LATENTKV: PRESERVING REASONING ABILITY IN CONTINUAL LATENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent reasoning carries intermediate computation in continuous hidden states rather than generating an explicit chain of thought, solving problems with a few continuous thoughts and far fewer reasoning tokens. Continually adapting such models to new tasks while preserving previously acquired reasoning abilities poses a further challenge: because the states carry the computation, the final output does not directly reveal how internal reasoning changes as the model is updated. Answer-based replay shapes these states only through the answer loss and does not distinguish how different state changes affect it; matching historical states constrains every state component to its historical value and may over-restrict adaptation to the new task. The key question is therefore which state perturbations affect the old-task answer loss locally. Through local perturbation analysis, we find that (i) projecting KV-state changes onto the sensitivity direction of the answer loss yields a signal correlated with the change in old-task loss, and (ii) sensitivity computed on the current model predicts this change more accurately than sensitivity inherited from the historical model. We propose LATENTKV, a continual-learning method for latent reasoning that combines historical-KV replay with answer-sensitive state constraints using a single LoRA adapter and a fixed replay budget. For replay, LATENTKV guides old samples with a surrogate gradient conditioned on stored historical KV. For constraint, it periodically refreshes both the reference KV and the answer-loss sensitivity, and applies a symmetric projection penalty that limits state change along answer-sensitive directions. When continually adapting from GSM8K to SVAMP, LATENTKV raises old-task latent accuracy from 26.80% to 52.13% over sequential fine-tuning, while achieving 69.09% accuracy on the new task, a decrease of 3.89 percentage points. Code is available at `https://anonymous.4open.science/r/latentkv/`.

1 INTRODUCTION

Chain-of-thought (CoT) improves performance on multi-step problems by generating intermediate steps, but unrolling these steps token by token also increases the output length and decoding cost (Wei et al., 2022). Latent reasoning carries intermediate computation in continuous hidden states, offering an alternative path toward more compact reasoning. COCONUT feeds hidden states back as input for subsequent reasoning (Hao et al., 2025), and CODI distills an explicit chain of thought into a few continuous thoughts (Shen et al., 2025). We consider sequential tasks that continually update the same LoRA adapter, without adding parameters or modules for each task. The parameters changed by new-task learning also support the model's existing continuous computation.

The Problem. How can a latent reasoning model continually learn new tasks while preserving its existing reasoning abilities?

Latent reasoning feeds each step's hidden state into the next, so the content of this feedback can matter as much as the number of steps. On a model adapted from ProsQA to GSM8K, we retain all six steps and their key–value (KV) cache updates but replace the feedback inputs with a fixed vector. New-task GSM8K accuracy drops from 60.60% to 25.00% (Section 5.5). Keeping the same number of steps is therefore insufficient: what the model passes between steps matters for its answer.

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.

When the model learns a new task, both the intermediate states and the way subsequent computation uses them change. Answer replay supervises the model with the answer loss on old samples (Chaudhry et al., 2019b) but does not explicitly constrain how the internal states evolve. Directly matching historical states may restrict changes that barely affect old-task answers yet are useful for adapting to the new task. We focus on KV states at continuous-thought positions, which later steps and answer decoding read through attention. KV provides a concrete interface for both replay and state protection. Stored KV retains the context of earlier computation; gradients with respect to KV describe how perturbing that context affects the answer locally.

We examine this sensitivity before and after diagnostic updates on the new task, comparing the KV projection signal with the old-task answer-loss change. Figure 1 distinguishes historical, current, and periodically cached sensitivities. At the four shown checkpoints, current sensitivity has higher correlation than historical sensitivity, but the cached direction can lag behind both. The current-versus-historical ordering also appears on an Answer Replay trajectory (Table 11). Historical sensitivity can miss changes in how the current model uses its states. This motivates revisiting the protection signal; separate component comparisons test refreshing its reference point and direction together.

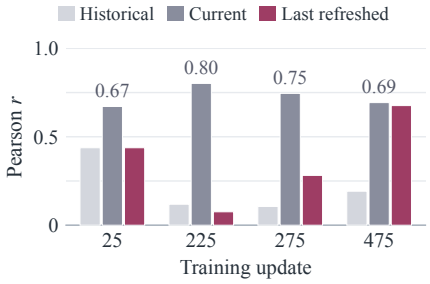

Figure 1: Pearson correlation with old-task loss change. Protocol B, GSM8K to ProsQA, seed 42, 64 probes. Last refreshed is the cached signal used by the constraint.

We propose LATENTKV, which separates the historical information used for replay from the reference used to constrain ongoing state changes. Surrogate-gradient replay keeps the current model's forward prediction but constructs its backward update through a proxy computation conditioned on fixed historical KV. A separate constraint penalizes the normalized squared projection of KV displacement onto an answer-sensitive direction, leaving orthogonal displacement unpenalized by this term. The reference follows adaptation, so the penalty limits displacement since each sample's latest refresh rather than its entire departure from the old model. Fixed historical KV continues to supply the replay context. Both components operate only during training. Deployment uses the base model with the updated LoRA adapter.

We evaluate LATENTKV on both old-task retention and new-task learning (Section 5). On GSM8K-to-ProsQA adaptation, LATENTKV achieves 69.62% accuracy averaged across the two tasks. It exceeds Answer Replay by 4.79 percentage points and DER++ (replay with historical-logit matching) by 4.23 points. Relative to Answer Replay, new-task accuracy improves by 10.80 points while old-task accuracy is 1.21 points lower; relative to DER++, old-task accuracy improves by 7.78 points while new-task accuracy remains comparable.

① We examine **what changes inside a continual latent reasoner**: feedback interventions reveal dependence on continuous computation, and KV diagnostics track changes in local answer sensitivity.
② We separate **historical context from the state-protection reference**, combining fixed-KV surrogate replay with a jointly refreshed projection penalty and leaving inference unchanged.
③ We demonstrate **improved retention–adaptation trade-offs** against Answer Replay and DER++, and identify the contribution of joint refreshing through component comparisons.

2 RELATED WORK

Latent reasoning. Implicit chain-of-thought methods internalize textual steps through distillation (Deng et al., 2024; 2025), and later work studies how to construct and compress continuous thoughts (Wei et al., 2026; Xu et al., 2025; Su et al., 2025). COCONUT feeds the last hidden state back as the next input (Hao et al., 2025), and CODI self-distills explicit reasoning into continuous thoughts (Shen et al., 2025). Our 3B experiments start from a CODI checkpoint. Additional steps alone need not imply sequential reasoning (Pfau et al., 2024; Liang & Pan, 2026; Dilgren & Wiegreffe, 2026), so we first verify that the continuous feedback matters for the task (Section 5.5) before asking how to preserve it.

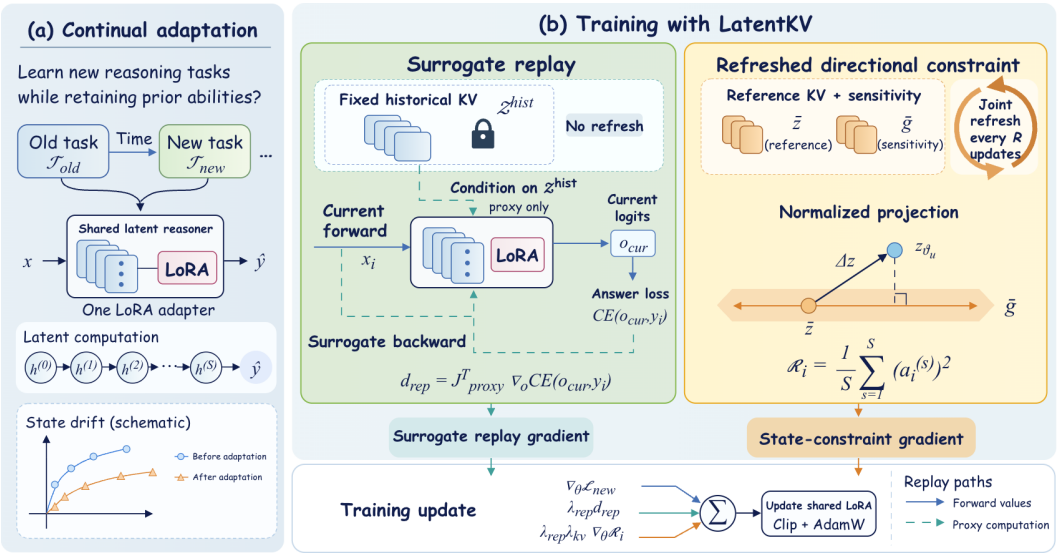

Figure 2: Overview of LATENTKV. (a) Sequential tasks update one LoRA adapter, changing the latent computation (schematic). (b) Replay combines current forward values with historical-KV-conditioned surrogate gradients. Jointly refreshed reference KV and sensitivity define a normalized, symmetric projection penalty. The weighted new-task, replay, and constraint gradients are combined, clipped, and passed to AdamW.

Continual adaptation and latent memory. Forgetting in language models can reflect disrupted task alignment rather than lost knowledge (Zheng et al., 2025). Existing approaches to continual latent reasoning add structure: typed latent operators in TYLER (Lin et al., 2026), generated latent memories in MEMGEN (Zhang et al., 2025), and experiential memories, latent contexts, or task prompts in related methods (Dorovatas et al., 2026; Anand et al., 2026; Razdaibiedina et al., 2023). We instead examine how continuous computation changes when all tasks update the same LoRA adapter, without adding an inference module.

Replay and distillation. Beyond replaying old samples (Chaudhry et al., 2019b), many methods add a signal from the old model, such as historical logits (Buzzega et al., 2020), output distillation (Li & Hoiem, 2017), generated samples (Sun et al., 2020), or a demonstration-conditioned teacher (Shenfeld et al., 2026). Latent Replay and REMIND store intermediate activations and feed them into later layers (Pellegrini et al., 2020; Hayes et al., 2020). Latent Replay keeps the cache valid by slowing learning below the cached layer. LATENTKV also stores intermediate states, but uses them to condition a surrogate backward path. The forward prediction comes from the current model, whose shared adapter continues to change across tasks.

Importance and state constraints. Regularization methods differ in what they protect. EWC and SI weight parameter changes by estimated importance (Kirkpatrick et al., 2017; Zenke et al., 2017), gradient and subspace methods restrict the update direction (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a; Wang et al., 2023; Liang & Li, 2024), and GPM derives its protected subspace from activations (Saha et al., 2021). PODNet and SEEKR constrain pooled features and selected attention distributions instead (Douillard et al., 2020; He et al., 2024). Our constraint also acts on internal states, but penalizes only the projection of the KV displacement onto the answer-loss sensitivity. Refreshing protection information has precedents: Online EWC updates its importance estimate at task boundaries (Schwarz et al., 2018), whereas LATENTKV refreshes the reference KV and the sensitivity together every R updates within a task.

3 PROBLEM FORMULATION

Continual learning setting. Tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$, with training data $\mathcal{D}_t$, arrive sequentially. All tasks update the same LoRA adapter (Hu et al., 2021) on a frozen base model; θ_t denotes its parameters after task t. Our main experiments use two tasks, with three-task sequences in Appendix E.3. Learning $\mathcal{T}_2$ uses $\mathcal{D}_2$ and a fixed replay subset $\mathcal{M}_1 \subseteq \mathcal{D}_1$, $|\mathcal{M}_1| \leq B$. The budget B is the number of retained samples. Selection and replay order are given in Appendix A.

Latent reasoning and KV states. For a question x, the model takes S continuous thought steps before generating an answer. With L Transformer layers, define

$$Z_\theta^{(s)}(x) = \left\{ K_\theta^{(s,\ell)}(x),\, V_\theta^{(s,\ell)}(x) \right\}_{\ell=1}^{L}, \qquad Z_\theta(x) = \left\{ Z_\theta^{(s)}(x) \right\}_{s=1}^{S}, \tag{1}$$

where K and V are the keys and values at thought position s. Later thought steps and answer decoding read these states through attention. Our constraint covers thought positions in all layers, excluding the question prefix. For each replay sample (x_i, y_i), LATENTKV additionally stores $Z_i^{\text{hist}} = Z_{\theta_1}(x_i)$ and holds it fixed during adaptation. For a common KV width across layers, this cache contains $2|\mathcal{M}_1|SLd_{\text{kv}}$ scalars, where d_{kv} is the number of KV heads times the head dimension. Reference KV and sensitivities add storage. Precision and byte counts are given in Appendix F.3.

Answer loss and sensitivity. At step s, replace only the KV slice inserted into the cache with z, holding the parameters and other interface inputs fixed, then continue the remaining thought steps and answer decoding. For a target $y = (y_1, \ldots, y_m)$, the token-averaged loss is

$$\mathcal{L}_{\text{ans}}^{(s)}(z, \theta; x, y) = -\frac{1}{m} \sum_{j=1}^{m} \log p_\theta^{(s)}(y_j \mid x, z, y_{<j}) \,. \tag{2}$$

For any s, substituting $z = Z_\theta^{(s)}(x)$ recovers the ordinary answer loss $\mathcal{L}_{\text{ans}}(\theta; x, y)$. Its local sensitivity at that interface is

$$g_\theta^{(s)}(x, y) = \nabla_z \mathcal{L}_{\text{ans}}^{(s)}(z, \theta; x, y)\Big|_{z = Z_\theta^{(s)}(x)} \,. \tag{3}$$

A diagnostic update $\delta\theta$ induces $\delta Z^{(s)} = Z_{\theta+\delta\theta}^{(s)}(x) - Z_\theta^{(s)}(x)$. Because states across steps are computationally dependent and parameters also affect answers through other paths, $\sum_s \langle g_\theta^{(s)}, \delta Z^{(s)} \rangle$ is a directional proxy, not a first-order expansion of the full loss change (Section 4.1).

Evaluation metrics. Let $A_{t,i}$ be accuracy on task i's designated evaluation set after learning task t. Unless stated otherwise, evaluation uses S thought steps and greedy decoding, including after explicit CoT training. For two tasks, old-task retention $A_{2,1}$ and new-task performance $A_{2,2}$ give $A_{\text{avg}} = (A_{2,1} + A_{2,2})/2$. Longer sequences average over all learned tasks. Where initial scores are available, forgetting $F = A_{1,1} - A_{2,1}$ is computed per seed on the same split and decoding protocol before averaging and taking the standard deviation; negative values indicate improvement (Table 16).

4 METHOD

LATENTKV combines historical-KV-conditioned surrogate-gradient replay with a jointly refreshed directional constraint (Figure 2(b)). During second-task adaptation, ϑ_u denotes the shared LoRA parameters before update u, with $\vartheta_0 = \theta_1$; θ_t still denotes task endpoints.

4.1 ANSWER-SENSITIVE DIRECTIONS AND LOCAL DIAGNOSIS

At a checkpoint θ, a diagnostic update $\delta\theta$ induces the state changes defined in Section 3 and the loss change $\delta\mathcal{L}_i = \mathcal{L}_{\text{ans}}(\theta + \delta\theta; x_i, y_i) - \mathcal{L}_{\text{ans}}(\theta; x_i, y_i)$. For the same changes, we compare three sensitivity signals:

$$q_i(\psi) = \sum_{s=1}^{S} \left\langle g_\psi^{(s)}(x_i, y_i),\, \delta Z_i^{(s)} \right\rangle, \qquad \psi \in \{\theta, \theta_1, \vartheta_r\}. \tag{4}$$

These give $q_{\mathrm{cur},i}$, $q_{\mathrm{hist},i}$, and $q_{\mathrm{rec},i}$, respectively; ϑ_r is the most recent refresh checkpoint on a LATENTKV trajectory. The fixed probes lie outside the replay set, with sensitivities cached separately at the training refresh cadence. At each checkpoint, Pearson and Spearman correlations with $\delta\mathcal{L}_i$ measure the association across probes (Section 5.3; Appendix D). Correlation measures the information in the projection signal; the component comparisons test its use as a training penalty.

4.2 SURROGATE-GRADIENT REPLAY CONDITIONED ON HISTORICAL KV

Surrogate replay lets the current model determine its prediction while retaining historical context in the update. A current path and a proxy path use the same ϑ_u and share the question-prefix computation. At each thought step, the proxy path mixes its newly produced KV $\widehat{Z}_i^{(s)}$ with the fixed historical state:

$$\widetilde{Z}_i^{(s)} = (1-\alpha_u)\widehat{Z}_i^{(s)} + \alpha_u \,\mathrm{sg}\Big(Z_i^{\mathrm{hist},(s)}\Big), \tag{5}$$

where sg stops gradients. Only KV at continuous thought positions is mixed.

Splicing values and gradients. Let $h_{\mathrm{cur}}^{(s)}, \widehat{h}^{(s)}$ be the current and proxy hidden states, and $o_{\mathrm{cur}}, o_{\mathrm{proxy}}$ their answer logits. We splice both as

$$\begin{aligned} \widetilde{h}^{(s)} &= \mathrm{sg}\Big(h_{\mathrm{cur}}^{(s)}\Big) + \widehat{h}^{(s)} - \mathrm{sg}\Big(\widehat{h}^{(s)}\Big), \\ \widetilde{o} &= \mathrm{sg}(o_{\mathrm{cur}}) + o_{\mathrm{proxy}} - \mathrm{sg}(o_{\mathrm{proxy}}). \end{aligned} \tag{6}$$

The next proxy step receives the current hidden-state value with proxy gradients, while its cache retains the mixed KV. The token-averaged replay loss $\mathcal{L}_{\mathrm{rep}} = \mathrm{CE}(\widetilde{o}, y_i)$ has the same value as ordinary answer replay, with surrogate gradient $d_{\mathrm{rep}} = J_{\mathrm{proxy}}^{\top}\nabla_o \mathrm{CE}(o_{\mathrm{cur}}, y_i)$. Here J_{proxy} is the answer-logit Jacobian with respect to LoRA parameters supplied by the spliced graph. This straight-through approximation generally differs from the true gradient of the normal forward answer loss.

Releasing the historical context. As adaptation proceeds, we gradually release the historical context with $\alpha_u = \frac{1}{2}[1+\cos(\pi u/(U-1))]$, for $u = 0, \ldots, U-1$ and $U > 1$. At $\alpha_0 = 1$, thought-position KV comes entirely from the fixed cache, cutting its parameter-gradient path. Replay gradients arise from the question prefix and answer decoding under that context. The historical weight reaches zero only at the final update. With dropout disabled, both paths then coincide up to numerical precision and the replay component reduces to standard answer replay. After the shared prefix, replay runs a current pass without a backward graph and a differentiable proxy pass, both through thought steps and answer decoding. The constraint requires a separate differentiable current-state pass. Periodic refreshes also compute KV and sensitivities. Appendix D.3 examines the resulting gradient directions; Section 5.5 gives the measured cost.

4.3 JOINTLY REFRESHED DIRECTIONAL PROJECTION CONSTRAINT

The penalty acts on the normal latent trajectory. At refresh checkpoint ϑ_r, its reference point and direction are computed together:

$$\bar{Z}_i^{(s)} = \mathrm{sg}\Big(Z_{\vartheta_r}^{(s)}(x_i)\Big), \qquad \bar{g}_i^{(s)} = \mathrm{sg}\Big(g_{\vartheta_r}^{(s)}(x_i, y_i)\Big). \tag{7}$$

Both remain fixed between refreshes. We initialize them for all replay samples, then refresh the next segment's samples every R updates. Historical KV for surrogate replay stays fixed. Refreshing resets each affected sample's displacement to zero. The penalty therefore constrains recent changes rather than all change since the old task; the fixed historical cache continues to supply that older context to replay.

Flatten current KV, reference KV, and sensitivity in the same order across all layers and KV heads. For displacement $\Delta_i^{(s)} = z_{\vartheta_u}^{(s)}(x_i) - \bar{z}_i^{(s)}$, the normalized projection is

$$a_i^{(s)} = \frac{\Big\langle \bar{g}_i^{(s)}, \Delta_i^{(s)} \Big\rangle}{\max(\|\bar{g}_i^{(s)}\|_2, \varepsilon)\ \max(\mathrm{RMS}(\bar{z}_i^{(s)}), \varepsilon)}, \tag{8}$$

Algorithm 1 LatentKV adaptation

Require: Replay set $\mathcal{M}_1$ with fixed Z^{hist}, initial adapter θ_1
1: Initialize $\vartheta_0 = \theta_1$ and reference KV and sensitivity for $\mathcal{M}_1$.
2: **for** $u = 0, \ldots, U-1$ **do**
3: Take a new-task batch and the next replay sample i; set cosine weight α_u.
4: At $u = R, 2R, \ldots$, refresh both references for the next replay segment.
5: Compute current values; run the proxy with KV mixing (5).
6: Splice at each thought step and at the logits (6) to obtain d_{rep}.
7: Compute $\mathcal{R}_i$ (9) in a separate current-state pass.
8: Form d_u (10), clip its norm at 2.0, and take an AdamW step.
9: **end for**

where $\text{RMS}(v) = \|v\|_2/\sqrt{n_s}$ for a state with n_s scalars. Sensitivity normalization removes gradient magnitude as an implicit weight across samples and steps. Reference RMS adjusts for state scale. The symmetric penalty is

$$\mathcal{R}_i(\vartheta_u) = \frac{1}{S}\sum_{s=1}^{S}\left(a_i^{(s)}\right)^2. \tag{9}$$

Squaring each step separately avoids cross-step cancellation. Each term is a quadratic form of rank at most one, with no penalty on orthogonal displacement. The penalty is symmetric in sign, so it can also restrict locally beneficial changes. The reference KV and sensitivity are detached, so the penalty backpropagates only through current KV and requires no second-order derivatives.

Component variants. Splice-only sets $\lambda_{\text{kv}} = 0$, and Replay + projection pairs the jointly refreshed penalty with standard answer replay. With surrogate replay retained, Fixed reference keeps both quantities at θ_1, and Refresh KV only and Refresh sensitivity only update one of them. The full method, Joint refresh, updates both.

4.4 Optimization

Algorithm 1 summarizes training. With the protocol-specific new-task loss $\mathcal{L}_{\text{new}}$ (Section 5.1), the gradient signal is

$$d_u = \nabla_\theta \mathcal{L}_{\text{new}} + \lambda_{\text{rep}} d_{\text{rep}} + \lambda_{\text{rep}}\lambda_{\text{kv}} \nabla_\theta \mathcal{R}_i, \tag{10}$$

evaluated at $\theta = \vartheta_u$. The default uses $R = 50$, $\lambda_{\text{rep}} = 0.02$, $\lambda_{\text{kv}} = 0.05$, and $\varepsilon = 10^{-8}$. The main comparison uses $\lambda_{\text{kv}} = 0.005$, with the other settings unchanged (Section 5.2). The effective penalty coefficient is $\lambda_{\text{rep}}\lambda_{\text{kv}}$; its gradient contribution also depends on the scale of $\nabla_\theta \mathcal{R}_i$. At inference, only the updated model is needed, with no historical cache, reference sensitivities, or extra computation branch.

5 Experiments

5.1 Setup

The 3B model starts from a six-step CODI checkpoint (Shen et al., 2025) on Llama-3.2-3B (Grattafiori et al., 2024); Qwen3-4B/8B/14B (Yang et al., 2025) use latent-curriculum initialization. We study arithmetic reasoning on GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021), and logical reasoning on ProsQA (Hao et al., 2025) and ProntoQA (Saparov & He, 2022). All latent-model comparisons update one shared adapter. Replay-based methods retain 200 old examples.

Training and comparison. CoT-supervised adaptation (Protocol A) trains on textual reasoning but evaluates through six continuous steps. Latent-curriculum adaptation (Protocol B) trains through progressively longer continuous computation and supplies the feedback and sensitivity analyses. Methods share initialization, training data, and update budgets within each comparison. Baselines include Answer Replay, parameter regularization with EWC (Kirkpatrick et al., 2017), historical-logit replay with DER++ (Buzzega et al., 2020), and selected-attention distillation with SEEKR (He

et al., 2024). Fixed-default comparisons also include Sequential FT and component-wise KV-L2 matching. Appendix A specifies these adaptations and training schedules.

Evaluation. We measure old- and new-task accuracy after adaptation and weight the tasks equally. Multi-seed entries give the mean and sample standard deviation; bold marks the largest value, not statistical significance. For the main ProsQA comparison, each method selects among three coefficients on seed-11 development data, then fixes its setting across seeds and test evaluation (Appendix B).

5.2 Retention and Adaptation

Answer Replay and DER++ favor different parts of the retention–adaptation trade-off (Table 1). Relative to Answer Replay, LatentKV learns ProsQA better: new-task accuracy improves by 10.80 points, with a 1.21-point reduction on old GSM8K. Relative to DER++, it retains 7.78 points more old-task accuracy at similar new-task accuracy. The resulting two-task average is 69.62%, above Replay and DER++ by 4.79 and 4.23 points. Old GSM8K accuracy also exceeds each run's starting point, rising from 43.47% to 48.98% on average. The higher average comes from combining strong new-task learning with old-task accuracy above initialization.

Table 1: CoT-supervised adaptation (A), 3B. Accuracy (%), mean ± sample standard deviation over three seeds. ProsQA: test with selected coefficients, except † (dev, fixed default). SVAMP: dev, fixed defaults. Bold marks block maxima, excluding †.

Method	Old task	New task	Avg
	GSM8K→ProsQA		
Sequential FT†	0.07±0.12	83.67±10.84	41.87±5.37
Answer Replay	**50.19±1.12**	79.47±8.60	64.83±3.98
EWC (adapted)	39.55±3.64	66.73±12.40	53.14±7.26
DER++ (adapted)	41.19±0.74	89.60±4.04	65.40±1.82
SEEKR (adapted)	43.11±2.28	77.53±7.14	60.32±3.56
LatentKV	48.98±0.72	**90.27±4.50**	**69.62±2.30**
	GSM8K→SVAMP		
Sequential FT	26.80±1.00	**72.98±3.30**	49.89±1.33
Answer Replay	**53.13±1.14**	69.26±2.02	**61.19±0.44**
EWC (adapted)	41.07±2.91	61.97±3.41	51.52±2.88
DER++ (adapted)	41.33±1.36	57.93±3.77	49.63±2.32
SEEKR (adapted)	40.60±3.83	71.52±1.70	56.06±2.69
KV-L2 matching	51.53±3.78	69.26±2.97	60.39±2.55
LatentKV	52.13±1.15	69.09±1.96	60.61±1.53

Latent-curriculum adaptation in the reverse order starts from a nearly solved old task. ProsQA accuracy starts at 99.44% and finishes at 98.56%, while new-task GSM8K reaches 57.93%, close to Sequential FT's 58.20%. Sequential FT retains only 5.33% on ProsQA. Compared with Replay, LatentKV improves both tasks at each of three seeds and reduces mean forgetting from 1.78 to 0.89 points (Appendix E). Here the model preserves an established ability while continuing to learn.

The advantage depends on the adaptation problem and the constraint strength. On GSM8K to SVAMP, LatentKV retains far more than Sequential FT but finishes 0.58 average points below Answer Replay. On ProsQA with fixed default coefficients, its test average is 54.30% versus Replay's 62.74%. The weaker selected penalty permits better new-task learning; this sensitivity to its strength is examined in Appendix B.

5.3 What Joint Refreshing Adds

Joint refreshing lets the reference move with adaptation while replay retains its fixed historical context. With surrogate replay held in place, adding this constraint improves both tasks: old-task accuracy rises by 1.73 points and new-task accuracy by 8.78, for a 5.26-point average gain (Table 2). Neither partial refresh matches joint refreshing, and refreshing KV alone underperforms leaving both quantities fixed. Joint refreshing changes both the direction and the interval over which displacement accumulates. The ablation tests these effects together. Paired gains over each component variant are positive at all three seeds, with unequal margins (Appendix C).

Table 2: GSM8K→ProsQA, Protocol A, 3B, three seeds. Accuracy (%, dev). Top: selected baselines (Replay weight 0.01). Bottom: component comparisons at replay weight 0.02 and projection weight 0.005 before removal or freezing; variants are not separately tuned.

Method	Old task	New task	Avg
Answer Replay	**53.53±0.31**	76.67±8.82	65.10±4.41
DER++ (adapted)	44.07±1.33	**89.22±5.32**	66.64±2.93
Splice-only	50.87±2.50	79.67±12.41	65.27±5.32
Fixed reference	49.53±5.11	73.00±23.25	61.27±14.17
Refresh KV only	49.47±0.81	60.22±41.74	54.84±20.99
Refresh sensitivity only	49.93±2.47	73.11±11.65	61.52±6.02
Joint refresh (LatentKV)	52.60±1.59	88.44±6.83	**70.52±2.68**

Historical conditioning alone changes the trade-off: at the same replay weight, Splice-only gains 5.18 old-task points over ordinary Replay but loses 4.87 on the new task (Appendix B). Joint refreshing improves on Splice-only at the selected setting, although the corresponding ordinary Replay + projection comparison is still needed to isolate the surrogate path's contribution to the full method. Direction choice is a separate question: under dynamic gradient-norm matching, random directions outperform own-sample sensitivity, leaving answer specificity unestablished (Appendix C). A separate refresh-interval comparison favors $R = 50$ among $25/50/100$ in both orders of the examined latent-curriculum runs (Appendix C.3).

5.4 Extension to Larger Models

The Qwen3 comparisons show the same pattern as the main 3B result: the larger gain is in learning ProsQA, with much smaller changes in old GSM8K accuracy (Table 3). New-task improvements range from 9.00 to 11.00 points across 4B, 8B, and 14B, while old-task differences range from -1.20 to $+2.00$. The benefit therefore extends beyond the 3B initialization in this task order. Each scale has only one paired run, and reversing the order lowers new-task accuracy at 4B and 8B (Table 15).

Table 3: Qwen3 adaptation from GSM8K to ProsQA under Protocol A, one paired run per scale. Accuracy (%, dev).

Model	Method	Old GSM8K	New ProsQA	Avg
Qwen3-4B	Answer Replay	33.80	39.33	36.57
	LatentKV	35.80	**50.33**	**43.07**
Qwen3-8B	Answer Replay	39.00	66.00	52.50
	LatentKV	37.80	**75.00**	**56.40**
Qwen3-14B	Answer Replay	48.60	88.00	68.30
	LatentKV	48.80	**97.67**	**73.23**

5.5 Continuous Computation and Efficiency

Keeping six latent steps while replacing their feedback inputs separates the value of the feedback from the mere presence of extra computation. After latent-curriculum adaptation from ProsQA to GSM8K, fixed feedback lowers new-task accuracy from 60.60% to 25.00%; channel rolling lowers it to 34.40% (Figure 3, left). Old ProsQA accuracy changes little. The continuous feedback is useful, but its importance depends on the task. Varying the number of steps gives a complementary result: GSM8K accuracy rises from 39.40% at zero steps to 60.60% at six, where the advantage over Replay reaches 3.20 points. Zero-step inference generates more text and takes longer, showing that fewer latent steps need not mean less total decoding work (Appendix F).

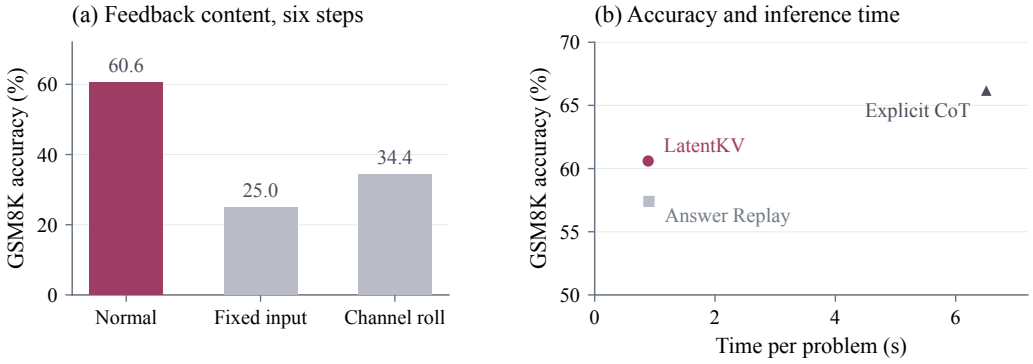

Figure 3: Continuous feedback and compact inference after ProsQA-to-GSM8K adaptation (Protocol B, 3B; dev). Left: GSM8K accuracy with six steps in all conditions. Right: accuracy on all evaluation examples versus synchronized batch-one model time on 64 examples, measured on the same physical A100-80GB over three interleaved repetitions. CoT is an independently trained system; timing details are in Appendix F.

The practical benefit is retaining compact inference after adaptation (Figure 3, right). Six-step LatentKV gives up 5.60 GSM8K accuracy points relative to explicit CoT for a $7.32\times$ speedup (0.889 versus 6.509 seconds) and 93.15% fewer generated text tokens. Replay takes 0.902 seconds: LatentKV preserves the latent model's inference cost, while the speedup over CoT comes from the reasoning format. The extra work occurs during training, with 1.48–$1.51\times$ Replay's elapsed time over the first 60 updates and 537 MiB of cached states and sensitivities (Appendix F.3).

6 Conclusion and Limitations

LatentKV separates the historical context used for replay from the reference used to regulate state changes. The main comparison improves new-task learning over Replay and retention over DER++; reverse latent-curriculum adaptation also preserves a high-accuracy old task. Joint refreshing improves both tasks over surrogate replay alone. These results suggest that historical context and a local protection reference need not remain identical: the former can stay fixed while the latter follows adaptation.

Limitations. Gains depend on coefficients, task order, and supervision. Fixed defaults and some reversed or latent-curriculum comparisons underperform Replay, while longer streams show no consistent average gain (Appendix E). The direction controls do not establish an advantage for own-sample sensitivity, and the selected setting lacks the ordinary Replay + projection comparison needed to isolate the surrogate path. Three training seeds, single-run larger models, and short task sequences limit conclusions about robustness and broader benchmarks such as TRACE (Wang et al., 2024). Training adds state storage and computation; the timing window and single-GPU measurements do not establish full-run costs or scaling to larger replay buffers.

AI USE STATEMENT

We used generative AI tools for language editing, drafting, structural revisions, and translation between Chinese and English, as well as feedback on experimental design and interpretation of results. The authors reviewed the AI-assisted content and take responsibility for the final text, research conclusions, and accompanying materials.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the problem and method; Algorithm 1 summarizes training. Section 5.1 and Appendix A describe the training and evaluation settings; Appendix B lists the coefficient candidates and selection procedure. The appendices provide per-seed results, ablations, and computational costs. Code is linked in the abstract.

REFERENCES

Emile Anand, Abdullah Ateyeh, Xinyuan Cao, and Max Dabagia. Continuous latent contexts enable efficient online learning in transformers, 2026. URL https://arxiv.org/abs/2605.09867.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation, 2024. URL https://openreview.net/forum?id=9cumTvvlHG.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step, 2025. URL https://openreview.net/forum?id=fRPmc94QeH.

Connor Dilgren and Sarah Wiegreffe. Are latent reasoning models easily interpretable? *arXiv preprint arXiv:2604.04902*, 2026.

Vaggelis Dorovatas, Nancy Kalaj, and Rahaf Aljundi. Continual self-improvement with lightweight experiential latent memories. *arXiv preprint arXiv:2606.17803*, 2026.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European conference on computer vision*, pp. 86–102. Springer, 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.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language model to reason in a continuous latent space, 2025. URL https://openreview.net/forum?id=tG4SgayTtk.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European conference on computer vision*, pp. 466–483. Springer, 2020.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. SEEKR: Selective attention-guided knowledge retention for continual learning of large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3266, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.190. URL https://aclanthology.org/2024.emnlp-main.190/.

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. *arXiv preprint arXiv:2106.09685*, 2021.

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

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

Jia Liang and Liangming Pan. Do latent-cot models think step-by-step? a mechanistic study on sequential reasoning tasks. *arXiv preprint arXiv:2602.00449*, 2026.

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.

Hanyu Lin, Min Cai, Jiawei Wen, and Haodi Zhang. Tyler: Typed latent reasoning for language models – when to think, what to compute, and how much to allocate, 2026. URL https://arxiv.org/abs/2606.16360.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10203–10209. IEEE, 2020.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*, 2023.

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

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pp. 4528–4537. PMLR, 2018.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qA6FgH0nnZ.

Dijia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57144–57163. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/su25g.html.

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 Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models, 2024. URL https://openreview.net/forum?id=xelrLobW0n.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-cot: Supervised implicit chain-of-thought. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=6YRJ4jmVQl.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1137. URL https://aclanthology.org/2025.acl-long.1137/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *International Conference on Learning Representations*, volume 2025, pp. 67081–67146, 2025.

A Training and Evaluation Details

Two adaptation protocols. Protocol A adapts a latent reasoner with textual reasoning supervision. Starting from a task-specific CoT-trained adapter on the latent backbone, it trains with explicit CoT and evaluates with six continuous steps followed by greedy decoding. The 3B comparisons use seeds 11/22/33, effective new-task batch size 32, at most 4,000 training examples per task, two adaptation epochs, maximum sequence length 1,024, and AdamW with learning rate 3×10^{-4}. Adaptation takes 236 updates for GSM8K to SVAMP and 250 for the reverse order and GSM8K to ProsQA.

Protocol B trains both tasks through a curriculum of 0, 2, 4, and 6 continuous steps. Component and diagnostic experiments use 500 adaptation updates, with six-step old-task trajectories for replay and state protection throughout. Initialization, data selection, and update budgets are shared within each comparison. Protocol A supplies the main performance comparison; Protocol B supplies the feedback and sensitivity analyses under latent-curriculum training. These analyses characterize a different training condition from the main comparison.

Replay memory. The default buffer contains 200 old training examples, replayed one per update in cyclic order. After removing duplicate questions, development-question overlap, and overlength examples, we sort identifiers by SHA-256 hash and take the first 200. SVAMP additionally requires valid equations. Optimization seeds change the starting offset but keep the buffer fixed. Buffer-selection controls shuffle prepared indices before selection (Table 18); three-task streams divide the same total budget as evenly as possible between learned tasks. This budget counts examples, with additional storage for historical KV, reference KV, and sensitivities (Appendix F.3).

Baseline adaptations. EWC estimates diagonal empirical Fisher importance from 200 old examples and penalizes adapter-parameter changes, without answer replay. DER++ combines historical-logit mean-squared error with old-answer cross-entropy. Our SEEKR adaptation adds logit and attention distillation to answer replay, selecting at most 24 layers and 128 heads from at most 16 answer-query positions. KV-L2 adds component-wise historical KV matching to answer replay. Fixed defaults are 10^4 for EWC, 0.5 for each DER++ term, and logit/attention weights $0.5/1.0$ for SEEKR. Answer replay, SEEKR, KV-L2, and LatentKV use replay weight 0.02; KV-L2 and LatentKV use internal state-penalty weight 0.05. LatentKV uses $R = 50$ and global gradient-clipping threshold 2.0. Appendix B gives the selected coefficients. Equal coefficients need not produce equal auxiliary-gradient norms.

Evaluation. Development evaluations contain 500 GSM8K, 300 ProsQA, and 206 SVAMP examples. The main test comparison uses 1,319 GSM8K and 500 ProsQA examples after overlap removal. Development data inform method design and coefficient selection; the selected setting is fixed for testing. Unless specified otherwise, decoding uses six continuous steps and is greedy. We average old/new accuracy within each run, then compute the mean and sample standard deviation across seeds. Paired gains use unrounded values. Strict accuracy requires an extractable answer. Default Sequential FT on GSM8K to ProsQA fails the answer-format or numeric-extraction checks for 1,496 of 1,500 old-task predictions, so its near-zero score combines reasoning and answer-format failures.

B Main Comparison and Coefficient Choice

B.1 Retention and Adaptation after Coefficient Selection

Each method chooses among three candidates by its seed-11 old/new development average, breaking ties in favor of the default (Table 5). The choice is fixed for seeds 22/33 and all test evaluations. The selected projection weight is 0.005, the smallest candidate, with effective weight $\lambda_{\text{rep}}\lambda_{\text{kv}} = 10^{-4}$. Search budgets are equal; DER++ and SEEKR keep their stated weight ratios rather than tuning each loss weight independently.

The main comparison separates two forms of improvement. Replay gains more on old GSM8K ($F = -6.72$) but learns ProsQA less well; DER++ reaches similar ProsQA accuracy while losing old-task accuracy ($F = +2.27$). LatentKV raises GSM8K from 43.47% to 48.98% ($F = -5.51$), finishing above its own starting accuracy at every seed (Table 4). Its paired average gains are $+5.02$,

$+0.43$, and $+8.94$ over Replay, and $+7.11$, $+0.20$, and $+5.36$ over DER++. All signs agree, though three seeds do not establish statistical significance.

Table 4: GSM8K to ProsQA, Protocol A, 3B. Accuracy (%, test) with the chosen coefficients fixed across seeds. Initial GSM8K accuracy is $42.84/41.77/45.79\%$ for seeds 11/22/33. F is initial minus final old-task accuracy in percentage points; negative values indicate improvement. Avg weights the tasks equally.

Method	Seed	Old GSM8K	New ProsQA	Avg	F
Answer Replay	11	50.80	83.40	67.10	-7.96
Answer Replay	22	48.90	85.40	67.15	-7.13
Answer Replay	33	50.87	69.60	60.24	-5.08
EWC (adapted)	11	35.86	66.60	51.23	$+6.97$
EWC (adapted)	22	39.65	54.40	47.03	$+2.12$
EWC (adapted)	33	43.14	79.20	61.17	$+2.65$
DER++ (adapted)	11	42.00	88.00	65.00	$+0.83$
DER++ (adapted)	22	40.56	94.20	67.38	$+1.21$
DER++ (adapted)	33	41.02	86.60	63.81	$+4.78$
SEEKR (adapted)	11	44.28	84.20	64.24	-1.44
SEEKR (adapted)	22	40.49	78.40	59.44	$+1.29$
SEEKR (adapted)	33	44.58	70.00	57.29	$+1.21$
LATENTKV	11	49.43	94.80	72.12	-6.60
LATENTKV	22	49.36	85.80	67.58	-7.58
LATENTKV	33	48.14	90.20	69.17	-2.35

B.2 How the Constraint Weight Changes the Trade-off

Within the coefficient sweep, reducing the projection weight from 0.05 to 0.005 raises new-task accuracy from 80.67% to 95.33%, while old-task accuracy stays within 49.60–52.20% (Table 5). A weaker penalty leaves more room for adaptation without a comparable loss of retention. The component study in Section 5.3 tests whether the constraint still contributes at this weight.

Table 5: Coefficient sweep, Protocol A, GSM8K to ProsQA, seed 11. Accuracy (%, dev). Asterisk marks the selected candidate. We vary the replay weight for Answer Replay, the internal projection weight for LATENTKV (replay weight 0.02), parameter-penalty weight for EWC, tied logit/answer weights for DER++, and logit/attention weights for SEEKR.

Method	Coefficient	Old task	New task	Avg
Answer Replay	0.01^*	53.80	83.33	68.57
	0.02	49.40	83.67	66.53
	0.04	38.80	78.33	58.57
EWC (adapted)	100	0.00	90.33	45.17
	1000^*	37.20	63.00	50.10
	10000	48.40	6.67	27.53
DER++ (adapted)	0.05^*	45.20	86.67	65.93
	0.15	38.80	77.00	57.90
	0.5	36.20	60.67	48.43
SEEKR (adapted)	$(0.05, 0.1)^*$	48.00	85.00	66.50
	$(0.15, 0.3)$	40.80	86.67	63.73
	$(0.5, 1.0)$	48.60	81.00	64.80
LATENTKV	0.005^*	51.40	95.33	73.37
	0.015	49.60	88.00	68.80
	0.05	52.20	80.67	66.43

The optimum may lie outside this range. Fixed defaults favor Replay in both evaluations (Table 6), with the seed-33 model reaching only 22.40% on new-task test data. Raising its development de-

coding cap from 256 to 512 or 1,024 leaves accuracy at 23.67%; extra output length does not resolve the failure.

At the same replay weight 0.02, Splice-only improves old-task test accuracy by 5.18 points and reduces new-task accuracy by 4.87. Historical conditioning shifts the trade-off toward retention, leaving the average nearly unchanged. Replay + projection is evaluated at 0.05, but not at the selected weight 0.005; whether the surrogate path is needed in that full configuration remains open.

Table 6: Fixed-default GSM8K-to-ProsQA comparisons, Protocol A, 3B. Accuracy (%). Dev: mean $\pm$ sample standard deviation over seeds 11/22/33; test: individual runs. Results with the chosen coefficients are in Table 4.

Method	Old task	New task	Avg
	Dev		
Sequential FT	0.07±0.12	**83.67±10.84**	41.87±5.37
Answer Replay	46.67±4.23	80.44±5.87	63.56±5.04
EWC (adapted)	47.47±2.16	10.56±3.47	29.01±1.41
DER++ (adapted)	38.67±4.27	62.22±2.41	50.44±3.34
SEEKR (adapted)	47.00±2.95	71.00±21.49	59.00±9.90
KV-L2 matching	50.27±1.86	59.11±10.63	54.69±4.75
Splice-only	50.67±0.90	74.33±18.66	62.50±9.47
Replay + projection	47.47±2.66	65.11±19.41	56.29±8.41
Fixed reference	**51.20±2.16**	76.56±13.07	**63.88±5.75**
LATENTKV	50.47±3.35	58.67±30.64	54.57±16.97
	Test		
Answer Replay (seed 11)	44.73	85.00	64.87
Answer Replay (seed 22)	38.44	77.80	58.12
Answer Replay (seed 33)	46.25	84.20	65.22
Splice-only (seed 11)	48.82	89.40	69.11
Splice-only (seed 22)	47.84	85.80	66.82
Splice-only (seed 33)	48.29	57.20	52.75
Replay + projection (seed 11)	39.12	86.00	62.56
Replay + projection (seed 22)	46.10	54.00	50.05
Replay + projection (seed 33)	45.19	64.00	54.59
LATENTKV (seed 11)	49.05	79.80	64.43
LATENTKV (seed 22)	48.60	79.20	63.90
LATENTKV (seed 33)	46.78	22.40	34.59

C JOINT REFRESHING AND CONSTRAINT DIRECTION

C.1 UPDATING THE REFERENCE AND DIRECTION TOGETHER

Adding joint refreshing to Splice-only improves both task means, with most of the 5.26-point average gain coming from new-task learning. Thus, at this setting, the constraint does more than move accuracy from one task to the other. Joint refreshing exceeds all four ablations at every seed, although its gain over Splice-only ranges from 0.03 to 11.03 points (Table 7).

All variants use Protocol A, 250 updates, 200 replay examples, $R = 50$, $\lambda_{\text{rep}} = 0.02$, and $\lambda_{\text{kv}} = 0.005$ before removing or fixing a component. The full-method row reuses the selected-configuration runs, including the seed-11 run used for selection; variants are not separately tuned. This isolates changes to the adopted configuration, not the mechanism by which refreshing improves accuracy. The selected Answer Replay row in Table 2 instead uses weight 0.01, so its comparison with Splice-only does not isolate the surrogate path; the equal-weight comparison appears in Table 6.

C.2 ALTERNATIVE CONSTRAINT DIRECTIONS

These controls keep surrogate replay and joint refreshing, but scale constraint parameter gradients to 1% of the new-task gradient norm before global clipping, without differentiating through the scale. Cross-sample sensitivity pairs another example's direction within the refresh block with the target's

Table 7: Joint refreshing minus each component ablation, Protocol A, GSM8K to ProsQA. Accuracy differences (percentage points, dev), calculated before rounding. The last three columns give paired two-task-average differences.

Comparator	Old	New	Avg	Seed 11	Seed 22	Seed 33
Splice-only	+1.73	+8.78	+5.26	+4.70	+0.03	+11.03
Fixed reference	+3.07	+15.44	+9.26	+1.23	+1.60	+24.93
Refresh KV only	+3.13	+28.22	+15.68	+2.20	+5.83	+39.00
Refresh sensitivity only	+2.67	+15.33	+9.00	+4.90	+10.10	+12.00

Table 8: Gradient-norm-matched controls under Protocol A, GSM8K to ProsQA, three seeds. Accuracy (%, dev). All arms retain surrogate replay and joint refreshing; dynamic scaling makes them distinct from the fixed-coefficient full method.

Variant	Old task	New task	Avg
Own-sample sensitivity	51.00 ± 1.56	71.00 ± 18.70	61.00 ± 9.98
Cross-sample sensitivity	51.20 ± 1.91	78.44 ± 8.70	64.82 ± 3.80
Random direction	53.00 ± 3.42	77.89 ± 9.18	65.44 ± 5.82
KV-L2	51.67 ± 1.29	62.67 ± 43.33	57.17 ± 22.30

reference KV. Random directions are independent Gaussian tensors scaled to the reference sensitivity norms. Zero gradients cannot attain the target; mean observed ratios are 0.00976–0.00980.

Random directions average 4.44 points above own-sample sensitivity, with positive gains at all three seeds; cross-sample directions also have a higher mean (Table 8). The sample's own direction therefore has no demonstrated advantage under norm matching. Because this controller replaces fixed coefficients, the comparison neither ranks these alternatives against the full method nor explains its gain as an effect of answer specificity.

C.3 Refresh Interval

Refreshing changes both the age of the sensitivity and the point from which displacement is measured. A shorter interval therefore also shortens the period over which displacement accumulates. Table 9 compares three intervals on the latent-curriculum trajectories. The intermediate interval gives the highest average in both task orders, although $R = 50$ and $R = 100$ are close in the reverse order. This single-seed comparison supports the adopted interval within the tested range; it does not establish that more frequent refreshing improves adaptation.

Table 9: Refresh intervals on the diagnostic trajectories: Protocol B, 3B, seed 42, 500 updates, $\lambda_{\text{rep}} = 0.02$, $\lambda_{\text{kv}} = 0.05$. Accuracy (%, dev).

	GSM8K → ProsQA			ProsQA → GSM8K		
R	Old	New	Avg	Old	New	Avg
25	24.80	99.67	62.23	94.67	57.60	76.13
50	36.00	99.67	67.83	96.00	58.80	77.40
100	26.60	100.00	63.30	96.33	58.20	77.27

D Sensitivity during Adaptation

D.1 Current and Cached Directions

Figure 1 uses a Protocol B, seed-42 GSM8K-to-ProsQA trajectory with $R = 50$; Table 10 includes all 13 checkpoints and the reverse-order comparison. Current sensitivity correlates more strongly

with old-answer loss changes than historical sensitivity at the four plotted checkpoints. The cached signal can lag behind: at update 225 its correlation is 0.076, versus 0.802 for the current signal, yet the two are close at update 475. Both caches are 25 updates old. The usefulness of a cached direction thus depends on the intervening model changes as well as its age. At update 225, Spearman correlation also favors current over historical sensitivity (0.732 versus 0.177), as does every leave-one-probe-out Pearson calculation.

At each checkpoint, we restore optimizer moments and the scheduled learning rate, clip a new-task mini-batch gradient, and take one AdamW step without fresh replay or state-penalty gradients. We correlate projected state changes with actual old-answer loss changes across 64 fixed old-training probes outside replay. Historical sensitivity comes from the old-task endpoint; recent sensitivity comes from the last refresh. The signals share the same checkpoint and update, with training state restored afterward and final accuracy evaluated separately under the corresponding training setup. Reverse-order correlations can be negative. These dependent checkpoints characterize a local empirical signal, not a complete first-order decomposition or a general test for harmful updates.

Table 10: Pearson correlation with old-answer loss change after one restored new-task AdamW step. Protocol B, 3B, seed 42, $R = 50$, 64 training probes outside replay. Age counts updates since refreshing. Recent is the periodically cached sensitivity labeled Last refreshed in Figure 1.

Update	Age	Current	Recent	Historical
GSM8K → ProsQA				
0	0	0.620	0.619	0.619
1	1	0.714	0.718	0.718
25	25	0.671	0.438	0.438
49	49	0.805	0.804	0.804
225	25	0.802	0.076	0.119
250	0	0.469	0.469	-0.006
251	1	0.713	0.320	0.053
275	25	0.745	0.282	0.106
299	49	0.432	0.159	0.001
450	0	0.411	0.411	0.006
451	1	0.615	0.619	0.081
475	25	0.693	0.676	0.192
499	49	0.860	0.856	0.046
ProsQA → GSM8K				
25	25	0.460	0.011	0.011
225	25	-0.085	0.008	-0.124
475	25	-0.691	-0.677	0.025

D.2 Sensitivity on Baseline Trajectories

Current sensitivity also has higher Pearson and Spearman correlations at all three forward Answer Replay checkpoints (Table 11). Historical directions can become less informative even without our constraint. The reverse-order Replay and Sequential FT trajectories lack this consistent ordering.

D.3 How Historical Conditioning Changes the Replay Gradient

The surrogate path leaves the answer prediction unchanged but can alter both the direction and magnitude of its parameter gradient. To measure this effect, we compute ordinary and surrogate replay gradients on four old-task training probes outside the replay buffer. Each diagnostic arm starts from the same model, AdamW moments, and random state, adds the same new-task gradient, and takes one update. Before addition, the surrogate replay gradient is scaled to the ordinary replay gradient norm; both receive weight 0.02 and global clipping at 2. These diagnostic updates omit the state-constraint gradient. Old-answer loss is measured on the remaining 60 probes, and new-task loss on the shared training mini-batch.

Table 11: Sensitivity diagnosis on baseline trajectories, Protocol B, seed 42, 64 probes per checkpoint. H/C denote historical/current sensitivity. Checkpoints on one trajectory are dependent.

Method	Update	r_{H}	r_{C}	ρ_{H}	ρ_{C}
GSM8K → ProsQA					
Answer Replay	25	0.122	0.643	0.226	0.389
Answer Replay	225	0.287	0.561	0.082	0.479
Answer Replay	475	0.185	0.749	0.225	0.382
Sequential FT	25	0.043	-0.047	0.073	-0.233
Sequential FT	225	-0.429	0.353	-0.136	0.287
Sequential FT	475	0.172	0.411	-0.003	0.357
ProsQA → GSM8K					
Answer Replay	25	0.266	0.372	0.337	0.428
Answer Replay	225	0.045	-0.067	0.029	0.112
Answer Replay	475	0.011	-0.377	0.330	-0.029
Sequential FT	25	0.067	0.112	-0.206	0.077
Sequential FT	225	0.121	-0.083	0.124	-0.084
Sequential FT	475	0.246	0.265	-0.059	0.220

Table 12 shows all three checkpoints in both task orders at $R = 50$. Historical conditioning changes the replay direction most clearly in the forward order, where the cosine is 0.522 at update 225. By update 475, the historical weight has fallen to 0.0057 and the directions are nearly aligned. This is the gradual transition toward ordinary replay induced by annealing. Directional agreement alone does not determine the AdamW step's effect on other old examples: the surrogate arm yields a lower old-task loss change at one of these six checkpoints and at three of the 18 checks across all three refresh intervals. These local checks characterize the surrogate update; the component experiments assess its effect over training.

Table 12: Replay-gradient diagnosis on Protocol B trajectories, 3B, seed 42, $R = 50$. Cosine and norm ratio (surrogate/ordinary) are measured before norm matching. Loss columns give 10^3 times the change after one AdamW update, with lower values indicating a better local loss outcome. Both Ordinary and Proxy include the same new-task gradient.

			Old loss change		New loss change	
Update	Cosine	Norm ratio	Ordinary	Proxy	Ordinary	Proxy
GSM8K → ProsQA						
25	0.7849	0.646	0.9093	0.3586	-7.0177	-6.9160
225	0.5216	0.768	-11.5266	-9.5423	-11.0975	-10.9557
475	0.9996	1.013	0.7591	3.0612	0.0009	0.0003
ProsQA → GSM8K						
25	0.8809	0.930	-6.7119	-6.5938	-38.2770	-38.4274
225	0.9759	1.019	-1.3518	-0.8968	-43.4444	-43.9865
475	0.9991	0.890	-0.4468	-0.0539	-9.4425	-9.5445

E Task Order and Transfer beyond the Main Setting

E.1 Learning through a Latent Curriculum

ProsQA-to-GSM8K adaptation tests retention from a high old-task starting accuracy of 99.44%. Under Protocol B, LatentKV retains 98.56% while reaching 57.93% on new GSM8K (Table 13). Forgetting is 0.89 points, compared with Replay's 1.78; each seed improves on Replay on both tasks. Sequential FT reaches similar new-task accuracy but retains only 5.33% on ProsQA. Here the 1.31-point average gain accompanies preservation of an already learned task. The opposite order

favors Replay: ProsQA learning is nearly saturated, leaving old-GSM8K retention to determine the difference. The seed-42 test comparison has the same asymmetry (Table 14).

Table 13: Latent-curriculum adaptation (Protocol B), 3B, seeds 11/22/42. Accuracy (%, dev), mean ± sample standard deviation, after 500 adaptation updates.

	ProsQA → GSM8K			**GSM8K → ProsQA**		
Method	Old	New	Avg	Old	New	Avg
Sequential FT	5.33±3.61	58.20±1.00	31.77±2.18	20.13±17.89	99.89±0.19	60.01±8.92
Answer Replay	97.67±1.67	56.20±1.74	76.93±0.35	34.00±5.30	99.22±0.19	66.61±2.74
Fixed reference	98.00±1.33	57.47±1.47	77.73±1.31	33.67±6.40	99.67±0.33	66.67±3.29
LATENTKV	98.56±1.64	57.93±2.23	78.24±1.06	29.47±3.56	98.89±1.35	64.18±1.34

Table 14: Protocol B, seed 42. Accuracy (%, test). Initial denotes the pre-adaptation model.

	GSM8K → ProsQA			**ProsQA → GSM8K**		
Method	Old	New	Avg	Old	New	Avg
Initial	57.54	0.00	28.77	99.20	17.21	58.21
Sequential	39.35	99.20	69.27	10.80	56.18	33.49
Replay	39.35	98.80	69.07	95.60	55.34	75.47
EWC	54.74	99.80	77.27	5.60	54.89	30.25
DER++	29.26	97.60	63.43	72.40	57.39	64.90
LatentKV	20.77	99.20	59.99	96.60	55.72	76.16

E.2 LARGER MODELS

Across Qwen3 sizes, the forward comparisons primarily improve new-task learning (Table 3). Reversing the order reduces new-task accuracy at 4B/8B and gives a small gain at 14B (Table 15), so model size alone does not determine the benefit. Each scale has one paired run: transfer seed 11 at 4B, and transfer seed 22 with initialization seed 42 at 8B/14B. Both 4B methods use adaptation learning rate 3×10^{-4}.

Table 15: Reverse-order Qwen3 comparison: ProsQA to GSM8K, Protocol A, six latent steps. Accuracy (%, dev). One paired run per scale; forward results are in Table 3.

Size	Method	Old ProsQA	New GSM8K	Avg
4B	Answer Replay	99.00	64.60	81.80
4B	LATENTKV	99.33	49.00	74.17
8B	Answer Replay	99.00	85.20	92.10
8B	LATENTKV	99.00	82.00	90.50
14B	Answer Replay	99.67	86.40	93.03
14B	LATENTKV	99.67	87.00	93.33

E.3 TASK PAIRS AND LONGER STREAMS

The six-order Protocol A study predates the fixed-default baseline batch and main comparison (Table 16). Its pair-averaged gains are $+4.86$, $+0.14$, and -5.68 points for GSM8K/ProsQA, GSM8K/ProntoQA, and ProsQA/ProntoQA. The near-zero six-order mean (-0.23) reflects opposing effects, rather than a small common benefit across pairs. Per-seed forgetting is $F = A_{1,1} - A_{2,1}$, with initial accuracies 46.47 ± 0.76, 60.56 ± 32.20, and 99.50 ± 0.50 on GSM8K, ProsQA, and ProntoQA. Negative values denote improvement; variation in initial ProsQA accuracy also widens its forgetting distribution.

Table 16: Six task orders under Protocol A, 3B, seeds 11/22/33. Accuracy (%, dev), mean ± sample standard deviation. F is forgetting in percentage points.

Method	Old	New	Avg	$F \downarrow$
GSM8K → ProsQA				
Answer Replay	45.20±5.33	69.22±9.67	57.21±3.83	1.27±4.80
LATENTKV	49.47±4.09	83.22±5.80	66.34±4.94	−3.00±3.36
ProsQA → GSM8K				
Answer Replay	80.11±3.10	39.33±2.72	59.72±0.77	−19.56±31.45
LATENTKV	80.67±2.89	39.93±2.84	60.30±0.96	−20.11±31.19
GSM8K → ProntoQA				
Answer Replay	38.93±4.37	67.33±3.79	53.13±2.83	7.53±3.71
LATENTKV	39.60±5.72	67.50±5.77	53.55±1.13	6.87±6.18
ProntoQA → GSM8K				
Answer Replay	87.67±3.06	41.60±2.50	64.63±0.70	11.83±3.25
LATENTKV	87.33±3.01	41.67±2.70	64.50±0.23	12.17±3.25
ProntoQA → ProsQA				
Answer Replay	81.00±7.37	55.11±20.72	68.06±8.09	18.50±7.86
LATENTKV	78.83±6.81	40.11±26.96	59.47±12.47	20.67±7.18
ProsQA → ProntoQA				
Answer Replay	72.67±1.15	60.17±4.04	66.42±2.18	−12.11±32.66
LATENTKV	69.44±3.86	57.83±7.69	63.64±3.68	−8.89±28.43

A third task. In the GSM8K, AQuA, ProsQA stream, LATENTKV retains 7.80 more first-task points, but lower final-task accuracy offsets this gain (Table 17). The reverse stream also has a lower average. These seed-11 Protocol A runs carry each method's adapted parameters forward and split a total replay budget of 200. Retention gains thus persist on one early task without improving performance across the whole stream.

Table 17: Three-task Protocol A streams, 3B, seed 11. Stage is the number of learned tasks; T_1, T_2, T_3 follow the displayed stream order. Avg includes only learned tasks.

		GSM8K → AQuA → ProsQA				**ProsQA → AQuA → GSM8K**			
Stage	Method	T_1	T_2	T_3	Avg	T_1	T_2	T_3	Avg
2	Answer Replay	52.80	23.62	—	38.21	83.33	29.53	—	56.43
2	LATENTKV	50.80	25.59	—	38.20	82.00	27.56	—	54.78
3	Answer Replay	25.40	26.77	24.00	25.39	84.00	27.95	35.40	49.12
3	LATENTKV	33.20	25.20	18.00	25.47	84.33	26.38	34.00	48.24

E.4 CHANGING THE REPLAY EXAMPLES

ProsQA-to-GSM8K average gains persist across three buffer selections (0.70–2.13 points), whereas the forward order favors LATENTKV in only one draw (Table 18). Buffer composition matters alongside task order. Draw IDs change the old examples only; model and optimization seed 42 remain fixed.

Table 18: Buffer selection under Protocol B, seed 42, 200 replay examples. Accuracy (%, dev). Each cell is old/new/two-task average; draw IDs are buffer-selection seeds.

	ProsQA → GSM8K		**GSM8K → ProsQA**	
Draw	Answer Replay	LATENTKV	Answer Replay	LATENTKV
default	97.67 / 57.00 / 77.33	99.33 / 59.60 / 79.47	35.20 / 99.33 / 67.27	30.20 / 99.67 / 64.93
11	97.00 / 57.20 / 77.10	96.00 / 59.60 / 77.80	33.00 / 99.00 / 66.00	38.00 / 99.67 / 68.83
22	98.33 / 56.40 / 77.37	98.67 / 59.40 / 79.03	33.40 / 99.67 / 66.53	26.40 / 98.67 / 62.53

F CONTINUOUS FEEDBACK AND COMPUTATIONAL COST

F.1 FEEDBACK CONTENT

Replacing feedback content while keeping all six forward passes reduces GSM8K accuracy for both Replay and LATENTKV, but changes ProsQA much less (Table 19). The passes alone are insufficient: GSM8K uses information carried between them. This task dependence motivates examining continuous computation separately from the eventual answer, without selecting a particular protection rule.

Table 19: Feedback interventions under Protocol B, 3B, seed 42. Accuracy (%, dev). Each condition uses the same trained model and six continuous steps. Fixed input removes state-dependent feedback; channel rolling perturbs state channels.

		GSM8K → ProsQA			**ProsQA → GSM8K**		
Method	Task	Normal	Fixed input	Channel roll	Normal	Fixed input	Channel roll
LATENTKV	GSM8K	23.00	11.20	11.60	60.60	25.00	34.40
LATENTKV	ProsQA	98.67	98.67	98.67	96.33	96.33	95.67
Replay	GSM8K	40.00	24.20	20.80	57.40	27.00	32.20
Replay	ProsQA	98.33	98.67	83.00	96.00	95.33	94.67

F.2 CONTINUOUS STEPS AND SAME-DEVICE INFERENCE TIME

Continuous steps add computation but can shorten subsequent text generation. On new GSM8K, moving from zero to six steps raises LATENTKV accuracy from 39.40% to 60.60% while reducing time from 1.531 to 0.889 seconds; old ProsQA stays within 96.33%–97.00% (Table 20). Zero-step decoding keeps latent-interface boundary tokens and is not a separately trained direct-answer model.

Table 20: ProsQA to GSM8K, Protocol B, 3B, seed 42. Accuracy (%, dev) uses all examples; synchronized seconds and text tokens use 64 per task. Cap: 256 tokens. Timing: mean ± SD of three repetition means. Dashes denote unreported combinations.

		GSM8K (new)			**ProsQA (old)**		
Method	Steps/mode	Acc.	Seconds	Tokens	Acc.	Seconds	Tokens
Replay	latent0	41.60	1.412 ± 0.006	21.55	—	—	—
Replay	latent2	46.40	0.649 ± 0.004	7.16	—	—	—
Replay	latent4	54.60	0.775 ± 0.005	7.16	—	—	—
Replay	latent6	57.40	0.902 ± 0.010	7.16	96.00	1.174 ± 0.021	11.70
LatentKV	latent0	39.40	1.531 ± 0.013	23.33	96.67	0.814 ± 0.020	11.84
LatentKV	latent2	45.20	0.654 ± 0.007	7.17	97.00	0.928 ± 0.023	11.66
LatentKV	latent4	55.40	0.774 ± 0.009	7.19	96.67	1.041 ± 0.007	11.66
LatentKV	latent6	60.60	0.889 ± 0.002	7.14	96.33	1.155 ± 0.006	11.66
CoT	explicit	66.20	6.509 ± 0.074	104.28	97.67	2.803 ± 0.014	45.42

Six-step inference trades 5.60 GSM8K accuracy points for a 7.32× speedup over explicit CoT in this order; the forward order has a much larger old-GSM8K gap (Table 21). These separately trained systems share data exposure and update counts, not training FLOPs. The speed difference concerns reasoning format rather than the protection term. CoT uses full-chain replay and a 256-token cap; its reverse-order GSM8K accuracy is 16.00% at 64 tokens, 56.40% at 128, and 66.20% at 256.

Table 21: GSM8K to ProsQA, Protocol B, 3B, seed 42. Accuracy (%, dev) uses all examples; synchronized seconds and text tokens use 64 per task. Cap: 256 tokens. Timing: mean ± SD of three repetition means. Dashes denote unreported combinations.

		GSM8K (old)			**ProsQA (new)**		
Method	Steps/mode	Acc.	Seconds	Tokens	Acc.	Seconds	Tokens
Replay	latent0	23.60	1.724 ± 0.010	26.42	—	—	—
Replay	latent2	35.40	0.646 ± 0.002	7.11	—	—	—
Replay	latent4	36.40	0.769 ± 0.009	7.12	—	—	—
Replay	latent6	40.00	0.892 ± 0.005	7.11	98.33	1.188 ± 0.006	11.69
LatentKV	latent0	22.80	1.400 ± 0.020	21.36	—	—	—
LatentKV	latent2	23.40	0.641 ± 0.007	7.11	—	—	—
LatentKV	latent4	23.80	0.770 ± 0.004	7.12	—	—	—
LatentKV	latent6	23.00	0.886 ± 0.008	7.12	98.67	1.178 ± 0.009	11.69
CoT	explicit	66.80	6.520 ± 0.039	104.81	99.33	2.836 ± 0.052	45.45

Accuracy uses all evaluation examples. For each order, timing uses one physical A100-80GB, the same 64 examples per task, and three interleaved repetitions. CUDA-synchronized, greedy batch-one time includes continuous computation and text generation, excluding loading, tokenization, scoring, and I/O. Repetitions add no examples or training seeds. Text-token counts include answers and EOS, excluding continuous steps. No historical caches or proxy computation enter inference.

F.3 Training and Storage

Over the first 60 Protocol B updates, LatentKV takes 1.48–1.51× Replay's elapsed time with similar peak GPU allocation (Table 22). Active KV/sensitivity files occupy 537.11 MiB. Cache storage grows linearly with the replay buffer, thought steps, layers, and KV width; low GPU overhead therefore does not imply low storage cost as these grow.

Each order is profiled on one physical GPU, including reference initialization, refreshing, and checkpoint time. Reference generation uses BF16 autocasting; tensors retain their dtypes, with FP32 projection sums and norms. Serialized size counts shared paths once and includes metadata, excluding replay text, model/optimizer state, and archives. Fixed reference still computes refreshes in this control implementation. Its timing is therefore not optimized, and the 60-update window measures early training rather than full-run cost.

Table 22: Training cost over the first 60 Protocol B updates. Peak is peak allocated GPU memory; Cache is serialized storage for the active KV/sensitivity files.

	GSM8K → ProsQA				**ProsQA → GSM8K**			
Method	Minutes	Opt. time (s/update)	Peak (GiB)	Cache (MiB)	Minutes	Opt. time (s/update)	Peak (GiB)	Cache (MiB)
Sequential FT	15.13	12.33	12.45	0.00	13.61	12.02	11.88	0.00
Answer Replay	16.61	13.67	12.45	0.00	15.46	13.57	13.40	0.00
Fixed reference	24.30	15.74	12.47	537.11	23.30	15.88	13.39	537.11
LatentKV	24.65	16.00	12.47	537.11	23.33	15.79	13.39	537.11