When Skills Remain but Access Fails: Factorized Skill Memory for Continual Skill Internalization

Anonymous authors
Paper under double-blind review

Abstract

Skill internalization aims to convert externally specified procedural skills into persistent model capabilities, allowing language-model agents to execute them without retrieving or reinjecting skill descriptions at inference time. Yet when new skills are learned continually, whether previously internalized skills remain usable is still poorly understood. We study forgetting in continual skill internalization. Using correct-skill-address reactivation as a diagnostic, we identify a hierarchical pattern: closed-book execution can collapse while supplying the correct skill address substantially restores learned behavior, and continued optimization later degrades execution even under the correct skill address. We call these two regimes *skill-addressing forgetting* and *skill-execution forgetting*. Guided by this finding, we introduce **Factorized Skill Memory (FSM)**, which separates continual skill memory into a plastic *Address Memory* and a stable *Execution Memory*. At each stage, FSM trains a new LoRA executor using only current-stage data, freezes it after acquisition, and appends it to Execution Memory; in parallel, Address Memory incrementally learns to map current requests to executors without replaying historical requests or execution trajectories. At inference time, Address Memory selects one executor from the raw request, and only that frozen executor performs a single language-model generation. On a five-stage benchmark built from 100 TaskBench skills, FSM achieves 65.95% mean exact match and 93.38% shard-routing accuracy across two task orders, improving over Progressive-LoRA by 14.3 percentage points while remaining within 1.9 points of Replay, which retains historical training examples. These results support an Address–Execution view of continual skill retention, in which stable procedural content and reliable access are distinct design problems. Code is available at `https://anonymous.4open.science/r/fsm-anonymous-release-F7A3/`.

1 Introduction

Large language model agents increasingly rely on reusable **Skills** to organize tool use, environment interaction, and multi-step action generation (Yao et al., 2023; Schick et al., 2023; Shen et al., 2024). A Skill packages procedural guidance into a workflow: which operations to invoke, how their inputs are constructed, and how intermediate steps depend on one another (Wang et al., 2023a; Yang et al., 2026). Most skill-based systems still keep this guidance outside the model and retrieve the relevant Skill text at inference time. Execution therefore depends both on retrieval and on the model's ability to follow the retrieved description. Long contexts are used unevenly, irrelevant context can distract reasoning, and detailed Skill documentation may not reliably improve execution (Liu et al., 2024; Shi et al., 2023; Li et al., 2026). Recent work on **Skill Internalization** moves this procedural guidance into model parameters, allowing learned behavior to be used without repeatedly injecting the Skill description (Lu et al., 2026; Lin et al., 2026; Zhang & Qi, 2026; Yu et al., 2026).

We consider agents that acquire new Skills over time. In this continual-learning setting, disjoint Skill families arrive sequentially, the model internalizes these Skills in sequence, and the final model receives only the user request at inference time, without a Skill family label, Skill name, or retrieved Skill card, while retaining the capabilities of all earlier Skills (Parisi et al., 2019; De Lange et al., 2022; van de Ven & Tolias, 2019). Existing skill-internalization studies do not establish whether

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.

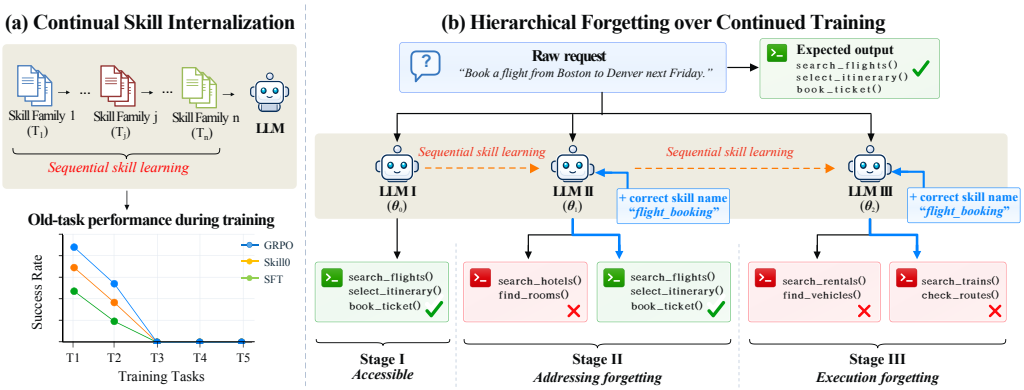

Figure 1: Continual skill internalization and hierarchical forgetting. **(a)** Sequential learning can erase old-task closed-book performance within a few optimization steps. **(b)** A matched request-level example exposes two stages of forgetting. During addressing forgetting, the correct Skill name recovers execution that the raw request no longer reaches. Continued optimization can later damage execution under the correct skill address itself.

earlier Skills remain usable after later Skills are learned. During continual Skill Internalization training, old-Skill closed-book performance can fall to zero within only a few optimization steps after training moves to a new Skill family. Successful internalization does not by itself ensure retention.

The collapse in closed-book performance hides a more structured failure. At a checkpoint that fails all 408 old-task requests, supplying the **correct skill address**, given by the benchmark's exact procedure identifier or identifiers required by the request, restores execution performance from 0 to 69.85%. Because this address also reveals the target node identity, we use it as a diagnostic rather than an information-neutral access probe. With further optimization, execution under the correct skill address degrades, and the same ordering appears across SFT, GRPO, and SKILL0. We call the two regimes skill-addressing forgetting and skill-execution forgetting: closed-book execution fails first while execution under the correct skill address remains recoverable, before that recoverability itself erodes.

Guided by this finding, we introduce **Factorized Skill Memory (FSM)**, which separates continual skill memory into a plastic **Address Memory** and a stable **Execution Memory**. The reactivation diagnostic exposes the Address–Execution split at individual-skill granularity; FSM operationalizes the same split more coarsely, using stage-level shards that each contain multiple Skills. At each stage, FSM trains and freezes one new LoRA executor while Address Memory learns request-to-shard associations without retaining historical requests or execution trajectories. The selected executor still identifies and composes the required Skills within its shard. At inference, Address Memory selects one shard from the raw request, and only that frozen executor performs a language-model generation.

We make three contributions:

- We construct a five-stage benchmark for **continual skill internalization** by partitioning 100 TaskBench Skills into disjoint Skill families and evaluating sequential acquisition, retention, and autonomous closed-book access.
- We identify a **hierarchical forgetting** pattern under canonical-name reactivation: closed-book execution can collapse while execution under the correct skill address remains recoverable, before that recoverability also erodes.
- We introduce **FSM**, which separates plastic Address Memory from frozen Execution Memory. Across two task orders, FSM reaches 65.95% mean performance and 93.38% shard-routing accuracy, improving over the strongest replay-free parameter-isolation baseline by 14.3 points while remaining within 1.9 points of Replay.

2 Related Work

Skill Internalization. Agent Skills are commonly represented as procedural artifacts that are retrieved and inserted into the model context at inference time (Li et al., 2026; Jiang et al., 2026).

Recent work studies how to convert this guidance into persistent model behavior. SKILL0 and SkillC learn to execute Skills with progressively less dependence on explicit Skill context (Lu et al., 2026; Lin et al., 2026). Another line of work parameterizes procedural knowledge directly: Skill-to-LoRA and ParametricSkills map Skills into adapter parameters, while LatentSkill and PEAM move textual or experiential Skill information into in-weight representations (Zhang & Qi, 2026; Zhao et al., 2026; Yu et al., 2026; Guo et al., 2026). LifeSkill further studies online lifelong adaptation from Skill-guided experience (Mao et al., 2026). Together, these methods show that procedural guidance can be transferred from context into model parameters, and several introduce mechanisms for accumulating such capabilities over time. Our focus is the failure structure that emerges when internalized Skills are learned sequentially without Skill identity at test time. We distinguish loss of request-side access from loss of execution itself, and use this Address–Execution hierarchy to determine what should remain plastic and what should be protected during continual internalization.

Continual Learning. Continual learning studies how models acquire new tasks without overwriting earlier competence. Replay preserves old examples, regularization constrains parameter changes, and architectural methods isolate or expand parameters for new tasks (Kirkpatrick et al., 2017; Li & Hoiem, 2018; Lopez-Paz & Ranzato, 2017; Rebuffi et al., 2017b; Rusu et al., 2016). The last family is especially relevant when different tasks require distinct behaviors. Adapter-based continual learning allocates task-specific parameters (Madotto et al., 2021); EPI combines private parameter-efficient modules with a non-parametric task identifier so that the appropriate module can be recovered without storing old training data (Wang et al., 2023c); and MoCL continually adds and composes modules to preserve knowledge while supporting transfer (Wang et al., 2024). FSM falls within this modular parameter-isolation line, with its decomposition motivated by the failure structure of continual Skill internalization. In continual Skill internalization, our reactivation experiments show that request-to-Skill access can fail while conditional execution remains intact. FSM therefore isolates the acquired procedural execution in frozen multi-Skill shards and treats access as the updated memory state.

3 Continual Skill Internalization

Problem setting. We consider a sequence of stages $t = 1, \ldots, T$, where each stage introduces a previously unseen skill set $\mathcal{S}_t$ and corresponding request–workflow supervision. The skill sets are pairwise disjoint, and after stage t the desired competence includes all skills in $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_t$. For a request x, successful closed-book execution requires the model to produce the correct request-conditioned workflow using only the raw request. Training may use stage-local workflow supervision and, for skill-guided baselines, textual skill cards. Evaluation is autonomous and closed-book: the model receives only x, not a stage label, skill name, or catalog. This setting tests *acquisition* of new skills and *retention* of earlier skills under autonomous inference.

Benchmark. We instantiate this stream using the public Microsoft TaskBench benchmark (Shen et al., 2024), partitioning its 100 canonical skills into five disjoint semantic families (Table 1). We use each family as a unit of continual acquisition: a capability package containing several related procedures. The five packages contain 40, 11, 21, 9, and 19 canonical skills. Requests can require multi-node workflows composed from several skills in the same package. Each stage contains 1,000 training requests, for 5,000 in total, and the five held-out sets contain 2,186 requests. The complete skill inventory and output contract appear in Appendix A.

Table 1: TaskBench benchmark statistics. Each stage uses 1,000 training requests.

Task	Semantic family	# Skills	Eval
T1	Daily Life	40	864
T2	Language / Speech	11	408
T3	Retrieval / Understanding	21	332
T4	Vision / Document	9	299
T5	Generation / Editing	19	283
Total	–	100	2,186

Metrics. Our primary metric is **exact-match task success (EM)**, which requires the complete prescribed workflow output to be correct; for TaskBench this is the benchmark's canonical graph exact match, but we refer to it simply as EM in the main text. The T1–T5 columns and their task-macro average all report this same quantity. We use three secondary diagnostics to interpret failures: **Tool-choice accuracy** checks whether the ordered tool sequence is correct while ignoring request-specific values; **Argument-value F1** compares predicted and target $(\text{skill}, \text{argument}, \text{value})$

tuples; and **BWT** measures final old-task EM minus EM immediately after that task was learned, with zero indicating no forgetting; additional benchmark-specific metrics appear in the appendix.

Direct continual optimization rapidly eliminates old-task closed-book performance. Before introducing any mechanism for preservation, we first measure how direct continual optimization behaves when training moves away from an acquired task. Immediately after learning the old task, EM is 77.45% for SFT, 84.31% for GRPO (Shao et al., 2024), and 84.07% for the SKILL0 (Lu et al., 2026) curriculum. After optimization switches to the next task, old-task EM falls to zero at observed steps 8, 6, and 10, respectively, and remains there over the rest of the measured interval (Figure 2). Across hundreds of held-out requests, none of the old-task outputs remain exactly correct. Under the standard closed-book criterion, all three procedures therefore exhibit catastrophic forgetting of the previously internalized skill. This endpoint shows that retention has failed without revealing which part of the acquired capability was lost. Section 4 tests this distinction directly.

4 When Forgotten Skills Come Back

4.1 Correct-Skill-Address Reactivation

Section 3 showed that old-task closed-book performance can fall to zero after continual optimization moves to a new task. That endpoint does not reveal whether the old procedure has been erased or whether the model has lost access to behavior that is still present. We distinguish these cases with a controlled reactivation intervention. We take a post-forgetting checkpoint that scores zero on all 408 closed-book T2 requests and supply only the *correct skill address*, given by the benchmark's exact procedure identifier or identifiers required by each request. On the same frozen model, **full-task EM recovers from 0 to 69.85%.** The address provides no argument values, workflow steps, or target serialization, so the model must still construct the request-conditioned workflow. Because it also reveals the gold node identity, we use the correct skill address as a diagnostic rather than a pure access probe. Additional controls are reported in Appendix B.1.

A frozen-base control further separates address recognition from learned execution. On 408 T2 requests, frozen Qwen3-8B (Yang et al., 2025) selects the correct skill on 92.65% of examples when given the correct skill address but obtains 0% full-task EM; by contrast, the post-forgetting checkpoint recovers 69.85% under the same address after request-only performance reaches zero. Recognizing the supplied address alone therefore cannot account for the recovered behavior. We use *Address* for access to the relevant procedural state and *Execution* for producing the request-conditioned workflow once that state is available. This reversal shows that substantial execution remains available once the correct procedure identity is supplied; we call this regime *skill-addressing forgetting*.

4.2 Execution under the correct skill address erodes later

To see what happens after this access failure, Figure 2 follows the first 20 optimizer steps of the T2→T3 transition, evaluating each checkpoint closed-book and with the correct skill address. When closed-book T2 first reaches zero, full-task EM under the correct skill address is still 76.96% for SFT, 84.07% for GRPO, and 77.70% for SKILL0. Because the paired measurements come from the same checkpoint, this gap shows a loss of autonomous closed-book access relative to the target-identity-assisted condition while conditional execution remains available. Further T3 optimization then reduces performance under the correct skill address from 77.45% to 58.82% for SFT, from 84.56% to 57.60% for GRPO, and from 85.29% to zero for SKILL0. We call this later failure *skill-execution forgetting*: even the correct address can no longer recover the earlier behavior.

The bottom matrices show the same ordering across other continual-learning methods: closed-book performance can collapse before execution under the correct skill address. This diagnostic operates at individual-skill granularity. FSM applies the same functional separation at shard granularity: routing selects a multi-skill capability package, while the selected executor still performs within-shard skill selection, composition, and execution.

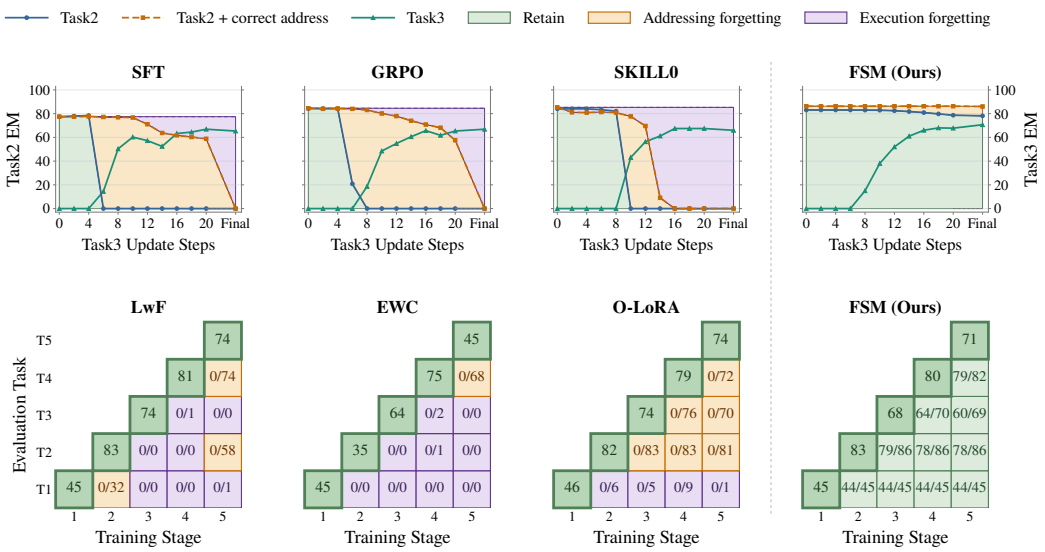

Figure 2: Hierarchical forgetting during continual skill internalization. **Top:** during the T2→T3 transition, closed-book Task2 EM can collapse while Task2 remains recoverable when the correct skill address is supplied; with continued optimization, execution under the correct skill address can also erode. The frozen-executor diagnostic preserves the old task while learning Task3. **Bottom:** five-stage evaluation matrices for four continual-learning conditions. Off-diagonal cells report closed-book / + Correct Skill Address EM. Across both rows, pale green denotes retained performance, pale yellow denotes addressing forgetting, and pale purple denotes execution forgetting. Diagonal cells show task-acquisition EM. The FSM column is our method.

5 Factorized Skill Memory

Section 4 separates request-side access from execution at individual-skill granularity. FSM preserves this functional split without allocating one executor per skill: related skills are grouped into stage-level execution shards, and Address Memory learns only which shard to read. Formally,

$$\mathcal{M}_t = (A_t, \mathcal{E}_t),$$

where A_t is **Address Memory** and $\mathcal{E}_t = \{E_1, \ldots, E_t\}$ is **Execution Memory**. Address remains plastic as new skill families arrive, whereas Execution Memory is append-only. At stage t, FSM trains and freezes a fresh executor E_t on $\mathcal{S}_t$; current requests update Address with shard identity t and are then discarded. Historical requests and workflow trajectories are never replayed.

Each shard contains 9–40 canonical skills, so shard routing only narrows the execution space to a capability package. It does not replace within-shard skill selection: the chosen executor must still recover the required skill sequence, request-specific arguments, and dependencies from the raw request. At inference, Address selects one shard and only that executor performs language-model generation. Figure 3 summarizes the write-and-read process; Appendix A provides further details.

5.1 Execution Memory: Immutable Parametric Shards

Let B denote the shared frozen base model. At stage t, FSM initializes a fresh LoRA executor $E_t = B \oplus \Delta_t$, trains Δ_t only on current-stage skill-execution supervision, freezes it after acquisition, and appends it to the bank, $\mathcal{E}_t = \mathcal{E}_{t-1} \cup \{E_t\}$. No subsequent stage updates an earlier Δ_j. This boundary follows the failure pattern in Section 4: conditional execution can survive after request-side access has failed, so FSM protects the parameters that store execution and places subsequent adaptation in Address Memory. Low-rank adapters provide a compact implementation of these isolated parametric writes (Hu et al., 2022; Rebuffi et al., 2017a; Houlsby et al., 2019).

Each shard stores a family of related skills rather than a single procedure. For example, the T1 shard contains 40 canonical skills; selecting T1 only determines which execution memory to read, while the

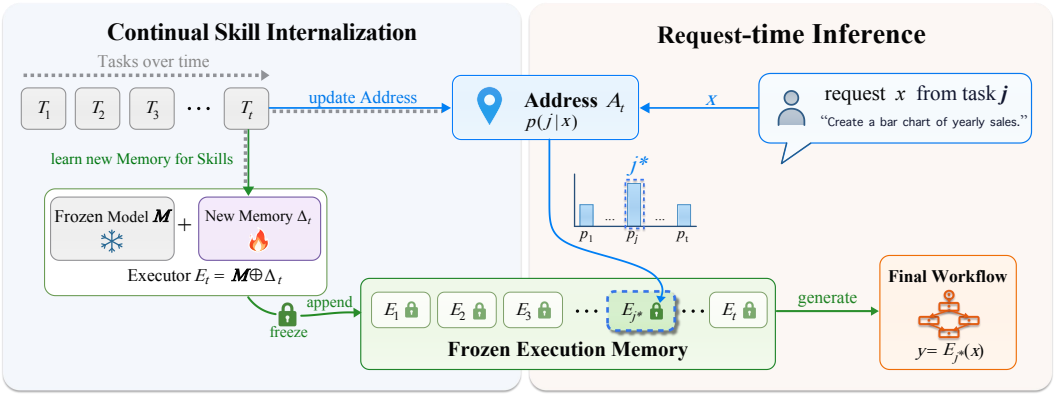

Figure 3: Factorized Skill Memory (FSM). **Left: memory write.** At stage t, current requests update Address Memory while a fresh executor $E_t = \mathcal{B} \oplus \Delta_t$ is trained from the frozen base and a stage-specific LoRA. The completed executor is frozen and appended to Execution Memory as a new skill-family shard. **Right: memory read.** Address maps the raw request to $p(j \mid x)$, selects $j^\star$, and reads one frozen execution shard. Only $E_{j^\star}$ performs language-model generation.

executor still chooses and composes the required skills and produces the complete request-conditioned graph. The shared 8.2B backbone is stored once. Under our rank-16 LoRA configuration on the $q/k/v/o$ projections, each Qwen3-8B shard adds 15.34M parameters, about 0.19% of the backbone, and the complete five-shard bank adds 76.68M parameters, about 0.94%. The five BF16 LoRA deltas occupy about 146 MiB, and inference activates only the selected LoRA.

5.2 Address Memory: Continual Associative Indexing

With earlier execution shards fixed, the continually updated state is the request-to-shard index

$$p_{\text{Address}}(j \mid x), \qquad j \in \{1, \ldots, t\}.$$

The test request contains no task label, correct skill address, or skill catalog, so shard identity must be recovered from request text alone. Our deployed Address maps request text into a fixed-dimensional hashing representation and updates an incremental Naive Bayes classifier using only the current request pool with label t. After the update, those requests are discarded; the retained state consists only of class and feature statistics. Address Memory can therefore absorb new request-to-shard associations without retaining historical requests while the executor bank stays fixed.

5.3 Single-Shard Memory Read

At inference, FSM performs a hard memory read:

$$j^\star = \arg\max_j p_{\text{Address}}(j \mid x), \qquad y = E_{j^\star}(x).$$

The read is single-shard but multi-skill: Address selects one capability package, and the chosen executor can compose several canonical skills from that package while resolving request-specific arguments and graph dependencies. The language-model cost is one executor generation per request. Because Address uses hashing plus Naive Bayes, shard selection requires neither an additional language-model forward pass nor bank-wide scoring.

6 Main Results

6.1 Experimental setup

We evaluate whether continually internalized skills remain usable after later skill families are acquired. Our primary experiments use Qwen3-8B with rank-16 LoRA on the $q/k/v/o$ projections; Qwen3-4B provides an additional model-scale robustness check. Each stage executor is trained for six epochs

Table 2: Main comparison across both task orders. Metrics are on a 0–100 scale; Order 2 values are placed by task identity. Tool Choice is ordered tool-list exact match, Arg. F1 compares argument-value tuples, Avg. is task-macro EM, and BWT is final minus post-acquisition EM over the first four arrival stages. Indented `w/ correct skill address` rows report percentage-point changes from the closed-book row above. A dash denotes an unreported metric.

Method	Tool Choice	Arg. F1	EM						BWT
			T1	T2	T3	T4	T5	Avg.	
Base + Skill	49.57	52.24	19.91	22.79	50.30	29.77	47.70	34.09	–
Base + Oracle Address+Fields	90.71	81.01	23.84	70.10	59.94	69.23	56.89	56.00	–
Order 1: $T1 \to T2 \to T3 \to T4 \to T5$									
Replay (reference)	85.90	83.75	43.75	77.21	70.48	80.27	74.91	69.32	-1.76
Progressive-LoRA	65.54	65.89	31.13	66.67	32.83	76.25	53.00	51.98	-8.38
LwF (Li & Hoiem, 2018)	18.87	17.52	0.00	0.00	0.00	0.00	**73.85**	14.77	-70.76
w/ correct skill address	+45.91	+29.84	+1.16	+58.33	+0.00	+73.58	+3.54	+27.32	–
EWC (Kirkpatrick et al., 2017)	9.89	10.53	0.00	0.00	0.00	0.00	44.88	8.98	-54.94
w/ correct skill address	+21.73	+19.35	+0.00	+0.00	+0.00	+67.56	+6.71	+14.85	–
O-LoRA (Wang et al., 2023b)	18.45	17.53	0.00	0.00	0.00	0.00	**73.85**	14.77	-70.19
w/ correct skill address	+70.79	+50.42	+1.16	+81.13	+70.18	+72.24	-2.47	+44.45	–
FSM	**82.57**	**80.71**	**44.21**	**78.19**	**59.64**	**78.93**	70.67	**66.33**	**-3.71**
Order 2: $T3 \to T1 \to T5 \to T2 \to T4$									
Replay (reference)	81.70	81.03	42.94	74.02	68.67	78.26	68.20	66.42	17.98
Progressive-LoRA	64.72	65.21	31.13	66.67	31.93	75.59	51.24	51.31	-11.51
LwF (Li & Hoiem, 2018)	17.73	17.50	0.00	0.00	0.00	78.60	0.00	15.72	-19.05
w/ correct skill address	+63.67	+46.54	+0.00	+74.51	+59.34	+3.67	+59.36	+39.38	–
EWC (Kirkpatrick et al., 2017)	16.92	16.62	0.00	0.00	0.00	75.25	0.00	15.05	-62.50
w/ correct skill address	+24.19	+11.68	+3.12	+41.67	+0.00	+3.68	+0.00	+9.69	–
O-LoRA (Wang et al., 2023b)	17.39	17.36	0.00	0.00	0.00	**78.93**	0.00	15.79	-68.77
w/ correct skill address	+71.44	+52.37	+1.85	+83.33	+72.29	+0.67	+66.78	+44.98	–
FSM	**81.78**	**79.94**	**44.68**	**78.19**	**57.83**	77.59	**69.61**	**65.58**	**-5.11**

with learning rate 4×10^{-4} and effective batch size eight, then frozen permanently. Evaluation uses greedy decoding with thinking disabled and a 768-token generation cap. We report both task orders from Section 3. At test time, FSM receives only the raw request: its replay-free Address Memory selects one stage-level execution shard, and only that frozen executor generates the workflow.

Dataset. We use the TaskBench stream from Section 3: 100 skills in five disjoint families, 1,000 training requests per stage, and 2,186 held-out requests. We evaluate $[T1, T2, T3, T4, T5]$ and $[T3, T1, T5, T2, T4]$. Closed-book methods receive only the raw request; correct-skill-address rows supply the correct skill address only for diagnostic evaluation.

Baselines. We include three reference conditions with additional information or memory. **Base + Skill** gives the frozen base model the full skill catalog at inference, **Base + Oracle Address+Fields** supplies the correct skill address and gold argument fields, and **Replay** retains 100 examples from each previous task. Our replay-free continual baselines are **LwF** (Li & Hoiem, 2018), **EWC** (Kirkpatrick et al., 2017), **O-LoRA** (Wang et al., 2023b), and **Progressive-LoRA**. For LwF, EWC, and O-LoRA, the correct-skill-address rows reevaluate the same final checkpoints with the correct skill address and no retraining; these rows are diagnostic rather than deployment conditions. Appendix C reports Current-only SFT, GRPO (Shao et al., 2024), SKILL0 (Lu et al., 2026), and Parallel-Additive LoRA.

6.2 Overall and per-task performance

Table 2 evaluates the final checkpoints under both task orders. FSM averages 65.95% task-macro EM, 14.3 points above Progressive-LoRA and within 1.9 points of Replay. Its performance is also stable to task order, reaching 66.33% on Order 1 and 65.58% on Order 2. The correct-skill-address

Table 3: Qwen3-4B Order-1 results. Tool Choice and Arg. F1 are percentages; T1–T5 report final closed-book graph exact match, Avg. is the task-macro mean, and BWT is mean final-minus-acquisition EM over T1–T4. Correct-skill-address rows are percentage-point deltas from the row above.

Method	Tool Choice	Arg. F1	EM						BWT
			T1	T2	T3	T4	T5	Avg.	
Replay (reference)	86.58	84.05	43.87	80.15	71.08	79.26	74.20	69.71	-1.39
Progressive-LoRA	56.78	60.03	18.29	62.99	29.82	73.58	49.82	46.90	-7.84
LwF (Li & Hoiem, 2018)	18.80	17.53	0.00	0.00	0.00	0.00	73.85	14.77	-70.31
w/ correct skill address	+17.73	+15.21	+0.00	+0.00	+0.00	+70.23	-0.35	+13.98	–
EWC (Kirkpatrick et al., 2017)	8.20	9.03	0.00	0.00	0.00	0.00	38.52	7.70	-37.64
w/ correct skill address	-8.20	-9.03	+0.00	+0.00	+0.00	+0.00	-38.52	-7.70	–
O-LoRA (Wang et al., 2023b)	18.45	17.53	0.00	0.00	0.00	0.00	72.79	14.56	-69.96
w/ correct skill address	+66.79	+47.28	+0.00	+71.57	+71.39	+67.56	+2.12	+42.53	–
FSM	**82.60**	**80.48**	**44.79**	**79.17**	**57.53**	**79.93**	**71.73**	**66.63**	**-3.58**

Table 4: Per-task Graph EM for selector variants across both task orders. EPI and AdapterCL use the same frozen execution banks as FSM. Macro is the task average.

Order	Task	Oracle	Self-route	EPI	AdapterCL	FSM
Order 1	T1	45.02	42.01	35.18	41.25	**44.21**
	T2	83.58	58.82	59.41	64.19	**78.19**
	T3	75.60	56.02	51.09	**62.89**	59.64
	T4	80.27	77.26	63.94	70.74	**78.93**
	T5	76.33	**72.44**	54.84	64.24	70.67
	Macro	72.16	61.31	52.90	60.66	**66.33**
Order 2	T1	45.49	42.82	35.55	41.25	**44.68**
	T2	83.33	56.62	59.41	63.97	**78.19**
	T3	73.80	57.23	49.88	**63.16**	57.83
	T4	78.93	**77.93**	62.88	70.74	77.59
	T5	75.27	**73.14**	53.99	64.87	69.61
	Macro	71.36	61.55	52.34	60.80	**65.58**

diagnostic recovers several otherwise weak closed-book checkpoints, showing that low autonomous performance can coexist with substantial execution under the correct skill address, as observed in Section 4.

LwF, EWC, and O-LoRA lose closed-book performance on most earlier tasks, whereas FSM retains much more of the stream. Its BWT is -3.71 on Order 1 and -5.11 on Order 2, versus -8.38 and -11.51 for Progressive-LoRA. On Order 2, the correct-skill-address diagnostic raises LwF and EWC to 55.10% and 24.74%, showing that some failures still preserve execution under the correct skill address.

Table 3 provides a Qwen3-4B robustness check. FSM reaches 66.63% task-macro EM with BWT -3.58, close to the Qwen3-8B Order 1 result of 66.33% and -3.71. Because archived 4B baselines use different training recipes, this comparison is descriptive, but the similar FSM result suggests that the factorized design is not specific to the 8B backbone.

Selector variants. To characterize request-to-shard selection with a fixed execution bank, we compare Oracle routing, confidence-based Self-route, the replay-free FSM Address, EPI task identification (Wang et al., 2023c), and AdapterCL perplexity routing (Madotto et al., 2021).

FSM reaches 93.38% mean routing accuracy, compared with 85.15% for AdapterCL and 74.28% for EPI; with the same frozen execution banks, their mean Graph EM is 65.95%, 60.73%, and 52.62%, respectively. Figure 4 shows the task-level routing differences.

Oracle routing reaches 71.76% mean EM, compared with 65.95% for FSM and 61.43% for Self-route. With the execution bank fixed, the 5.81-point FSM–Oracle gap reflects remaining shard-access error.

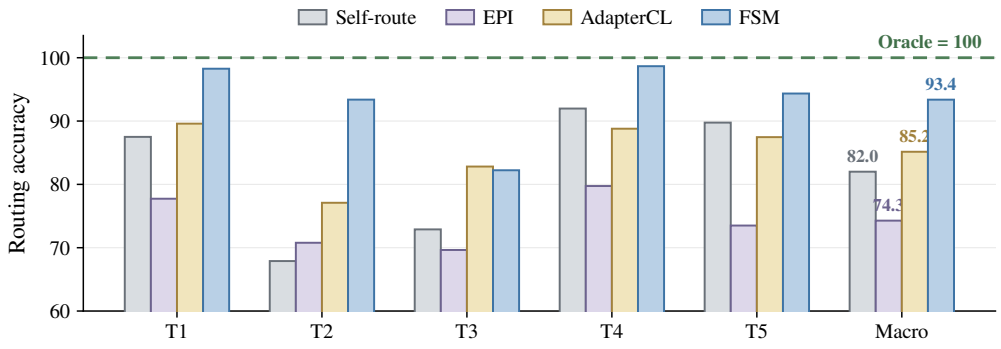

Figure 4: Two-order mean routing accuracy by task for Self-route, EPI, AdapterCL, and FSM. The dashed horizontal line marks Oracle routing at 100%. Macro is the task average.

Oracle itself still lies 28.24 points below perfect EM after the correct shard is supplied, because the selected executor must recover the skill sequence, arguments, dependencies, and complete workflow. Thus both access and within-shard execution contribute materially to the final error.

The task-level pattern makes this split concrete. T2 is mostly access-limited: Oracle reaches 83.58% / 83.33%, FSM 78.19% in both orders, and Self-route 58.82% / 56.62%. T1 is mostly execution-limited because FSM nearly matches an Oracle that reaches only 45.02% / 45.49%. T3 is mixed: Oracle remains above 73%, while all learned selectors leave a larger gap; AdapterCL is stronger than FSM on this family. T4 again lies close to Oracle, whereas Self-route slightly exceeds FSM on T5. These differences show why a single final EM score is insufficient: some families are dominated by access error and others by the execution quality of the selected shard. FSM's macro advantage comes mainly from recovering the access-sensitive cases while remaining competitive elsewhere.

With the execution bank fixed, offline Address reaches 69.15% mean EM and 97.03% routing accuracy, versus 65.95% and 93.38% for sequential Address (Appendix B.5). This 3.20-point difference serves as a historical-request reference rather than an isolated replay effect, because the two selectors also differ in both representation and classifier family (hashing+NB versus TF–IDF+LR). The stronger offline selector closes a noticeable portion of the remaining Oracle gap, indicating that improved request-to-shard access still offers meaningful headroom beyond the deployed sequential Address.

FSM avoids bank-wide language-model scoring: Address performs a lightweight request-to-shard lookup, activates one frozen LoRA, and uses only one executor generation for each incoming request. Each rank-16 execution shard adds 15.34M parameters; five shards add 76.68M, or only 0.94% of the shared 8.2B backbone. The backbone itself is stored only once, and only the selected LoRA is activated during inference. As Execution Memory grows, storage therefore increases with the number of frozen shards, while the per-request language-model cost remains a single executor generation.

7 Conclusion

Correct-skill-address reactivation reveals a hierarchical forgetting pattern: closed-book execution can collapse while execution under the correct skill address remains substantially recoverable, and continued optimization can later erase that recoverability as well. FSM follows this distinction by keeping Address Memory plastic and storing acquired procedures in frozen, multi-skill Execution Memory shards that remain unchanged during later stages. Across two five-stage task orders, FSM reaches 65.9593.38percentage points while remaining within 1.9 points of Replay. These results support a simple design principle for continual skill memory: preserve acquired procedural content while allowing access to adapt as the skill stream grows. They also separate two remaining sources of error: Address determines which execution shard is reached, whereas the selected shard determines how well the requested workflow is executed. This decomposition makes retention failures more interpretable and suggests where additional capacity or supervision is most useful.

AI Use Statement. AI-assisted tools were used for language polishing, data cleaning and consistency checks, and figure preparation and refinement. All scientific claims, experimental results, analyses, and final manuscript content were reviewed and verified by the authors.

Reproducibility Statement. We provide the protocol, prompts, metrics, training and inference settings, FSM procedure, and supporting controls, with code and artifacts at `https://anonymous.4open.science/r/fsm-anonymous-release-F7A3/`.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3366–3375, 2017.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446.

Yuchen Guo, Junli Gong, Weicheng Wang, Hongmin Cai, Yiu-ming Cheung, and Weifeng Su. Peam: Parametric embodied agent memory through contrastive internalization of experience in minecraft. *arXiv preprint arXiv:2605.27762*, 2026.

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, 2019.

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

Zhiyuan Jiang, Fangrui Huang, Hanwen Xing, Xander Wu, Yipeng Gao, Rui Cao, Mengdi Wang, Shilong Liu, and Yijiang Li. Demystifying agent skills: Why they work—until they don't. *arXiv preprint arXiv:2608.14036*, 2026.

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

Xiangyi Li et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

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

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. Skillc: Learning autonomous skill internalization in llm agents via contrastive credit assignment. *arXiv preprint arXiv:2605.27899*, 2026.

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.

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

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7452–7467, 2021. doi: 10.18653/v1/2021.emnlp-main.590.

Bo Mao, Jie Zhou, Yutao Yang, Xin Li, Xian Wei, Qin Chen, Xingjiao Wu, and Liang He. Learning while acting: A skill-enhanced test-time co-evolution framework for online lifelong learning agents. *arXiv preprint arXiv:2606.04815*, 2026.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. doi: 10.1016/j.neunet.2019.01.012.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30, 2017a.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017b.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schuetze. Rehearsal-free modular and compositional continual learning for language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 469–480. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-short.39. URL https://aclanthology.org/2024.naacl-short.39/.

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

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10933–10946. Association for Computational Linguistics, 2023c. doi: 10.18653/v1/2023.acl-long.612.

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

Cehao Yang, Xiaojun Wu, Honghao Liu, Xueyuan Lin, Chengjin Xu, Xuhui Jiang, Yuanliang Sun, Wenjie Zhang, Zhichao Shi, Yijie Xu, Jia Li, Hui Xiong, and Jian Guo. A survey of agent skills: Toward procedural infrastructure for llm agents. *Preprints*, 2026. doi: 10.20944/preprints202605.1276.v1.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Aofan Yu, Chenyu Zhou, Tianyi Xu, Zihan Guo, Rong Shan, Zhihui Fu, Jun Wang, Weiwen Liu, Yong Yu, Weinan Zhang, and Jianghao Lin. Latentskill: From in-context textual skills to in-weight latent skills for llm agents. *arXiv preprint arXiv:2606.06087*, 2026.

Tianyi Zhang and Zhonghao Qi. Skill-to-lora: From using skills to learning behaviors for token-efficient llm agents. *arXiv preprint arXiv:2606.16769*, 2026.

Xuan Zhao, Haonan He, Qingyu Yang, Minglei Li, Jingqi Ye, Zelin Tan, Bo Wan, and Peng Ye. Parametric skills. *arXiv preprint arXiv:2606.30015*, 2026.

A EXPERIMENTAL DETAILS AND REPRODUCIBILITY

This appendix records the benchmark protocol, evaluation contract, metrics, FSM implementation, information budgets, and safeguards used for the main results.

A.1 CONTINUAL INTERNALIZATION PROTOCOL

At stage t, the learner receives request–graph pairs from a previously unseen skill set $\mathcal{S}_t$ and then moves to $\mathcal{S}_{t+1}$. The skill sets are pairwise disjoint, and earlier skills remain part of the target competence. For a request x, the model must produce an executable graph g: nodes specify canonical skills and arguments, and directed edges encode producer–consumer dependencies. Training may use graph supervision and, for skill-guided baselines, textual skill cards. Evaluation is closed-book: the model receives only x, with no stage label, skill name, catalog, demonstration, retrieved memory, or target graph.

We evaluate two complete task orders, $O1 = [T1, T2, T3, T4, T5]$ and $O2 = [T3, T1, T5, T2, T4]$. The primary metric is exact match of the canonical skill graph. Component metrics separate errors in skill selection, argument values, and dependency edges. Backward transfer measures how performance on earlier tasks changes after later stages. All reported aggregates are task macros, so each family has equal weight despite different held-out set sizes.

A.2 BENCHMARK INVENTORY AND CLOSED-BOOK CONTRACT

Table A1 gives the five-family split. Each family contains 1,000 training requests, and final task scores use its complete held-out set. The headline aggregate is the macro mean across families.

ID	Family	Skills	Train	Held-out
T1	Daily Life	40	1,000	864
T2	Language/Speech	11	1,000	408
T3	Retrieval/Understanding	21	1,000	332
T4	Vision/Document	9	1,000	299
T5	Generation/Editing	19	1,000	283
Total	Five disjoint families	100	5,000	2,186

Table A1: Benchmark inventory. Train and held-out counts are request–graph pairs; the two orders reuse the same held-out sets.

The closed-book system contract requires exactly one JSON object containing `task_nodes` and `task_links`. Nodes must be topologically ordered and include a canonical task identifier with named arguments. Links specify directed producer–consumer dependencies. The user message contains only the original request; skill names, family IDs, catalogs, workflows, demonstrations, retrieval results, and target graphs are absent. We use greedy generation with thinking disabled and a 768-token cap. Evaluation parses and canonicalizes the response before computing graph and component metrics. No test-informed repair is applied.

A.3 CLOSED-BOOK PROMPT AND AN EXAMPLE RECORD

Figure A1 reproduces the system instruction and user template used for all five stages. The model uses its chat template with thinking disabled. Gold graphs are training targets and evaluation references, never closed-book inputs. Both task orders use the same prompt and held-out requests; only the acquisition order changes.

```
[system]
You are a task-automation planner.
Return exactly one JSON object and no other text.
The JSON object must have exactly these two keys in this order:
    task_nodes, task_links.
task_nodes is a non-empty topologically ordered list. Every node has
    exactly task and arguments.
task is an exact canonical skill identifier. arguments is a list of
    objects with exactly name and value.
task_links is a list of objects with exactly source and target. Use <
    node-i> when an argument consumes node i.
Use the minimum skills needed. Do not emit markdown, explanation, a
    skill list, a plan, or text before or after JSON.

[user]
Plan the following request:
{user_request}
```

Figure A1: Closed-book TaskBench prompt template. The system instruction is reproduced from the benchmark records; the user request is substituted into the final line. No skill card, gold skill name, family label, or target graph appears in the input.

Illustrative training pair. The following T1 Daily Life training record requires two skills and one dependency link. The graph is pretty-printed for readability; the original target is the same JSON object in compact form.

```
Request: I want to pay my electricity bill and then, attend an online
    meeting about 'Electricity Bill Management'.

Gold graph:
{"task_nodes": [
  {"task": "TaskBench::t1_daily_life::daily_bill_payment",
   "arguments": [{"name": "bill", "value": "electricity"}]},
  {"task": "TaskBench::t1_daily_life::attend_meeting_online",
   "arguments": [{"name": "topic", "value": "Electricity Bill Management
    "}]}
 ],
 "task_links": [
  {"source": "TaskBench::t1_daily_life::daily_bill_payment",
   "target": "TaskBench::t1_daily_life::attend_meeting_online"}
 ]}
```

A.4 METRIC DEFINITIONS

For a parsed prediction $\hat{g} = (\hat{V}, \hat{E}, \hat{\phi})$ and target $g = (V, E, \phi)$, Graph EM is

$$\mathbf{1}[\mathrm{canon}(\hat{g}) = \mathrm{canon}(g)].$$

Tool-choice accuracy is exact match of the ordered tool-name sequence. Argument-value F1 compares multisets of task, argument-name, and argument-value tuples. These diagnostics distinguish choosing the correct procedure from filling its arguments correctly.

For finer analysis, node-set exactness compares the canonical node multiset, argument-value exactness requires exact equality of the value-tuple multiset, and edge exactness compares the directed edge multiset. A prediction can match the nodes and still fail Graph EM because of one incorrect value or link. Edge exactness is not a standalone success measure: single-node targets have no gold edges, so an empty predicted edge set can be edge-exact even when the node is wrong. Route accuracy is

$$|D|^{-1} \sum_{(x,j)\in D} \mathbf{1}[\hat{j}(x) = j],$$

Reproducibility item	Frozen definition in the current candidate
Executor checkpoint	Rank-16 LoRA on $q/k/v/o$, learning rate 4×10^{-4}, six epochs, epoch 6 exported.
Execution-memory storage	Shared 8.2B backbone plus 15.34M LoRA parameters per shard; five shards add 76.68M parameters (0.94% of the backbone).
Sequential Address	Request text only; fixed-dimensional hashing features plus incremental Naive Bayes; update on current stage only.
Retained Address state	Compact class/feature statistics; no old requests or workflow targets are replayed.
Inference	Hard request-level routing to one frozen executor; one language-model generation per request.
Reference selector	Offline TF–IDF + logistic regression uses all historical requests.

Table A2: Protocol choices that define the deployed FSM method and its reference variants.

Algorithm 1 Factorized Skill Memory: replay-free memory update and read

Require: Frozen base model B; stage data $\{\mathcal{D}_t\}_{t=1}^{T}$
$\mathcal{E}_0 \leftarrow \varnothing$; initialize Address Memory A_0
for $t = 1, \ldots, T$ **do**
    Initialize a fresh LoRA Δ_t on the frozen base B
    Train Δ_t using only current-stage supervision $\mathcal{D}_t$
    $E_t \leftarrow B \oplus \Delta_t$; freeze E_t
    $\mathcal{E}_t \leftarrow \mathcal{E}_{t-1} \cup \{E_t\}$
    $A_t \leftarrow \mathrm{UpdateAddress}(A_{t-1}, \mathrm{Req}(\mathcal{D}_t), t)$
    Discard the stage-t requests after updating A_t
end for

function INFER(x)
    $p_j \leftarrow p_{\mathrm{Address}}(j \mid x; A_T)$ for $j = 1, \ldots, T$
    $j^\star \leftarrow \arg\max_j p_j$
    return $\mathrm{Generate}(E_{j^\star}, x)$
end function

which measures executor selection rather than graph correctness. It is distinct from tool-choice accuracy within the generated graph. For continual accuracy matrix $a_{t,k}$, backward transfer is

$$\mathrm{BWT} = \frac{1}{T-1} \sum_{k=1}^{T-1} (a_{T,k} - a_{k,k}),$$

where negative values indicate forgetting relative to performance immediately after task k was learned. Task-macro averages assign equal weight to all five families.

A.5 FSM IMPLEMENTATION AND INFORMATION BUDGETS

Each executor is fixed after its stage finishes. Address is then updated without replaying old requests. Qwen3-8B has 36 transformer layers with hidden size 4096, 32 query heads, and 8 key/value heads (Yang et al., 2025). For rank $r = 16$, the LoRA tensors on $q/k/v/o$ contribute $36r[(4096 + 4096) + 2(4096 + 1024) + (4096 + 4096)] = 15{,}335{,}424$ parameters per shard. Five shards therefore contribute 76.68M parameters, or about 146 MiB when stored as BF16 tensors. Hard routing activates one shard per request, so deployed inference still uses one language-model generation.

A.6 INFORMATION BUDGETS AND COMPARATOR ROLES

The selectors differ in the information they receive. Oracle uses the source family label to measure conditional execution. Self-route sees executor generations and token scores but no family label. FSM receives the current request pool during training, retains only classifier statistics, and runs one selected executor at test time. FSM-Offline revisits the full historical request set. Replay uses old request–workflow pairs during training.

Table A3: Frozen Qwen3-8B with the correct skill address only (percent).

Task	Requests	Full-task EM	Skill selection
T1	864	7.87	83.22
T2	408	0.00	92.65
T3	332	0.00	87.05
T4	299	0.00	93.65
T5	283	0.00	94.35
Macro	–	**1.57**	–

These budgets define the comparisons: Oracle–FSM includes residual addressing and executor error. FSM-Offline–FSM is a historical-request reference rather than an isolated replay ablation, because the two Address variants also use different feature representations and classifiers.

A.7 Evaluation safeguards and artifact provenance

The evaluation packet contains 50 rollout files across the two task orders, with 25 task–executor cells per order. Address uses only request text at inference; family labels are training supervision only, and held-out graphs are not used for model or hyperparameter selection. Each rollout is checked for row count, closed-book status, the 768-token cap, and token-score conventions. Separate result summaries track learned Address, Sequential Address, semantic controls, and template-group controls. Self-route additionally retains per-request candidate indices for the coverage–selection analysis in Appendix B.

B Additional analysis and controls

These controls test the access–execution split, alternative Address representations, and routing as the executor bank grows.

B.1 Correct-skill-address reactivation controls

At the frozen T2 checkpoint used in the main text, closed-book full-task EM is 0% over 408 requests and rises to 69.85% when the correct skill address is supplied. The address contains no request-specific values, intermediate results, workflow steps, demonstration, or target serialization; the model still reconstructs the request-conditioned workflow. Across all requests, evaluation under the correct skill address reaches 70.10% argument-value exactness and 69.85% full-task EM.

The recovery is specific to the correct skill address. An incorrect skill address from the same family yields 6.86% full-task EM, while a semantic description, an unrelated valid address, token-matched random text, and a generic instruction all remain at 0%. These controls rule out generic non-target cues, but the correct skill address still reveals target node identity. An arbitrary-alias control would be needed to separate procedure access from this tool-selection information.

Frozen-base correct-skill-address control. Table A3 evaluates frozen Qwen3-8B on all 2,186 held-out requests with only the correct skill address. No argument-field names, workflow, or answer are supplied.

The base model identifies the intended skill on 87.05–94.35% of T2–T5 requests but obtains 0% full-task EM on all four families; T1 reaches 7.87% EM. Address recognition alone therefore does not explain the T2 reactivation result.

B.2 Coverage–selection decomposition

Table A4 separates self-routing coverage from selection. $O = C$ marks requests for which the matching executor is correct and at least one candidate in the bank is correct; $C - R$ is the loss from confidence-based selection among those candidates.

Table A4: Current self-routing coverage and selection decomposition (percent). The table applies only to self-routing; learned Address did not retain per-request candidate indices.

Order/task	$O = C$	Self-route R	All wrong	$C - R$
O1/T1	45.02	42.01	54.98	3.01
O1/T2	83.58	58.82	16.42	24.75
O1/T3	75.60	56.02	24.40	19.58
O1/T4	80.27	77.26	19.73	3.01
O1/T5	76.33	72.44	23.67	3.89
O2/T1	45.49	42.82	54.51	2.66
O2/T2	83.33	56.62	16.67	26.72
O2/T3	73.80	57.23	26.20	16.57
O2/T4	78.93	77.93	21.07	1.00
O2/T5	75.27	73.14	24.73	2.12

Question	Observed evidence	What it measures
Is execution retained?	Oracle Graph EM is 71.76% across two orders.	Conditional execution under perfect shard access.
Is access unreliable?	Self-route is 61.43% while candidate coverage is 71.76%.	Selection loss is measured only for self-route.
Can request text recover access continually?	FSM reaches 65.95% and 93.38% route accuracy.	Address itself remains sequence-sensitive.
What is available with historical requests?	Offline Address reaches 69.15%.	Reference selector with historical-request access.

Table A5: Measurements supporting the Address–Execution decomposition.

Observed pattern	Primary diagnosis	Next measurement
Low oracle and low routing	Conditional execution is weak.	Revisit stage training, capacity, or data before tuning Address.
High oracle and low self-route	Usable skills exist but are selected unreliably.	Compare request-only routing rules.
Sequential below offline	Historical-request reference is stronger, but the cause is confounded by selector design.	Match representation and classifier under online/offline updates.
Offline close to oracle	A stronger historical-request selector leaves a smaller access gap.	Separate update regime from representation/classifier choice.

Table A6: Error decomposition induced by Oracle, Self-route, and learned Address.

B.3 Evidence for the access–execution split

Table A5 aligns each intervention with the part of the pipeline it measures: name cues and Oracle expose conditional execution, whereas Self-route and Address test recovery of the appropriate executor from the raw request.

B.4 Factorized error decomposition

Oracle, Self-route, and FSM separate executor error from access error. Oracle uses the matching executor, Self-route uses generation confidence, and FSM learns request-to-shard access under the replay-free sequential constraint.

B.5 Representation and calibration controls

Table A7 compares replay-free Address with offline and representation controls. Because the deployed sequential Address uses hashing plus Naive Bayes while the offline variants use different representations and classifiers, these rows are reference comparisons rather than isolated replay ablations. The template-group split removes exact value, filename, and number matches before fitting the offline classifier.

B.6 Training sensitivity check

Table A8 reports a shared T1/T5 screen over learning rate, batch size, and epoch. Both model sizes select 4×10^{-4} with batch eight; the 8B batch-32 condition is worse.

Table A7: Representation and routing controls (percent). Training-request classification and final-test route accuracy are distinct quantities.

Address/control	Route acc.	EM	Evaluation scope
FSM	93.38	65.95	replay-free, two-order mean
Lexical TF–IDF (offline)	97.03	69.15	historical requests retained
E5-base-v2	94.22	66.61	final test, two-order mean
Qwen3-4B mean-pool	90.92	64.20	final test, two-order mean
Template-group split	97.09	–	997 held-out train rows
Ordinary five-fold split	97.26	–	training requests

Table A8: Controlled T1/T5 validation screen (percent). The screen used 800/200 training/validation rows per task and epochs 1–3.

Task	4B best	8B best	8B batch-32	Selected recipe
T1 EM	47.5	44.5	32.5	4×10^{-4}, batch 8
T5 EM	77.5	76.5	71.0	4×10^{-4}, batch 8

Order prefix	Oracle	FSM	Route accuracy	Available tasks
O1/$n = 1$	45.02	45.02	100.00	T1
O1/$n = 2$	64.30	63.77	98.88	T1–T2
O1/$n = 3$	68.07	63.59	94.30	T1–T3
O1/$n = 4$	71.12	66.70	94.67	T1–T4
O1/$n = 5$	72.16	66.33	93.38	T1–T5
O2/$n = 1$	73.80	73.80	100.00	T3
O2/$n = 2$	59.64	59.41	99.39	T3–T1
O2/$n = 3$	64.85	62.10	96.64	T3–T1–T5
O2/$n = 4$	69.47	63.19	92.73	T3–T1–T5–T2
O2/$n = 5$	71.36	65.58	93.38	T3–T1–T5–T2–T4

Table A9: Sequential Address prefix results (percent). Oracle is the conditional reference at each prefix; route accuracy is the replay-free Address route audit.

The controls separate routing from execution: frozen representations remain below the lexical Address, and route accuracy exceeds end-to-end EM when the selected executor still makes execution errors.

Model-scale and representation robustness. The 8B oracle average is 71.76%, close to the historical 4B value of 71.94%; the cross-scale comparison is descriptive because the training recipes differ. In the controlled T1/T5 screen, both sizes select the same learning rate and batch size, with 4B/8B validation EM of 47.5/44.5 on T1 and 77.5/76.5 on T5. E5-base-v2 and Qwen3-4B mean-pool representations remain below the sparse lexical Address, and the template-group split reaches 97.09% Address accuracy versus 97.26% under ordinary five-fold splitting.

B.7 Sequential Address prefixes

Table A9 tracks Sequential Address after every prefix. Final route accuracy is 93.38% in both orders; O1 drops most at T2–T3, whereas O2 drops more after T3–T5. The executor bank is fixed throughout.

B.8 Self-route prefix trajectories

Table A10 tracks confidence-based routing as the frozen bank grows. Each row evaluates the tasks available at that prefix; Table A9 gives the corresponding learned-Address trajectory.

B.9 Historical prefix and prompt controls

The earlier Qwen3-4B one-order audit produced routed task rows of 44.68 at stage 1; 44.21/83.09 at stage 2; 43.87/72.79/62.35 at stage 3; 43.87/68.14/60.84/78.60 at stage 4; and 43.87/61.52/57.23/77.93/71.73 at stage 5. In the same protocol, five prompt-order variants reached

Table A10: Current self-confidence Graph EM by order prefix (percent). Dashes mark task families that had not entered the stream.

Order	Prefix	T1	T2	T3	T4	T5
O1	$n = 1$	45.02	–	–	–	–
	$n = 2$	42.82	82.60	–	–	–
	$n = 3$	42.36	72.06	61.45	–	–
	$n = 4$	42.01	64.71	60.54	77.93	–
	$n = 5$	42.01	58.82	56.02	77.26	72.44
O2	$n = 1$	–	–	73.80	–	–
	$n = 2$	44.33	–	72.59	–	–
	$n = 3$	43.52	–	68.07	–	74.20
	$n = 4$	43.06	64.22	58.43	–	73.50
	$n = 5$	42.82	56.62	57.23	77.93	73.14

final Graph EM of 60.28, 61.51, 59.55, 61.22, and 59.67, while request-only self-routing reached 62.45. None of the prompt variants exceeded request-only self-routing.

Backbone	Request only	Gold names/fields	Full catalog
Qwen3-4B	0.00	7.01	18.31
Qwen3-8B	0.00	56.00	34.09

Table A11: No-adapter skill-information controls (Graph EM, percent). Oracle membership and the complete catalog alter the inference information boundary and are not deployment baselines.

Branch	Evidence level	Outcome and disposition
Current-only SFT, GRPO/HC-GRPO, replay, LwF, EWC, OPSD, and adapter variants	Complete five-task endpoints	Most direct updates overwrite old tasks; representative endpoints are reported in the main results and the analyses below.
SKILL0 TaskBench curriculum	Complete lower triangles	Current-task acquisition succeeds, but request-only old-task Graph EM collapses; exact names temporarily reactivate intermediate checkpoints.
One-epoch SFT warm start before GRPO/SKILL0	Completed audited cells	Warm start enables current-task learning but does not prevent cross-task overwrite.
Pure on-policy SKILL0 pilot	Completed T1 pilot	Protocol assistance creates a weak learning signal, but strict closed-book Graph EM remains zero.
No-LoRA skill-information controls	Complete 30-cell grid	Canonical skill metadata exposes latent capability but changes the inference information boundary.
T2→T3 learning-rate sweep	Complete 8B checkpoint grid	Lower learning rates delay the switch but do not yield a useful retention–plasticity operating point.
Frozen executors plus sequential Address	Complete two-order result	Promoted to FSM: execution is frozen and Address is updated without old-request replay.

Table A12: Training branches and the evidence they provide for the final design.

C Complete results and ablations

C.1 No-adapter information controls

Table A11 evaluates the base models before adapter training under three inputs: request only, gold skill names with argument-field names, and the complete 100-skill catalog. None supplies argument values, dependencies, workflow order, demonstrations, or the target graph.

Canonical metadata exposes substantial latent capability, especially at 8B. Because that metadata is supplied at inference, these rows measure available information rather than persistent learning.

D Training dynamics and additional analysis

This appendix records the training dynamics and capacity analyses behind the factorized design.

D.1 Memory growth and modularity

Freezing executors prevents later stages from overwriting their parameters and makes conditional retention directly measurable. The bank therefore grows with the number of stages, as in other progressive and expert-bank methods (Rusu et al., 2016; Aljundi et al., 2017). Address is updated without old requests or graphs and retains compact class statistics; deployed FSM selects one executor before a single language-model generation.

D.2 Training branches that motivate factorization

Table A12 summarizes the completed branches that led to the final factorization.

D.3 Sequential SFT, GRPO, and SKILL0 acquisition

All three direct sequential recipes learn the current family and lose earlier families. Table A13 reports acquisition on the diagonal and the final closed-book endpoint. SFT and Dense GRPO use Qwen3-8B; the SKILL0 trajectory uses Qwen3-4B.

Recipe	S1/T1	S2/T2	S3/T3	S4/T4	S5/T5	Old tasks after S5	Prompt
Current-only SFT (8B)	45.02	83.58	75.90	80.60	76.33	all 0	request only
GRPO (8B)	40.86	79.90	73.19	79.93	71.73	all 0	request only
SKILL0	46.06	83.33	77.11	82.27	79.15	all 0	request only
SKILL0	46.88	83.09	76.51	80.60	79.86	all 0	full skill card

Table A13: Closed-book or supplied-skill Graph EM (percent) on the acquired task at each stage. The final old-task statement is evaluated after S5. The 4B SKILL0 rows are not capacity-matched to the 8B baselines.

Method	First request-only zero	Skill address at step 126	Request-only at 126
SFT-8B	8	54.66	0.00
GRPO-8B	6	37.01	0.00
SKILL0-4B	18	82.84	0.00

Table A14: Extended T2 retention trajectories (Graph EM, percent). Supplying the correct skill address continues to recover behavior long after autonomous closed-book access reaches zero.

Supplying the full skill card creates only a temporary retention window: T2 reaches 58.33% at S3 and 0% at S4, with all old tasks at 0% after S5. The correct-skill-address audit follows the same pattern, reaching 52.94% at S3 and 0% at S4.

D.4 Full forgetting trajectories and correct-skill-address reactivation

The extended T2 audit evaluates every even optimizer step from 0 through 126 under request-only and correct-skill-address conditions. SFT and GRPO use Qwen3-8B checkpoints; the SKILL0 trajectory uses Qwen3-4B. The curves compare the timing of access and execution failure.

Figure 2 shows the early transition; Table A14 extends it to step 126.

D.5 Warm starts and pure on-policy acquisition

One SFT epoch produces a strong current-task policy. Warm-started GRPO moves T1 from 39.93% after SFT to 41.20% after one GRPO epoch. In the warm-started SKILL0 five-task sequence, final request-only scores are 0/0/0/0/72.44 versus 0/0/0/0/69.61 for one-epoch current-only SFT: the final task improves slightly while old-task retention remains zero. In the zero-SFT on-policy T1 pilot, direct skill-free training yields constant within-group rewards. A staged protocol raises recall Graph EM from 4.17% to 6.94% after 40 steps and 7.18% after 100 steps, with 22.34% selection exactness, 52.39% node F1, and 28.21% argument-value F1 at step 100. Strict closed-book Graph EM remains 0.

D.6 Learning-rate sweep from the post-T2 checkpoint

We test whether a lower T3 learning rate can preserve T2 under current-only SFT. Every arm starts from the same post-T2 Qwen3-8B checkpoint (T2 Graph EM 77.45%), trains on 1,000 T3 examples for one epoch (125 optimizer steps, global batch eight), uses no old-task replay, and follows the same held-out request-only evaluation protocol.

The coarse grid shows an abrupt transition between retaining T2 and acquiring T3. We evaluate checkpoints every five steps for the two learning rates nearest that transition.

At 10^{-6}, performance changes from T2=79.41/T3=0 at step 30 to T2=0/T3=18.98 five updates later. At 5×10^{-7}, step 60 retains T2=67.89 while T3 Graph EM is only 2.11%. Lower learning rates delay the switch but do not produce a checkpoint with both substantial T2 retention and a learned T3 policy.

Learning rate	Best observed step	T2 Graph EM	T3 Graph EM	T2–T3 mean	Outcome
2×10^{-5}	100	0.00	74.40	37.20	T3 learned; T2 erased
1×10^{-5}	125	0.00	75.60	37.80	T3 learned; T2 erased
5×10^{-6}	50	0.00	76.81	38.40	best T3; T2 erased
2×10^{-6}	125	0.00	73.19	36.60	T3 learned; T2 erased
1×10^{-6}	25	79.17	0.00	39.58	T2 retained before switch
5×10^{-7}	50	78.43	0.00	39.22	T2 retained before switch
2×10^{-7}	75	78.19	0.00	39.09	T3 never acquired
1×10^{-7}	50	77.94	0.00	38.97	T3 never acquired

Table A15: Best two-task mean observed in each learning-rate arm (percent). For the four higher rates, every checkpoint with nonzero T3 has zero T2. For the two lowest rates, T3 remains zero for the full epoch.

LR	Step	T2 Graph EM	T3 Graph EM	Mean	T2 node F1	T3 node F1
1×10^{-6}	25	79.17	0.00	39.58	94.45	0.00
	30	79.41	0.00	39.71	95.45	0.00
	35	0.00	18.98	9.49	0.00	28.22
	40	0.00	39.76	19.88	0.00	56.35
	45	0.00	51.20	25.60	0.00	70.15
	50	0.00	57.23	28.61	0.00	77.34
5×10^{-7}	50	78.43	0.00	39.22	94.35	0.00
	55	78.19	0.00	39.09	94.56	0.00
	60	67.89	2.11	35.00	81.23	2.72
	65	0.25	10.24	5.24	0.25	16.94
	70	0.00	12.65	6.33	0.00	19.86
	75	0.00	15.36	7.68	0.00	24.24

Table A16: Five-step zoom around the learning-rate transition (percent). The two overlapping checkpoints occur before T3 is meaningfully acquired.