FUNCTIONAL MEMORY COMPILATION: BEHAVIOR-LEVEL MIGRATION OF AGGREGATE LORA MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

When a language model is upgraded, facts accumulated in its memory adapter may become inaccessible through the successor. Migration is difficult when the original interaction history is unavailable. Parameter mappings need not preserve individual answers, while response distillation requires target gradient training. We study migration from an already-trained aggregate low-rank adaptation (LoRA) memory. The source model and retained questions remain accessible, but the questions' original answers are unavailable. We propose Functional Memory Compilation (FMC), which transfers the source's observable answers rather than its parameter coordinates. FMC recovers responses at retained questions and retokenizes them for the successor. It turns selected answer positions into joint constraints on the target's output. Batched forward computation followed by one analytic low-rank fit installs a target-native output-head adapter without backpropagation or optimizer steps. Across the evaluated memory tasks, FMC improves task scores and lowers fixed-pool target installation cost over same-content distillation at the evaluated limited training budget.

1 INTRODUCTION

A model upgrade need not coincide with a reset of everything an assistant has learned in service. Parameter-efficient adapters separate task state from a backbone, and low-rank adaptation (LoRA) stores that state as a compact matrix residual (Houlsby et al., 2019; Hu et al., 2022). Context distillation, document internalization, and online agent memory use such residuals to retain information beyond a prompt (Caccia et al., 2025; Charakorn et al., 2026; Ren et al., 2026). Consider an assistant whose adapter remembers many corrected facts. A new backbone should preserve those corrections, even when the original interaction history is unavailable (Figure 1). We study this post-hoc setting: an already-trained aggregate memory adapter survives, but the original interactions cannot be replayed for target training. Retained answer-free queries still allow us to read accessible memories from the source model.

An adapter's effect depends on its backbone's representations and vocabulary. Cross-LoRA maps parameter subspaces, while Command-V transfers interventions through activation correspondences (Xia et al., 2025; Wang et al., 2025a). Such mappings transform internal updates without directly specifying which answer should remain accessible at each memory query. Response-based transfer makes the desired answers explicit: Trans-LoRA and TiTok train target adapters using source responses (Wang et al., 2024b; Jung & Kim, 2026). Recovered responses offer a route around the missing history, but installing them by distillation still requires iterative target optimization. Our goal is to reduce this adaptation work when carrying existing memories across a model upgrade. We therefore ask whether source-exposed answers can be installed through batched forward computation and one joint analytic fit. The resulting target-native adapter should remain usable after retiring the source model.

FMC uses the responses exposed by the source memory as a migration specification. It retokenizes these values for the successor, collects frozen target representations at selected answer positions, and requests sparse logit corrections. A standard reduced-rank ridge solution (Mukherjee & Zhu, 2011) fits the constraints jointly and produces one target output-head LoRA. The key separation

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.

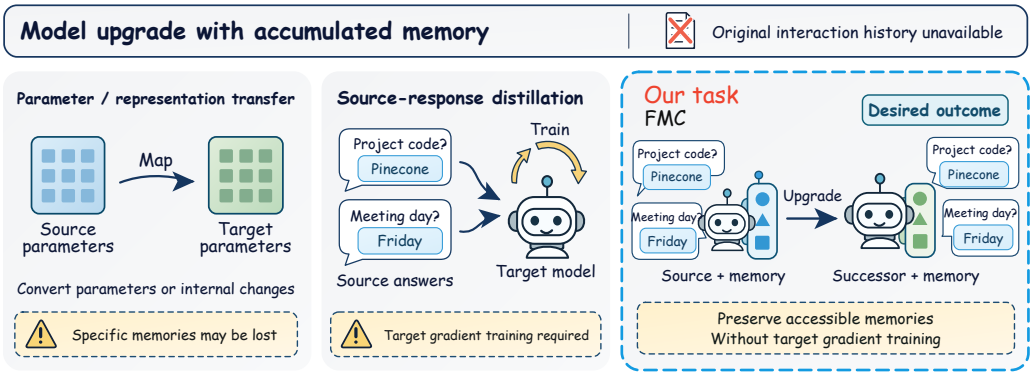

Figure 1: Memory migration without the original interaction history. Parameter or representation mapping may lose individual answers; response distillation uses target gradient training. FMC instead compiles source-exposed answers into a successor adapter. The illustrative right panel shows the desired memory continuity.

is between a model-dependent representation of memory and its observable answer at a retained address. Responses cross the model boundary; hidden coordinates do not. Recovering an answer identifies what to preserve, while the joint fit balances its competing demands with other memories in the target representation space.

The target output head makes this separation computationally useful. Its update does not change the hidden representations of a fixed input sequence. Selected positions can therefore be collected in batches before solving a single joint problem. Answer-position policies connect this formulation to classification and bounded conversational value generation.

Experiments on KnowEdit, LoCoMo, and MemoryAgentBench (Zhang et al., 2024; Maharana et al., 2024; Hu et al., 2026) show higher task scores and lower fixed-pool installation cost than matched-content distillation under the evaluated training budget. Our contributions are threefold:

- We propose FMC, a behavior-level framework for migrating an already-trained aggregate LoRA memory without the original interaction history. It transfers question–answer associations recovered at retained answer-free queries rather than mapping source parameter coordinates.
- We design a unified constraint construction that expresses recovered answers as target-side output requirements. Task-specific answer-position policies connect the same formulation to binary classification, fixed-label classification, and bounded answer generation.
- We develop a batched analytic compiler that collects frozen target representations and jointly fits the resulting constraints into one output-head LoRA. Installation requires no target backpropagation or optimizer updates.

Section 2 reviews related work, Section 3 presents FMC, and Section 4 reports experiments and analyses. Sections 5 and 6 discuss limitations and conclude the paper.

2 Related Work

2.1 Parametric Memory and Editing

Parametric memory methods study how information is written, organized, and accessed within a model. ROME and MEMIT modify factual associations, while MEND learns to generate edits (Meng et al., 2022; 2023; Mitchell et al., 2022a). SERAC and GRACE retain explicit or latent side memories (Mitchell et al., 2022b; Hartvigsen et al., 2023). Temp-LoRA stores context in temporary adapters, while MELO encodes edits in dynamically indexed LoRA blocks (Wang et al.,

2024c; Yu et al., 2024). MoRAM accumulates rank-1 memory experts for continual learning (Lu et al., 2026). WISE and MEMOIR manage access and interference (Wang et al., 2024a; 2025b). Empirical studies characterize LoRA's memory capacity and recall (Back et al., 2026; Xu et al., 2026). AdapterFusion, LoraRetriever, and parametric memory decoding compose or route adapter state (Pfeiffer et al., 2021; Zhao et al., 2024; Ji et al., 2026). Our setting begins after memory writing: the resulting aggregate adapter must migrate to a different backbone, with no original history available for replay.

2.2 Parameter and representation transfer

Cross-model adapter methods make internal updates portable at different stages. Trans-PEFT and LoRA-X constrain source adaptation to support subsequent transfer (Gu et al., 2025; Farhadzadeh et al., 2025). LoRASuite adds target fine-tuning to adapter mappings, while PortLLM studies patches for evolving models (Li et al., 2025b; Khan et al., 2025). Cross-LoRA projects a trained update using base-weight singular subspaces, and Command-V transfers residual interventions through activation profiles and linear converters (Xia et al., 2025; Wang et al., 2025a). MindBridge and OnceEdit build systems for sharing edits across models (Li et al., 2025a; Ma et al., 2025). Related model-merging approaches align parameter symmetries or combine updates relative to a common pretrained reference (Ainsworth et al., 2023; Ilharco et al., 2023). Mapping an internal update can support memory transfer, but does not directly specify which answer should remain accessible at each memory query.

2.3 Response-based transfer

Behavioral transfer instead uses source outputs to supervise a successor. Trans-LoRA retains a few task examples and a discriminator trained during source adaptation to filter generated inputs (Wang et al., 2024b). It then trains the target on source-LoRA responses. TiTok selects samples and masks training losses using source-LoRA on/off log-probability differences, with mask alignment across tokenizers (Jung & Kim, 2026). These methods establish response-based transfer and token selection; FMC's distinction is the target installation mechanism. It converts answers at retained addresses into one joint analytic output-head fit, rather than iteratively optimizing a target training loss. Our matched-content baseline compares these installation choices on the same recovered pairs (Appendix B.2). Thus the question addressed here is how to compile individually exposed memories into a shared target adapter. This must require neither the original history nor target optimizer steps.

3 Functional Memory Compilation

3.1 Overview

Memory migration must address two challenges: the successor uses its own parameter coordinates, and many source answers must coexist in one compact update. FMC therefore separates the process into three components (Figure 2). Source readout recovers portable answer strings at retained queries (Section 3.2). Target constraint construction expresses those answers as desired token-score changes using the successor's own representations (Section 3.3). Analytic installation fits all constraints jointly and produces one target-native output-head adapter (Section 3.4). The output-head restriction keeps representations fixed during fitting, making batched collection and an analytic solution possible. Algorithm 1 in Appendix G gives the complete procedure; Section 4.1 defines the task interfaces used to evaluate the installed memory.

3.2 Source readout: recover transferable answers

Without the original history, observable source answers provide the content to transfer. The inputs are a source model f_s, its trained aggregate memory LoRA M_s, a frozen target f_t, and retained answer-free addresses $\mathcal{Q} = \{q_i\}$. An address is a question that can be asked of the source, not its original answer or interaction transcript. For example, it may ask for a person's employer. The response comes from $f_s + M_s$, even if it differs from the benchmark reference. Width, depth, and tokenization may differ between models. We recover address–response pairs $\mathcal{B} = \{(q_i, y_i)\}$ and

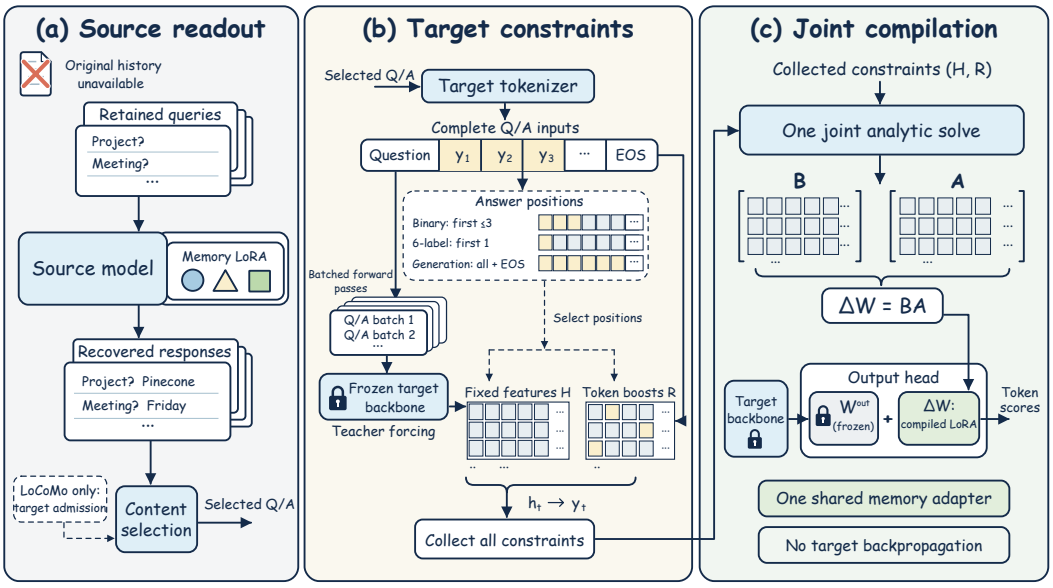

Figure 2: Functional Memory Compilation. (a) Retained answer-free queries recover source responses. (b) Target retokenization and batched teacher forcing collect fixed features H and sparse token boosts R at task-specific positions. The illustrated y_t denotes $v_{i,t}$ for one answer. Columns of R index only requested tokens. (c) One joint reduced-rank ridge fit maps the compact solution back to the vocabulary and installs a single output-head LoRA. Original target weights remain frozen.

select $\mathcal{A} \subseteq \mathcal{B}$ for a declared output interface. These pairs specify accessible source behavior rather than exhaustive or necessarily correct knowledge.

Addresses must survive the upgrade. Benchmark preprocessing constructs them from public records or source-writer questions and discards the original answers from migration supervision. Both FMC and matched distillation receive values read anew from the source, without directly consulting evaluation references. The no-history restriction applies to target migration; source training and address retention precede it (Appendix A.2).

For open-ended values, we test prompt sensitivity by greedily decoding under a canonical prompt $\pi_0(q_i)$ and an answer-only realization $\pi_1(q_i)$. Both use bounded output lengths. Cleaning removes completed reasoning blocks, truncates at role delimiters, and trims whitespace. We accept nonempty responses that agree after case folding and whitespace normalization, excluding unknown-like values. Accepted responses retain canonical casing. This checks prompt stability, not factual correctness. For six-label classification, the largest mean conditional label log-probability selects a valid label without an independent correctness check. Read rules and address selectors are frozen on development data.

3.3 Target constraints: express answers in the successor

Source token IDs and hidden coordinates need not have the same meaning in the successor. We therefore retokenize each recovered string as $(v_{i,1}, \ldots, v_{i,L_i})$ in the target vocabulary and collect its target representations. Only the output projection receives a LoRA update (Hu et al., 2022). For row-vector hidden state h, logits become $h(W_t^{\text{out}} + BA)^\top$, with $\text{rank}(BA) \leq r_t$. All original target parameters remain frozen. This restriction keeps hidden states unchanged for a fixed input prefix, allowing representation collection to be separated from fitting.

Teacher forcing supplies the recovered preceding tokens and collects the final hidden state $h_{i,t}$ before predicting $v_{i,t}$. Each selected position contributes $(h_{i,t}, v_{i,t}, \beta_{i,t})$, requesting a positive residual $\beta_{i,t} = s\omega_{i,t}$ in that token's logit. This uses only the recovered string, avoiding a full-vocabulary teacher distribution and its cross-tokenizer alignment. When fitted accurately, the requested boost increases the recovered token's score relative to competing tokens. Whether it changes the decision

also depends on the base model's score gap and the residual fitting error (Appendix D.2). The fixed-response square loss is therefore a tractable surrogate for behavior preservation, distinct from cross-entropy training.

Position choice follows what the output interface must recover. A label needs its decision token, whereas literal value generation needs the answer trajectory and its stopping point. For binary answer comparison, the fixed policy uses $P_i = \{1, \ldots, \min(3, L_i)\}$ with boosts $(96, 48, 48)$. Six-label classification uses $P_i = \{1\}$ with boost 32. Conversational values use the entire answer plus one end-of-sequence (EOS) token, each with boost 32. The target ranks are respectively 32, 64, and 128, with LoRA scale $\alpha = r_t$.

The classification profiles use source-side selection only: all 247 stable KnowEdit responses, and 144 MemoryAgentBench responses selected from 900 addresses by the fixed label-aware rule. They perform no target trial writes or generation. All profiles retain complete teacher-forced inputs, so position selection changes the regression size rather than the number of traversals per example.

Full-answer generation introduces more constraints per memory and competition within the shared update. LoCoMo therefore retains a target-admission rule: temporary isolated updates and bounded generations estimate each source-safe candidate's readback gain and constraint-count cost. Eligible candidates are packed under a fixed allocation budget; these costs are selection heuristics, not reserved ranks in the final adapter. This selects 250 of 352 source-safe responses across nine streams. The final compiler writes *every* answer position plus EOS for each selected response, including values already readable from the target base. It then discards the temporary candidate parameters. Distilled LoRA receives the same selected pool and serving addresses. Appendix A specifies the rule and development comparison.

The three-position classification policy limits the constraint count while allowing early score corrections to change candidate ranking. Generation instead follows prefixes that can leave the fitted trajectory after an error, motivating full-answer and EOS constraints. More positions also create competing requests within the shared low-rank update. The position and independent-source analyses in Appendix C examine this coverage–interference tradeoff.

3.4 Analytic installation: fit a shared low-rank update

All selected behaviors must share one adapter, so we fit their constraints jointly rather than applying a sequence of isolated final writes. Stack the n hidden keys into $H \in \mathbb{R}^{n \times d_t}$ and let $\mathcal{T}$ contain their m unique desired tokens. The compact response matrix $R \in \mathbb{R}^{n \times m}$ has $R_{j,\tau_j} = \beta_j$ and zeros elsewhere, where τ_j indexes row j's desired token in $\mathcal{T}$. We solve

$$\min_{\operatorname{rank}(W) \le r_t} \|HW - R\|_F^2 + \lambda \|W\|_F^2, \tag{1}$$

with $W \in \mathbb{R}^{d_t \times m}$ and $\lambda = \rho \max\{\operatorname{mean}(\operatorname{diag}(HH^\top)), 10^{-12}\}$. Ridge regularization penalizes large updates and makes the Gram matrix invertible; the rank constraint matches the compact LoRA representation. At each key, other tokens in $\mathcal{T}$ have requested residual zero, while output-update rows outside $\mathcal{T}$ remain zero after installation. The joint fit accounts for constraints that request different tokens at similar hidden states, but can still leave conflicts. Rank limits update dimension, not a guaranteed number of preserved facts.

For nonempty constraints, standard reduced-rank ridge regression (Mukherjee & Zhu, 2011) gives

$$G = H^\top H + \lambda I, \qquad Z = G^{-1/2} H^\top R, \qquad W^\star = G^{-1/2} [Z]_{r_t}, \tag{2}$$

where $[Z]_{r_t} = P_r \Sigma_r Q_r^\top$ is a best rank-r_t approximation and $r = \min(r_t, \operatorname{rank}(Z))$. Completing the square reduces the objective to $\|G^{1/2} W - Z\|_F^2$ plus a constant. Invertibility of $G^{1/2}$ preserves rank, yielding the classical optimum.

Let $S \in \{0, 1\}^{|\mathcal{V}_t| \times m}$ scatter compact token columns into the target vocabulary. A LoRA factorization is

$$A = \Sigma_r^{1/2} P_r^\top G^{-1/2}, \qquad B = S Q_r \Sigma_r^{1/2}. \tag{3}$$

Then $BA = SW^{\star\top}$. We evaluate the solve through a thin singular value decomposition of H, use float32 arithmetic, and serialize in the serving dtype. The scale $\alpha / r_t = 1$ preserves the derived

update. The closed-form optimum applies to the fixed surrogate in exact arithmetic; task accuracy is evaluated separately.

The compact token matrix avoids fitting a dense residual over the entire vocabulary. For $H = UDV^\top$, the whitened response is represented by $D(D^2 + \lambda I)^{-1/2}U^\top R$ in the row space of H. No dense $d_t \times d_t$ inverse is required. Collection processes all input tokens, while fitting scales with the numbers of constraints and distinct requested tokens. Appendix D gives the thin-factor construction and explains the roles of boost scale, regularization, and effective rank.

Using the compiled memory. The installed adapter defines a target conditional distribution that can score candidate answers or decode a value autoregressively. At each generated step, the frozen backbone processes the current prefix and the updated output head scores the next token. Thus frozen features during fitting do not imply a fixed generation trajectory: each emitted token determines the next input prefix. Serving no longer requires the source model or a table of recovered values. A task may invoke the adapter directly or retrieve a retained address before decoding its value; these readout choices do not change the compilation objective.

4 EXPERIMENTS

We organize the experiments section around four research questions (RQs).
RQ1. How well does FMC preserve memory after a model upgrade?
RQ2. How much does target installation cost?
RQ3. What roles do source memory and address–answer correspondence play?
RQ4. Does FMC transfer across model pairs and source memories?
The following subsections address these questions in turn through comparative evaluations and targeted analyses.

4.1 TASKS, ACCESS, AND CONTROLS

We evaluate migration from Qwen3.5-2B to Qwen3.5-4B (Qwen Team, 2026) on three task interfaces. We instantiate KnowEdit WikiData-counterfact (Zhang et al., 2024) as binary classification between supplied new and old answers. Excluding 40 development facts from 408 leaves 736 test paraphrases, two per fact. All 408 facts are available to the source writer before migration. The target is evaluated on held-out paraphrases rather than new facts. Success requires strictly greater mean token log probability for the new answer; ties count as failures. MemoryAgentBench (MAB) TREC-Coarse (Hu et al., 2026) measures six-label classification on official queries 40–99, with the first 40 reserved for development. Its 6,600 demonstrations create the source memory before migration. On LoCoMo (Maharana et al., 2024), we report query-weighted Token F1 over 1,201 questions from nine streams; stream 3 is development-only.

Task readout. KnowEdit and MAB activate the installed adapter directly and compare the conditional scores of their candidate answers or labels. LoCoMo questions need not match a retained address. Its common reader therefore separates finding a memory, decoding its value, and answering the natural question. All states share the same 250 selected answer-free addresses. A frozen BM25 lexical retriever (Robertson & Zaragoza, 2009) selects at most one address, whose value is decoded by the evaluated model state. The target base then answers the natural question using the decoded pair as temporary context, with the memory adapter disabled. Both generation stages are greedy and bounded to 32 tokens. The threshold and stopping rules are specified in Appendix E. FMC serving retains the address index but accesses neither the source model nor an external value table.

The compiled profiles use complete source answers as input. KnowEdit constrains at most three answer positions, MAB constrains its label position, and LoCoMo constrains every answer position plus an end-of-sequence token. Output-head ranks are 32, 64, and 128, respectively; ridge ratio is 10^{-3}. Profiles and development choices are detailed in Appendices A and A.1.

Comparison states. Source Base and Source Memory are pre-upgrade references; on LoCoMo, they decode values for the common target-base reasoner. Target Base has no memory update. Cross-LoRA (Xia et al., 2025) transfers source weight geometry. Our Command-V LoRA port (Wang

Table 1: Primary model upgrade. All LoCoMo states use the shared reader. Bold marks the best migrated parameter state; source references and reader-only controls are separate comparisons. Dashes denote inapplicability.

Group	State	KnowEdit Accuracy (%) ↑	LoCoMo F1 (%) ↑	MAB Accuracy (%) ↑
Source references	Source Base	12.23	3.94	18.33
	Source Memory	97.83	8.84	95.00
Target states	Target Base	10.87	4.10	23.33
	Cross-LoRA	9.78	4.03	21.67
	Command-V LoRA port	34.92	2.92	28.33
	Distilled LoRA	34.92	6.10	60.00
	FMC	**53.53**	**8.74**	**90.00**
Reader-only controls	Selected-address-only	—	5.66	—
	Recovered text (same values)	—	8.74	—

et al., 2025a) adapts intervention transfer to an aggregate source-memory residual, using converters fitted on 1,030 task-independent prompts. Appendix B details these baseline adaptations and their retained addresses.

Distilled LoRA trains an independently initialized, rank-matched output-head LoRA (Hu et al., 2022) on the same recovered strings as FMC. It uses source-response supervision (Kim & Rush, 2016) with cross-entropy (CE) loss. We report two epochs, representation batches of 32, optimizer batches of four, and minimum-training-loss checkpoint selection before task evaluation. This compares installation on fixed content at one training budget; longer training can reverse the ranking. Appendix B records the recipe and budget-selection history.

LoCoMo also includes two reader-only controls. Selected-address-only supplies the retrieved question without its value; recovered text supplies the same source value that FMC aims to decode. Both use the identical final reasoner. These controls separate the contribution of the value from that of its address and test whether parameter readback preserves the usefulness of the recovered content.

4.2 RQ1: How well does FMC preserve memory?

Under the evaluated two-epoch distillation budget, FMC achieves the highest scores among migrated parameter states on all three tasks (Table 1).

FMC answers 394/736 KnowEdit and 54/60 MAB queries correctly, improving over Distilled LoRA by 18.61 and 30.00 percentage points. For these fixed states, we use a paired bootstrap over 368 KnowEdit facts, keeping both paraphrases together. The resulting 95% interval for the gap is $[13.45, 23.51]$ points. LoCoMo improves by 2.64 F1 points. The source-reference gaps reflect the full recovery-and-installation pipeline rather than compilation alone.

To separate coverage from downstream accuracy, we group KnowEdit test facts by whether their canonical address entered the compiled pool. On the 450 paraphrases of included facts, FMC scores 64.44% versus 45.11% for Distilled LoRA. On the remaining 286, it scores 36.36% versus 18.88%. Thus the advantage appears in both groups, while aggregate accuracy also depends on source-address coverage (Appendix C.5). Coverage identifies which facts supply explicit constraints; paraphrase accuracy measures the resulting update under changed wording. Keeping these stages distinct helps interpret the gap between source memory and its compiled successor.

Figure 3 examines fact-level coverage and question-level differences in these saved predictions. FMC answers both KnowEdit paraphrases correctly for more facts, but its successes are not a superset of Distilled LoRA's successes on that task.

At LoCoMo's 250 selected canonical addresses, FMC reproduces all recovered source values verbatim. Normalized exact match is 250/250 for FMC, 40/250 for Distilled LoRA, and 1/250 for Target Base. This tests faithful joint installation of the admitted source values. The pool was selected using temporary FMC writes, so sharing it holds content fixed but does not make admission method-neutral. The result is conditional on those 250 of 352 source-safe candidates.

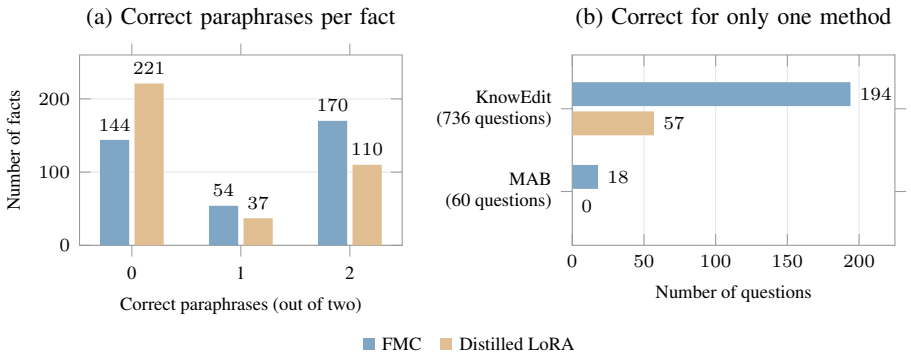

Figure 3: Prediction structure of the fixed FMC and Distilled LoRA states (post-hoc). (a) All 368 test facts in KnowEdit, grouped by the number of their two paraphrases answered correctly. (b) Questions correct only for the indicated method, among all 736 KnowEdit and 60 MAB questions. Counts follow the existing scoring rules.

Table 2: Fixed-pool target installation time: median of six runs. Speedup is the Distilled LoRA/FMC ratio of unrounded medians.

Task	Distilled LoRA Time (s) ↓	FMC Time (s) ↓	Speedup ↑
KnowEdit	5.10	**3.44**	1.48×
MAB	3.25	**2.34**	1.39×
LoCoMo	7.44	**6.30**	1.18×

The LoCoMo text control tests whether parameter readback preserves the values' downstream usefulness. Selected questions alone yield 5.66 F1; adding their values yields 8.74. With recovered text, all 1,201 final strings and token sequences are identical to FMC. The compiled values therefore support the same answers as recovered text under this reader. Remaining errors arise in shared content, address coverage, retrieval, or reasoning. This establishes parameter installation and preserved downstream utility, not superiority to text storage (Appendix E).

4.3 RQ2: How much does target installation cost?

FMC reduces fixed-pool installation time by 32.58% on KnowEdit, 28.11% on MAB, and 15.30% on LoCoMo (Table 2). We report medians of six installations per method from a fixed source pool on the same RTX 4090 after warmup. Each includes encoding, representation collection, fitting, and saving; LoCoMo sums nine streams per repeat. Appendix B reports the timing ranges and repeat protocol.

KnowEdit collects its 705 constraints with eight production forwards and one solve; MAB uses five forwards and one solve. Distilled LoRA makes 124 and 72 optimizer updates, respectively. Both methods collect complete source answers in batches of 32; Distilled LoRA retains optimizer batches of four. Thus parallel representation collection is matched independently of the optimization batch.

A complementary MAB control holds the keys and square-loss targets fixed and changes only the solver. Both solutions classify 54/60 queries correctly. The iterative route costs 24.02 seconds including all three prescribed learning rates, versus 2.34 for the analytic route. This separates the benefit of solving the same local objective from the comparison with CE distillation (Appendix D.3).

4.4 RQ3: Ablation studies—which components matter?

We intervene on the source-recovery pipeline and on address–answer correspondence. Removing source memory before recovery lowers scores on all three tasks; shuffling a fixed recovered pool lowers KnowEdit accuracy.

Table 3: Source-recovery pipeline ablation. MAB fixes addresses; KnowEdit and LoCoMo reapply recovery and selection. Both rows are compiled target states, distinct from the source references in Table 1.

Variant	KnowEdit Accuracy (%) ↑	LoCoMo F1 (%) ↑	MAB Accuracy (%) ↑
w/o source memory	40.76	4.46	20.00
FMC	53.53	8.74	90.00

The source-recovery ablation queries the source base without its memory, then reapplies recovery, selection, and compilation (Table 3). MAB holds all 144 addresses fixed, directly testing the effect of recovered labels. KnowEdit recovers 282 usable source-base responses versus 247 source-memory responses; LoCoMo selects 89 versus 250. These two tasks measure the combined effect of source memory on recovered content and admission.

The address–answer shuffling ablation preserves all 247 KnowEdit addresses, their order, and the complete-answer multiset. Accuracy nevertheless drops in all three prescribed pairings: $32.70 \pm 2.16\%$ (mean $\pm$ standard deviation) versus 53.53% for the original pairing. Correct pairing thus contributes beyond retaining the answers alone on this source pool (Appendix C.2).

4.5 RQ4: Does FMC transfer across models and sources?

We test FMC on two additional model pairs using the same decision profiles: Qwen3-1.7B→Qwen3-4B (Yang et al., 2025) and Qwen3-1.7B→SmolLM3-3B (Bakouch et al., 2025). Its advantage over Cross-LoRA persists on both pairs. KnowEdit/MAB scores are 46.60/85.00 and 55.43/85.00, versus Cross-LoRA's 19.16/23.33 and 7.34/20.00. These checks test portability against parameter transfer; matched distillation is evaluated on the primary pair. Each successor supplies its own frozen representations for the joint fit; no source-to-target hidden-coordinate map is reused. The answer-position profiles and analytic installation rule remain fixed across these backbones. Across three independently trained source memories, FMC obtains $51.95 \pm 1.50\%$ on KnowEdit and $84.44 \pm 4.81\%$ on MAB (mean $\pm$ standard deviation). Full-answer fitting improves two KnowEdit sources but harms the third, supporting source-dependent position tradeoffs rather than a universally best profile. Appendices C and F report these analyses.

5 Scope and Limitations

The evidence covers fixed task interfaces and 1.7B–4B models. Migration depends on source-accessible answers, which may be incomplete or incorrect. LoCoMo further depends on target admission and a shared reader. Behavior on unrelated requests while the adapter is active remains unmeasured; applications must decide when to invoke it. Broader training-budget and selection comparisons remain open.

6 Conclusion

We presented FMC, a behavior-level approach to migrating aggregate LoRA memory without the original interaction history. FMC turns source-exposed answers into target-side constraints. Batched representation collection and one joint analytic fit then produce a target-native output-head adapter. Installation requires no target backpropagation or optimizer updates, and the resulting memory remains usable after retiring the source model.

Across the evaluated tasks, FMC improves task scores and lowers fixed-pool installation cost relative to matched-content distillation under the evaluated budget. Exact readback and the recovered-text control further show that selected conversational values can be installed jointly while retaining their downstream usefulness. Together, these findings establish analytic compilation as a viable mechanism for memory continuity across model upgrades.

AI USE STATEMENT

ChatGPT and Codex assisted with method and experiment design, implementation, analysis, and manuscript writing. OpenAI tools also assisted figure creation. Qwen3.5-4B generated LoCoMo source-writing supervision, as described in Appendix H. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Section 3 and the appendix specify the method, experimental settings, and evaluation protocols. The supplementary code provides the compiler and usage instructions; model checkpoints and benchmark data must be obtained separately.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, S. K. Hong, Youngjune Gwon, and Sungjin Ahn. Understanding LoRA as knowledge memory: An empirical analysis. In *International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2603.01097. URL https://arxiv.org/abs/2603.01097.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulić, and Alessandro Sordoni. Training plug-and-play knowledge modules with deep context distillation. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ghyyHZYORi.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-LoRA: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*, 2026. doi: 10.48550/arXiv.2602.15902. URL https://arxiv.org/abs/2602.15902.

Farzad Farhadzadeh, Debasmit Das, Shubhankar Borse, and Fatih Porikli. LoRA-X: Bridging foundation models with training-free cross-model adaptation. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2501.16559. URL https://arxiv.org/abs/2501.16559.

Naibin Gu, Peng Fu, Xiyu Liu, Ke Ma, Zheng Lin, and Weiping Wang. Adapt once, thrive with updates: Transferable parameter-efficient fine-tuning on evolving base models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14765–14783. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.719. URL https://aclanthology.org/2025.acl-long.719/.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/95b6e2ff961580e03c0a662a63a71812-Abstract-Conference.html.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=DT7JyQC3MR.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Fengxian Ji, Zhuohan Xie, Jingpu Yang, Fan Zhang, Zirui Song, and Xiuying Chen. Parametric memory decoding for zero-shot routing in LoRA-based external parametric memory. *arXiv preprint arXiv:2607.04118*, 2026. URL https://arxiv.org/abs/2607.04118.

Chanjoo Jung and Jaehyung Kim. TiTok: Transfer token-level knowledge via contrastive excess to transplant LoRA. In *The Fourteenth International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2510.04682. URL https://arxiv.org/abs/2510.04682.

Rana Muhammad Shahroz Khan, Pingzhi Li, Sukwon Yun, Zhenyu Wang, Shahriar Nirjon, Chau-Wai Wong, and Tianlong Chen. PortLLM: Personalizing evolving large language models with training-free and portable model patches. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gyHoR6uFhU.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327. Association for Computational Linguistics, 2016. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Shuaike Li, Kai Zhang, Qi Liu, and Enhong Chen. MindBridge: Scalable and cross-model knowledge editing via memory-augmented modality. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11999–12013. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.findings-acl.621. URL https://aclanthology.org/2025.findings-acl.621/.

Yanan Li, Fanxu Meng, Muhan Zhang, Shiai Zhu, Shangguang Wang, and Mengwei Xu. LoRASuite: Efficient LoRA adaptation across large language model upgrades. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 170149–170176. Curran Associates, Inc., 2025b. doi: 10.52202/085713-5664. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f8887787de8f39c9a3eb28e1afbe22ce-Paper-Conference.pdf.

Haodong Lu, Chongyang Zhao, Minhui Xue, Lina Yao, Kristen Moore, and Dong Gong. Little by little: Continual learning via incremental mixture of rank-1 associative memory experts. In *International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2506.21035. URL https://arxiv.org/abs/2506.21035.

Weitao Ma, Xiyuan Du, Xiaocheng Feng, Lei Huang, Yichong Huang, Huiyi Zhang, Xiaoliang Yang, Baohang Li, Xiachong Feng, Ting Liu, and Bing Qin. One for all: Update parameterized knowledge across multiple models with once edit. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16021–16034. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.780. URL https://aclanthology.org/2025.acl-long.780/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 15817–15831. PMLR, 2022b. URL https://proceedings.mlr.press/v162/mitchell22a.html.

Ashin Mukherjee and Ji Zhu. Reduced rank ridge regression and its kernel extensions. *Statistical Analysis and Data Mining*, 4(6):612–622, 2011. doi: 10.1002/sam.10138. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC3444519/.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39/.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Tao Ren, Weiyao Luo, Hui Yang, Rongzhi Zhu, Xiang Huang, Yuchuan Wu, Bingxue Chou, Jieping Ye, Jiafeng Liang, Yongbin Li, and Yijie Peng. Scaling self-evolving agents via parametric memory. *arXiv preprint arXiv:2606.04536*, 2026. doi: 10.48550/arXiv.2606.04536. URL https://arxiv.org/abs/2606.04536.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://www.nowpublishers.com/article/DownloadEBook/INR-019.

Barry Wang, Avi Schwarzschild, Alexander Robey, Ali Payani, Charles Fleming, Mingjie Sun, and Daphne Ippolito. Command-V: Pasting LLM behaviors via activation profiles. *arXiv preprint arXiv:2506.19140*, 2025a. doi: 10.48550/arXiv.2506.19140. URL https://arxiv.org/abs/2506.19140.

Ke Wang, Yiming Qin, Nikolaos Dimitriadis, Alessandro Favero, and Pascal Frossard. MEMOIR: Lifelong model editing with minimal overwrite and informed retention for LLMs. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 89143–89170. Curran Associates, Inc., 2025b. doi: 10.52202/085713-2982. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/809a560bf60984be05180c071c352eaa-Abstract-Conference.html.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 53764–53797. Curran Associates, Inc., 2024a. doi: 10.52202/079017-1703. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/60960ad78868fce5c165295fbd895060-Abstract-Conference.html.

Runqian Wang, Soumya Ghosh, David Cox, Diego Antognini, Aude Oliva, Rogerio Feris, and Leonid Karlinsky. Trans-LoRA: Towards data-free transferable parameter efficient finetuning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 61217–61237. Curran Associates, Inc., 2024b. doi: 10.52202/079017-1957. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/708fdc7911f11585ee7161518e509ae6-Paper-Conference.pdf.

Yan Wang, Dongyang Ma, and Deng Cai. With greater text comes greater necessity: Inference-time training helps long text generation. In *Conference on Language Modeling*, 2024c. doi: 10.48550/arXiv.2401.11504. URL https://arxiv.org/abs/2401.11504.

Feifan Xia, Mingyang Liao, Yuyang Fang, Defang Li, Yantong Xie, Weikang Li, Yang Li, Deguo Xia, and Jizhou Huang. Cross-LoRA: A data-free LoRA transfer framework across heterogeneous LLMs. *arXiv preprint arXiv:2508.05232*, 2025. doi: 10.48550/arXiv.2508.05232. URL https://arxiv.org/abs/2508.05232.

Ziwen Xu, Haiwen Hong, Linsong Yu, Benglei Cui, Longtao Huang, Hui Xue, and Ningyu Zhang. How LoRA remembers? a parametric memory law for LLM finetuning. *arXiv preprint arXiv:2605.30260*, 2026. doi: 10.48550/arXiv.2605.30260. URL https://arxiv.org/abs/2605.30260.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: Enhancing model editing with neuron-indexed dynamic LoRA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19449–19457, 2024. doi: 10.1609/aaai.v38i17.29916. URL https://ojs.aaai.org/index.php/AAAI/article/view/29916.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024. doi: 10.48550/arXiv.2401.01286. URL https://arxiv.org/abs/2401.01286.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4447–4462. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.263. URL https://aclanthology.org/2024.findings-acl.263/.

Table 4: Evaluation workloads and installation profiles. The ridge ratio is 10^{-3}; target LoRA alpha equals its rank. Full+EOS includes all recovered answer tokens and one tokenizer end-of-sequence target.

Task	Test queries	Rank	Source rows	Positions	Logit boosts
KnowEdit	736	32	247	first $\min(L, 3)$	96, then 48
LoCoMo	1,201	128	250	full+EOS	32
MAB	60	64	144	label position zero	32

Table 5: Source writing, preceding migration. LoCoMo has N_s cached supervision pairs in stream s. These are source-training costs, not target installation steps.

Workload	Rank/alpha	Batch	Epochs	Supervision	Steps
KnowEdit	32/32	4	32	408 canonical pairs	3,264
LoCoMo	32/32	4	32	per-stream QA pairs	$32\lceil N_s/4 \rceil$
MAB	64/64	16	2	6,600 demonstrations	826

A Reproducible Benchmark Profiles

This appendix specifies the batched FMC implementation used in the main results. The target representation-collection batch size is 32, distinct from Distilled LoRA's optimization batch size of four. Source memories were trained before migration. Target supervision consists of responses recovered from those memories, not their original writing histories. Classification uses fixed candidate sets; LoCoMo uses address-mediated question answering.

KnowEdit compares the mean conditional log-probability of the new and original answer: the new answer is correct only when its score is strictly greater. Each of 408 facts has a canonical question and two paraphrases. The primary partition excludes the actual 40 development fact IDs and all canonical forms, leaving $368 \times 2 = 736$ questions. It is not obtained by discarding the first 40 rows. MAB chooses the first maximum-scoring label in the ordered set $\{0, 1, 2, 3, 4, 5\}$. Its primary partition contains query indices 40–99; all-100 scores are development-inclusive diagnostics. LoCoMo excludes development stream 3 and evaluates all 1,201 questions in the other nine streams.

FMC encodes each complete prompt and recovered answer in their original order, without length sorting or truncation. Answer position p uses the hidden representation at prompt length minus one plus p. Restricting fitted positions therefore does not shorten the backbone input. A single traversal collects the representations in batches, followed by one global reduced-rank ridge solve per target state. KnowEdit and MAB use no target trial writes, generation-based admission, or priority refinement. LoCoMo retains target-side selection; only its subsequent fixed-pool installation is a single global fit.

A.1 Source Memory Writers

The source writer adapts Q/K/V/O attention projections at zero-indexed layers $\{3, 7, 11, 15, 19, 23\}$. It uses AdamW with learning rate 10^{-3}, zero weight decay, and a fixed unshuffled example order. KnowEdit and LoCoMo mask prompt and padding positions. They supervise the complete encoded assistant response, including its termination suffix, with maximum encoded length 256. MAB supervises the numeric label token with maximum encoded length 128. The answer-free address list is retained at source export; original supervision values are not passed to target migration.

The 32-epoch source writers retain the epoch with minimum training loss, without evaluation feedback. Qwen3-source model-upgrade checks retain the same task-specific writing recipes and layer indices. Changing the source checkpoint can change which responses are reliably exposed; these conditions do not assume that every source produces the primary pair's 247 KnowEdit responses.

Table 6: Source recovery and fixed target content. Stable and safe are separate LoCoMo stages. All selected rows, including one base-inherited row, enter its final full-answer installation.

Task	Candidates	Stable/read-valid	Safe	Selected	Constraints
KnowEdit	408	247	247	247	705
LoCoMo, nine streams	783	369	352	250	2,353
MAB	900	900	900	144	144

A.2 Source Recovery and Address Selection

An address contains a key and the question used to access a memory, but no answer. KnowEdit supplies canonical questions, LoCoMo supplies instruction strings extracted from source sessions, and MAB supplies demonstration inputs. KnowEdit and LoCoMo read two fixed source prompt realizations. Cleaning removes completed reasoning blocks and role-delimited suffixes, then trims whitespace. Accepted responses agree after case folding and whitespace normalization, are nonempty and not unknown-like, and pass the existing control-marker rejection rule. Canonical casing is preserved for target tokenization. Stability does not establish correctness.

For the primary KnowEdit source, all 247 stable, eligible responses pass the text rule. The source-only projection exactly matches the installed strings and order. It contains 22 development facts and 225 test facts. Another 143 test facts have no selected canonical source address, although the same aggregate target state answers all test questions. No target trial filter is needed to obtain this pool.

MAB reads the largest conditional likelihood among six labels, so all source predictions are syntactically valid. Its margin is the difference between the predicted label's score and the strongest alternative. From 900 source-read inputs, the default selector retains the 96 largest margins per predicted label. It then greedily chooses 64 with low maximum token-set Jaccard similarity to already chosen examples. Finally, it restores arrival order and keeps the first 24 per label. Arrival order resolves diversity ties. Neither original demonstration labels nor official test answers select these 144 source-predicted labels.

LoCoMo rejects 17 control-marker outputs from 369 stable responses, retaining 352. Target-side selection evaluates base readback and isolated temporary updates at each address. Fidelity, value containment, and isolated constraint-count cost determine admission under the frozen per-candidate cap of 32 and per-stream allocation budget of 128. Candidates are packed by cost and arrival order; these costs are selection heuristics, not ranks reserved inside the final adapter. A value already readable from the base can be inherited. The formal run selects 250 rows, including one inherited row. It recompiles every selected row with full-answer-plus-EOS constraints rather than reusing trial-fitted positions.

This selection performs 352 base probes and 2,536 isolated updates with readback, totaling 2,888 generation calls. Target selection, resets and index preparation cost 2,626.5045 s; projection of cached source fields adds 0.0350 s. These are real shared costs, not a free input to either installation method. The rule was fixed on development stream 3 before confirmation-stream evaluation.

The development comparison used all 179 questions in stream 3 and the same full-answer-plus-EOS compiler for both pools. Source-only admission retained 39 addresses and obtained 8.4328 F1; target-side selection retained 25 and obtained 10.9856. The predeclared rule retained source-only admission if its F1 was at least the target-selected score, otherwise target selection. It therefore selected the latter without a test-driven tolerance. This comparison was specified after historical runs were available but before either new condition was executed. It changes the address index as well as content, and is not a single-position ablation. A descriptive paired-question interval for the 2.5528-point difference is $[-0.7504, 6.0807]$; it does not establish a robust advantage across conversations. The nine other streams provide the separate confirmation workload. These experiments reuse existing benchmark partitions whose earlier results were available; they are not an unopened test set.

Table 7: Fixed-pool installation variability over six runs, in seconds: median [minimum, maximum]. LoCoMo sums nine streams within each repeat before summarizing. Shared preparation and diagnostic costs are separate.

Workload	FMC	Distilled LoRA
KnowEdit	3.4358 [3.4298, 3.4628]	5.0963 [5.0787, 5.1133]
MAB	2.3355 [2.3087, 2.3433]	3.2487 [3.1981, 3.2867]
LoCoMo	6.2981 [6.1811, 6.3773]	7.4359 [7.3730, 7.6961]

B Matched-Content Training and Cost Accounting

Distilled LoRA trains the same target output head on exactly the source strings and order used by FMC. These comprise 247 KnowEdit responses, 144 MAB labels, and 250 LoCoMo responses partitioned into nine states. Both methods retain the same rank and alpha within each task. Source recovery and LoCoMo target-side pool selection are shared. Target construction does not replay the original history or directly consult evaluation references; KnowEdit evaluates paraphrases of facts available to the source writer.

Classification reference states retain their fixed source checkpoint, target configuration, and scoring protocol. The Command-V LoRA port uses retained answer-free address sets of 408 KnowEdit and 900 MAB entries. Its aggregate intervention is an adaptation of the original method. LoCoMo references are decoded again through the shared value-reading interface rather than reusing scores from a different reader.

The target recipe is fixed at two epochs of AdamW, learning rate 10^{-3}, betas $(0.9, 0.999)$, epsilon 10^{-7}, and zero weight decay. Representations collected in batches of 32 are split back into the original ordered groups of four for optimization. Cached hidden states preserve sequence shapes, batch padding, causal label shifting, and full-vocabulary cross-entropy (CE) loss over the encoded assistant response and termination suffix. Checkpoint selection minimizes mean training-batch loss within the two-epoch budget, before task evaluation. Every independent timing repeat restores the same blank factors and random state, constructs a fresh cache and optimizer, and performs only those two epochs.

KnowEdit makes $2\lceil 247/4 \rceil = 124$ optimizer steps; MAB makes $2(144/4) = 72$. LoCoMo stream sizes 47/20/48/21/24/28/21/16/25 produce 130 steps in total. Same-collection-batch online and cached implementations pass a fixed two-update comparison of loss, gradients and factor updates before production. This does not prove that arbitrary batch shapes yield identical training trajectories.

Distilled LoRA's maximum encoded lengths remain 256 for KnowEdit and LoCoMo and 128 for MAB. In MAB the 144 source strings are identical, but encoded token sequences are not. Distilled LoRA includes an assistant suffix and truncates four prompt tokens in total per epoch under the fixed length rule. All label supervision remains present. FMC uses the full input, whose longest sequence is 130 tokens. This was part of the frozen recipe; the accuracy gap cannot be attributed to those four tokens without a separate intervention.

The comparison concerns a specific two-epoch budget, not the best attainable target training. This reporting scope was chosen after inspecting earlier budget experiments, which showed that longer training can reverse comparisons. Those results remain in the research archive; they do not establish that two epochs are optimal.

B.1 Installation Windows and Repeats

Production installation includes restoring the blank state, complete encoding, representation collection or cache construction, solve or optimization, factor installation where applicable, and one state serialization. Model loading, source acquisition, numerical checks, complete warmup runs, hashing and evaluation are separate. CUDA synchronization bounds timed accelerator phases. Allocated and reserved memory peaks are checked separately; allocated memory is not added to reserved memory.

KnowEdit alternates methods across six matched rounds on one loaded target. MAB records separate contiguous groups of six independent installations, not interleaved pairs. LoCoMo records six fixed-pool installations per stream and method. Repeated states are sealed and compared before official evaluation. When their factors agree exactly, the predeclared first measured state supplies quality, not the fastest repeat. These repeats estimate timing variability for a fixed source, not independent source-seed uncertainty.

For LoCoMo, adding the once-measured shared pool cost of 2,626.5395 s gives composites of 2,632.8376 s for FMC and 2,633.9754 s for Distilled LoRA. These combine one selection measurement with an installation median; they are not six end-to-end migrations. Source writing and source-model rereading are not assigned zero cost when cached contracts are reused.

FMC installation makes eight forwards and one solve for KnowEdit, five forwards and one solve for MAB, and eleven forwards and nine solves across LoCoMo. These installations use zero target backward passes and optimizer steps. Temporary LoCoMo selection updates and implementation checks are additional work. Production installation peaks meet the fixed 18 GiB limit. The separate full/head-cut checks in the frozen MAB and LoCoMo FMC runs did not record their own peaks. They therefore do not establish a measured whole-process memory bound.

B.2 Relationship to Response-Transfer Pipelines

Distilled LoRA combines training on source-generated response strings, as in sequence-level knowledge distillation (Kim & Rush, 2016), with low-rank adaptation (Hu et al., 2022). Its implementation uses the recovered hard responses rather than a teacher soft-token distribution.

The comparison fixes recovered supervision, output-head modules, and adapter rank to study two installation choices: joint residual fitting and iterative cross-entropy optimization. It deliberately separates this stage from generating a new query distribution. Trans-LoRA's original pipeline retains a few seed examples and a discriminator trained alongside source adaptation (Wang et al., 2024b). These artifacts are not part of our exported source memory. TiTok uses source-adapter on/off scores for sample and token selection, and can also operate on external rather than synthetic data (Jung & Kim, 2026). Its selective-training mechanism could be adapted to recovered pairs, but is not evaluated here. Accordingly, Distilled LoRA is a controlled response-training baseline. It is not a reproduction of either full method or evidence that their additional components would be ineffective.

LoCoMo admission is also a separate comparison choice. Both installers receive the same content and addresses, but membership is determined by FMC-style temporary writes. The source-only development comparison in Appendix A.2 motivated this choice. It does not substitute for a method-neutral or distillation-specific admission comparison on the nine evaluation streams.

C Constraint and Source-Content Analyses

C.1 KnowEdit Answer Positions

This analysis changes only fitted answer positions, holding complete source inputs, order, batch size 32, rank/alpha 32, ridge ratio and the 96/48 boost schedule fixed. No condition adds EOS. All states are sealed before evaluation; K3 reproduces every candidate score of the primary result.

Figure 4 separates the position sweep from source variability. Sources A, B and C use writer seeds 41, 42 and 43, respectively, and expose 243, 247 and 154 valid responses. For the main source, K1/K3/full fit 247/705/1,692 constraints and take 3.4310/3.5298/3.5807 s. The K3/full constraint counts are 707/1,816 for Source A and 462/1,680 for Source C. Each state uses one solve and eight complete-input forwards for Sources A/B or five for C. Timings are single conditional installations; the companion plot data retain their exact values and counts.

K3 improves over K1 by 11.68 percentage points (paired fact-cluster 95% interval $[7.47, 16.17]$). Full improves over K3 by 5.57 points ($[1.63, 9.51]$), but 56 questions are correct only under K3. Intervals use 10,000 paired resamples of 368 facts with seed 20260909, retaining each fact's two paraphrases together. They condition on this source state, are not multiplicity-adjusted, and do not replace independent-source checks.

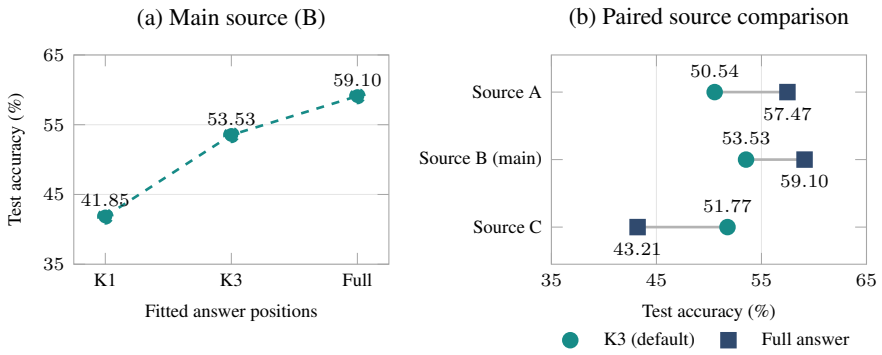

Figure 4: KnowEdit answer-position controls on the same 736 test paraphrases. Left: K1, K3 and full-answer fitting for the main source. Right: paired K3/full comparisons for three independently trained sources; Source B is the main source. Full-answer fitting helps Sources A and B but hurts Source C.

Table 8: KnowEdit address–answer pairing on 736 test paraphrases. Each shuffled pairing retains the same addresses and complete-answer multiset.

Pairing	Accuracy (%) ↑
Original pairing	53.53
Shuffled pairing 1	31.39
Shuffled pairing 2	35.19
Shuffled pairing 3	31.52

Full gains 51 questions for Source A but loses 63 for Source C; K3 remains the fixed default. The Source C pool contains 75 distinct complete answer strings across 154 addresses, including one repeated four-word answer shared by 44 addresses. Source A has 227 distinct answers across 243 addresses. Stable responses need not be useful responses. Amplifying repeated source content is a possible explanation; this comparison does not identify its cause.

C.2 Address–Answer Binding

Three predeclared shuffled pairings retain the 247 addresses, order and complete-answer multiset while changing correspondence. Shuffled pairings 1, 2, and 3 use permutation seeds 20260909, 20260910, and 20260911, respectively. All use K3, 705 constraints, rank 32, and representation batches of 32. Each has zero fixed indices and zero unchanged answer strings. All states precede the same classification evaluation.

Shuffled-pairing accuracy is $32.70 \pm 2.16\%$ (sample standard deviation across mappings). Correspondence contributes to classification, but this does not imply perfect binding or complete-answer generation.

C.3 Contribution of the Source Memory

Source-base compilation queries the unadapted source at the same candidate addresses and applies the frozen source-validity and installation rules. It is not direct Source Base evaluation.

KnowEdit source base produces 282 responses and 545 K3 constraints, obtaining 300/736 against source memory's 394/736. The pools share 175 addresses. Its single installation takes 3.8188 s, with nine forwards and one solve, not a six-repeat timing estimate. MAB keeps its addresses fixed and replaces their source-predicted labels. It retains all rows even when base labels are unbalanced. It obtains 12/60 versus 54/60, with a single source-base installation of 2.7408 s. LoCoMo applies fresh target selection to 89 safe source-base responses. All 89 are selected across the nine streams, with counts 6/2/6/16/9/12/4/17/17. It obtains 18/1,201 exact matches versus 60/1,201 for source memory.

Table 9: Source-memory contribution. KnowEdit and LoCoMo change the selected content pool and are pipeline controls, not same-content value substitutions. MAB fixes all 144 addresses.

Task	Base rows	Memory rows	Base score	Memory score
KnowEdit accuracy (%)	282	247	40.76	53.53
MAB accuracy (%)	144	144	20.00	90.00
LoCoMo F1 (%)	89	250	4.4606	8.7352

Table 10: MAB selectors on the fixed 60 questions. Full rule denotes the default 144-address selector, not writing all 900 inputs.

Selector	Accuracy (%) ↑
Random, repeat 1	86.67
Random, repeat 2	83.33
Random, repeat 3	86.67
Balanced random, repeat 1	88.33
Balanced random, repeat 2	86.67
Balanced random, repeat 3	93.33
Balanced margin	85.00
Full rule (default)	90.00

Its different contents and index preclude attributing the difference solely to parametric installation quality.

C.4 MAB Address Selection

All eight selectors draw from the same 900 source-read candidates and retain 144 examples. Each independently makes five full-input forwards and one solve, rather than collecting a union with different batch grouping. Random and balanced-random repeats 1, 2 and 3 use sampling seeds 0, 1 and 2, respectively.

One balanced-random condition exceeds the default by two questions; the default does not dominate every selector. It remains unchanged rather than being selected again using test results. The full-rule control reproduces the main factors in the recorded CPU comparison.

C.5 KnowEdit Coverage of the Compiled Pool

We partition the saved primary predictions by whether each test fact's canonical address occurs in the 247-row compilation input. The pool contains 225 test facts and 22 development facts; the remaining 143 test facts have no compiled canonical address. Each test fact contributes its two paraphrases. Table 11 uses the same first-repeat FMC and two-epoch Distilled LoRA states as Table 1, without changing predictions or rerunning either model.

FMC answers 290 included-fact and 104 other paraphrases correctly; Distilled LoRA answers 203 and 54. These sum exactly to 394/736 and 257/736. The within-group advantages are 19.33 and 17.48 percentage points. This descriptive split separates address coverage from overall task accuracy, but does not isolate a causal generalization mechanism. The two fact groups may differ in difficulty, and a shared parameter update can affect facts outside its input pool. These remain new-versus-old answer decisions, not measurements of exact source-response reproduction.

D Reduced-Rank Solution and Effective Capacity

D.1 Thin-Factor Realization

Reduced-rank ridge is a standard constrained regression problem, not a new theorem (Mukherjee & Zhu, 2011). Using the row-stacked keys of the main text, write $H = U_h \Sigma_h V_h^\top$ and define

$$D = (\Sigma_h^2 + \lambda I)^{1/2}, \qquad T = D^{-1} \Sigma_h U_h^\top R. \tag{4}$$

Table 11: KnowEdit accuracy by compiled-address coverage, computed from the fixed primary predictions. Membership refers to the canonical address of a test fact; evaluation uses its two paraphrases.

Canonical address	Facts	Queries	FMC (%)	Distilled LoRA (%)
In compiled pool	225	450	64.44	45.11
Outside compiled pool	143	286	36.36	18.88

Writing $T = P\Gamma Q^\top$ yields

$$W^\star = V_h D^{-1} P_r \Gamma_r Q_r^\top. \tag{5}$$

With S embedding compact response columns into the full vocabulary, compatible factors are

$$A = \Gamma_r^{1/2} P_r^\top D^{-1} V_h^\top, \qquad B = S Q_r \Gamma_r^{1/2}. \tag{6}$$

Thin factors have width at most $\min(n, d_t)$, avoiding an explicit full hidden-dimension inverse. In exact arithmetic this is Eq. 2. Regression uses float32; installation uses the target serving dtype, with unused rank zero-padded. Rounding can change the realized correction. Only selected vocabulary rows receive nonzero updates; full-vocabulary CE is a different objective.

D.2 From Residual Fitting to Token Decisions

The sparse residual objective has a direct, conditional interpretation at a fixed target prefix. Let z be the base logits and $\delta = h(BA)^\top$ the realized vocabulary-wide correction. For desired token v and boost $\beta > 0$, define the error vector $e = \delta - \beta \mathbf{e}_v$. Here $\mathbf{e}_v$ is the unit vector for v. Set $\epsilon = \|e\|_\infty$. For every competing token $a \neq v$,

$$(z_v + \delta_v) - (z_a + \delta_a) = z_v - z_a + \beta + e_v - e_a \geq z_v - z_a + \beta - 2\epsilon. \tag{7}$$

Consequently, a sufficient condition for the desired token to be the unique greedy choice is

$$\beta > \max_{a \neq v} (z_a - z_v) + 2\epsilon. \tag{8}$$

Suppose this holds at every recovered-answer prefix, including EOS. If the decoding rules and length limit permit that trajectory, induction gives exact greedy readback from the same prompt. This algebraic condition explains the roles of the base score gap, boost, and fitting error; it is not an empirical certificate for the reported states. Minimizing the joint square loss alone does not establish sufficiently small errors at every prefix. Fitting only the first three positions likewise does not guarantee either complete generation or ranking by mean full-answer log probability.

The three fitting controls act on different parts of this tradeoff. Fix H, the requested-token set, $\lambda > 0$, and the rank budget. With a scale-consistent truncated singular subspace, multiplying all boosts by a positive scalar multiplies an exact solution by the same scalar. Requested corrections and fitting errors scale together, so a larger boost is not an unconditional remedy; serving-dtype rounding can further break exact scaling. Without rank truncation, ridge regression applies the filter $\sigma/(\sigma^2 + \lambda)$ along singular directions of H, suppressing poorly conditioned directions. With truncation, the retained directions come from the whitened response $G^{-1/2} H^\top R$. These properties motivate the controls but do not replace empirical sensitivity tests. In the six-label task, the update has at most six nonzero vocabulary rows and effective rank at most six, despite its nominal rank of 64.

D.3 Matched Local Objective

The MAB control fixes 144 behaviors, keys, sparse responses, ridge ratio and rank, replacing the analytic solver with direct optimization of Eq. 1. Six response tokens and target rank 64 make the rank constraint non-binding. The dense compact update starts at zero and is optimized with Adam at fixed learning rates $0.003, 0.01, 0.03$. Convergence checks occur every 50 steps, after at least 250, with relative tolerance 10^{-5} and at most 10,000 steps per candidate. Selection uses only the final normalized local objective.

The selected learning rate is 0.03. All four states are saved before evaluation; only the objective-selected iterative state and closed-form state are scored. Conditional cost charges the shared 2.2070

Table 12: Same-matrix MAB solver control. All three iterative candidates are charged. These are single-run costs, not a comparison against CE training.

Procedure	Correct / 60	Solve (s)	Conditional cost (s)
Closed-form ridge	54	0.0102	2.3398
Iterative ridge, all candidates	54	21.3603	24.0172

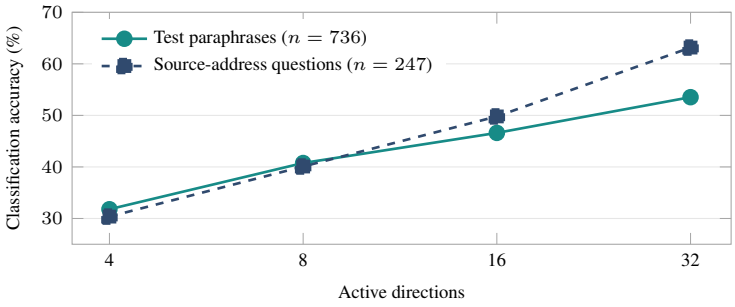

Figure 5: KnowEdit capacity within the fixed K3 solution. Both series compare new versus original candidate answers. Test paraphrases exclude development facts; source-address questions are the canonical forms of the 247 compiled addresses, an in-pool diagnostic. Each tick doubles the number of active directions.

s representation preparation once to each procedure, then its solves, installations and saves. The optimizer acts on a fixed local matrix, not on the target language model. This tests solving the constraints, not the best attainable Distilled LoRA result.

D.4 Fixed-Solution Capacity

KnowEdit keeps the same 247 responses, 705 constraints and singular ordering, retaining the first 4, 8, 16 or 32 directions. It does not recollect keys, re-solve or choose a new rank using test scores. All four states and a blank are saved first; rank 32 reproduces every main-run candidate score.

All files retain physical rank 32 and alpha 32; inactive B columns are zero. Each is 16,059,501 bytes including metadata, with 16,056,320 bfloat16 factor bytes. The theoretical minimal active-factor payload scales with retained rank, but no compressed on-disk layout is measured here.

E LoCoMo Shared-Interface Results

States share the 250-address index, BM25 top-1 route, minimum score 4 fixed on the development stream, canonical value prompt and saved blank Qwen3.5-4B final reasoner. Each state decodes the selected address; final reasoning disables the memory adapter and uses that decoded pair as context. Value decoding and answering each permit 32 tokens with explicit EOS IDs 248044 and 248046. An unaccepted route supplies empty context. Retrieval, value decoding, and final reasoning incur serving costs in addition to the installation costs reported in the main timing comparison.

FMC, Distilled LoRA and shared Target Base obtain 60, 30 and 13 exact matches out of 1,201, with F1 values 8.7352, 6.0971 and 4.1012 percent. FMC exceeds Distilled LoRA on each stream in this fixed run, while absolute QA performance remains low. These are known-memory readback measurements, not a guarantee of unconstrained conversation recall.

Recovered text and FMC produce identical final strings and final answer token sequences on all 1,201 questions, not just equal averages. This shows no quality advantage over storing those values as text and does not alone independently establish parameter-storage mechanisms. The selected-address-only control differs from FMC on 358 final strings.

Across nine streams, compact UTF-8 JSON indices occupy 26,641 bytes; addresses plus source values occupy 40,014 bytes. These are content sizes, not complete model, runtime-index or deployment

Table 13: New shared-interface LoCoMo F1 by held-out stream. Macro means weight streams equally; the final row weights questions equally.

Stream	Questions	Addresses	FMC	Distilled LoRA	Target Base
0	137	47	7.51	4.45	4.02
1	72	20	7.19	4.70	3.03
2	136	48	14.40	9.24	4.37
4	162	21	5.95	3.97	3.80
5	114	24	11.90	8.79	5.89
6	137	28	9.14	6.58	2.04
7	165	21	6.70	4.53	2.64
8	135	16	11.05	9.86	8.72
9	143	25	5.70	3.43	2.68
Stream macro	—	—	8.84	6.17	4.13
Question-weighted	1,201	250	8.74	6.10	4.10

Table 14: Readout controls on the same new LoCoMo index and final reasoner. Text values are source-recovered responses, not the original history or evaluation answers.

Read condition	Token F1 (%)	Exact / 1,201
Selected-address-only	5.6645	20
Source Base decoded value	3.9375	14
Source Memory decoded value	8.8432	62
Cross-LoRA decoded value	4.0307	14
Command-V LoRA port	2.9150	12
FMC-decoded value	8.7352	60
Recovered text (same values)	8.7352	60

sizes. Each FMC factor payload occupies 61.25 MiB. Final answering in the single text-control measurement takes 295.3087 s versus 341.0842 s for address-only, with 9.4877 s shared loading. These omit parameter-value decoding and are not complete migration timings.

Source Base and Source Memory reuse the Qwen3.5-2B source with its existing memory adapter off and on, respectively; neither condition performs new training. Each first decodes all 250 selected addresses across nine streams. All decoded values are sealed before loading the 1,201 evaluation questions. The source model is unloaded before the same saved blank Qwen3.5-4B reasoner answers them through the shared index. Their query-weighted F1 scores are 3.9375 and 8.8432%, with 14 and 62 exact matches; stream-macro F1 is 3.9659 and 8.9507%. These are shared-reader source references, not newly trained target deployments. The temporary predecoded cache separates value decoding from final reasoning; its staged timings are not on-demand serving measurements.

The Command-V LoRA port reuses its existing aggregate intervention and converters, without training or refitting. It injects only at the final prompt position during value decoding and removes the intervention before the same blank target performs final reasoning. All 250 decoded values are sealed before the 1,201 questions are loaded. Its query-weighted F1 is 2.9150%, with 12 exact matches; stream-macro F1 is $3.0582 \pm 2.1417\%$ (sample standard deviation across nine streams). These shared-index results replace, rather than reuse, earlier reader measurements.

E.1 Canonical Memory Readback

The canonical-only check directly asks each of the 250 selected addresses with empty context. It keeps the saved FMC or Distilled LoRA adapter active, or restores the saved Target Base blank. All 19 physical state files are checked before generation. It uses the same 32-token budget and explicit stopping set; reference strings are the source responses used for installation. All three conditions and nine streams are retained, for 750 calls.

FMC reproduces all selected source answers under this fixed readback. Selection already depended on target-side trial fidelity, so the result is conditional on the admitted pool. It does not establish

Table 15: Readback at the 250 selected source addresses. Literal and normalized complete matches are distinct. These are not the 1,201 held-out natural questions.

State	Literal exact / 250	Normalized exact / 250	LoCoMo F1 (%)
FMC	250	250	100.0000
Distilled LoRA	36	40	33.8615
Target Base	1	1	2.1870

Table 16: B32 FMC across three independently trained source memories ($n = 3$ per task). Each task uses its fixed partition; KnowEdit sources expose different valid pools.

Task	Source A	Source B (main)	Source C
KnowEdit K3, correct / 736	372	394	381
MAB, correct / 60	49	54	49

perfect recovery of all 352 safe source responses, nor perfect held-out QA. The question-answering scores remain those in Table 13. The different natural-question direct interface is outside this check.

F Independent Sources and Model Upgrades

Source robustness changes the trained source, rather than repeating installation of one state. Sources A, B and C use writer seeds 41, 42 and 43 with the same task-specific recipe; Source B supplies the main results. Figure 4 shows their paired position comparisons.

Across these three source memories, mean accuracy and sample standard deviation are $51.95 \pm 1.50\%$ for KnowEdit K3 and $84.44 \pm 4.81\%$ for MAB. These describe source variation, separately from repeated installation timings.

Qwen3-1.7B to Qwen3-4B obtains 343/736 (46.60%) on KnowEdit and 51/60 (85.00%) on MAB. Qwen3-1.7B to SmolLM3-3B obtains 408/736 (55.43%) and 51/60 (85.00%). The KnowEdit source projection contains 406 safe responses, not the primary pair's 247. Position, boost, ridge and rank policies are copied before target evaluation; no target test score selects a new policy. These checks show transfer across model pairs, with a notably weaker KnowEdit condition on Qwen3-4B. They do not supply Distilled LoRA comparisons on those target models.

Each upgrade and source-confirmation condition records one full-input installation and separate model loading/evaluation costs, not six-repeat timing. Reused baseline states must match source checkpoint, target configuration and partition. Historical results with weaker input-hash records are identified in the artifact ledger rather than described as freshly reproduced tensors.

G Implementation and Reproduction

Algorithm 1 summarizes the compiler. Content selection precedes the final joint fit; the task-specific rules are given in Appendix A.

Each checkpoint retains its tokenizer and chat template. Backbone inference is bfloat16 without quantization; the solve is float32. Timings use one RTX 4090 without competing compute jobs. Full-forward versus early-stop output-head checks verify representations; they do not shorten input strings.

Our Cross-LoRA implementation fixes same-index attention correspondence, Q/K/V/O modules and rank-320 alignment bases (Xia et al., 2025). Model-pair caches are separate from transferred memory states. The Command-V port uses relative-depth mapping and bidirectional pseudoinverse converters. Source LoRA-on/off residuals at retained addresses are fitted as one rank-8 intervention (Wang et al., 2025a). Its converter is frozen from the first 1,030 public Alpaca prompts without benchmark-specific fitting. LoCoMo changes the state used to decode an address, not the final blank reasoner. Neither method trains on formal evaluation queries.

Algorithm 1 Batched functional memory compilation

Require: Source model and memory, retained addresses, target backbone, fixed task profile
1: Recover source responses and apply the fixed content-selection rule
2: Retokenize selected responses with the target tokenizer
3: **for** each batch of at most 32 complete address–response inputs **do**
4: Collect frozen final hidden states at the profile's answer positions
5: Append hidden keys and desired token residuals to (H, R)
6: **end for**
7: Solve the joint rank-constrained ridge objective in Eq. (1)
8: Scatter token rows into the target vocabulary; install and save one LoRA
9: Freeze the state and serving index before task evaluation

Table 17: Base checkpoints. Revision prefixes identify full hashes in resolved configurations.

Checkpoint	Width	Layers	Heads/KV	Vocabulary	Revision
Qwen3.5-2B	2,048	24	8/2	248,320	`15852e8c1636`
Qwen3.5-4B	2,560	32	16/4	248,320	`851bf6e806ef`
Qwen3-1.7B	2,048	28	16/8	151,936	`70d244cc86cc`
Qwen3-4B	2,560	36	32/8	151,936	`1cfa9a720891`
SmolLM3-3B	2,048	36	16/4	128,256	`a07cc9a04f16`

New runners bind source contracts, selected addresses, parameters, dependencies and state hashes before loading official evaluation records. Classification artifacts retain 1,224 KnowEdit rows and the evaluated MAB partition (60 primary rows; 100 in development-inclusive controls), candidate scores, and separately recomputed primary partitions. LoCoMo stores query IDs, references, route decisions, predictions, raw tokens and stopping reasons. CPU arithmetic rechecks are not new model execution; file hashing is distinct from CPU tensor comparison. Failed invocations and superseded protocols remain archived without overwriting states.

The supplementary code provides source readout, batched constraint collection, reduced-rank ridge compilation, adapter serialization, and target readout. Its instructions specify input formats, commands, and model prerequisites; benchmark records and source memory states are obtained separately. Evaluation references are not direct inputs to target construction; source training on the underlying memory and subsequent source recovery remain explicit upstream prerequisites.

H Exact Prompt and Source-Write Interfaces

Shared model prompt. Source reads, target value reads, candidate scoring and final reasoning use the pinned chat template with `add_generation_prompt=True` and `enable_thinking=False`. The system message is:

> You are a memory assistant. Answer using the current context and any persistent memory made available or encoded by the memory system. When memories conflict, prefer the information with the most recent timestamp. Give the shortest direct answer and no explanation. If the available context and memory do not support an answer, reply unknown.

The user message is exactly `Context:`, a newline, the context string, two newlines, `Question:`, a newline, and the question. Source canonical reads and teacher-forced compilation use empty context. The stability view prepends `Answer the following memory question with only the answer:` and a newline. Source read limits are 16 tokens for KnowEdit and 32 for LoCoMo.

FMC separately encodes the complete prompt and answer before concatenation. KnowEdit adds no termination constraint; LoCoMo adds one tokenizer EOS target at the representation after the full answer, not after a truncated prefix. It differs from the full assistant suffix supervised by Distilled LoRA. For LoCoMo reasoning, a usable pair is `Relevant memory 1`, newline, `Question:` and its address, newline, `Answer:` and its value. Top-1 routing supplies at most one pair; an

unaccepted route gives empty context. Decoding and answering both use 32 tokens and the explicit stopping set.

LoCoMo history to source supervision. Before source writing, Qwen3.5-4B converts each public session to at most four QA pairs using deterministic decoding, a 512-token limit and `qa_json_v4`. This checkpoint differs from the Qwen3.5-2B writer. Only session records enter conversion, not official queries. The ten-stream cache contains 272 event entries; session-text hashes and its manifest bind provenance. The exact conversion instruction is:

> You are converting one public memory event into QA supervision. The event may come from a synthetic benchmark. Artificial names and codes are intentional and must be treated as valid record values. Use only the public event text below; do not infer missing facts. If the event explicitly states that an entity has a value or association, you MUST produce one question-answer pair for that association. Produce only forward factual questions, with no inverse, yes/no, or paraphrase duplicates. Extract up to 4 self-contained factual question-answer pairs and preserve exact names, dates, places, quantities, preferences, codes, and relationship updates. Example event: Alice's contact key is X-123. Example output: {"pairs":[{"instruction":"What is Alice's contact key?","answer":"X-123"}]}. Second example event: I work for Northstar Labs. Second example output: {"pairs":[{"instruction":"Where does the speaker work?","answer":"Northstar Labs"}]}. Output only one JSON object with this schema: {"pairs":[{"instruction":"...","answer":"..."}]}. Use {"pairs":[]} only for an event containing no explicit stable fact, such as a pure opinion.
>
> Event text: [session text]

The rendered instruction uses ordinary JSON double quotes; the typography adds no tokens. Embedded image payloads are omitted while surrounding text is retained. The writer deduplicates nonempty pairs, trains under Table 5, and exports distinct instructions as answer-free addresses. Migration reads values anew from source memory. The supervision cache is a source-training input, not an external value store available to migration. Preprocessing and source training precede and are excluded from target installation.