SELF-CONSOLIDATING LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) increasingly receive information as streams of passages, conversations, and long-context workflows. While longer context windows expose more evidence, they do not ensure that useful information is preserved and reused. We study continual context consolidation: writing current context into model weights while limiting interference with previously consolidated information. We propose **S**elf-**Co**nsolidating **L**anguage Models (SCoL), a post-training framework in which, given current context, an LLM learns to generate textual update instructions specifying which of its own Transformer layers should be updated. Because committed updates change the model that later generates future selections, we train SCoL with meta-reinforcement learning over an evolving model state. We instantiate SCoL with supervised QA rewards on SQuAD knowledge incorporation and intrinsic likelihood-based rewards for LongBench v2 long-context consolidation. On SQuAD, SCoL improves acquisition over the evaluated baselines and retention over sequential finetuning. On LongBench v2, we meta-train SCoL only on short contexts. It outperforms the evaluated baselines on short contexts and generalizes to longer contexts while remaining competitive with in-context methods. Analysis of learned selection patterns shows that SCoL encourages the LLM to generate sparse update locations that align with layers of high Fisher information, suggesting that the model learns to route plasticity toward loss-sensitive regions while limiting interference.

1 INTRODUCTION

Large language models increasingly operate over streaming context, including multi-turn conversations, tool-using agents, document workflows, and long-horizon interactive tasks. In these settings, the model must not only answer from the current prompt, but also preserve earlier information for later use. Longer context windows help, but they do not guarantee reliable memory: models can underuse distant evidence and degrade as context length and reasoning complexity increase (Liu et al., 2024; Hsieh et al., 2024). Prior work manages long-context information through retrieval or external memory (Lewis et al., 2020; Borgeaud et al., 2022; Wang et al., 2023a), attention-routing or memory mechanisms (Mohtashami & Jaggi, 2023), compression (Chevalier et al., 2023; Jiang et al., 2023), and agentic memory stores (Packer et al., 2023; Zhong et al., 2024). These methods are effective, but they often depend on retrieval quality, auxiliary modules, lossy compression, or repeated inference-time processing of persistent context. We study a complementary direction: using the model weights as an internal memory substrate. Motivated by theories of memory consolidation, where experience is integrated into long-term internal representations rather than kept only as transient context (McClelland et al., 1995; Kumaran et al., 2016; Dudai et al., 2015), we ask whether useful context can be written into parameters so it remains available after the original context is gone.

This reframes long context inference as a continual learning problem. The model must consolidate a stream of contexts into weights while preserving previously consolidated knowledge. However, sequential neural network updates are prone to catastrophic interference (McCloskey & Cohen, 1989). Classical methods reduce interference by estimating parameter importance, for example with Fisher based regularization (Kirkpatrick et al., 2017), but reliable Fisher or Hessian style estimation is costly and implementation sensitive at LLM scale (van de Ven, 2025). We therefore ask whether the model itself can determine where adaptation should occur.

Inspired by SEAL (Zweiger et al., 2025), we propose *Self-Consolidating Language Models* (SCoL), a post-training framework in which an LLM learns to generate textual update instructions specifying

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.

which of its own parameters to adapt for a given context. We then attach LoRA adapters to the selected modules and perform a LoRA update (Hu et al., 2022). Unlike SEAL, which focuses on generating self edits and update directives, our focus is where consolidation should occur under a drifting policy—-each committed update changes the model that will choose future updates, so the policy itself is part of the evolving state. We trained SCoL with meta-reinforcement learning whose reward favors acquisition of the current context while penalizing forgetting of earlier contexts.

We instantiate SCoL under two reward settings: a supervised QA reward for SQuAD-style knowledge incorporation (Rajpurkar et al., 2016), and an intrinsic likelihood-based reward for unlabeled long-context streams evaluated on LongBench v2 (Bai et al., 2025). On SQuAD, SCoL improves acquisition over the evaluated baselines and retention over sequential finetuning. Its learned selections are sparse and align with high-Fisher information layers, suggesting that SCoL routes plasticity toward loss-sensitive regions while limiting interference. On LongBench v2, SCoL is meta-trained only on short streams and generalizes to longer evaluation streams without full-context access.

Our contributions are:

- We recast long-context inference as continual context consolidation, where useful information from a stream of contexts is written into model weights while preserving previously consolidated knowledge.
- We propose SCoL, a post-training framework that trains an LLM with meta-reinforcement learning to generate textual descriptions of where in its own weights to update, and instantiate SCoL with supervised QA rewards and intrinsic likelihood-based rewards, covering both labeled knowledge incorporation and unlabeled long-context consolidation.
- We evaluate SCoL on SQuAD and LongBench v2, showing a favorable acquisition–retention trade-off on SQuAD, sparse selections aligned with Fisher information, and generalization from short meta-training streams to longer LongBench v2 contexts.

2 Related work

Continual learning for large language models Continual learning studies how models acquire new tasks, domains, or knowledge without overwriting prior behavior. In LLMs, this appears in factual and lifelong model editing (Meng et al., 2022; Mitchell et al., 2022; Wang et al., 2024a; Chen et al., 2024), continual domain adaptation (Gururangan et al., 2020), and continual instruction or alignment tuning (Shi et al., 2025; Chen et al., 2026). Existing methods typically control forgetting by estimating parameter importance or preserving prior behavior through regularization (Kirkpatrick et al., 2017; Li & Hoiem, 2018), or by restricting updates to low-rank, orthogonal, or modular parameter subspaces (Hu et al., 2022; Wang et al., 2023b; Wang & Li, 2024). These approaches are effective but often depend on fixed update rules, replay or distillation signals, hand-designed parameter partitions, or explicit importance estimates that can be costly at LLM scale (van de Ven, 2025). Our work instead learns a context-conditioned update policy: rather than computing weight importance directly, we post-trained the LLM to decide where new contextual knowledge should be inserted so adaptation improves incorporation while minimizing forgetting.

Inference with long context Long-context inference asks whether an LLM can reliably use information that appears far back in the prompt. Although context windows continue to grow, models can still underuse distant evidence and degrade as contexts become longer or more complex (Liu et al., 2024; Hsieh et al., 2024). Existing approaches improve access to long-range information through retrieval or external memory (Lewis et al., 2020; Guu et al., 2020; Borgeaud et al., 2022; Wang et al., 2023a), special attention or memory mechanisms (Mohtashami & Jaggi, 2023; Wang et al., 2023a), context compression (Rae et al., 2019; Chevalier et al., 2023; Jiang et al., 2023), or agentic memory systems (Zhong et al., 2024; Packer et al., 2023; Park et al., 2023; Shinn et al., 2023). These methods are effective but often keep information outside the model weights or compress it before reuse, making performance depend on retrieval quality, memory management, or lossy summarization. We study a complementary direction: consolidating context-derived knowledge into localized parameter updates. This reframes long-context inference as continual knowledge incorporation, where the model must retain useful information from a stream of contexts without overwriting previously consolidated knowledge.

Self-adaptation with reinforcement learning Reinforcement learning is often used to optimize LLMs as policies over text for alignment, reasoning, or task-level rewards (Ouyang et al., 2022; DeepSeek-AI, 2025). In self-adapting LMs, the target instead shifts toward learning *how the model should change* in response to new information, connecting to meta-learning and expert-iteration views of controlling an inner improvement process (Anthony et al., 2017). Recent work learns such adaptation strategies: Transformer-Squared learns task-conditioned self-adaptation with RL (Sun et al., 2025), and SEAL trains an LLM to generate self-edits and update directives (Zweiger et al., 2025). CaMeLS is especially close to our setting because it meta-learns online adaptation over context, but it does so by training a separate small autoregressive model to assign token-level loss weights during fine-tuning (Hu et al., 2023). By contrast, we proposed a post-training paradigm in which the LLM itself learns to propose *where in its own weights* to update, without explicitly computing parameter importance. Since each committed update changes the model used for future contexts, we frame the problem as a meta-RL problem over a drifting model state, with the goal of continual knowledge consolidation under forgetting constraints.

3 Self-Consolidating Language Models

We study a continual consolidation setting in which a single LLM ingests a stream of contexts and must internalize each into its weights, without replay and without retrieval at inference time. Section 3.1 formalizes continual context consolidation as a reinforcement learning problem whose reward decomposes into an acquisition term and a forgetting term. Section 3.2 presents SCoL, a meta-reinforcement learning procedure that trains the LLM to generate textual update-location selections under a drifting model state. Section 3.3 instantiates the reward under supervised and sparse-supervision regimes. The full procedure is given in Algorithm 1 in Appendix A.

3.1 Problem setup: continual context consolidation

Let π_{θ_0} denote a base language model (our policy) with parameters $\theta_0 \in \mathbb{R}^d$, and let $\mathcal{C} = (c_1, c_2, \ldots, c_T)$ be a stream of contexts, where each $c_t \in \mathcal{X}^*$ is a token sequence. We use *context* generically: it can be a text passage, a multi-turn interaction history, or a long-context window. At step t, the current model $\pi_{\theta_{t-1}}$ receives c_t and generates a *textual* action

$$a_t \sim \pi_{\theta_{t-1}}(\cdot \mid c_t), \tag{1}$$

which is parsed into a structural update units, and in our main instantiation the units are Transformer layers. We then update the current model by:

$$\Delta\theta_t = \mathrm{Adapt}(\theta_{t-1}, c_t, a_t), \qquad \theta_t = \theta_{t-1} \oplus \Delta\theta_t. \tag{2}$$

The prompting template and parsing procedure are described in Section B. The weights θ_t carry all accumulated history. No retrieval index or auxiliary memory is maintained for generation. During training, we keep a past-context set $\mathcal{D}_{<t}$ containing only the material needed to *evaluate* forgetting from steps $1, \ldots, t-1$. At inference time, the model relies on θ_t alone. After each consolidation step, the environment returns

$$r_t(\theta_t; c_t, \mathcal{D}_{<t}) = u(\theta_t; c_t) - \lambda f(\theta_t; \mathcal{D}_{<t}), \tag{3}$$

where u measures acquisition of the current context, f measures drift on previously consolidated material, and $\lambda \geq 0$ trades off acquisition and retention. We keep u and f abstract here and instantiate them in Section 3.3.

The learning objective is to learn a *pre-adaptation* policy π_{θ_0} whose own sequential updates produce high cumulative reward:

$$\max_{\theta_0} \mathcal{J}(\theta_0) = \mathbb{E}_{\substack{a_t \sim \pi_{\theta_{t-1}}(\cdot|c_t) \\ \theta_t = \theta_{t-1} \oplus \mathrm{Adapt}(\theta_{t-1}, c_t, m_t)}} \left[\sum_{t=1}^{T} r_t(\theta_t; c_t, \mathcal{D}_{<t}) \right]. \tag{4}$$

The key point is that the policy parameters drift during the stream: the action at step t is sampled from $\pi_{\theta_{t-1}}$, whose parameters already contain all committed updates from earlier contexts. Thus, the goal is not to learn a fixed context-to-update mapping, but to meta-learn an initialization whose rolled-out updates continue to produce good future update selections.

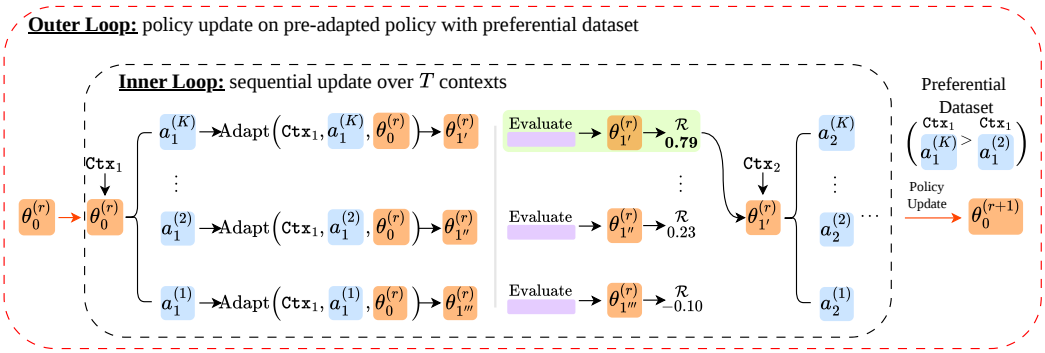

Figure 1: Training pipeline of the Self-Consolidating Language Model (SCoL). **Inner loop (black):** at each context Ctx_t, the current model state $\theta_{t-1}^{(r)}$ samples K textual actions, adapts a candidate model, and scores it with the reward in Equation (3). The highest-reward candidate is committed, advancing the running model state. **Outer loop (red):** after the stream is exhausted, the induced preferential dataset trains the round-start policy $\theta_0^{(r)}$ with IPO, producing $\theta_0^{(r+1)}$ for the next round.

Several nearby paradigms share aspects of this problem but differ in a central architectural commitment. RAG (Lewis et al., 2020) and memory-augmented architectures (Packer et al., 2023; Chevalier et al., 2023) handle new information by expanding an external store while leaving model parameters fixed. Knowledge-editing methods (Meng et al., 2022; 2023; Mitchell et al., 2022; Wang et al., 2024b) target isolated factual updates rather than streaming accumulation and general context learning. Batch fine-tuning (Howard & Ruder, 2018; Raffel et al., 2020) requires simultaneous access to the full $\mathcal{C}$. SCoL instead commits each incoming context into the model weights online, while learning where such updates should occur.

3.2 Learning to Learn without Forgetting with Meta-Reinforcement Learning

SCoL parameterizes the adaptation decision as text generation. For each context, the LLM emits a textual list of update locations. We then attach LoRA modules (Hu et al., 2022) to the selected structural units and perform a CLM update on c_t:

$$\Delta\theta^\star(a_t) = \arg\min_{\Delta\theta_{a_t}} \mathbb{E}_{x\sim c_t}\left[-\log p_{\theta_{t-1}+\Delta\theta_{a_t}}(x)\right]. \tag{5}$$

This yields the candidate post consolidation model

$$\theta_t^{(a)} = \theta_{t-1} \oplus \Delta\theta^\star(a). \tag{6}$$

Across candidates, the inner-loop hyperparameters are fixed; candidates differ only in which structural units receive adapters. Thus, the learned decision is where to adapt, not how strongly to adapt. At each context, we sample K candidate textual actions from the current, already-drifted policy:

$$a_t^{(1)}, \ldots, a_t^{(K)} \sim \pi_{\theta_{t-1}}(\cdot \mid c_t). \tag{7}$$

Each action is parsed, adapted, and scored, and we commit the highest-reward candidate:

$$r_t^{(k)} = r_t\left(\theta_t^{(a_t^{(k)})}; c_t, \mathcal{D}_{<t}\right), \quad k^\star = \arg\max_k r_t^{(k)}, \quad \theta_t \leftarrow \theta_t^{(a_t^{(k^\star)})}. \tag{8}$$

This committed update changes the model that will generate the next action: π_{θ_t} is both the consolidated model and the policy used at the next context. The K candidate scores also induce preference data. For each context, we construct

$$\mathcal{P}_t = \left\{(c_t, a^w, a^l) : r_t(a^w) > r_t(a^l)\right\}, \tag{9}$$

where a^w and a^l denote higher- and lower-reward textual actions for the same context. Across the full inner stream, we collect $\mathcal{P}^{(r)} = \bigcup_{t=1}^{T} \mathcal{P}_t$.

After a full round of T sequential consolidations, we update the *pre-adaptation* parameters $\theta_0^{(r)}$, not the final drifted parameters $\theta_T^{(r)}$. This is the meta-learning step: the preference data are generated by

rolling the current policy through a stream of self-induced updates, but the outer update improves the round-start model so that its future rollouts generate better update-location selections. We use Identity Preference Optimisation (IPO) (Azar et al., 2024):

$$\mathcal{L}_{\mathrm{IPO}}(\theta) = \mathbb{E}_{(c,a^w,a^l)\sim\mathcal{P}^{(r)}}\left[\left(\log\frac{\pi_\theta(a^w \mid c)}{\pi_{\mathrm{ref}}(a^w \mid c)} - \log\frac{\pi_\theta(a^l \mid c)}{\pi_{\mathrm{ref}}(a^l \mid c)} - \frac{1}{2\beta}\right)^2\right], \tag{10}$$

with $\pi_{\mathrm{ref}} = \pi_{\theta_0^{(r)}}$ fixed to the round-start policy. The minimizer becomes the next round-start policy,

$$\theta_0^{(r+1)} = \arg\min_\theta \mathcal{L}_{\mathrm{IPO}}(\theta). \tag{11}$$

The next round begins from $\theta_0^{(r+1)}$ and produces a fresh drifting trajectory $\theta_1^{(r+1)}, \ldots, \theta_T^{(r+1)}$. We use IPO rather than supervised fine-tuning on the top-1 action because the action space consists of low-entropy structured strings. In preliminary experiments, SFT-style reinforcement such as ReST (Gulcehre et al., 2023) quickly concentrated probability mass on a few repeated layer lists. IPO instead stops updating once the chosen–rejected log-ratio reaches the finite margin $1/(2\beta)$, which helps preserve diversity among plausible update-location selections. Additional comparisons between outer-loop update algorithms are provided in Section B.11.

3.3 Reward instantiations and reward sparsity

The abstract reward of Equation (3) admits two instantiations, distinguished by whether a context arrives with downstream supervision.

When c_t is accompanied by a set of held-out queries $\mathcal{Q}_t = \{(q, a)\}$ tied to the context, both reward terms are measured in the same units of downstream performance. The acquisition term is the accuracy on $\mathcal{Q}_t$ of the candidate post-consolidation model. The forgetting term is the accumulated degradation on past query sets, measured against each past context's own first-consolidation baseline. Concretely, for each past step $s < t$ we cache the accuracy $b_s = \mathrm{Acc}_{\mathcal{Q}_s}(\theta_{s+1})$ attained immediately after c_s was first consolidated, and we penalise any drop from this baseline under the candidate state $\theta_t^{(a)}$. That is, $\mathcal{D}_{<t} = \{(c_s, \mathcal{Q}_s, b_s)\}_{s<t}$ in this regime, and

$$r\big(a; c_t, \theta_t\big) \;=\; \mathrm{Acc}_{\mathcal{Q}_t}\big(\theta_t^{(a)}\big) \;-\; \frac{\lambda}{t-1}\sum_{s=1}^{t-1}\Big[b_s \;-\; \mathrm{Acc}_{\mathcal{Q}_s}\big(\theta_t^{(a)}\big)\Big]. \tag{12}$$

Many settings of interest do not provide such supervision. Contexts stream in without labels; downstream performance is unavailable, noisy, or delayed. In this regime $\mathcal{D}_{<t} = \{c_s\}_{s<t}$, and we instantiate both reward terms intrinsically. A well-consolidated model θ' should raise the likelihood of c_t relative to the pre-adaptation model θ_t,

$$u_{\mathrm{intrinsic}}(\theta'; c_t) \;=\; \log p_{\theta'}(c_t) \;-\; \log p_{\theta_t}(c_t), \tag{13}$$

and maintain the likelihood of $x \in \mathcal{D}_{<t}$ as measured by the output-distribution drift on previously-consolidated material. Because $\mathcal{D}_{<t}$ is an empirical collection of past contexts rather than samples drawn from the model, absolute log-likelihood differences can disproportionately weight passages with higher baseline likelihood or lower entropy, leading to uneven retention where some contexts dominate the forgetting signal. To ensure uniform retention across previously consolidated material, we instead measure forgetting in relative terms by normalizing the change in log-likelihood by its pre-adaptation value. This yields a notion of fractional degradation that treats each passage comparably, regardless of scale, and better aligns with the objective of preserving all contexts without bias. Concretely, we define

$$\widetilde{f}_{\mathrm{intrinsic}}(\theta'; \mathcal{D}_{<t}) \;=\; \mathbb{E}_{x\sim\mathcal{D}_{<t}}\left[\frac{\log p_{\theta_t}(x) - \log p_{\theta'}(x)}{\big|\log p_{\theta_t}(x)\big|}\right] = \mathbb{E}_{x\sim\mathcal{D}_{<t}}\left[\frac{u_{\mathrm{intrinsic}}(\theta'; x)}{\log p_{\theta_t}(x)}\right], \tag{14}$$

While this normalization ensures that each passage contributes comparably, it removes the original scale of log-likelihood differences. To restore a meaningful magnitude to the forgetting signal, we rescale by the average pre-adaptation log-likelihood over $\mathcal{D}_{<t}$, yielding

$$f_{\mathrm{intrinsic}}(\theta'; \mathcal{D}_{<t}) \;=\; \widetilde{f}_{\mathrm{intrinsic}}(\theta'; \mathcal{D}_{<t}) \cdot \mathbb{E}_{x\sim\mathcal{D}_{<t}}\left[\big|\log p_{\theta_t}(x)\big|\right], \tag{15}$$

In both regimes the forgetting term is a drop relative to a per-context baseline evaluated on the same test material, differing only in whether that material is labeled queries or the raw context. Substituting Equations (13) and (15) into Equation (3) yields

$$r_{\text{sparse}}(a; c_t, \theta_t) \;=\; u_{\text{intrinsic}}(\theta_t^{(a)}; c_t) \;-\; \lambda\, f_{\text{intrinsic}}(\theta_t^{(a)}; \mathcal{D}_{<t}), \tag{16}$$

applicable to rolling-window long-context consolidation (Chen et al., 2023; Chevalier et al., 2023), memory-compaction in agentic interaction histories (Packer et al., 2023; Mu et al., 2023), and any setting in which a context arrives unlabeled.

We then evaluate our self-consolidating LMs framework under the above two reward instantiations: context that comes with immediate downstream supervisions (Section 4), and context that has no downstream supervisions (Section 5).

4 Continual Knowledge Injection

4.1 Experiment Setup

Task and dataset. We evaluate continual knowledge incorporation, in which a language model is updated in sequence on a stream of SQuAD (Rajpurkar et al., 2016) passages and held responsible for answering a short question-answer set attached to each passage. We prepare a disjoint training split used for meta-training the policy and a validation split reserved for sequential-update evaluation. Additional dataset details are provided in Section B.7.

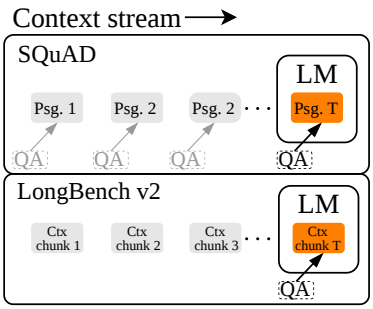

Figure 2: Illustrations of experiment setup under two reward settings. **Top**: each context (SQuAD passages) comes with an environment reward (downstream QA). **Bottom**: QA is evaluated at the end of the context stream.

Policy, inner adaptation, and meta update. We choose the policy as a `Qwen2.5-7B-Instruct` model: at each context, the model is prompted to emit a list of up to $B = 10$ transformer-layer indices. Additional results for finer granularities (i.e, projection layers) are discussed in Section B.12. Our inner adaptation follows SEAL setting: for every candidate action, we attach a LoRA adapter to the selected layers and run the CLM update of Equation (5) on a set of implications pre-generated once per passage by a frozen SEAL-pretrained model, for 10 epochs at learning rate 2×10^{-4} and batch size 1. During meta-training we stream $T = 50$ passages per round; at each passage we sample $K = 10$ candidates actions from the current policy, score each candidate under of Equation (12) using a frozen `Qwen2.5-7B-Instruct` as a judge, commit the argmax to the running weights per Equation (8), and accumulate pairwise comparisons of reward scores into a round-level buffer. At the end of each round the buffer is used to reinforce the *pre-adaptation* base policy with IPO (Equation (10)), producing the base policy that seeds the next round; we report $R = 2$ such rounds. During evaluation, LLM only emits the selection once (i.e. $K = 1$) and commits the update. Prompts and additional training detailes are in Sections B.1 to B.5.

Baselines and metrics. We compare against three baselines and three SCoL variants. All methods use the same pre-generated implications and inner LoRA update procedure, so the comparison isolates the effect of *where* adaptation is applied. **QA Prompting** prompts the base model with the question only. **Batch TTT** trains a LoRA adapter jointly on all evaluation passages in an offline pass, serving as a batch-access upper bound. **SEAL**$_{\text{continual}}$ applies the SEAL knowledge-incorporation procedure to the continual stream and serves as the continual LoRA fine-tuning baseline. **SCoL**$_{\lambda=0}$ removes the forgetting term from our reward, isolating its contribution. **SCoL** uses the full reward with $\lambda = 1$.

At evaluation we apply each method to a fixed validation stream of $N = 100$ passages and record the sequential-update accuracy matrix $M \in [0, 1]^{N \times N}$ whose entry $M_{i,j}$ is the accuracy on the j-th passage's queries after the model has been edited through the i-th passage (for $j \leq i$). From M we report two metrics for each round: **immediate acquisition**, the mean of the diagonal, measuring how well each update works at the moment it is committed and **retention**, the mean of the final-row entries on past passages, measuring accuracy on everything learned earlier once the full stream has been processed. Metric formulas and baseline details are given in Sections B.6 and B.8.

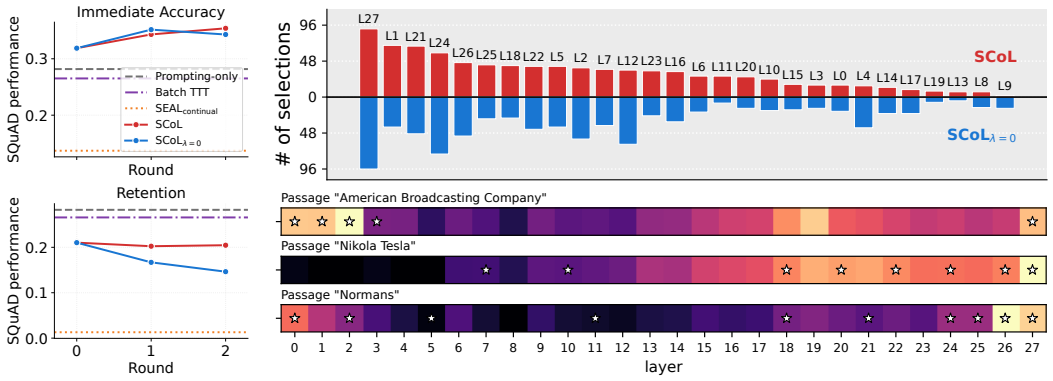

Figure 3: **Left:** immediate accuracy (top) and retention (bottom) across IPO updates. **Top Right:** top layer selection frequencies after the final IPO update, with the full reward variant in red and the acquisition only variant in blue. **Bottom Right:** examples of per-layer weight importance measured by Fisher information (brighter is higher) with SCoL's selected layers marked by white stars.

4.2 Results and Discussions

Table 1 shows that both SCoL and SCoL$_{\lambda=0}$ exceed all three baselines in immediate accuracy, indicating that the learned policy improves acquisition beyond prompting, batch TTT, and sequential updating. QA Prompting achieves limited accuracy because many SQuAD questions require passage-specific information that general knowledge alone cannot reliably provide. Batch TTT performs worse than prompting only, despite seeing all 100 passages and implications jointly, with an immediate accuracy drop from 28.17% to 26.52%. We interpret this as a declarative training, query evaluation mismatch. The adapter is trained on passage text and implication strings under language modeling, but is later queried through QA. This resembles the reversal curse (Berglund et al., 2024), where facts learned in one direction are not reliably retrieved in another. SEAL collapses under sequential updating, consistent with compounding drift in parameter efficient continual tuning (Ren et al., 2024; Wang et al., 2023b). We refer to Appendix B.15 for additional ablation studies on QA instruction following and knowledge retention in sequential updates. Retention clarifies the role of the forgetting term. Prompting only and Batch TTT are no update reference points whose retention equals to their immediate score. SEAL retains only 1.30% accuracy after sequential incorporation. In contrast, SCoL$_{\lambda=0}$ achieves stronger immediate acquisition, but its retention remains limited at 14.64%. Using forgetting reward f recovers retention to 20.46% while also improving immediate accuracy. This suggests that explicitly rewarding retention can substantially reduce catastrophic forgetting in a self-improving language model framework (Zweiger et al., 2025). Figure 3 further shows that retention decays monotonically without f, whereas SCoL keeps retention nearly stationary and continues improving acquisition. The layer selection patterns in Figure 3 indicate that the learned policy is not selecting arbitrary update sites. Both SCoL and SCoL$_{\lambda=0}$ concentrate their selections in the last layer ($L27$). This agrees with prior evidence that late blocks in autoregressive Transformers act as storage and retrieval sites for factual associations (Geva et al., 2021; Meng et al., 2022). We further test a fixed-selection ablation that always updates the last 10 layers. It achieves 28.78% immediate accuracy and 20.10% retention, close to but below SCoL's 20.46%. This supports the view that late layers preserve factual associations, and that SCoL learns to exploit this structure, while its stronger immediate accuracy suggests that context-dependent layer selection is more robust than a fixed late-layer strategy. This combination of higher acquisition and comparable retention yields a favorable acquisition–retention trade-off under the same 10-layer update budget. We refer to Appendix B.16 for additional middle-10 results and paired comparisons.

Table 1: Comparison of immediate accuracy and retention across methods on continual knowledge incorporation over 100 SQuAD passages.

Method	**Immediate Acc.**	**Retention**
Prompting-only	28.17%	–
Batch TTT	26.52%	–
SEAL$_{\text{continual}}$	13.70%	1.30%
SCoL$_{\lambda=0}$	34.30%	14.64%
SCoL	**35.40%**	**20.46%**

At the same time, both variants avoid several early and middle layers, with $L8$ and $L9$ rarely selected and changing little across IPO rounds (see Figure 7). We refer to Appendix B.9 for additional results and standard errors.

Fisher alignment. We then ask what the LLM is learning to select, and how these generated layer selections contribute to retention. Since Fisher information measures where the current passage loss is most sensitive in the running model, it provides a direct diagnostic for whether SCoL routes adaptation toward steep layerwise directions while keeping updates sparse. We measure layerwise Fisher information on the running model's base projection weights from each passage's implication lines, with details in Appendix B.14. Given the layer selection set $S(c)$ generated for context c, we measure alignment as $\mathrm{recall@}k(c) = |S(c) \cap \mathrm{Top}_{k(c)}(F(c))|/k(c)$, where $k(c) = |S(c)|$ and $F(c)$ is the layerwise Fisher vector. A random selection of 10 layers (out of 28 total layers) has expected recall $10/28 = 35.7\%$. Table 2 shows that SCoL reaches $45.1\%_{\pm 1.6\%}$, while $\text{SCoL}_{\lambda=0}$ reaches $41.0\%_{\pm 1.3\%}$, both clearly above the random selection baseline. This provides evidence that the learned selections are not merely produced by the prompt format and do not degenerate to random sparse choices, and shows that IPO reinforces the LLM to generate update locations that follow high sensitivity layers for the current passage. Figure 3 shows SCoL selections overlaid on Fisher information heatmaps for sampled SQuAD passages, where the generated layer selections concentrate on high (bright) Fisher layers. As a result, SCoL updates layers that are most responsive to the current passage, but does so under a sparse selection pattern that leaves most layers untouched. Such selective plasticity matches the principle behind continual learning methods that reduce catastrophic interference by restricting adaptation to sparse, task specific update pathways (Zenke et al., 2017; Mallya & Lazebnik, 2018; Serrà et al., 2018).

Table 2: Layerwise Fisher alignment on 100 SQuAD passages. Values report `recall@k(c)` against the Fisher top-$k(c)$ layers for learned and random selections.

	`recall@k(c)`	
Method	Learned selection	Random selection
SCoL	$\mathbf{45.1}_{\pm 1.6}$	35.7
$\text{SCoL}_{\lambda=0}$	$41.0_{\pm 1.3}$	35.7

5 Long-context consolidation

5.1 Experiment setup

Task, dataset, and evaluation. In many real world settings, models often receive information as a continuous stream (e.g., daily updates to knowledge base) without intermediate supervision, and must answer queries based on accumulated context. To simulate this setting, we use LongBench-v2 (Bai et al., 2025), where each example consists of a long context paired with a question requiring reasoning over the full passage. We treat each passage as a sequential stream of fixed-length (2048-token) segments. The objective is to consolidate information from the stream into model parameters, such that the model can answer the final query without access to the original context. To evaluate both consolidation and generalization, we consider two regimes: (1) **short**, with context lengths of 16k–32k tokens, and (2) **long**, with context lengths of 32k–64k tokens. The model is trained only on the short regime and evaluated on both. We evaluate on 40 short passages and 20 long passages. Due to the sparsity of the reward signal, evaluation is performed by sampling 10 candidate answers after consolidation and measuring mean accuracy. Additional details are provided in Section C.2.

Training procedure. We use `Qwen2.5-7B-Instruct` as our policy and reuse hyperparameters from Section 4. Meta-training consists of 2 rounds, each processing 5 passages (or streams). For each passage, at each step, we sample $K = 10$ candidate actions, perform inner-loop adaptation and use the intrinsic reward defined in Equation (16) to select the highest-scoring candidate. The selected adapter is merged into the running model (Equation (8)) before proceeding to the next segment.

Baselines. We compare methods across three settings: **(1) In-context. Base**: the base model answers questions without additional context. **Full Context**: the base model is provided the entire passage. **Summarization**: the base model is given a concise summarization of the passage by `Kimi-K2.6` (Moonshot AI, 2026). **(2) Test-time training. Batch TTT**: a single LoRA adapter is trained on the full passage before querying. **Sequential FT**: Sequentially LoRA Fine-Tuning the

model over passage segments without any module selection. **(3) Self-consolidation. SCoL**: our framework.

5.2 Results and discussions

Table 3 summarizes performance across both context regimes. SCoL achieves the strongest performance in the short-context setting at 42.3%, outperforming all baselines. In the long-context regime, SCoL achieves 37.0%, outperforming Sequential FT and remaining competitive with Full Context and Summarization. These results demonstrate that explicit consolidation into model weights effectively integrates information across context lengths without requiring full access to the sequence. Batch TTT achieves the highest accuracy in this regime at 41.5%. In contrast, Sequential FT exhibits a performance collapse as context length increases ($36.5\% \rightarrow 23.0\%$), underscoring the challenge of catastrophic forgetting under naïve online updates. SCoL consistently outperforms Sequential FT, confirming that structured consolidation is critical for stable continual adaptation. In the in-context setting, providing full context or a Kimi-K2.6 summary improves over the base model, but these gains diminish as context length increases. This degradation, consistent with prior findings that context length alone can hurt performance (Du et al., 2025), emphasizes the limitations of relying solely on context or lossy summarization and motivates the transition to persistent parameter-based consolidation.

Table 3: Long-context consolidation comparison across two context-length regimes (short: 16k–32k and long: 32k–64k tokens).

Method	short	long
Base	32.5%	29.5%
Full Context	38.0%	35.5%
Summarization	39.3%	36.0%
Batch TTT	<u>39.5%</u>	**41.5%**
Sequential FT	36.5%	23.0%
SCoL	**42.3%**	<u>37.0%</u>

Overall, weight consolidation provides a competitive alternative to in-context methods, with SCoL leading on short contexts and Batch TTT achieving the highest accuracy on long contexts. Notably, although SCoL is meta-trained on short-context examples, it generalizes effectively to the long-context regime, handling sequences up to twice the training length while remaining competitive with methods that operate on full context. This suggests that our consolidation mechanism captures length-agnostic structure and can extrapolate beyond its training distribution. We hypothesize that further gains for SCoL could be achieved by scaling meta-training to longer sequences or dynamically adapting the strength of the forgetting term λ as a function of context length. We refer to section C.3 for additional results, discussions, and standard errors.

6 Conclusion, limitations, and future work

We introduced SCoL, a post-training framework that treats long-context inference as continual context consolidation. Instead of keeping new information only in the prompt or an external memory, SCoL post-trains the LLM to generate textual descriptions of where the current context should be written into its own weights. We formulate consolidation as a meta-reinforcement learning problem, where each committed update changes the model state that will generate future update selections. SCoL improves acquisition and retention over sequential finetuning on SQuAD and outperforms Sequential FT in both LongBench v2 regimes. Under the same update budget, SCoL also achieves higher immediate accuracy than fixed layer selection on SQuAD while maintaining comparable retention. An open-book QA probe on held-out passages further shows that SCoL largely preserves its ability to answer questions from the supplied context after sequential updates. Analysis of learned selection patterns shows that SCoL encourages sparse update locations that align with layers assigned high Fisher information, suggesting that the model learns to route plasticity toward loss-sensitive regions while limiting interference. SCoL also remains competitive with in-context methods on longer LongBench v2 streams, even though meta-training uses only short streams.

Limitations. SCoL can exhibit mode collapse over additional rounds of IPO training, concentrating on repeated layer selections and reducing context-dependent diversity (see Appendix B.13). Future work should study policy update algorithms with improved diversity regularization, such as KL control or batch-level entropy penalties.

AI USE STATEMENT

In this work, we used generative AI tools for codebase organization and tracking experimental results. We have not used generative AI tools for generating the core research ideas or conducting autonomous research, and tasks involving formal proofs are not applicable to this work. Additionally, we used generative AI tools for manuscript writing and proofreading. We have reviewed all AI-assisted work. The authors checked the AI-generated text and code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We conduct our experiments on publicly available benchmarks using existing language models, without involving human participants. Because knowledge stored in model weights can persist beyond the original context, applications should consider data privacy and the risk of retaining inaccurate or biased information.

REPRODUCIBILITY STATEMENT

We describe the learning procedure in Section 3, with pseudocode in Appendix A. Appendices B and C provide the data processing, training configurations, and evaluation details needed to reproduce the experiments.

REFERENCES

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017. URL https://arxiv.org/abs/1705.08439.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024. doi: 10.48550/arXiv.2310.12036. URL https://arxiv.org/abs/2310.12036.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on "A is B" fail to learn "B is A". In *International Conference on Learning Representations*, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International*

Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Hongyang Chen, Zhongwu Sun, Hongfei Ye, Kunchi Li, and Xuemin Lin. Continual learning in large language models: Methods, challenges, and opportunities. *arXiv preprint arXiv:2603.12658*, 2026. URL https://arxiv.org/abs/2603.12658.

Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue'. Lifelong knowledge editing for LLMs with retrieval-augmented continuous prompt learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13565–13580, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.751. URL https://aclanthology.org/2024.emnlp-main.751/.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023. URL https://arxiv.org/abs/2306.15595.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

DeepSeek-AI. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. Context length alone hurts llm performance despite perfect retrieval. *arXiv preprint arXiv:2510.05381*, 2025.

Yadin Dudai, Avi Karni, and Jan Born. The consolidation and transformation of memory. *Neuron*, 88 (1):20–32, 2015. doi: 10.1016/j.neuron.2015.09.004.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023. doi: 10.48550/arXiv.2308.08998. URL https://arxiv.org/abs/2308.08998.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740/.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 328–339, 2018. URL https://arxiv.org/abs/1801.06146.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

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 (ICLR)*, 2022. URL https://arxiv.org/abs/2106.09685.

Nathan Hu, Eric Mitchell, Christopher Manning, and Chelsea Finn. Meta-learning online adaptation of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4418–4432, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.268. URL https://aclanthology.org/2023.emnlp-main.268/.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

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.

Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20 (7):512–534, 2016. doi: 10.1016/j.tics.2016.05.004.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2018.

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Zhi-Quan Luo, and Ruoyu Sun. Preserving diversity in supervised fine-tuning of large language models. *arXiv preprint arXiv:2408.16673*, 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995. doi: 10.1037/0033-295X.102.3.419.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989. doi: 10.1016/S0079-7421(08)60536-8.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022. URL https://arxiv.org/abs/2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.07229.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2110.11309.

Amirkeivan Mohtashami and Martin Jaggi. Random-access infinite context length for transformers. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://openreview.net/forum?id=7eHn64wOVy.

Moonshot AI. Kimi k2.6 tech blog: Advancing open-source coding, 2026. URL https://www.kimi.com/blog/kimi-k2-6. Accessed: 2026-05-05.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. URL https://arxiv.org/abs/2304.08467.

Laura O'Mahony, Léo Grinsztajn, Hailey Schoelkopf, and Stella Biderman. Attributing mode collapse in the fine-tuning of large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.18290.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Weijieying Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. Analyzing and reducing catastrophic forgetting in parameter efficient tuning, 2024. URL https://arxiv.org/abs/2402.18865.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4548–4557. PMLR, 2018.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Qi Sun, Edoardo Cetin, and Yujin Tang. Transformer-squared: Self-adaptive llms. *arXiv preprint arXiv:2501.06252*, 2025. URL https://arxiv.org/abs/2501.06252.

Gido M. van de Ven. On the computation of the fisher information in continual learning. *arXiv preprint arXiv:2502.11756*, 2025. URL https://arxiv.org/abs/2502.11756. To appear in the ICLR 2025 Blogpost Track.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797, 2024a.

Renzhi Wang and Piji Li. LEMoE: Advanced mixture of experts adaptor for lifelong model editing of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2551–2575, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.149. URL https://aclanthology.org/2024.emnlp-main.149/.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*, 2024b. URL https://arxiv.org/abs/2310.16218.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://openreview.net/forum?id=BryMFPQ4L6.

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, Singapore, 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JsNUE84Hxi.

A TRAINING ALGORITHM

Algorithm 1 gives the complete training procedure described in Section 3.2, including the sequential inner updates and the outer policy update.

Algorithm 1 Self-Consolidating LM Training: Learning to Learn Without Forgetting

Require: $\theta_0^{(1)}$, stream $\mathcal{C} = (c_1, \ldots, c_T)$, K, λ, rounds R
Ensure: trained policy $\pi_{\theta_0^{(R+1)}}$

for $r = 1, \ldots, R$ **do**
  $\theta_0^{(r)} \leftarrow \theta_0^{(r)}$; $\mathcal{P}^{(r)} \leftarrow \varnothing$
  for $t = 1, \ldots, T$ **do**
    $a_t^{(1)}, \ldots, a_t^{(K)} \sim \pi_{\theta_{t-1}^{(r)}}(\cdot \mid c_t)$
    for $k = 1, \ldots, K$ **do**
      $\Delta_t^{(k)} \leftarrow \textsc{Adapt}(\theta_{t-1}^{(r)}, c_t, a_t^{(k)})$
      $\theta_t^{(r,k)} \leftarrow \theta_{t-1}^{(r)} \oplus \Delta_t^{(k)}$
      $r_t^{(k)} \leftarrow u(\theta_t^{(r,k)}; c_t) - \lambda f(\tilde{\theta}_t^{(r,k)}; \mathcal{D}_{<t})$
    end for
    append $\{(a^w, a^l, c_t) : r_w > r_l\}$ to $\mathcal{P}$
    $k^\star \leftarrow \arg\max_k r_t^{(k)}$
    $\theta_t^{(r)} \leftarrow \tilde{\theta}_t^{(r,k^\star)}$
  end for
  $\theta_0^{(r+1)} \leftarrow \arg\min_\theta \mathcal{L}_{\mathrm{IPO}}(\theta; \mathcal{P}^{(r)})$
end for
return $\pi_{\theta_0^{(R+1)}}$

B CONTINUAL KNOWLEDGE INJECTION: IMPLEMENTATION DETAILS

This appendix collects the reproducibility details for the experiments of Section 4.

B.1 SELECTION PROMPT

The selection prompt is a Qwen chat-template instance shown in Figure 4. The prompt is parameterised by the passage title, the passage context, the implication string, the per-round budget B, and the maximum layer index. We instantiate $B = 10$ and the maximum layer index at 27 for Qwen2.5-7B.

```
<|im_start|>system
You are an assistant that selects which Transformer layers to update
so the model best memorizes the given passage and its implications.
<|im_end|>
<|im_start|>user
{title}
{context}

Implications:
{implications}

Select up to {budget} layer indices most critical for encoding this
knowledge. Each layer is an integer in [0, {max_layer}]. Be selective.
Output ONLY comma-separated integers, no descriptions.<|im_end|>
<|im_start|>assistant
```

Figure 4: Selection prompt template.

The model's response is parsed deterministically: all integers in the output are extracted by regex, filtered to the range $[0, L-1]$, deduplicated preserving first-seen order, and truncated to the budget

B. Each retained layer index is expanded to its seven projection matrices for adapter attachment. Malformed outputs that yield an empty parse produce an empty action, which in turn produces a near-zero reward under the inner adaptation of Equation (5); such rollouts are retained in the preference buffer but are, by construction, almost always the rejected element of any pair that includes them.

B.2 Implication generation

Implications are generated by the SEAL-pretrained Qwen checkpoint after its second self-training round, held frozen throughout all our experiments. The generation prompt is shown in Figure 5.

```
<|im_start|>system
You are an assistant tasked with analyzing the provided passage and
producing a list of implications derived directly or indirectly from
the content.
<|im_end|>
<|im_start|>user
{title}
{context}<|im_end|>
<|im_start|>assistant
```

Figure 5: Implication-generation prompt template, used unmodified from SEAL.

Sampling uses temperature 0.7, top-p 0.95, a maximum of 512 new tokens, and a single candidate per passage. The resulting implication string is cached per split and reused verbatim across all runs (ours and baselines), so implication-generation stochasticity does not contribute to variance between methods.

For inner adaptation, each cached implication string is split on `--` and newlines, capped at 30 sequences, and each sequence is templated as `{title}\n{sequence}`. The raw passage context is appended as an additional sequence, so the final training set for one rollout contains at most 31 sequences. This templating is inherited from SEAL.

B.3 Inner LoRA configuration

Table 4 lists the inner adaptation hyperparameters.

Table 4: Inner LoRA configuration for one candidate rollout.

Field	Value
Rank r	32
Alpha α	64
Dropout	0.0
Optimiser	AdamW (Loshchilov & Hutter, 2019)
Learning rate	2×10^{-4}
Epochs	10
Batch size	1
Gradient accumulation	1
Warmup / scheduler	none

The commit policy merges the winning candidate's adapter into the base weights at the end of each context and discards the others, so the running base model θ_t at step t is a single fully-consolidated Qwen2.5-7B checkpoint rather than a stack of adapters.

B.4 Outer IPO configuration

Table 5 lists the outer preference-optimisation configuration.

The reference policy π_{ref} for the round-r IPO update is the round-start base model $\pi_{\theta^{(r)}}$ before any rollout-induced drift, matching the formulation in Section 3.2. After the IPO step, the updated base

Table 5: Outer-loop IPO configuration.

Field	Value
Loss	IPO (Azar et al., 2024)
β	0.5
Preference-margin threshold	0.05 (pairs with $\lvert r_w - r_l \rvert < 0.05$ dropped)
Pairs per context	all $\binom{K}{2} = 45$ surviving pairs
Optimiser	AdamW (Loshchilov & Hutter, 2019)
Learning rate	5×10^{-6}
Gradient accumulation	4
Epochs per round	2
Reference-policy snap	at start of each round; re-anchored after update

model becomes $\pi_{\theta^{(r+1)}}$ and is used both as the new rollout substrate for round $r+1$ and as the new reference policy for that round's outer update.

B.5 Reward and judge

The QA judge is a Qwen2.5-7B-Instruct model served as a separate vLLM instance alongside the editable model. Each judgement receives a single (question, gold answer, student answer) triple through the prompt in Figure 6 and returns a binary decision.

```
You are a grading assistant. Your job is to determine whether a
student's answer correctly answers the question based solely on the
provided gold answer. Do not use any outside knowledge. The student
answer can include additional information, but it must at least
fully convey the gold answer and must not contradict it. Ignore
style, phrasing, or extra details that do not affect correctness.
Respond ONLY with 'yes' or 'no'.
Question: {question}
Gold answer: {gold}
Student answer: {pred}
Is the student answer correct based solely on the gold answer?
Respond 'yes' or 'no'.
```

Figure 6: Judge prompt.

The judge decodes greedily with a maximum of 8 new tokens. A regex of the form `\b(yes|no)\b` is run over the output and the last match is taken as the decision. Outputs whose last match is not "yes" (including outputs that produce no match at all) are scored zero. Accuracy on a single $\mathcal{Q}_t$ is the mean of the binary decisions over the roughly five questions in that passage.

B.6 Metric formulas

All main-text metrics are summary statistics of the sequential-edit accuracy matrix $M \in [0,1]^{N \times N}$, with $N = 100$ the length of the evaluation stream. Writing $M_{i,j}$ for the accuracy on $\mathcal{Q}_j$ after editing through c_i ($j \le i$), we report

$$\begin{aligned} \text{Immediate acquisition} &= \tfrac{1}{N} \textstyle\sum_i M_{i,i}, \\ \text{Retention} &= \tfrac{1}{N-1} \textstyle\sum_{j<N-1} M_{N-1,j}, \end{aligned}$$

B.7 Stream and round configuration

Table 6 lists the continual-stream configuration.

The training cache contains 250 shuffled passages; Section 4 uses the first two disjoint slices of 50 passages each. The remaining 150 passages are reserved for extended-rounds diagnostics (Section B.13) and are not used by any main-text result.

Table 6: Stream configuration for Section 4.

Field	Value
Training passages per round T	50
Rounds reported in main text R	2 (plus round 0)
Rollouts per context K	10
Selector sampling temperature	1.0
Shuffle seed	42 (applied once before slicing)
Round-to-passage mapping	R1: passages $[0, 50)$; R2: passages $[50, 100)$
Evaluation stream length	100 passages (fixed order)
Past-query subsampling	none

B.8 Baseline implementation details

QA Prompting. Prompting the base model with question only.

Batch LoRA FT. LoRA attached to all 28 layers, trained jointly on the concatenated implications of all 100 evaluation passages for 10 epochs at learning rate 2×10^{-4}.

SEAL. The SEAL knowledge-incorporation method applied to our continual stream, implemented as a sequential LoRA update per passage with the inner LoRA configuration of Table 4.

SCoL, both variants. Identical configurations; the only difference is the value of the reward weight λ.

B.9 Additional results

See standard errors in Table 7.

Table 7: Standard errors across three inner-training seeds for the round-2 SQuAD results, reported in percentage points.

Method	Immediate acc.	Retention
SEAL	1.7	1.2
SCoL$_{\lambda=0}$	2.0	1.6
SCoL	0.8	1.3

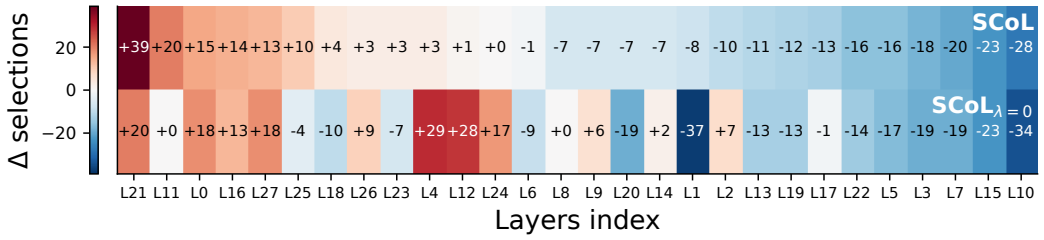

Figure 7: Per-layer change in selection count between the initial and final IPO updates. Layers are sorted by SCoL's largest gain (left) to largest drop (right). SCoL (top, full reward) concentrates onto a handful of layers and abandons most others, while SCoL$_{\lambda=0}$ (bottom, acquisition only) shifts selection mass differently.

B.10 Compute resources and compute overheads

All experiments in this section used NVIDIA RTX PRO 6000 Blackwell GPUs with 96 GB of memory. On one GPU, meta-training took about 4.3 hours per round for T=50 passages and K=10 candidates

Table 8: Outer loop algorithm sweep.

Algorithm	Key hyperparameters	Imm. Acc.	Retention	uniq/100	top1%
IPO	β=0.5	36.60%	25.23%	65	8
IPO	β=0.5, ep= 2, ga= 4, lr= 5×10^{-6}	38.00%	24.87%	73	5
IPO	β=0.1	33.90%	18.38%	81	3
IPO	β=0.01	34.20%	25.26%	79	7
DPO	β=0.1	35.90%	14.91%	83	8
DPO	β=0.5	33.70%	22.17%	77	6
ReST	lr= 5×10^{-5}, ga= 2, k=1	35.90%	26.25%	59	9
ReST	lr= 3×10^{-4}, ga= 10, k=1	41.00%	11.28%	67	7
ReST	lr= 3×10^{-4}, ga= 2, k=3	12.90%	7.52%	14	61

per passage, giving a total of 8.6 GPU-hours for two rounds. Evaluating the full sequential accuracy matrix over N=100 passages took 50 to 110 minutes on one GPU. During meta-training, candidates for each passage share the same starting weights, so they can be adapted and scored in parallel when sufficient devices or memory are available. Updates across passages remain sequential because each committed update determines the starting state for the next passage. LongBench v2 documents form independent streams and can also be processed in parallel. At deployment, the model generates a short list of layers and performs one LoRA update per context (K=1), without candidate search, reward evaluation, or a replay buffer. Generating the selection adds a small cost to sequential fine-tuning, with a total of about 15–20 seconds per context compared with 12–15 seconds for sequential FT or SEAL on one GPU. Queries then run closed-book, so their prompt length does not grow with the amount of consolidated text.

B.11 Outer Loop Algorithm Sweep: IPO, DPO, and ReST

We sweep three outer loop policy update algorithms while holding the inner loop, reward, stream, and rollout budget fixed. Each selector is trained on one round of T=50 passages with K=10 rollouts per passage, layer granularity, λ=1, and evaluated on the same N=100 validation stream. We compare ReST (Gulcehre et al., 2023), DPO (Rafailov et al., 2023), and IPO from Eq. equation 10 (Azar et al., 2024). For each algorithm, we sweep learning rate, gradient accumulation, and the algorithm specific parameters. All other hyperparameters match Table 5.

Following the mode concentration concern in Section 3.2, we report selector uniqueness as a diagnostic. For each evaluated selector, we count the number of *distinct* selection completions emitted across the 100 validation passages, denoted uniq/100. We also report top1%, the share of the most common completion. uniq/100 is necessary but not sufficient for a useful selector, since high uniqueness can also reflect noisy selections. We therefore read it jointly with immediate accuracy.

At β=0.5, IPO gives 36.60% to 38.00% immediate accuracy and 24.87% to 25.23% retention, with 65 to 73 distinct completions. DPO produces 77 to 83 distinct completions, but both DPO settings have lower immediate accuracy and retention than IPO at β=0.5. ReST varies more across configurations. Its best immediate accuracy of 41.00% comes with only 11.28% retention. Another setting reaches 26.25% retention with 35.90% immediate accuracy and 59 distinct completions. In the third setting, the most common selection accounts for 61% of outputs, leaving only 14 distinct completions and 7.52% retention. This concentration on a few selections is consistent with mode collapse under cross entropy updates (O'Mahony et al., 2024; Li et al., 2024). We therefore use IPO with β=0.5 for the main experiments to balance acquisition, retention, and selection diversity.

B.12 Granularity Comparison: Layer, Module, and Per Projection

The action space in Eq. equation 1 ranges over L structural units of the base model. We instantiate three choices in the same codebase, with budgets chosen to keep the per rollout adapter parameter count comparable. The layer setting uses L=28 and budget B=10, where each selected layer attaches LoRA adapters to seven projection modules. The module setting uses L=56 and budget B=20, where each layer is split into attention and MLP units. The per projection setting uses L=196 with budgets B=70 and B=100, where each q, k, v, o, gate, up, and down projection is a separate slot.

Table 9: Round 2 granularity comparison. SCoL uses the full reward. SCoL$_{\lambda=0}$ removes the forgetting term from the reward.

Granularity	B	SCoL			SCoL$_{\lambda=0}$		
		Imm.	Ret.	uniq/100	Imm.	Ret.	uniq/100
Layer	10	35.40%	20.46%	82	34.30%	14.64%	85
Module	20	35.10%	20.21%	100	36.61%	19.45%	95
Per projection	70	24.70%	16.60%	99	31.40%	14.50%	96
Per projection	100	29.80%	20.00%	98	29.50%	17.00%	95

All settings share the inner LoRA configuration in Table 4, the IPO configuration in Table 5, the same stream, $K=10$, and both reward variants.

Layer and module granularity yield comparable Round 2 immediate accuracy, around 35.00% across both reward variants, but the layer configuration is more stable across rounds and is therefore used in the main text. Per projection granularity underperforms by roughly five to ten absolute points in immediate accuracy, despite its larger nominal action space. Since uniq/100 remains high, often above 90, the failure is better explained as selector *noise* rather than selector collapse.

This pattern suggests that action space expressivity is bottlenecked by structured generation. As granularity becomes finer, the policy must emit longer and more constrained outputs, moving from comma separated layer indices to module pairs and then projection pairs. With budgets of 70 or 100, the policy must produce a long list of valid pairs whose indices and projection names obey the parser grammar. A 7B base policy struggles with this output length and schema fidelity, so the effective entropy of useful rollouts shrinks even as the nominal action space grows.

Two future directions follow. First, the gap between layer and per projection granularity may narrow with stronger base policies, since long structured generation is plausibly scaling sensitive (Wei et al., 2022). We state this only as a hypothesis. Second, the outer loop update can be made more diversity preserving at extreme structured lengths, either through entropy regularization (Li et al., 2024) or through an objective that scores list *compositions* rather than independent positions. We leave both directions to future work.

B.13 Extended Round Dynamics and Mode Collapse

We also continue the outer loop beyond the two rounds reported in the main text. This experiment is diagnostic rather than a headline result. We run two additional rounds, R3 and R4, for layer and module granularity under both reward variants. The inner loop, IPO objective, stream, and budget are unchanged. After each round, we record uniq/100 and immediate accuracy.

Table 10: Extended round diagnostics. Each cell reports immediate accuracy and uniq/100.

Granularity and reward	R1	R2	R3	R4
Layer, SCoL	34.32% / 64	35.40% / 82	33.53% / 39	36.52% / 5
Layer, SCoL$_{\lambda=0}$	35.16% / 85	34.30% / 85	37.50% / 69	39.13% / 33
Module, SCoL	27.27% / 93	35.10% / 100	6.68% / 99	10.97% / 63
Module, SCoL$_{\lambda=0}$	31.36% / 96	36.61% / 95	34.29% / 76	27.62% / 7

With layer selection, SCoL's distinct completions fall from 82 at R2 to 5 at R4. The acquisition-only variant retains 33 distinct completions at R4, showing that selection diversity declines faster under the full reward. Module selection also becomes unstable, with the full reward variant's immediate accuracy falling to 6.68% at R3.

Round 2 should therefore be read as an early stopping point for selector quality, not as the asymptotic behavior of the fixed objective. With the same IPO loss and buffer construction, later rounds concentrate probability mass on a small number of high reward actions observed during training and reduce exploration. This is the mode concentration mechanism flagged in Section 3.2 and observed in cross entropy and preference style fine tuning of LLMs (O'Mahony et al., 2024; Li et al., 2024).

For this reason, the main text reports R0 to R2 and treats further round scaling as a regularization problem for future work.

B.14 Layerwise Fisher computation

For each passage c, let $\mathcal{I}(c)$ denote its generated implication lines. Fisher scores are computed on the running consolidated base model before attaching a new LoRA adapter for the passage. For each implication line $x \in \mathcal{I}(c)$, we compute the autoregressive sum loss

$$\mathcal{L}(x;\theta) = -\sum_{t=1}^{|x|} \log p_\theta(x_t \mid x_{<t}),$$

where θ denotes the current consolidated base parameters. We estimate diagonal Fisher importance by squared gradients:

$$F_j(c) = \frac{1}{|\mathcal{I}(c)|} \sum_{x \in \mathcal{I}(c)} \left(\frac{\partial \mathcal{L}(x;\theta)}{\partial \theta_j} \right)^2 .$$

Let $\mathcal{P}_\ell$ be the parameters of the seven base projection matrices in layer ℓ that are eligible for LoRA attachment. The layerwise Fisher score is

$$F_\ell(c) = \sum_{j \in \mathcal{P}_\ell} F_j(c), \qquad \ell \in \{0, \ldots, 27\}.$$

This yields a 28 dimensional Fisher vector $F(c)$ for each passage. These scores are used only for analysis and are never provided to the LLM when it generates layer selections.

B.15 QA-format forgetting and replay

We use an open-book QA probe to test whether sequential updates impair the model's ability to answer questions. Because each prompt includes the relevant passage, the probe helps distinguish loss of QA instruction-following ability from forgetting of consolidated knowledge. The probe contains 115 questions from 25 SQuAD passages held out from meta-training and the 100-passage consolidation stream. Table 11 shows that SEAL's probe accuracy falls from 93.2% to 0.0% after 100 passages, while SCoL retains 90.1%. The meta-trained initialization alone does not explain this difference, as forcing SCoL to update all layers reduces its probe accuracy from 94.1% to 3.1%. With the same total number of gradient updates as SEAL, Batch TTT preserves 93.8% accuracy by training jointly on the passages. These results show that SCoL largely preserves QA instruction-following ability during sequential consolidation. We then repeat the original 100-passage evaluation with QA replay to test whether supplying the format recovers performance. In-context replay adds three QA examples from the held-out passages to the evaluation prompt. For in-weights replay, we train a separate LoRA adapter on held-out passage-QA tuples and their contexts using the CLM objective. This adapter is applied only for evaluation and discarded afterward, so QA training does not enter the committed consolidation stream. As shown in Table 12, in-weights replay raises SEAL's retention from 1.30% to 6.98%. Although replay recovers some of SEAL's lost performance, its retention remains below SCoL's 20.46% without replay and 27.98% with the same replay.

Table 11: Open-book QA accuracy (%) on 115 questions from 25 held-out SQuAD passages as consolidation progresses. SEAL starts from a fresh Qwen model, while both SCoL variants start from the same meta-trained checkpoint. Dashes mark unreported intermediate values.

	Consolidated passages				
Method	0	25	50	75	100
$\text{SEAL}_{\text{continual}}$	93.2	$15.0_{\pm 6.4}$	$1.4_{\pm 0.8}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
SCoL	94.1	$94.4_{\pm 0.6}$	$94.1_{\pm 0.5}$	$91.6_{\pm 1.0}$	$90.1_{\pm 1.9}$
SCoL (all layers)	$94.1_{\pm 0.0}$	$23.7_{\pm 3.1}$	$2.8_{\pm 2.4}$	$0.8_{\pm 0.8}$	$3.1_{\pm 3.1}$
Batch TTT	94.1	—	—	—	$93.8_{\pm 0.3}$
Meta-trained base (no updates)	94.1	—	—	—	94.1

Table 12: Effect of QA replay on the 100-passage evaluation from Table 1. In-context replay provides three held-out QA examples in the prompt. In-weights replay applies a separate QA adapter only during evaluation. Both metrics are percentages.

Method	Immediate accuracy	Retention
Batch TTT	$26.52_{\pm 0.2}$	—
+ in-context replay	$26.12_{\pm 0.5}$	—
+ in-weights replay	$28.81_{\pm 1.4}$	—
$\text{SEAL}_{\text{continual}}$	$13.70_{\pm 1.7}$	$1.30_{\pm 1.2}$
+ in-context replay	$11.27_{\pm 2.2}$	$0.94_{\pm 0.5}$
+ in-weights replay	$14.32_{\pm 1.7}$	$6.98_{\pm 0.8}$
SCoL	$35.40_{\pm 0.8}$	$20.46_{\pm 1.3}$
+ in-context replay	$\mathbf{36.60}_{\pm 1.3}$	$21.91_{\pm 0.4}$
+ in-weights replay	$35.44_{\pm 0.5}$	$\mathbf{27.98}_{\pm 1.0}$

B.16 Fixed-layer controls

We extend the fixed last-10 ablation with a Middle-10 control that updates layers 9–18, keeping the 10-layer budget and all other settings unchanged. Table 13 reports mean performance and paired differences across the same seeds. SCoL achieves higher immediate accuracy than both fixed controls with comparable retention. Relative to $\text{SCoL}_{\lambda=0}$, adding the forgetting-aware reward raises mean retention from 14.64% to 20.46% and mean immediate accuracy from 34.30% to 35.40%.

Table 13: Fixed-layer controls under a 10-layer update budget. All entries show mean ± SE across seeds. Method scores are percentages, and paired differences are in percentage points.

Method	Immediate accuracy	Retention
SCoL	$\mathbf{35.40}_{\pm 0.79}$	$\mathbf{20.46}_{\pm 1.28}$
$\text{SCoL}_{\lambda=0}$	$34.30_{\pm 2.00}$	$14.64_{\pm 1.61}$
Last-10	$28.78_{\pm 1.22}$	$20.10_{\pm 0.30}$
Middle-10	$25.31_{\pm 1.53}$	$19.77_{\pm 0.73}$
Paired comparison	Δ immediate accuracy	Δ retention
SCoL − Last-10	$+6.62_{\pm 2.00}$	$+0.36_{\pm 1.57}$
SCoL − Middle-10	$+10.09_{\pm 2.15}$	$+0.69_{\pm 1.58}$
$\text{SCoL} - \text{SCoL}_{\lambda=0}$	$+1.10_{\pm 2.55}$	$+5.82_{\pm 2.09}$

C Long-context Consolidation

C.1 Hyperparameters

C.1.1 Inner LoRA configuration

We retain the same LoRA configuration from Section B.3 and Table 4. Additionally, all our chunks are of a fixed 2048 token length calculated using the `Qwen2.5-7B` tokenizer and we create a training set by further chunking these into 128 token size subchunks.

C.1.2 Outer IPO configuration

We retain mostly the same IPO configuration from Section B.4 and Table 5. We only change the preference margin threshold from 0.05 to 0.2. That is, given candidates c_1, c_2 with rewards r_1, r_2 respectively and $r_1 > r_2$, we put $\{c_1, c_2\}$ in the IPO buffer iff $r_1 - r_2 > 0.2$.

C.2 Implementation Details

SCoL. We retain the majority of the setup from Section B, with the modifications presented in Section C.1

In-context settings. We use the base `Qwen2.5-7B` model with temperature $= 1$ implemented using vLLM.

For **Summarization**, we prompt `Kimi-K2.6` using the template shown in Figure 8. We set `maxTokens` to 10,000 to discourage degenerate behavior where the model copies large portions of the original passage instead of compressing it.

```
Please provide a dense and concise summary of the following passage in
under
{maxTokens} tokens keeping all the important
information:\n\n{context}\n\nSummary:
```

Figure 8: Summarization prompt used for Kimi-K2.6.

Test-time training. For both Batch TTT and Sequential FT, we attach LoRA adapters to all 28 transformer layers. Training is performed either jointly on the full passage (Batch TTT) or incrementally over sequential chunks (Sequential FT). Each passage is evaluated using freshly initialized LoRA adapters trained under the same inner-loop test-time training configuration as given in Section C.1.

C.2.1 Dataset Processing

Our goal is to evaluate whether long contexts can be continually consolidated into model parameters. We therefore use LongBench-v2, which contains naturally long documents (e.g., books and multi-section texts) paired with reasoning-based question-answer tasks.

Each document is treated as an independent context and processed as a stream of 2048-token segments, simulating a setting where information arrives sequentially. For training, we use 10 passages with lengths between 16k and 32k tokens. We sample 40 different passages for evaluation and an additional 20 passages of lengths between 32k and 64k tokens.

Tokenization and segmentation are performed using the `Qwen2.5-7B` tokenizer. All dataset splits are created with a fixed random seed of 42 for reproducibility.

C.2.2 Compute Resources

All experiments for this section are conducted on NVIDIA H200 GPUs with 8 CPUs. Meta-training (2 rounds over 5 passages each) takes approximately 8 hours. Evaluation over the full benchmark (40 short and 20 long passages) requires approximately 6 hours.

C.3 Additional Results

Passage Log-likelihoods. We further compare the final Log-Likelihoods between batch TTT, sequential FT, and SCoL in Table 14. Consistent with the main results, Batch TTT achieves the highest likelihood in both regimes (-1.26), indicating that full-sequence optimization yields the best fit to the data distribution. In contrast, Sequential FT performs substantially worse (-2.32 short, -2.51 long), reinforcing that naïve online updates lead to degraded representations and poor predictive calibration. SCoL consistently improves over Sequential FT in both regimes (-1.44 vs. -2.32 in short, -1.82 vs. -2.51 in long), indicating that structured consolidation mitigates the degradation in likelihood associated with continual updates. Notably, the gap between SCoL and Batch TTT is smaller in the short-context regime (0.18) than in the long-context regime (0.56), mirroring the trends observed in accuracy.

Table 14: Long-context consolidation comparison of final log-likelihoods across short and long regimes.

Method	**short** (↑)	**long** (↑)
Batch TTT	**-1.26**	**-1.26**
Sequential FT	-2.32	-2.51
SCoL	-1.44	-1.82

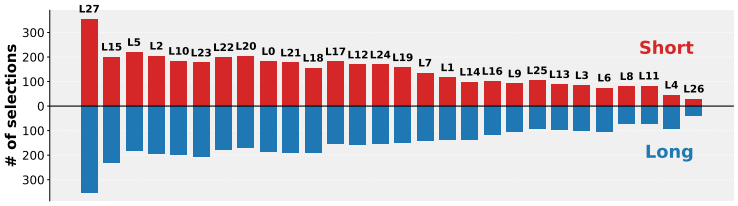

Figure 9: Layer selections for SCoL for the long context setting for short and long regimes.

We hypothesize that this widening gap reflects the same stability–plasticity tradeoff observed earlier. In the short-context setting, SCoL's forgetting-aware objective enables updates that remain close to the optimal batch solution, yielding likelihoods comparable to Batch TTT. However, as context length increases, the constraint imposed by the forgetting term may limit the model's ability to fully adapt to the extended sequence, leading to lower likelihood relative to Batch TTT.

Overall, these results support the conclusion that SCoL provides a strong approximation to batch retraining while maintaining stability, substantially improving over naïve sequential updates, though some loss in optimality remains when full-context adaptation is required.

Layer selection frequencies. We also show the top layer selection frequencies for our Long Context Consolidation Setting in Figure 9. Similar to the results in Section 4, we find that L27 is the selected most often but here we find that L26 and L25 are selected relatively less. Additionally, the middle layers are selected least often matching the SQuAD setting.

Table 15: Long-context consolidation results (mean ± standard error) across two context-length regimes (short: 16k–32k and long: 32k–64k tokens).

Method	short	long
Base	$32.5_{\pm 0.4}\%$	$29.5_{\pm 0.5}\%$
Full Context	$38.0_{\pm 0.5}\%$	$35.5_{\pm 0.8}\%$
Summarization	$39.3_{\pm 0.5}\%$	$36.0_{\pm 0.7}\%$
Batch TTT	$39.5_{\pm 0.5}\%$	$\mathbf{41.5}_{\pm 0.5}\mathbf{\%}$
Sequential FT	$36.5_{\pm 0.8}\%$	$23.0_{\pm 1.0}\%$
SCoL	$\mathbf{42.3}_{\pm 0.8}\mathbf{\%}$	$37.0_{\pm 1.0}\%$

Full results table with standard errors. We report mean accuracy and standard errors across the 10 sampled evaluation answers in Table 15. Each standard error is $\sqrt{v/10}$, where v is the sample variance of the answer scores.

D Broader Impacts

SCoL consolidates streaming context through persistent weight updates, which may support long-term agent memory without keeping the original context in the prompt. However, this shift from transient context to persistent weights introduces privacy risks, as consolidated sensitive information may be harder to redact. Furthermore, continual self-adaptation may inadvertently reinforce biases or internalize misinformation present in the input stream.

E Licenses

We provide details regarding the licenses of external assets used in this work, presented in Table 16.

Asset	URL	License
Datasets		
SQuAD (Rajpurkar et al., 2016)	Link	Creative Commons Attribution-ShareAlike 4.0 International
LongBench-v2 (Bai et al., 2025)	Link	MIT License
Models		
Qwen Models (Bai et al., 2023)	Link	Apache License 2.0*

Table 16: Licenses for assets used in our experiments. *The majority of the Qwen series (e.g., Qwen1.5, Qwen2, Qwen2.5) are licensed under Apache 2.0, with a few exceptions for specific legacy sizes governed by the Tongyi Qianwen License Agreement.