpreted as descriptive run-level uncertainty summaries rather than high-powered significance tests.

For individual-task accuracy forgetting, the three factor-holdout protocols all have positive point-estimate gains, but their run-clustered intervals include zero. For individual-task NLL, the clearest incremental improvement appears under old-task holdout, with gain 0.084391 and interval [0.022042, 0.143365]. The current-task gain is also positive in point estimate but its interval slightly crosses zero. Task-order and double holdouts do not provide stable evidence of an incremental gain.

Table 9: Conditional contribution of activation geometry for individual-old-task fixed-anchor forgetting. Positive MAE gain means that adding activation geometry reduces OOF MAE. Intervals are 95% paired run-cluster bootstrap intervals over the six observed complete runs.

Held out	Accuracy gain	95% interval	NLL gain	95% interval
Current task	0.010666	[−0.003980, 0.025860]	0.040074	[−0.001492, 0.083581]
Old task	0.016693	[−0.001668, 0.033245]	0.084391	[0.022042, 0.143365]
Task order	0.010802	[−0.009262, 0.030866]	0.016734	[−0.047760, 0.083912]
Order + seed	−0.019396	[−0.102036, 0.040697]	−0.248652	[−0.804113, 0.147352]

Table 10: Conditional contribution of activation geometry for mean-old-task fixed-anchor forgetting. Old-task holdout is undefined after aggregation. Intervals are 95% paired run-cluster bootstrap intervals over six complete runs.

Held out	Accuracy gain	95% interval	NLL gain	95% interval
Current task	0.000953	[−0.011514, 0.012166]	−0.070038	[−0.122929, −0.011248]
Task order	0.004139	[−0.005469, 0.012598]	−0.005318	[−0.059145, 0.040758]
Order + seed	−0.006767	[−0.095587, 0.079287]	0.046952	[−0.161655, 0.341530]

The double-holdout result is substantially more variable than the other protocols. This is expected from its small fitting set: after excluding both one order and one seed, only two complete runs remain available for fitting in each fold. We therefore use the double split primarily as a stringent stress test rather than as the main basis for inference.

For completeness, Table 10 gives the same conditional comparison after averaging forgetting over old tasks.

The aggregate conditional results are weaker. Mean accuracy gains are close to zero, and under current-task holdout the mean-NLL fit becomes worse after adding activation geometry. This pattern suggests that the main diagnostic value of activation geometry in the present experiment lies in resolving which particular historical task is being damaged, rather than in improving an already controlled estimate of average forgetting across all old tasks.

E FOUR-TASK QWEN BENCHMARK

We use the same three task orders throughout the four-task experiments: Order 1 is DBP–AMZ–YAH–AGN, Order 2 is DBP–AMZ–AGN–YAH, and Order 3 is YAH–AMZ–AGN–DBP. This numbering is reused in the Qwen and Llama tables below.

E.1 OVERALL COMPARISON

Table 11 consolidates the core comparison and the additional Qwen2.5-7B baselines on the same three task orders. Replay-64 provides a complementary comparison because it preserves historical information by retaining labeled examples rather than by constraining the parameter update. Under the fixed 64-example capacity, replay dramatically improves over Sequential LoRA, confirming that historical information is valuable. Nevertheless, it remains below HARD AG-LORA in final average accuracy and has higher forgetting on every task order. Its three-order mean is 77.82% AA, 5.36 points of forgetting, and -5.36 points BWT, compared with 80.96% AA, 0.95 points of forgetting, and -0.36 points BWT for HARD AG-LORA. For Replay-64, BWT is exactly the negative of forgetting in these runs because each historical task reaches its best accuracy immediately after it is learned. O-LoRA and HARD AG-LORA remain close overall, but O-LoRA achieves this by increasing adapter rank with task count whereas HARD AG-LORA keeps rank fixed.

The supplementary methods differ in capacity or implementation and should not all be interpreted as strictly matched comparisons. The InfLoRA-Qwen adaptation (Liang & Li, 2024) uses the full current-task training stream to estimate covariance, initializes a safe input factor, maintains historical subspaces using DualGPM, adds a rank-8 block per task, freezes historical B blocks, and targets `k_proj`/`v_proj`; its effective rank is 32 after four tasks. The evaluated CLoRA configuration

Table 11: Qwen2.5-7B four-task results across three task orders. Accuracy is in percent and forgetting/BWT are percentage points. Orders 1–3 are defined in the text. The upper block contains the core comparison; the lower block contains supplementary methods and controls with different capacity or implementation choices. Bold marks the best observed value within each order and summary row of each comparison block. ± values are sample standard deviations across task orders.

Method	Order	Final AA ↑	Forgetting ↓	BWT ↑	Final rank
Core comparison					
Sequential	1	39.87	55.39	−55.39	8
Replay-64	1	77.65	5.57	−5.57	8
O-LoRA	1	80.37	2.11	−2.06	32
HARD AG-LORA	1	**80.84**	**1.53**	−1.50	8
Sequential	2	24.40	61.24	−61.24	8
Replay-64	2	78.87	3.26	−3.26	8
O-LoRA	2	**81.48**	**0.79**	−0.61	32
HARD AG-LORA	2	80.96	1.29	−1.26	8
Sequential	3	45.92	41.23	−41.23	8
Replay-64	3	76.95	7.25	−7.25	8
O-LoRA	3	80.33	0.69	−0.13	32
HARD AG-LORA	3	**81.09**	**0.04**	**1.67**	8
Sequential	mean ± order std.	36.73±11.10	52.62±10.29	−52.62 ± 10.29	8
Replay-64	mean ± order std.	77.82±0.97	5.36±2.00	−5.36 ± 2.00	8
O-LoRA	mean ± order std.	80.72±0.65	1.20±0.79	−0.93 ± 1.00	32
HARD AG-LORA	mean ± order std.	**80.96±0.12**	**0.95±0.80**	−0.36 ± 1.77	8
Supplementary methods and controls					
InfLoRA	1	**81.81**	**0.54**	−0.54	32
CLoRA	1	65.29	21.62	−21.62	8
Freeze-B	1	79.37	3.62	−3.57	8
REFRESH AG-LORA	1	77.71	4.82	−4.82	8
InfLoRA	2	**81.46**	**0.87**	−0.86	32
CLoRA	2	68.58	17.80	−17.80	8
Freeze-B	2	75.38	8.42	−8.42	8
REFRESH AG-LORA	2	79.51	2.67	−2.67	8
InfLoRA	3	**82.14**	**0.21**	−0.04	32
CLoRA	3	45.90	45.27	−45.27	8
Freeze-B	3	74.54	7.82	−7.82	8
REFRESH AG-LORA	3	81.57	1.12	−1.12	8
InfLoRA	mean ± order std.	**81.80±0.34**	**0.54±0.33**	−0.48 ± 0.41	32
CLoRA	mean ± order std.	59.93±12.26	28.23±14.88	−28.23 ± 14.88	8
Freeze-B	mean ± order std.	76.43±2.58	6.62±2.62	−6.60 ± 2.64	8
REFRESH AG-LORA	mean ± order std.	79.60±1.93	2.87±1.86	−2.87 ± 1.86	8

regularizes one rank-8 A/B pair against a fixed random orthogonal subspace ($k = 256$, $\lambda = 1$) without retaining historical data. Freeze-B retains one rank-8 adapter, freezes B after Task 1, and applies no subspace constraint. Both CLoRA and Freeze-B target `q_proj`/`v_proj`. REFRESH AG-LORA also keeps rank 8 and refreshes the historical activation coordinates at task boundaries. These capacity and target-layer differences limit a controlled interpretation of InfLoRA versus HARD AG-LORA.

InfLoRA attains 81.80% AA and 0.54 points of forgetting averaged across the three orders, compared with 80.96% and 0.95 points for HARD AG-LORA, at the cost of growing to rank 32 and using different target layers. Freeze-B's three-order means are 76.43% AA and 6.62 points of forgetting. The evaluated CLoRA configuration yields 59.93% AA and 28.23 points of forgetting on average, with substantial order variation, while REFRESH AG-LORA reaches 79.60% AA and 2.87 points of forgetting. Here ± denotes sample standard deviation across task orders, not a confidence interval. Replay-64 retains labeled text and executes old-example forwards throughout later training, whereas the activation sketch does not retain labeled text and does not add old-example forwards to every later optimizer step.

Table 12: Final per-task Qwen2.5-7B test accuracy (%). Columns use a fixed dataset order irrespective of training order. Orders 1–3 are defined in the text. Bold marks the best observed accuracy within each training-order block of the core comparison; supplementary methods are reported without additional emphasis.

Method	Order	DBpedia	Amazon	Yahoo	AG News
Core comparison					
Sequential	1	3.74	32.57	32.91	90.26
Replay-64	1	98.49	54.79	65.82	91.50
O-LoRA	1	96.00	**62.12**	**72.45**	90.89
HARD AG-LORA	1	**98.53**	60.97	72.33	**91.53**
Sequential	2	0.00	0.00	72.99	24.61
Replay-64	2	98.04	57.54	73.30	86.61
O-LoRA	2	96.74	**63.43**	**75.51**	90.22
HARD AG-LORA	2	**98.51**	59.53	75.46	**90.36**
Sequential	3	94.05	21.34	13.33	54.95
Replay-64	3	**98.87**	54.21	67.03	87.68
O-LoRA	3	98.67	60.22	71.72	90.68
HARD AG-LORA	3	**98.87**	**62.11**	**72.57**	**90.80**
Supplementary methods and controls					
InfLoRA	1	98.64	61.49	75.58	91.54
InfLoRA	2	98.59	61.17	75.28	90.80
InfLoRA	3	98.96	63.26	74.89	91.43
CLoRA	1	65.29	39.01	67.89	88.97
CLoRA	2	73.37	52.50	74.95	73.51
CLoRA	3	98.91	25.37	12.04	47.29
Freeze-B	1	97.68	55.93	72.29	91.55
Freeze-B	2	87.61	50.72	74.83	88.36
Freeze-B	3	98.37	59.87	63.07	76.84
REFRESH AG-LORA	1	98.21	51.91	69.91	90.83
REFRESH AG-LORA	2	95.99	58.72	74.86	88.46
REFRESH AG-LORA	3	98.95	62.76	73.62	90.93

E.2 FINAL TASK-LEVEL PERFORMANCE

Table 12 consolidates the final task-level accuracies for both comparison blocks. Sequential LoRA learns the final task strongly but earlier tasks collapse, most visibly DBpedia in Orders 1–2 and Yahoo/Amazon in Order 3. Both O-LoRA and HARD AG-LORA retain useful performance on every task.

E.3 MECHANISM ABLATION

We compare Freeze-B alone, a matched-per-layer-rank random projector, the SOFT AG-LORA quadratic penalty, Sequential LoRA, and HARD AG-LORA on Qwen Order 1. All methods use one rank-8 adapter and the shared full-data protocol. These controls separate the effect of freezing the LoRA output factor, merely restricting the update, softly penalizing activation-weighted drift, and protecting activation-informed directions exactly. The complete protocol is specified in Appendix C.

Table 13 reports aggregate and final task-level results in a single view. Freeze-B alone improves final AA from 39.87% to 79.37% and reduces forgetting from 55.39 to 3.62 points. The activation-informed projector further improves AA by 1.47 points and forgetting by 2.09 points. A matched-rank random projector reaches 77.11% AA and 6.33 points of forgetting, showing that the full gain is not explained merely by freezing B or shrinking the available update space. The evaluated SOFT AG-LORA implementation reaches 70.13% AA and 16.28 points of forgetting; this single-order result does not establish that other penalty weights or optimization settings would behave identically. Random projection particularly harms Yahoo retention: its final accuracy is 64.55%, compared with 72.29% for Freeze-B and 72.33% for HARD AG-LORA.

Table 13: Qwen Order-1 mechanism ablation on DBP–AMZ–YAH–AGN. All methods use one rank-8 adapter. Aggregate metrics and final task-level accuracies are shown together; bold marks the best observed value in each column except AG News, where Freeze-B is marginally higher than HARD AG-LORA.

Method	Final AA ↑	Forgetting ↓	BWT ↑	DBP	AMZ	YAH	AGN
Sequential / no projection	39.87	55.39	−55.39	3.74	32.57	32.91	90.26
SOFT AG-LORA	70.13	16.28	−16.28	76.41	48.46	64.97	90.67
Matched-rank random projection	77.11	6.33	−6.29	96.93	56.82	64.55	90.14
Freeze-B only	79.37	3.62	−3.57	97.68	55.93	72.29	**91.55**
HARD AG-LORA	**80.84**	**1.53**	−1.50	**98.53**	**60.97**	**72.33**	91.53

Implementation diagnostics. The implementation diagnostics support the intended comparison. All 56 LoRA layers have zero B drift in Freeze-B and random-projection runs. Random projection has constraint residuals around 10^{-8} and uses mean per-layer ranks 42.6, 46.8, and 48.4 on Tasks 2–4, matching the activation-informed projector's rank. Yet those random directions cover only 1.21–1.34% of historical activation energy. This provides direction-specific evidence for the activation-informed projector on the reported order.

F FOUR-TASK LLAMA REPLICATION

The backbone replication uses Meta-Llama-3.1-8B-Instruct on the same four task-aware classification datasets and task orders as the Qwen comparison. Sequential LoRA and HARD AG-LORA retain one rank-8 adapter; O-LoRA adds one rank-8 block per task and reaches rank 32.

Replication configuration. The Llama runs use BF16 without quantization, the full training set for one epoch per task, AdamW with zero weight decay, a 10^{-3} per-task learning rate, mini-batch size 16 with gradient accumulation 4, and LoRA on `q_proj`/`v_proj` with rank 8, $\alpha = 32$, and dropout 0.1. The base and reservoir seeds are 0 and 97, respectively; the geometry uses the same 256-source/64-row protocol as Qwen. Evaluation uses 7,600 examples per task after each stage with greedy decoding for 12 tokens. Sequential uses a different, explicitly scheduled learning-rate protocol in the reported comparison.

F.1 ORDER-LEVEL RESULTS

Table 14 reports the order-level results. The scheduled-LR Sequential reference leads all three metrics on Order 3. Within the matched fixed-10^{-3} pair, HARD AG-LORA leads O-LoRA on the first two orders and in the three-order aggregate.

F.2 FINAL TASK-LEVEL PERFORMANCE

Table 15 gives the final task-level accuracies. The largest HARD AG-LORA gains over O-LoRA occur on Amazon in the first two orders; Order 3 instead favors Sequential LoRA across all four tasks.

Table 14: Meta-Llama-3.1-8B four-task results. Accuracy is in percent and forgetting/BWT are percentage points. Orders 1–3 follow the definitions used for Qwen. Summary rows report the recorded three-order aggregates; order-level entries are displayed to two decimals. Bold marks the best observed value within each order and in the summary rows. Sequential uses a different, explicitly scheduled learning-rate protocol.

Method	Order	Final AA ↑	Forgetting ↓	BWT ↑	Final rank
Sequential (scheduled LR)	1	74.04	10.28	−8.89	8
O-LoRA	1	73.49	9.98	−9.15	32
HARD AG-LORA	1	**76.62**	**6.61**	−5.36	8
Sequential (scheduled LR)	2	74.92	9.08	−7.69	8
O-LoRA	2	72.61	11.20	−10.28	32
HARD AG-LORA	2	**79.09**	**3.64**	−2.34	8
Sequential (scheduled LR)	3	**81.24**	**1.34**	−1.00	8
O-LoRA	3	79.33	3.42	−3.42	32
HARD AG-LORA	3	79.81	3.51	−3.32	8
Sequential (scheduled LR)	three-order mean	76.73	6.90	−5.86	8
O-LoRA	three-order mean	75.14	8.20	−7.62	32
HARD AG-LORA	three-order mean	**78.51**	**4.58**	−3.67	8

Table 15: Final per-task Meta-Llama-3.1-8B test accuracy (%). Columns use a fixed dataset order. Orders 1–3 follow the definitions used for Qwen. Bold marks the best observed accuracy within each training-order block; Sequential uses its scheduled-LR protocol.

Order	Method	DBpedia	Amazon	Yahoo	AG News
1	Sequential (scheduled LR)	94.96	38.51	**71.49**	91.20
	O-LoRA	**95.34**	36.75	70.58	**91.30**
	HARD AG-LORA	95.17	**49.76**	70.74	90.80
2	Sequential (scheduled LR)	92.99	41.46	**75.99**	**89.24**
	O-LoRA	91.64	34.42	75.55	88.83
	HARD AG-LORA	**95.13**	**57.82**	75.22	88.17
3	Sequential (scheduled LR)	**99.01**	**62.34**	**73.03**	**90.58**
	O-LoRA	98.96	58.21	69.82	90.33
	HARD AG-LORA	98.72	61.38	69.70	89.45

G RETENTION TRAJECTORIES

Table 16 places the complete Order-1 HARD AG-LORA trajectories for Qwen and Llama side by side at the same stage-by-task resolution. In each panel, row s evaluates every task learned through stage s. The archived matrices also contain all scheduled-Sequential and HARD AG-LORA Llama trajectories and O-LoRA Order 1; O-LoRA Orders 2–3 are partial and retain only the learned-task diagonal and final row.

Table 16: Order-1 stagewise test accuracy (%) for HARD AG-LORA across the two backbones. Dashes denote tasks not yet learned.

Stage just completed	DBpedia	Amazon	Yahoo	AG News
(a) Qwen2.5-7B				
DBpedia	98.64	–	–	–
Amazon	98.74	62.25	–	–
Yahoo	98.43	61.03	75.42	–
AG News	98.53	60.97	72.33	91.53
(b) Meta-Llama-3.1-8B-Instruct				
DBpedia	94.63	–	–	–
Amazon	98.38	62.74	–	–
Yahoo	91.49	57.80	74.37	–
AG News	95.17	49.76	70.74	90.80

H FIFTEEN-TASK QWEN EXTENSION

We extend the Qwen2.5-7B evaluation to a 15-task sequence consisting of Yelp, Amazon, MNLI, CB, COPA, QQP, RTE, IMDB, SST-2, DBpedia, AG News, Yahoo, MultiRC, BoolQA, and WiC, in that order. After each task is learned, every task seen so far is evaluated. Thus, each complete trajectory contains $1 + \cdots + 15 = 120$ stage–task cells. The results below use the completed seed-0 Hard AG-LoRA and Sequential LoRA trajectories in the long15 appendix artifacts. They are a descriptive long-horizon stress test rather than a multi-seed inferential comparison. Sequential LoRA is initialized from the Hard AG-LoRA Task-1 adapter, so the two trajectories share the initial anchor.

The source runner's endpoint forgetting is historical-best forgetting, matching the forgetting definition used by the other Qwen tables. For completeness, a separate table below reports fixed-anchor forgetting as an additional diagnostic; it is not the primary endpoint or the standard cross-table forgetting metric. For task i, the fixed anchor is its accuracy immediately after task i is learned. A negative fixed-anchor value denotes a final improvement.

At the final stage, Hard AG-LoRA obtains 68.44% mean accuracy on the 14 old tasks and 64.42% on WiC, whereas Sequential LoRA obtains 10.14% and 53.45%, respectively. The final seen-task accuracy gap is therefore 55.15 percentage points in this paired seed-0 trajectory.

The complete 240-row accuracy/NLL matrix (two runs times 120 cells), stagewise summaries, and task-level acquisition/final NLL values are provided as the machine-readable appendix artifacts. These long-sequence files contain only accuracy and NLL trajectories; activation/Fisher/SVD metric-only OOF mappings remain sourced from the separate metric-analysis experiment.

Table 17: Qwen2.5-7B 15-task long-sequence endpoints. Accuracy is in percent, and forgetting/BWT are percentage points. Forgetting uses the same historical-best definition as the other Qwen tables.

Method	Final AA ↑	Hist.-best forgetting ↓	BWT ↑	Final NLL ↓	Rank
Hard AG-LoRA	68.18	17.81	−16.84	0.541	8
Sequential	13.03	73.06	−72.10	3.096	8

Table 18: Additional fixed-anchor diagnostic for the 15-task sequence. This quantity is reported separately from the standard historical-best forgetting used in the endpoint tables. Accuracy and forgetting are reported in percent and percentage points, respectively.

Task	Hard AG final	Sequential final	Hard AG-LoRA Forgetting	Sequential Forgetting
Yelp	62.53	0.00	−3.92	58.61
Amazon	55.61	0.00	9.33	60.68
MNLI	62.30	0.00	24.32	85.41
CB	50.00	0.00	37.50	89.29
COPA	89.00	0.00	9.00	96.00
QQP	73.96	36.12	11.51	27.97
RTE	56.68	0.00	30.32	86.64
IMDB	37.37	0.08	59.29	94.75
SST-2	82.45	0.00	13.07	94.50
DBpedia	95.96	0.00	2.68	98.49
AG News	79.41	0.00	11.54	91.00
Yahoo	71.91	0.00	3.51	75.29
MultiRC	67.90	43.42	15.18	32.22
BoolQA	73.15	62.39	12.42	18.56
WiC	64.42	53.45	0.00	0.00

I Related Work

Parameter-efficient adaptation. Parameter-efficient fine-tuning (PEFT) adapts frozen pretrained models through a small trainable state, including bottleneck adapters (Houlsby et al., 2019), continuous prefixes or prompts (Li & Liang, 2021; Lester et al., 2021), and low-rank weight updates. LoRA represents each update as a low-rank product (Hu et al., 2022); AdaLoRA allocates the rank budget by importance (Zhang et al., 2023), while QLoRA reduces training memory by backpropagating through a quantized backbone (Dettmers et al., 2023). These methods make adaptation cheaper on a fixed dataset, but do not specify how a shared adapter should preserve behavior when tasks arrive sequentially: bounding the update rank does not prevent later updates from changing earlier responses.

Continual learning and historical state. Continual-learning methods differ primarily in what they retain from the past. Replay preserves examples or generates pseudo-examples, as in GEM, LAMOL, Continual-T0, and LFPT5 (Lopez-Paz & Ranzato, 2017; Sun et al., 2020; Scialom et al., 2022; Qin & Joty, 2022). Regularization preserves parameter importance or reference outputs, as in EWC and Learning without Forgetting (Kirkpatrick et al., 2017; Li & Hoiem, 2018); projection methods instead summarize historical gradients, inputs, or representations, as in OGD, OWM, and GPM (Farajtabar et al., 2020; Zeng et al., 2019; Saha et al., 2021). A third line isolates tasks architecturally through pruning masks, adapters, or prompts (Mallya & Lazebnik, 2018; Madotto et al., 2021; Zhu et al., 2022; Razdaibiedina et al., 2023). ConPET and SAPT extend this approach with PET-module selection or shared attention between learning and selection (Song et al., 2023; Zhao et al., 2024). Replay retains rich historical supervision but repeatedly evaluates old objectives; task isolation avoids overwriting but usually introduces task-indexed state; geometric summaries are more compact, but their effectiveness depends on which subspace is protected.

Continual low-rank adaptation. Recent work combines LoRA with continual learning across domains. CoLoR studies domain-incremental transformers (Wistuba et al., 2023), C-LoRA regularizes sequential diffusion customization (Smith et al., 2024), Online-LoRA addresses task-free vision streams (Wei et al., 2025), and CL-LoRA separates shared and task-specific branches for class-incremental vision (He et al., 2025). These studies establish the breadth of continual low-rank adaptation, although their task assumptions, inference protocols, and capacity accounting differ from our sequential language setting.

The closest methods differ in how they define and retain safe update directions. O-LoRA freezes task-specific adapters and adds new orthogonal low-rank blocks (Wang et al., 2023); InfLoRA adds a branch selected from input and old-gradient subspaces (Liang & Li, 2024). Both therefore increase adapter capacity with the task sequence. N-LoRA reduces collisions between task parameters (Yang et al., 2025), Controlled LoRA regularizes LoRA null-space directions (Lu et al., 2025), and SLoRA filters update components using their similarity to the base-model subspace (Yang et al., 2026). These methods derive protection primarily from parameter or base-model geometry. Meta-UCF follows another route, using task embeddings and a shared hypernetwork to generate fixed-rank LoRA updates (Xiao et al., 2026). Collectively, they show that update geometry matters, but do not characterize which information about historical inputs is required to determine an update's historical response drift.

Historical functional geometry. An update direction cannot answer this question by itself. The same effective LoRA displacement can induce zero or maximal layer-local response drift, depending on the historical activation distribution. Replay retains enough information to revisit historical empirical risk, while OWM and GPM show that past inputs or representations can define protected subspaces (Zeng et al., 2019; Saha et al., 2021). We make this connection explicit for continual LoRA: the uncentered second moment of historical layer inputs is sufficient for the local quadratic response drift studied here, so retaining the full labeled dataset is unnecessary for this quantity. This statement concerns direct layer responses and does not identify end-to-end forgetting.

In contrast to growing-branch methods such as O-LoRA and InfLoRA, AG-LoRA updates one fixed-rank adapter. Unlike parameter-derived constraints in N-LoRA, Controlled LoRA, and SLoRA, its protected space is induced by historical activations; unlike Meta-UCF, it does not generate adapters from a task-conditioned hypernetwork. HARD AG-LoRA converts temporary prompt-only examples into a capped, task-count-independent activation sketch and constrains the task-start displacement of the effective LoRA weight. The stored-versus-refreshed comparison further asks whether this historical reference should remain fixed when a task is learned or move with the current model. Our focus is therefore not orthogonality alone, but which compressed historical information suffices to constrain layer-local response drift under fixed capacity.