Combating Catastrophic Forgetting in Continual Domain Adaptation via Knowledge Recasting

Zijie Liu
UNC at Chapel Hill
jesseliu@cs.unc.edu

Zhen Tan
Stevens Institute of Technology
ztan12@stevens.edu

Charles Fleming
Cisco Research
chflemin@cisco.com

Tianlong Chen
UNC at Chapel Hill
tianlong@cs.unc.edu

Abstract

Adapting language models to new domains during post-training risks degrading previously acquired capabilities, a phenomenon commonly known as *catastrophic forgetting*. Existing mitigation methods largely treat forgetting as a consequence of excessive model drift. They therefore constrain updates by preserving old outputs, replaying old data, or regularizing changes to parameters and representations so that the updated model remains close to its previous state. We take a *different view*: Rather than pulling the updated model back toward its old state, we translate the new domain into the model's pre-existing knowledge space, making unfamiliar knowledge learnable through familiar procedures. Therefore, we propose **Skill Schema Transport (SST)**, a post-training framework that transports new knowledge into the model's existing skill space. SST abstracts each example type into a skill schema, preserving its essential operations, dependencies, and verification steps while removing domain-specific surface forms. By recasting new examples as instances of reusable procedures, SST turns knowledge update from trajectory memorization into procedural reuse. In continual domain adaptation settings, SST consistently reduces forgetting while improving generalization to related tasks. Across two 3B instruction-tuned backbones, SST improves average final performance on streamed domains by 2.53-5.75 points over sequential post-training. It also lowers max-drop forgetting in all evaluated settings, with the largest reduction from 42.41 to 7.10 points on the Qwen-3B MATH stream, while substantially recovering the OOD degradation induced by sequential updates. Code is available at `https://github.com/jesseliu0913/Skill_Forgetting`.

1 Introduction

Post-training improves language models (LMs) through supervised fine-tuning [22], reinforcement learning from feedback [37, 4], and domain-adaptive continued training [19]. However, sequential post-training can also degrade capabilities acquired before the update, a phenomenon commonly known as *catastrophic forgetting* [35, 27, 34, 46, 54]. This risk becomes especially important under domain shift, where the update distribution introduces concepts, terminology, or reasoning contexts that differ from those emphasized by the pre-update model [46, 24, 62].

To clarify what existing mitigation strategies are designed to control, we view post-training under domain shift through the lens of model drift. As shown in Figure 2, knowledge updates can change a model in at least two distinct ways. *Surface drift* refers to changes in the model's observable behavior, such as its output distribution or final answers, whereas *representation drift* refers to changes in the

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

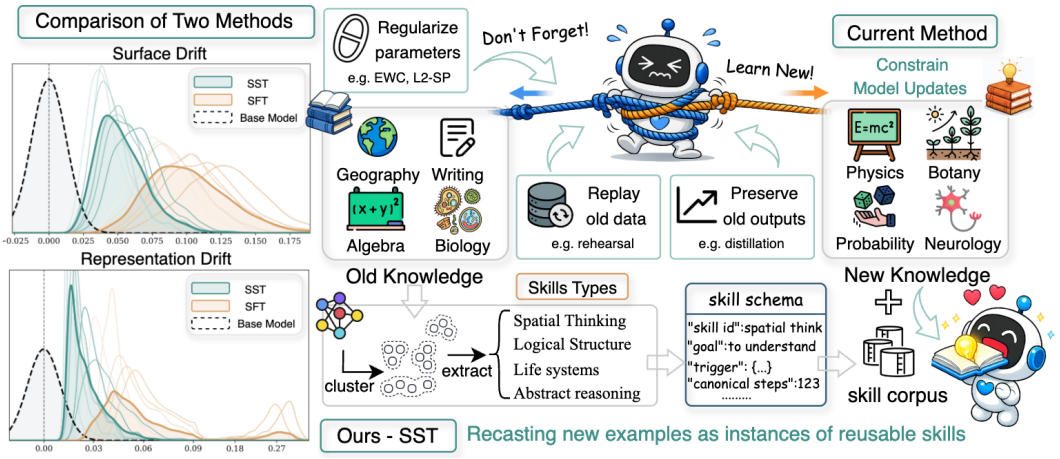


Figure 1: **Overview of Skill Schema Transport (SST).** SST recasts new-domain examples as instances of reusable skill schemas rather than anchoring old data, output distributions, or parameter values. The left panels illustrate the drift diagnostics used in our analysis, and the bottom pipeline shows how procedural skills are extracted, abstracted, stored, and used for schema-conditioned updates.

internal states that support those behaviors. These two forms of drift need not agree: the mapping from representations to outputs is many-to-one, so a model may reorganize its internal states while producing similar outputs, or substantially shift its output distribution while leaving representations relatively stable.

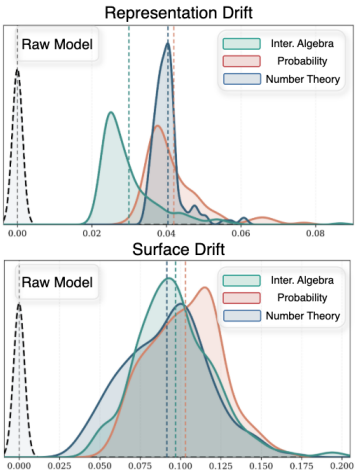


Figure 2: **Surface drift and representation drift need not align.** Domain-shifted updates induce different distributions of output-level and hidden-state change, motivating diagnostics that track both observable behavior and internal representations.

From this perspective, many forgetting-mitigation methods can be understood as different ways of anchoring the updated model to its pre-update behavior or state. Replay-based methods mix old examples with new training data so that the model continues to reproduce earlier task behavior during the update [43, 32, 24]. Distillation-based methods use the original model as a teacher and penalize deviations from its outputs [31]. Both families primarily constrain surface drift by preserving observable behavior on old examples. In contrast, regularization-based methods penalize changes to important parameters or restrict updates to protected subspaces [27, 60, 2, 52, 33], thereby limiting drift in the model's internal state or parameterization.

Because these approaches primarily anchor one aspect of the update, anchoring surface behavior alone need not control representation drift, and anchoring parameter or representation movement alone need not preserve task behavior. This points to a limitation of treating forgetting mitigation only as a problem of preventing change: the key question is not merely how much the model changes, but which aspects of the update are being constrained and whether those constraints help integrate the new domain.

Yet anchoring the model is not the same as integrating new knowledge. Under out-of-domain updates, useful adaptation requires change: the model may need to acquire, emphasize, or recombine concepts and reasoning contexts that were weakly represented in the pre-update distribution. *The key issue is therefore not how to minimize change, but how to organize it.* The organizing principle we propose is **procedural reuse**. If a new problem can be solved by a procedure the model already uses, then the desirable update is not to memorize a new trajectory or to pull the model back toward its old outputs. Instead, the update should connect the unfamiliar instance to a familiar procedure, expressing new knowledge in procedural coordinates the model can reuse. Inspired by this view, we ask the following research question:

RQ: *Can new-domain knowledge be expressed in the procedural language of the model's existing skills, so that unfamiliar problems become learnable through familiar procedures?*

To probe whether this premise is plausible, we analyze the error structure of a base model across MATH categories. The same failed examples are embedded and projected in Figure 3; coloring them by procedural error type reveals clusters that cut across the original domain labels. This diagnostic does not by itself prove that all domains share a complete skill library, but it suggests that observed failures are not purely domain-specific and motivates testing whether schema-level reuse can improve continual adaptation.

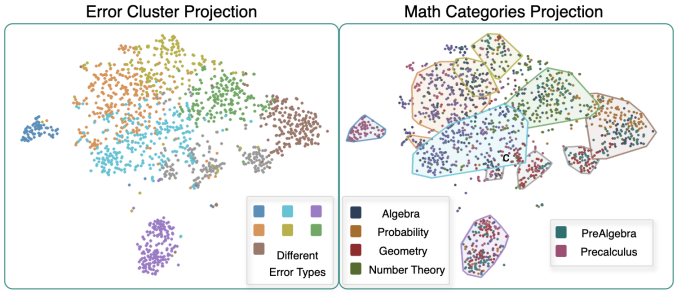


Figure 3: **Errors cluster by procedure rather than by domain label.** The same failed examples are projected and colored by procedural error type on the left and by MATH category on the right. Error clusters cut across category boundaries, motivating cross-domain skill schema reuse.

Based on this observation, we propose *Skill Schema Transport* (SST), a schema-conditioned post-training framework that transports new-domain examples into a reusable procedural schema space. A skill schema serves as the medium of this translation: it retains the operations, dependencies, and checks needed to solve a task, while abstracting away domain-specific wording, formats, and trajectories. SST uses these schemas to associate unfamiliar examples with familiar procedures, allowing a new-domain example to be interpreted as a transported instance of a reusable skill rather than as an isolated new trajectory, as illustrated in Figure 1. We evaluate SST in streaming knowledge-update settings, where new-domain knowledge is introduced sequentially and the model must retain capabilities learned from earlier domains. Across two 3B instruction-tuned backbones, a four-domain MATH stream, and a three-domain agent stream, SST improves average final performance on streamed domains by 2.53-5.75 points over the corresponding sequential post-training baseline. It also reduces max-drop forgetting in all evaluated settings and maintains or substantially recovers OOD benchmark performance relative to sequential updates. To summarize, our main contributions are:

- We reframe forgetting mitigation under domain shift as a problem of knowledge organization rather than only model constraint, shifting the burden of adaptation from limiting model movement to structuring how new examples are represented during training.
- We provide drift and error-structure diagnostics showing that surface and representation drift need not align, and that procedural failure patterns can cut across domain boundaries, motivating procedural reuse as an alternative to purely drift-based anchoring.
- We propose *Skill Schema Transport* (SST), a schema-conditioned post-training framework that extracts domain-agnostic skill schemas and transports new-domain examples into reusable procedural coordinates.
- Across two streaming settings comprising seven update domains and two 3B instruction-tuned backbones, SST improves streamed-domain performance by 2.53-5.75 points over sequential post-training, reduces max-drop forgetting in all evaluated settings, and maintains or substantially recovers OOD benchmark performance relative to sequential updates.

2 Related Work

Continual Learning for Sequential LLM Training. Continual learning studies how models acquire new capabilities without overwriting old ones, with catastrophic forgetting as a central failure mode [35, 42, 15, 16]. Existing methods mitigate forgetting through parameter consolidation, functional regularization, replay, and gradient constraints, including EWC, SI, MAS, and related approaches [27, 60, 2, 8, 44, 31, 41, 32, 9, 7, 38, 12]. Analogous ideas appear in sequential RL through replay, distillation, and policy regularization, while recent LLM post-training studies show that continual pretraining, instruction tuning, SFT, and RLHF can all induce forgetting under domain shift

[43, 26, 37, 4, 34, 54, 46, 62]. Parameter-efficient methods such as LoRA, O-LoRA, and CLoRA reduce interference by restricting adaptation to low-rank, orthogonal, or regularized subspaces, but they still mainly act as drift-control mechanisms [23, 52, 33]. Our work is complementary: instead of only constraining surface or representation drift, we ask whether new-domain knowledge can be transported into procedures already learned by the model, and analyze this using representation-similarity tools such as SVCCA, PWCCA, and CKA [40, 36, 28].

Skills, Temporal Abstraction, and Skill Discovery. The idea that complex behavior can be decomposed into reusable subroutines is central to hierarchical reinforcement learning, where Options, MAXQ, feudal RL, and Option-Critic define skills as temporally extended actions or subtasks that support behavioral reuse [49, 13, 11, 5, 3, 39]. A related line studies unsupervised skill discovery, where methods such as VIC, DIAYN, DADS, and OPAL learn diverse, predictable, or recurring primitives without task-specific supervision [18, 14, 45, 1]. We build on this abstraction view, but do not treat skills as options, policies, or transfer modules. Instead, we define skills as language-level schemas that preserve operations, dependencies, and checks while abstracting away domain-specific wording, formats, and trajectories, allowing us to test whether schema-based supervision induces less unnecessary representation drift during post-training [40, 28].

Skill-Augmented and Continually Improving Agents. Recent LLM-agent systems externalize reusable experience as memories, reflections, trajectories, executable programs, or skill libraries. ReAct interleaves reasoning with environment actions, while Reflexion and ExpeL improve agents through verbal feedback, episodic memory, or distilled experience without updating model weights [59, 47, 61]. Voyager extends this idea to open-ended lifelong learning with an executable skill library for embodied control [51]. More recent work makes skill learning explicit through recursive skill libraries, persistent skill memories, multimodal experience-skill stores, and skill-oriented benchmarks [55, 30, 25, 64]. These methods typically use skills as external memories, prompts, tools, or inference-time resources, whereas our work studies schema-conditioned sequential post-training with retrieval-free main evaluation: SST uses skill schemas during training to encourage new-domain knowledge to be internalized through reusable procedures.

3 Preliminary

We formalize the continual domain adaptation setting used throughout the paper. We first define the update and retention distributions, then define forgetting and drift diagnostics, and finally introduce the schema objects used by SST.

Definition 1 (Continual Learning on Shifting Domains)**.** *Let $\mathcal{M}_{\theta_0}$ be a pre-trained language model and $\mathcal{D}_1, \ldots, \mathcal{D}_T$ a sequence of update distributions, where each $\mathcal{D}_t$ is supported on a domain Ω_t (e.g., a subject area such as algebra, medicine, or law) that may differ from the domains seen during pre-training. At step t, the model is updated from θ_{t-1} to θ_t using examples drawn from $\mathcal{D}_t$, while retaining performance on a set of retention distributions $\mathfrak{R}_t \subseteq \{\mathcal{Q}_i : 1 \leq i < t\}$, where each $\mathcal{Q}_i$ is supported on the previously encountered domain Ω_i.*

The setting specifies where new learning occurs and where retention is measured. We next define forgetting as the maximum accuracy drop over the observed stream.

Definition 2 (Forgetting Gap)**.** *For a retention distribution $\mathcal{Q} \in \mathfrak{R}_t$, let $A_t(\mathcal{Q})$ denote the task accuracy on $\mathcal{Q}$ after update step t. If $\mathcal{Q}$ corresponds to a domain introduced at step $t_{\mathcal{Q}}$, we define its forgetting gap after the full stream as*

$$\mathcal{F}_{\mathcal{Q}} = \max_{t_{\mathcal{Q}} \leq t \leq T} A_t(\mathcal{Q}) - A_T(\mathcal{Q}). \tag{1}$$

We report average forgetting by averaging $\mathcal{F}_{\mathcal{Q}}$ over prior retention distributions, excluding the final streamed domain because it is not followed by a later update and therefore cannot be overwritten within the observed stream.

The forgetting gap measures the outcome of the update stream. To diagnose the updates themselves, we also track how much the model changes at the output and representation levels.

Definition 3 (Drift Diagnostics)**.** *For an input x, let $p_\theta(\cdot \mid x)$ denote the model's output distribution over the vocabulary and $h_\theta(x) \in \mathbb{R}^d$ a hidden representation extracted from a fixed intermediate layer. Let $\mathcal{Q}_X$ denote the marginal distribution over inputs x. The cumulative drift from θ_0 and the*

Input: problem q, transported skill schema $\widetilde{z}$, target response y.

[Problem q] Solve for u: $\frac{u}{2} + 3 = 7$.

[Transported skill schema $\widetilde{z}$] `Algebraic Substitution and Equation Solving`. *Canonical steps:* rewrite the relation; isolate the variable; solve; verify the result.

[Augmented training prompt]

Problem: Solve for u: $\frac{u}{2} + 3 = 7$. Relevant skill: Algebraic Substitution and Equation Solving. Use the following procedure: rewrite the relation, isolate the variable, solve, and verify the result.

[Target response y]

Subtract 3 from both sides to get $\frac{u}{2} = 4$. Multiply both sides by 2 to get $u = 8$.

Check: $\frac{8}{2} + 3 = 4 + 3 = 7$, so the solution is correct.

Figure 4: A schema-conditioned SFT training example in SST. The setup is identical to standard SFT except that the input prompt is augmented with a transported skill schema $\widetilde{z}$; the supervision target remains the gold response y only.

incremental drift from $\bar{\theta} = \theta_{t-1}$ are defined, respectively, as

$$G^0_{\text{surface}}(\theta; \mathcal{Q}) = \mathbb{E}_{x\sim\mathcal{Q}_X}\left[\text{JS}(p_\theta(\cdot|x) \,\|\, p_{\theta_0}(\cdot|x))\right], \quad G_{\text{surface},\bar{\theta}}(\theta; \mathcal{Q}) = \mathbb{E}_{x\sim\mathcal{Q}_X}\left[\text{JS}(p_\theta(\cdot|x) \,\|\, p_{\bar{\theta}}(\cdot|x))\right], \tag{2}$$

$$G^0_{\text{repr}}(\theta; \mathcal{Q}) = \mathbb{E}_{x\sim\mathcal{Q}_X}\left[\text{Dist}(h_\theta(x),\, h_{\theta_0}(x))\right], \qquad G_{\text{repr},\bar{\theta}}(\theta; \mathcal{Q}) = \mathbb{E}_{x\sim\mathcal{Q}_X}\left[\text{Dist}(h_\theta(x),\, h_{\bar{\theta}}(x))\right], \tag{3}$$

These drift quantities measure changes in the model, but they do not specify whether two examples use the same solution procedure. SST therefore uses a schema-level representation of procedural structure. We represent procedural structure via skill schemas. Let $\tau \in \mathcal{T}$ denote a reasoning trace for solving x, and let $\mathcal{B}_t$ denote the schema memory accumulated through step t.

Definition 4 (Skill Schema)**.** *A skill schema is an abstraction $z = \Gamma_\theta(\tau) \in \mathcal{Z}$, where $\Gamma_\theta : \mathcal{T} \to \mathcal{Z}$ maps a reasoning trace τ to a structured, domain-agnostic representation of its procedural steps.*

Table 1 shows one instantiated schema and its fields. It specifies what information is stored in a schema, including its goal, applicability conditions, canonical steps, failure modes, and compositional links. Figure 4 illustrates how a concrete problem is paired with a transported schema to form a schema-conditioned SFT example.

Definition 5 (Schema Transportability)**.** *A new example with schema $z = \Gamma_{\theta_{t-1}}(\tau)$ is* ϵ-schema-transportable *with respect to a nonempty schema memory $\mathcal{B}$ if*

$$\text{Dist}(z,\, \text{NN}_{\mathcal{B}}(z)) \le \epsilon, \qquad \text{NN}_{\mathcal{B}}(z) \in \arg\min_{z'\in\mathcal{B}} \text{Dist}(z,\, z'), \tag{4}$$

where $\text{NN}_{\mathcal{B}}(z)$ denotes the nearest neighbor of z in the schema memory $\mathcal{B}$ under Dist*, with ties broken deterministically. If $\mathcal{B} = \varnothing$, the example is treated as non-transportable.*

Transportability determines how new examples interact with the schema memory $\mathcal{B}_t$: transportable examples are attached to an existing procedure, while non-transportable examples introduce a new one. The maintenance operations on $\mathcal{B}_t$ are detailed in Section 4.

4 Skill Schema Transport

We propose *Skill Schema Transport* (SST), a post-training framework that connects new-domain knowledge to procedures learned from earlier domains, shown in Algorithm 1. Rather than constraining the model to reproduce previous outputs, SST uses skill schemas as procedural descriptors, allowing a new-domain example to be interpreted through a reusable skill whenever its schema matches one stored in memory.

Schema Extraction. For each training example $(x, y, \tau) \sim \mathcal{D}_t$, the reasoning trace τ is the ground-truth solution provided by the dataset. The skill schema $z = \Gamma_{\theta_{t-1}}(\tau)$ is extracted using the model prior to the current update step via a structured prompt (detailed in

Table 2: **Main streaming results.** All numbers are percentages after the full stream. *Stream Avg.*: average over streamed domains. *Forget.*: average max-drop over prior streamed domains. *OOD Avg.*: average over MMLU, GSM8K, PIQA, CSQA, and SciQ (flex score for GSM8K). Sequential baseline is Seq-SFT for MATH and Seq-RL for agent tasks. SST uses the same update objective as its sequential counterpart, conditioned on skill schemas. Forgetting is not applicable to base models.

MATH stream: Algebra → Counting & Probability → Geometry → Intermediate Algebra

		Streamed domains						OOD generalization					
Model	Method	Alg.	Count.	Geom.	Inter.	Stream Avg. ↑	Forget. ↓	MMLU	GSM8K	PIQA	CSQA	SciQ	OOD Avg. ↑
Qwen-3B	Base	44.99	28.48	22.34	9.19	26.25	–	65.45	63.00	77.69	75.54	92.50	74.84
Qwen-3B	Seq-SFT	26.62	21.31	18.16	13.73	19.96	42.41	57.18	7.58	74.43	61.59	93.80	58.92
Qwen-3B	SST	40.27	24.68	20.46	15.06	**25.12**	**7.10**	65.28	39.20	77.58	80.56	97.70	**72.06**
Llama-3B	Base	8.17	5.70	11.27	9.63	8.69	–	62.12	67.48	73.90	73.96	94.20	74.33
Llama-3B	Seq-SFT	30.03	20.16	18.66	12.33	20.30	4.39	60.43	46.40	73.72	71.01	95.10	69.33
Llama-3B	SST	31.76	21.73	20.88	16.94	**22.83**	**4.09**	60.58	61.84	74.32	70.60	94.30	**72.33**

Agent stream: AlfWorld → WebShop → Search

		Streamed domains					OOD generalization					
Model	Method	AlfWorld	WebShop	Search QA	Stream Avg. ↑	Forget. ↓	MMLU	GSM8K	PIQA	CSQA	SciQ	OOD Avg. ↑
Qwen-3B	Base	19.79	42.75	53.72	38.75	–	65.45	63.00	77.69	75.54	92.50	74.84
Qwen-3B	Seq-RL	66.97	91.33	60.81	73.04	1.47	65.42	63.00	77.69	78.54	94.70	**75.87**
Qwen-3B	SST	77.94	94.12	64.32	**78.79**	**0.77**	65.47	63.61	77.15	78.13	94.50	75.77
Llama-3B	Base	20.99	18.43	49.59	29.67	–	62.12	67.48	73.90	73.96	94.20	74.33
Llama-3B	Seq-RL	71.93	91.76	59.81	74.50	1.47	61.84	67.32	73.23	73.71	94.30	74.08
Llama-3B	SST	77.94	92.55	62.81	**77.77**	**1.27**	61.95	68.54	73.50	72.56	94.30	**74.17**

Appendix E). The extractor $\Gamma_{\theta_{t-1}}$ takes τ as input and produces a domain-agnostic schema $z \in \mathcal{Z}$ specifying the procedural structure of the solution, including operations, dependencies, and validation checks, while abstracting away domain-specific terminology and surface details.

Table 1: Example of a structured skill schema used.

Field	Description
Skill Name	Algebraic Substitution and Equation Solving
Goal	Solve algebra problems via rewriting, substitution, and branch verification.
Applicability	Equations, expressions, radicals, and symbolic substitutions.
Keywords	equation, substitute, solve, factor, expression
Structural Cues	functional relations; inequalities or intervals
Output Format	integer, fraction, radical, interval
Canonical Steps	(1) Rewrite the relation; (2) substitute or factor strategically; (3) solve candidate values; (4) check constraints.
Failure Modes	sign errors; dropped branches; unchecked extraneous solutions
Compositional Link	exact form and answer realization

Schema Memory Initialization and Skill Retrieval. The schema memory $\mathcal{B}_0$ is initialized from a fixed set of predefined skill families and seed skills before any examples from $\mathcal{D}_1$ are processed. During update step t, SST maintains a working memory $\bar{\mathcal{B}}$ initialized as $\mathcal{B}_{t-1}$. For each example in $\mathcal{D}_t$, the system extracts a candidate schema using $\Gamma_{\theta_{t-1}}$, retrieves the nearest skill from $\bar{\mathcal{B}}$, constructs the transported schema $\widetilde{z}$, and immediately applies a memory-update operation to $\bar{\mathcal{B}}$. After all examples from $\mathcal{D}_t$ are processed, the working memory is committed as $\mathcal{B}_t$. New skills are written into memory only if the same procedural signature receives at least $n_{\min} = 2$ supporting examples, preventing spurious entries from single occurrences. Schema transportability (Definition 5) is defined conceptually over the schema space $\mathcal{Z}$. In implementation, we operationalize this with a problem-to-skill retrieval proxy: the input problem x and each skill entry in the current working memory $\bar{\mathcal{B}}$ are encoded using `bge-small-en-v1.5` [56] and ranked by cosine similarity. Each skill is represented as a canonicalized natural-language block comprising five fields: Skill ID, Family, Goal, When to use, Steps, and Common failure modes. The top-k retrieved skill blocks ($k = 2$ in our experiments) are then used to condition schema extraction and the construction of the transported schema $\widetilde{z}$.

Schema-Conditioned Post-Training. At update step t, standard SFT optimizes

$$\mathcal{L}^t_{\mathrm{SFT}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}\left[-\log p_\theta(y \mid x)\right]. \tag{5}$$

When $\mathcal{D}_t$ is out-of-domain, this objective may cause the model to acquire isolated domain-specific procedures rather than expressing new knowledge in terms of procedures already present in its repertoire. SST processes examples sequentially with a working memory $\bar{\mathcal{B}}_{t,0} = \mathcal{B}_{t-1}$. For the j-th

example (x_j, y_j, τ_j), let $z_j = \Gamma_{\theta_{t-1}}(\tau_j)$ and assign

$$\widetilde{z}_j = \begin{cases} \mathrm{NN}_{\bar{\mathcal{B}}_{t,j-1}}(z_j), & \text{if } \bar{\mathcal{B}}_{t,j-1} \neq \emptyset \text{ and } \mathrm{Dist}\Big(z_j, \mathrm{NN}_{\bar{\mathcal{B}}_{t,j-1}}(z_j)\Big) \leq \epsilon, \\ z_j, & \text{otherwise.} \end{cases} \tag{6}$$

Here $\bar{\mathcal{B}}_{t,j-1}$ is the current working memory before processing example j, and $\mathrm{NN}_{\bar{\mathcal{B}}_{t,j-1}}(z_j)$ denotes the nearest neighbor of z_j in that memory under Dist. This yields the schema-conditioned dataset

$$\mathcal{D}_t^{\mathrm{SST}} = \{(x_j, \widetilde{z}_j, y_j)\}_{j=1}^{|\mathcal{D}_t|}$$

and objective

$$\mathcal{L}_{\mathrm{SST}}^t(\theta) = \mathbb{E}_{(x,\widetilde{z},y)\sim\mathcal{D}_t^{\mathrm{SST}}}\left[-\log p_\theta(y \mid x, \widetilde{z})\right]. \tag{7}$$

Here $\widetilde{z}$ serves solely as input context and is not a supervision target: the model is trained to generate y conditioned on both the problem x and the retrieved procedural context $\widetilde{z}$, but is not explicitly trained to predict $\widetilde{z}$ itself. Thus x provides instance-specific content, y provides the target response, and $\widetilde{z}$ provides the procedural coordinates through which the new example is expressed.

Algorithm 1 Skill Schema Transport (SST)

Require: Pre-trained model $\mathcal{M}_{\theta_0}$
Require: Update stream $\mathcal{D}_1, \ldots, \mathcal{D}_T$
Require: Distance Dist; thresholds ϵ, δ_{create}, δ_{merge}, δ_{drop}, $n_{\min}$, $k_{\min}$
Ensure: Updated model $\mathcal{M}_{\theta_T}$, schema memory $\mathcal{B}_T$

Initialize $\mathcal{B}_0$ from seed skill families
for $t = 1, \ldots, T$ **do**
  $\bar{\mathcal{B}} \leftarrow \mathcal{B}_{t-1}$; $\widetilde{\mathcal{D}}_t \leftarrow \emptyset$
  for each $(x, y, \tau) \sim \mathcal{D}_t$ **do**
    $z \leftarrow \Gamma_{\theta_{t-1}}(\tau)$
    $z^\star \leftarrow \arg\min_{z' \in \bar{\mathcal{B}}} \mathrm{Dist}(z, z')$
    $\widetilde{z} \leftarrow z^\star$ if $\mathrm{Dist}(z, z^\star) \leq \epsilon$ else z
    Add $(x, \widetilde{z}, y)$ to $\widetilde{\mathcal{D}}_t$
    $\bar{\mathcal{B}} \leftarrow \mathrm{MemUpdate}_{\Gamma_{\theta_{t-1}}}(\bar{\mathcal{B}}, z, \widetilde{z})$
  end for
  $\mathcal{B}_t \leftarrow \bar{\mathcal{B}}$
  $\theta_t \leftarrow \arg\min_\theta \mathcal{L}_{\mathrm{SST}}^t(\theta; \widetilde{\mathcal{D}}_t)$
end for

Schema Memory Maintenance. The schema memory $\mathcal{B}_t$ evolves across update steps without requiring replay access to $\mathcal{D}_{<t}$. Memory updates occur per example, not as a separate post-hoc phase. During stage t, all schema extraction and memory-operation proposals are produced by the pre-update model $\Gamma_{\theta_{t-1}}$; the updated model θ_t is obtained only after the schema-conditioned dataset $\widetilde{\mathcal{D}}_t$ has been constructed. After $\widetilde{z}$ is assigned for a given example, $\Gamma_{\theta_{t-1}}$ produces an update proposal and applies either a Reuse no-op or one of four memory-modifying operations. *Revise:* if the proposed schema matches an existing entry within the same skill family, the stored schema is updated with new keywords, canonical steps, or failure modes. *Create:* if $\bar{\mathcal{B}} = \varnothing$ or $\mathrm{Dist}(z, \mathrm{NN}_{\bar{\mathcal{B}}}(z)) > \epsilon$, the proposed schema is a candidate for a new entry; it is written into $\mathcal{B}_t$ only when the same procedural signature accumulates at least $n_{\min}$ supporting examples and the extractor's confidence exceeds a threshold δ_{create}. *Merge:* if two schemas share the same skill family and satisfy $\mathrm{Dist}(z, z') \leq \delta_{\mathrm{merge}}$, they are consolidated, retaining the more-retrieved entry. *Drop:* a schema whose utility score $u(z)$ falls below δ_{drop} is removed, provided it has been retrieved at least $k_{\min}$ times. The utility score aggregates behavioral signals including source correctness, retrieval support, retrieval precision. Implementation details and default hyperparameters are provided in Appendix D. Skill family membership is a discrete field assigned by the extractor at creation time from a fixed family list; it governs which skills are candidates for Revise and Merge, while Dist governs the within-family similarity threshold for Merge. Together, these operations allow $\mathcal{B}_t$ to grow, refine, and prune itself as new domains are encountered. A local linearized analysis in Appendix B shows that, under a schema-separable approximation, schema-conditioned directions need not increase the minimum shared-parameter displacement required by an update.

5 Experiments

Models and datasets. We evaluate SST on `Llama-3.2-3B-Instruct` [17] and `Qwen2.5-3B-Instruct` [57]. We consider two streaming settings: a mathematical reasoning stream [21] with four sequential domains (Algebra → Counting & Probability → Geometry → Intermediate Algebra), and an agent-task stream with three sequential domains (AlfWorld → WebShop → search-augmented QA [48, 58, 55]).

Training and inference. At each stream step, methods may only access the current domain plus any information explicitly permitted by their design. SST applies the same base objective as the

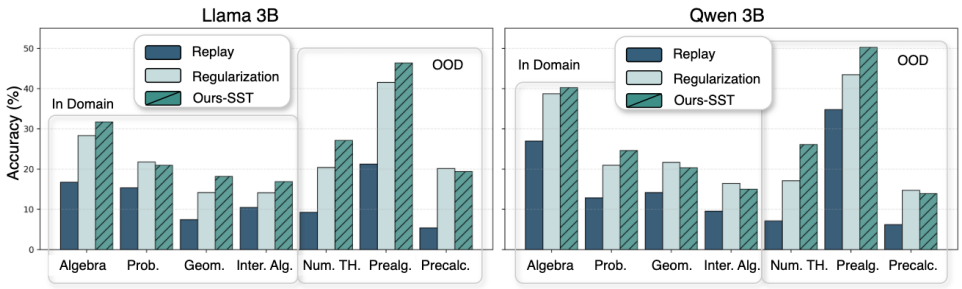


Figure 5: **Baseline comparison on streaming MATH.** Final accuracy after the four-domain stream. Left: seen domains. Right: held-out related MATH domains. SST yields a favorable retention-transfer trade-off on both models.

sequential baseline (SFT for MATH, RL for agent tasks), but rewrites updates through retrieved skill schemas. All main evaluations are retrieval-free at inference time, so reported gains reflect internalized knowledge rather than test-time prompting.

Evaluation. After each step we evaluate on all domains seen so far. We report average final performance and average max-drop forgetting. For a domain d introduced at step t_d, let $A_t(d)$ denote its accuracy after update step t. We define

$$F(d) = \max_{t_d \le t \le T} A_t(d) - A_T(d).$$

We average $F(d)$ over prior streamed domains, excluding the final domain because it is not followed by a later update. For the MATH stream, we additionally report transfer to held-out domains (Number Theory, Prealgebra, Precalculus) and preservation on general benchmarks (MMLU [20], GSM8K [10], PIQA [6], CommonsenseQA [50], SciQ [53]); the same benchmarks are used for the agent stream.

Baselines. We compare against SEQTRAIN (sequential, no mitigation), REPLAY (fixed-count rehearsal), and REGULARIZATION (L2 parameter anchoring to the base model). Unlike these, SST does not anchor to past examples or states; instead, it conditions new updates on reusable procedural schemas. We also include non-streaming experiments, Appendix A shows that the gain-drop pattern also holds for independent single-domain updates.

RQ1: Does SST reduce forgetting while preserving out-of-stream performance? Table 2 summarizes results across both streaming settings and backbones. In the MATH stream, Seq-SFT causes severe forgetting on Qwen3B (avg. max-drop 42.41 pts, final streamed average 19.96%); SST reduces max-drop to 7.10 pts and raises the streamed average to 25.12%, while recovering general benchmark performance from 58.92% to 72.06%. Llama3B follows the same direction: streamed average improves from 20.30% to 22.83% and general benchmarks from 69.33% to 72.33%. In the agent-task stream, SST improves final average task performance over Seq-RL by 5.75 pts on Qwen3B ($73.04\% \rightarrow 78.79\%$) and 3.27 pts on Llama3B ($74.50\% \rightarrow 77.77\%$), with forgetting reduced and general benchmark scores preserved. Across both streams, schema-conditioned updates improve streaming adaptation without increasing interference with out-of-stream capabilities. Appendix C provides the corresponding max-drop decomposition by domain.

RQ2: How does SST compare with standard forgetting-mitigation baselines? Replay and regularization preserve prior behavior by anchoring updates to earlier examples or model states. SST instead maps new examples into a reusable skill-schema space before updating, encouraging the model to relate new domains to previously acquired procedures. Figure 5 reports final accuracy after the four-domain MATH stream. Across both backbones, SST matches or improves over replay and regularization on streamed domains, with the largest advantage on held-out related MATH domains, where preserving old outputs is insufficient and transferring reusable procedures is required. This suggests

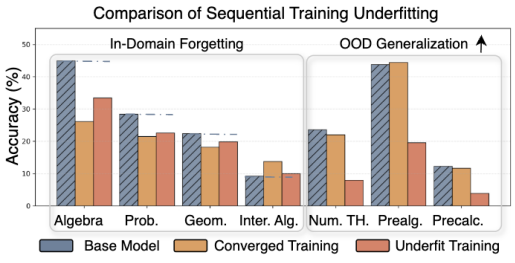


Figure 6: **Underfitting does not explain SST's retention.** Underfit SFT retains more earlier-domain accuracy only by failing to learn the final domain and degrading related OOD generalization.

schema transport does more than constrain model drift: by organizing updates around reusable skills, SST yields a favorable retention-transfer trade-off.

RQ3: Can forgetting be avoided simply by underfitting the update? A simpler explanation for improved retention is weaker optimization: undertrained updates may overwrite earlier domains less. To test this, we compare converged and underfit sequential SFT on Qwen-3B over the four-domain MATH stream. As shown in Figure 6, underfit SFT preserves some earlier-domain accuracy but at the cost of Intermediate Algebra performance and held-out MATH generalization (Number Theory, Prealgebra, Precalculus), a retention, plasticity trade-off rather than a solution to continual adaptation. SST targets a different mechanism: by conditioning updates on reusable skill schemas, it preserves prior procedures while still fitting new examples.

RQ4: Do later updates overwrite prior skills, or reinforce shared procedures? Final-checkpoint accuracy captures the net effect of the stream but not how skills evolve during adaptation. We evaluate each SST checkpoint on all streamed MATH domains after every update, testing whether shared procedural structure allows later updates to recover or reinforce earlier skills. For each prior domain d, max-drop forgetting is $F(d) = \max_{t_d \le t \le T} A_t(d) - A_T(d)$.

Figure 7 shows that SST keeps forgetting bounded: Qwen-3B drops by 8.68, 8.02, and 4.59 points on Algebra, Counting & Probability, and Geometry (avg. 7.10 pts), Llama-3B drops by 7.51, 4.04, and 0.71 points on Algebra, Counting & Probability, and Geometry, respectively, with an average max-drop of 4.09 points. Crucially, trajectories are not purely monotonic: Qwen-3B's Algebra accuracy partially recovers from 37.15% to 40.27% by the final checkpoint, consistent with later examples activating schemas shared with earlier domains. These trajectories are consistent with SST mitigating forgetting not by freezing prior behavior, but by allowing skills to be reused and sometimes reinforced as the stream grows.

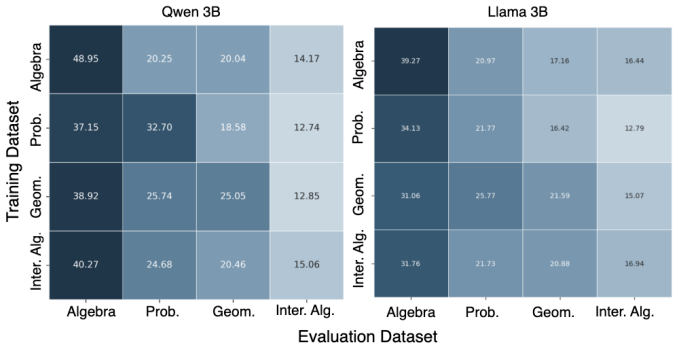


Figure 7: **SST checkpoint trajectories on streaming MATH.** Rows are checkpoints after each update and columns are evaluation domains. Partial recoveries after later updates are consistent with later domains reinforcing shared procedures.

RQ5: Is SST robust to the order in which domains arrive? Streaming adaptation is inherently order-sensitive: a method may appear effective only because the domains are presented in a favorable sequence. We therefore test SST on four permutations of the same MATH domains using the same training budget and evaluation protocol, and evaluate the final checkpoint on both streamed and held-out domains. Figure 8 shows that SST yields a broadly consistent performance profile across orders. Although some variation remains, especially on Algebra and Number Theory, no tested permutation causes a collapse on streamed domains, and held-out domains such as Prealgebra remain comparatively strong. This finite-order stress test does not prove full order invariance, but it reduces the concern that SST's retention-transfer trade-off is driven by a single hand-picked curriculum. Instead, schema transport appears to provide a stable procedural interface for integrating new domains as they arrive.

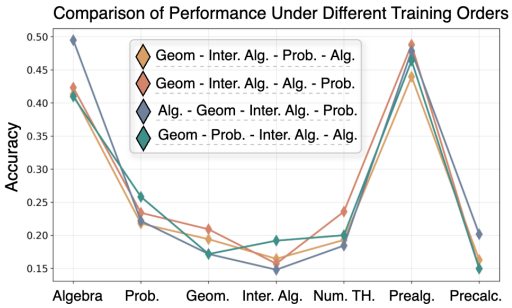


Figure 8: **Robustness to stream order.** SST is trained on four permutations of the same MATH stream and evaluated after the final update. Performance is broadly consistent across orders, indicating that SST does not rely on a single favorable domain sequence.

6 Conclusion

We proposed *Skill Schema Transport* (SST), a post-training framework that mitigates forgetting by expressing new-domain examples through reusable procedural schemas rather than constraining model drift. Across streaming math and agent-task settings, SST improves retention and related-domain transfer over sequential fine-tuning and standard forgetting-mitigation baselines. Further analyses reduce the likelihood that the gains are explained solely by underfitting or a favorable stream order, and checkpoint trajectories are consistent with the view that later domains can reuse and sometimes reinforce shared procedures. These results suggest that continual adaptation can be improved not only by limiting model change, but by representing new knowledge in procedural coordinates the model already understands.

Acknowledgments and Disclosure of Funding

This research was partially supported by the National Science Foundation (NSF) under Award No. IIS-2551752 and a Cisco Faculty Award.

References

[1] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[3] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[4] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[5] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(4):341–379, 2003.

[6] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

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

[8] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018.

[9] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

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

[11] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

[12] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš 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, 2021.

[13] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

[14] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[15] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[16] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[17] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[18] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[19] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 8342–8360, 2020.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[21] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[22] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[24] Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1416–1428, 2024.

[25] Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026.

[26] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *arXiv preprint arXiv:1902.00255*, 2019.

[27] 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.

[28] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR, 2019.

[29] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[30] Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

[31] Zhizhong Li and Derek Hoiem. Learning without forgetting. *arXiv preprint arXiv:1606.09282*, 2016.

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

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

[34] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[35] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[36] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in neural information processing systems*, 31, 2018.

[37] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[38] 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.

[39] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

[40] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

[41] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, 2017.

[42] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[43] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

[44] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

[45] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

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

[47] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

[48] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[49] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

[50] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, 2019.

[51] 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*, 2023.

[52] 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. *arXiv preprint arXiv:2310.14152*, 2023.

[53] Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, 2017.

[54] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.

[55] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

[56] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

[57] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[58] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[59] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

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

[61] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023.

[62] Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. *arXiv preprint arXiv:2501.13453*, 2025.

[63] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[64] Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

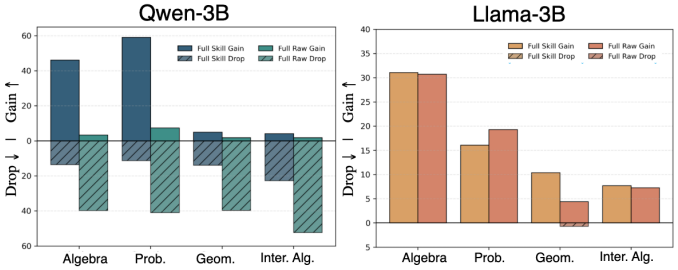


Figure 9: **Single-domain gain–drop diagnostic.** Each MATH domain is trained independently from the same base checkpoint. Upward bars show target-domain gain, and downward hatched bars show average drop on the other streamed MATH domains. Compared with FULL RAW, FULL SKILL yields a stronger gain–drop trade-off on both Qwen-3B and Llama-3B.

A Single-Domain Gain-Drop Diagnostic

The main experiments evaluate SST in streaming settings, where updates arrive sequentially and later domains may interact with earlier ones. To isolate the effect of schema conditioning from stream order and cumulative checkpoint dynamics, we additionally run a non-streaming single-domain diagnostic. For each target domain $d \in \{\text{Algebra}, \text{Counting \& Probability}, \text{Geometry}, \text{Intermediate Algebra}\}$, we initialize from the same base model and train only on examples from d. We compare raw full fine-tuning against skill-conditioned full fine-tuning using the same target-domain data and training budget. All evaluations in this diagnostic are retrieval-free at inference time; skill schemas are used only as training-time context.

Let $A_0(e)$ denote the base-model accuracy on evaluation domain e, and let $A_d^m(e)$ denote the accuracy after training on target domain d with method $m \in \{\text{RAW}, \text{SKILL}\}$. We measure the target-domain gain and cross-domain drop as

$$\text{Gain}_m(d) = A_d^m(d) - A_0(d),$$

$$\text{Drop}_m(d) = \frac{1}{|\mathcal{E}| - 1} \sum_{e \in \mathcal{E} \setminus \{d\}} [A_0(e) - A_d^m(e)]_+ ,$$

where $\mathcal{E}$ is the set of streamed MATH evaluation domains and $[\cdot]_+$ counts only performance decreases. In Figure 9, gains are plotted upward and drops are plotted downward as $-\text{Drop}$.

Figure 9 shows that schema conditioning improves the gain-drop trade-off even in isolated single-domain updates. Starting from the same base checkpoint, raw fine-tuning often provides weaker target-domain gains and/or larger drops on the remaining streamed MATH domains. In contrast, skill-conditioned fine-tuning achieves stronger or comparable in-domain gains with smaller cross-domain degradation. This complements the streaming results by showing that SST's advantage is not only a consequence of cumulative sequential adaptation, but also appears at the level of individual domain updates.

B Additional Details for the Linearized SST Analysis

B.1 A Linearized Drift Bound

We give a local linearized argument for why schema conditioning can reduce retention drift. The result concerns minimum shared-parameter displacement in a first-order logit-fitting problem, not

global convergence. Assume the model decomposes into shared parameters θ_{sh} and schema-specific parameters θ_{sc}, with logits

$$g_\theta(x, s) = f(W\phi_{\theta_{\mathrm{sh}}}(x) + Ue_{\theta_{\mathrm{sc}}}(s)) . \tag{8}$$

Under the following assumption, retention diagnostics evaluated with s_0 depend only on θ_{sh} to first order.

Assumption 1 (Null-schema inactivity)**.** *The schema embedding satisfies $e_{\theta_{\mathrm{sc}}}(s_0) \equiv 0$ for all θ_{sc}, so that*

$$\left.\frac{\partial g_\theta(x, s_0)}{\partial \theta_{\mathrm{sc}}}\right|_{\bar\theta} = 0. \tag{9}$$

At step t, let $\bar\theta = \theta_{t-1}$ and let $r \in \mathbb{R}^m$ be the desired first-order logit change for a mini-batch from $\mathcal{D}_t$. The linearized updates are

$$\Delta^{\mathrm{SFT}} = A\delta_{\mathrm{sh}}, \qquad \Delta^{\mathrm{SST}} = A\delta_{\mathrm{sh}} + S\delta_{\mathrm{sc}}, \tag{10}$$

where A, S are Jacobians w.r.t. shared and schema-specific parameters. Let $P_A = AA^\dagger$, $r_A = P_A r$, $\widetilde{S} = P_A S$, so that r_A is the component of r fitable by shared parameters and $\widetilde{S}$ captures how schema-specific directions can offload residual from θ_{sh}.

Define the minimum shared-parameter displacement under each objective as

$$\rho^2_{\mathrm{SFT}} = \min_{\delta_{\mathrm{sh}}} \left\{ \|\delta_{\mathrm{sh}}\|_2^2 : A\delta_{\mathrm{sh}} = r_A \right\}, \qquad \rho^2_{\mathrm{SST}} = \min_{\delta_{\mathrm{sh}}, \delta_{\mathrm{sc}}} \left\{ \|\delta_{\mathrm{sh}}\|_2^2 : A\delta_{\mathrm{sh}} + \widetilde{S}\delta_{\mathrm{sc}} = r_A \right\}. \tag{11}$$

Since retention diagnostics under s_0 depend only on θ_{sh}, δ_{sc} is free in ρ^2_{SST}.

Theorem 1 (SST cannot increase the minimum required shared-parameter change)**.** *Let $M = (AA^\top)^\dagger$ and $\widetilde{\mathcal{C}} = \mathrm{col}(\widetilde{S}) \subseteq \mathrm{col}(A)$. Then*

$$\rho^2_{\mathrm{SST}} = r_A^\top M r_A - r_A^\top M\widetilde{S}\left(\widetilde{S}^\top M\widetilde{S}\right)^\dagger \widetilde{S}^\top M r_A \le \rho^2_{\mathrm{SFT}}, \tag{12}$$

where $\|u\|_M^2 = u^\top M u$ for $u \in \mathrm{col}(A)$. The inequality is strict if and only if $\widetilde{S}^\top M r_A \neq 0$.

Proof. If $A = 0$, then $r_A = 0$ and the claim is trivial. Otherwise, let $A = U\Sigma V^\top$ be a compact SVD. Then $M = (AA^\top)^\dagger = U\Sigma^{-2}U^\top$. Hence $\langle u, v\rangle_M = u^\top M v$ is an inner product on $\mathrm{col}(A)$: for every nonzero $u = Uw \in \mathrm{col}(A)$, we have $u^\top M u = w^\top \Sigma^{-2} w > 0$.

For any $b \in \mathrm{col}(A)$, the minimum-norm solution of $A\delta_{\mathrm{sh}} = b$ is $A^\dagger b$. Therefore

$$\min_{\delta_{\mathrm{sh}}} \left\{ \|\delta_{\mathrm{sh}}\|_2^2 : A\delta_{\mathrm{sh}} = b \right\} = \|A^\dagger b\|_2^2 = b^\top (A^\dagger)^\top A^\dagger b = b^\top (AA^\top)^\dagger b = \|b\|_M^2. \tag{13}$$

The identity $(A^\dagger)^\top A^\dagger = (AA^\top)^\dagger$ follows directly from the same SVD. Taking $b = r_A \in \mathrm{col}(A)$ gives $\rho^2_{\mathrm{SFT}} = \|r_A\|_M^2$.

For SST, write $s = \widetilde{S}\delta_{\mathrm{sc}} \in \widetilde{\mathcal{C}}$. For any fixed s, the remaining residual assigned to shared parameters is $r_A - s \in \mathrm{col}(A)$, and the minimum shared-parameter norm is $\|r_A - s\|_M^2$ by the previous paragraph. Minimizing over $s \in \widetilde{\mathcal{C}}$ gives

$$\rho^2_{\mathrm{SST}} = \min_{s \in \widetilde{\mathcal{C}}} \|r_A - s\|_M^2. \tag{14}$$

Thus ρ^2_{SST} is the squared M-distance from r_A to $\widetilde{\mathcal{C}}$. The M-orthogonal projection onto $\widetilde{\mathcal{C}}$ is

$$P^M_{\widetilde{\mathcal{C}}} r_A = \widetilde{S}\left(\widetilde{S}^\top M\widetilde{S}\right)^\dagger \widetilde{S}^\top M r_A. \tag{15}$$

Therefore,

$$\rho^2_{\mathrm{SST}} = \left\| r_A - P^M_{\widetilde{\mathcal{C}}} r_A \right\|_M^2 \tag{16}$$

$$= \|r_A\|_M^2 - \left\| P^M_{\widetilde{\mathcal{C}}} r_A \right\|_M^2 \tag{17}$$

$$= r_A^\top M r_A - r_A^\top M\widetilde{S}\left(\widetilde{S}^\top M\widetilde{S}\right)^\dagger \widetilde{S}^\top M r_A. \tag{18}$$

This proves (12). Since $0 \in \widetilde{\mathcal{C}}$,

$$\rho_{\mathrm{SST}}^2 = \min_{s \in \widetilde{\mathcal{C}}} \|r_A - s\|_M^2 \le \|r_A\|_M^2 = \rho_{\mathrm{SFT}}^2. \tag{19}$$

The inequality is strict exactly when $P_{\widetilde{\mathcal{C}}}^M r_A \neq 0$, which is equivalent to r_A not being M-orthogonal to $\widetilde{\mathcal{C}}$, i.e., $\widetilde{S}^\top M r_A \neq 0$. □

Corollary 1 (Incremental retention-drift bound)**.** *Let $G_{\bar\theta}(\theta; \mathcal{Q})$ be a nonnegative incremental retention diagnostic evaluated under s_0 that depends locally only on θ_{sh}, satisfies $G_{\bar\theta}(\bar\theta; \mathcal{Q}) = 0$, and is $K_{\mathcal{Q}}$-Lipschitz in θ_{sh} near $\bar\theta$. For the minimum-shared-norm SST update,*

$$G_{\bar\theta}(\bar\theta + \delta^{\mathrm{SST}}; \mathcal{Q}) \le K_{\mathcal{Q}} \rho_{\mathrm{SST}} \le K_{\mathcal{Q}} \rho_{\mathrm{SFT}}. \tag{20}$$

For a raw-reference diagnostic G^0, this bounds only the increment $|G^0(\theta; \mathcal{Q}) - G^0(\bar\theta; \mathcal{Q})|$, not $G^0(\theta; \mathcal{Q})$ itself.

Proof. See Appendix B.2. □

Corollary 2 (Retention-loss-increase bound)**.** *Assume the retention loss $\mathcal{L}_{\mathcal{Q}}$ evaluated under s_0 depends locally only on θ_{sh} and is $K_{\mathcal{Q}}$-Lipschitz in θ_{sh} near $\bar\theta$. Define*

$$\Delta\mathcal{L}_{\mathcal{Q}}(\delta) = \mathcal{L}_{\mathcal{Q}}(\bar\theta + \delta) - \mathcal{L}_{\mathcal{Q}}(\bar\theta). \tag{21}$$

Then, for any minimum-shared-norm SST update,

$$\left[\Delta\mathcal{L}_{\mathcal{Q}}(\delta^{\mathrm{SST}})\right]_+ \le K_{\mathcal{Q}} \rho_{\mathrm{SST}} \le K_{\mathcal{Q}} \rho_{\mathrm{SFT}}. \tag{22}$$

Proof. See Appendix B.3. □

B.2 Proof of Corollary 1

By Assumption 1 and the local schema-separable decomposition, $G_{\bar\theta}(\cdot\,; \mathcal{Q})$ depends locally only on δ_{sh}. Since $G_{\bar\theta}(\bar\theta; \mathcal{Q}) = 0$, nonnegativity and Lipschitz continuity give

$$G_{\bar\theta}(\bar\theta + \delta; \mathcal{Q}) = \left|G_{\bar\theta}(\bar\theta + \delta; \mathcal{Q}) - G_{\bar\theta}(\bar\theta; \mathcal{Q})\right| \le K_{\mathcal{Q}} \|\delta_{\mathrm{sh}}\|_2. \tag{23}$$

For the minimum-shared-norm SST update, $\|\delta_{\mathrm{sh}}^{\mathrm{SST}}\|_2 = \rho_{\mathrm{SST}}$. Theorem 1 gives $\rho_{\mathrm{SST}} \le \rho_{\mathrm{SFT}}$.

B.3 Proof of Corollary 2

By Assumption 1, $\mathcal{L}_{\mathcal{Q}}$ evaluated under s_0 depends locally only on θ_{sh} and is $K_{\mathcal{Q}}$-Lipschitz. Therefore

$$\left|\mathcal{L}_{\mathcal{Q}}(\bar\theta + \delta) - \mathcal{L}_{\mathcal{Q}}(\bar\theta)\right| \le K_{\mathcal{Q}} \|\delta_{\mathrm{sh}}\|_2, \tag{24}$$

which gives

$$\left[\Delta\mathcal{L}_{\mathcal{Q}}(\delta)\right]_+ \le K_{\mathcal{Q}} \|\delta_{\mathrm{sh}}\|_2. \tag{25}$$

For the minimum-shared-norm SST update, $\|\delta_{\mathrm{sh}}^{\mathrm{SST}}\|_2 = \rho_{\mathrm{SST}}$. Theorem 1 gives $\rho_{\mathrm{SST}} \le \rho_{\mathrm{SFT}}$.

B.4 Exact interpolation versus projected fitable component

The main theorem compares the shared-parameter displacement needed for the component $r_A = P_A r$ that is fitable by the shared Jacobian. This is the appropriate comparison when the linearized system is interpreted in least-squares or projected-logit form. If instead one requires exact interpolation of the full residual r, the feasible sets should be written explicitly.

For exact SFT interpolation,

$$\rho_{\mathrm{SFT,exact}}^2 = \min_{\delta_{\mathrm{sh}}} \left\{ \|\delta_{\mathrm{sh}}\|_2^2 : A\delta_{\mathrm{sh}} = r \right\} \tag{26}$$

is finite if and only if $r \in \mathrm{col}(A)$. For exact SST interpolation,

$$\rho_{\mathrm{SST,exact}}^2 = \min_{\delta_{\mathrm{sh}}, \delta_{\mathrm{sc}}} \left\{ \|\delta_{\mathrm{sh}}\|_2^2 : A\delta_{\mathrm{sh}} + S\delta_{\mathrm{sc}} = r \right\} \tag{27}$$

is finite if and only if $r \in \mathrm{col}(A) + \mathrm{col}(S)$. Whenever this exact SST problem is feasible, it can be written as

$$\rho^2_{\mathrm{SST,exact}} = \min_{s \in \mathrm{col}(S):\ r-s \in \mathrm{col}(A)} \|r-s\|^2_M. \tag{28}$$

In the special case where $r \in \mathrm{col}(A)$, exact SFT is feasible and the exact SST expression reduces to

$$\rho^2_{\mathrm{SST,exact}} = \min_{s \in \mathrm{col}(S) \cap \mathrm{col}(A)} \|r-s\|^2_M \le \|r\|^2_M = \rho^2_{\mathrm{SFT,exact}}. \tag{29}$$

Thus exact interpolation also cannot increase the minimum required shared-parameter displacement. The projected theorem in the main text is slightly more permissive because it studies the shared-fitable component r_A and the projected schema directions $P_A S$; the exact full-residual theorem instead uses the intersection $\mathrm{col}(S) \cap \mathrm{col}(A)$ when $r \in \mathrm{col}(A)$.

B.5 Counterexample to the ordinary Euclidean-projection claim

The following example shows why the ordinary Euclidean projection P_S onto $\mathrm{col}(S)$ is not sufficient for the theorem.

Let

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}, \qquad M^{-1} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \tag{30}$$

and choose a full-row-rank matrix A such that $AA^\top = M^{-1}$, for example

$$A = \begin{pmatrix} \sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}. \tag{31}$$

Let

$$S = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad r = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{32}$$

The ordinary Euclidean projection of r onto $\mathrm{col}(S)$ is

$$P_S r = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \neq 0. \tag{33}$$

However,

$$S^\top M r = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0. \tag{34}$$

Therefore $P^M_{\mathrm{col}(S)} r = 0$ and there is no strict improvement:

$$\rho^2_{\mathrm{SFT}} = r^\top M r = 1, \qquad \rho^2_{\mathrm{SST}} = \min_{a \in \mathbb{R}} \begin{pmatrix} 1-a & 1 \end{pmatrix} M \begin{pmatrix} 1-a \\ 1 \end{pmatrix} = \min_{a \in \mathbb{R}} (1+a^2) = 1. \tag{35}$$

Moreover, fitting the ordinary Euclidean residual $(I - P_S)r = (0,1)^\top$ through A would require squared shared norm

$$(0,1) M (0,1)^\top = 2, \tag{36}$$

which is larger than the SFT cost 1. Hence the statement

$$\rho^2_{\mathrm{SST}} = \min_{\delta_{\mathrm{sh}}} \left\{ \|\delta_{\mathrm{sh}}\|_2^2 : A\delta_{\mathrm{sh}} = (I - P_S) r \right\} \tag{37}$$

is false in general. It becomes valid if $AA^\top$ is proportional to the identity, or if P_S is replaced by the M-orthogonal projection.

B.6 What the theorem does and does not prove

The theorem proves a statement about a local first-order interpolation problem: after linearizing the logits around the pre-update parameter $\bar{\theta}$, adding schema-specific tangent directions cannot increase the minimum amount of movement required in shared parameters to fit the same shared-fitable residual component. The theorem does not, by itself, prove that arbitrary nonconvex SGD will find the minimum-shared-norm solution, nor that every empirical forgetting metric, including accuracy max-drop, must decrease without the stated null-schema and Lipschitz assumptions. Theory-related claims should therefore be phrased as local upper bounds under explicit assumptions, while the actual retention behavior should be validated empirically.

Table 3: **Default training and evaluation settings.**

Category	Setting	Value
SFT hyperparameters		
SFT	Training framework	LlamaFactory
SFT	Maximum sequence length	4096
SFT	Precision	bf16
SFT	Scheduler	cosine
SFT	Warmup ratio	0.03
SFT	Per-device train batch size	2
SFT	Gradient accumulation steps	8
SFT	Effective batch size	16
SFT	Training epochs	5
SFT	Full fine-tuning learning rate	1×10^{-5}
SFT	Retrieved skills k	2
Evaluation settings		
Eval	Decoding	deterministic
Eval	Temperature	0.0
Eval	Top-p	1.0
Eval	Maximum new tokens	512
Eval	Inference engine	vLLM
Eval	Tensor parallel size	1
Eval	GPU memory utilization	0.8
Eval	Maximum number of sequences	64

C Additional Details for RQ1

Forgetting metric. For each streamed domain d, let t_d be the update step at which d is introduced and let T be the final update step. We compute max-drop forgetting as

$$F(d) = \max_{t_d \le t \le T} A_t(d) - A_T(d), \tag{38}$$

where $A_t(d)$ is the evaluation accuracy on domain d after update step t. We average $F(d)$ over prior streamed domains only. The final domain is excluded because it is not followed by a later update and therefore cannot be overwritten within the observed stream.

D Implementation Details

Sequential MATH adaptation. For the sequential skill-conditioned SFT setting, we train on four MATH domains in order: Algebra, Counting & Probability, Geometry, and Intermediate Algebra. After each stage, we evaluate the resulting checkpoint on all seven Hendrycks MATH categories: Algebra, Counting & Probability, Geometry, Intermediate Algebra, Number Theory, Prealgebra, and Precalculus. For SST, we construct skill-augmented training examples by retrieving the top-k skills from the schema memory, with $k = 2$ by default, and use the full skill prompt format described in Appendix E. Non-SST baselines use the same training examples and hyperparameters, but without the retrieved skill context.

SFT training configuration. We implement SFT using LlamaFactory [63] and run evaluation with vLLM [29]. Unless otherwise specified, we use bf16 training, cosine learning-rate scheduling, and a warmup ratio of 0.03. The default per-device batch size is 2 with gradient accumulation over 8 steps, giving an effective batch size of 16 on a single GPU. We train for 5 epochs by default. For full fine-tuning, we use a learning rate of 1×10^{-5};

RL implementation details. For the agent-stream experiments, we instantiate SST-RL as a schema-conditioned offline reinforcement-learning setup. In each agent domain, SST-RL augments the state prompt with the top retrieved skills from the schema memory and then optimizes the policy with a GRPO-style objective. Seq-RL and SST-RL use the same RL objective, reward functions, decoding settings, and training budget; the only difference is that SST-RL conditions the training prompt on retrieved skill schemas. In our implementation, the agent skill prompt uses top-k retrieval with $k = 4$.

Rewards are rule-based and combine action correctness with output-format compliance. The main reward is an exact-action reward: it assigns 1.0 when the generated action exactly matches the reference action and 0.0 otherwise. An auxiliary format reward adds 0.2 when the response follows the required tagged protocol, including the expected `<think>` and `<action>` fields. Unless otherwise specified, GRPO training uses learning rate 10^{-6}, one training epoch, per-device batch size 1, gradient accumulation 4, four sampled completions per prompt, maximum prompt length 4096, maximum completion length 512, temperature 0.9, $\beta = 0.0$, and a DAPO-style loss.

Agent evaluation. Agent evaluation is action-level rather than full episodic environment evaluation. The main metric reported for AlfWorld, WebShop, and Search is exact action-match accuracy under deterministic decoding. For Search, we additionally compute tool-selection accuracy, average query-token F1, and a soft search accuracy that requires both the correct tool and query-token F1 of at least 0.5. All agent results in Table 2 use retrieval-free inference unless explicitly stated otherwise. Thus, the agent-stream experiment tests whether schema-conditioned training improves sequential robustness in agent decision-making, rather than introducing a new RL objective or relying on test-time skill retrieval.

Evaluation configuration. We use deterministic decoding for all evaluations, with temperature 0.0 and top-$p = 1.0$, and generate up to 512 new tokens per example. For vLLM inference, we use tensor parallel size 1. Typical inference settings use `gpu_memory_utilization=0.8` and `max_num_seqs=64`. All main results reported in Table 2 use retrieval-free inference: the final fine-tuned model is evaluated with the raw solver prompt and without prepended skill blocks. Retrieval-enabled inference is used only as an optional diagnostic ablation, and any such results are reported separately when included.

Compute. All main SFT jobs are run on a single-GPU node, typically using $1\times$A100 GPU, 8 CPU cores, and 256GB host memory, with a 72-hour wall-clock limit for training jobs. Base-model evaluation jobs use the same single-GPU setup with a shorter time budget. In our local setup, the full Hendrycks MATH test set contains $5{,}000$ examples across seven subject categories.

Replay baseline. Our replay baseline uses fixed-count rehearsal rather than a dynamic reservoir buffer. At stream step t, the training set contains all current-domain examples together with a uniformly sampled subset from each previous domain. Unless otherwise specified, we sample 500 examples per previous domain using a fixed random seed and sample without replacement after shuffling. Replay examples are concatenated with current-domain examples without additional weighting. The replay ratio is therefore step-dependent:

$$\text{replay_ratio}_t = \frac{(t-1) \times 500}{|\mathcal{D}_t|}.$$

Regularization baseline. The regularization baseline is an L2-SP-style parameter anchoring baseline. It loads the original base model as an anchor and adds

$$\lambda \sum_{\theta_i \in \Theta_{\text{train}}} \|\theta_i - \theta_{0,i}\|_2^2$$

to the sequential SFT objective over trainable parameters. Unless otherwise specified, we use $\lambda = 10^{-4}$ and the same optimizer, sequence length, batch size, learning-rate schedule, and stream order as the sequential SFT baseline.

Error-clustering diagnostic. We construct Figure 3 by first evaluating the base model on MATH examples and retaining examples answered incorrectly under the same answer-normalization protocol used in Section 5. Each failed example is represented by the *problem text only* (i.e., excluding both the gold solution and the model-generated reasoning trace), embedded with the sentence embedding model `BAAI/bge-small-en-v1.5`, and projected to two dimensions with PCA using the top two principal components. Procedural error labels are assigned by *manual annotation* using the categories ALGEBRAIC MANIPULATION, CASE OMISSION / COUNTING BREAKDOWN, GEOMETRIC RELATION MISUSE, MODULAR / DIVISIBILITY REASONING, COORDINATE-MODELING ERROR, CONSTRAINT / INTERVAL CHECKING, and ANSWER-FORMAT / FINALIZATION. Domain labels are the original

MATH subject categories. To avoid circularity, the procedural labels are used only for post-hoc visualization and are not used as supervision, retrieval targets, memory entries, or training signals for SST. This diagnostic is intended only as motivating evidence that failure modes exhibit cross-domain procedural overlap, rather than as a quantitative claim about a complete latent error taxonomy.

E Prompting Details

Overview. We use three prompt types in the SST pipeline: a raw solver prompt for base and main retrieval-free evaluation, a skill-conditioned solver prompt for SST training, and a memory-update prompt for maintaining the skill memory. Unless explicitly stated otherwise, all reported main results use the raw solver prompt at inference time without retrieved skill blocks. In all skill-conditioned SFT examples, retrieved skills are provided only as input context; the supervision target remains the gold solution.

Raw solver prompt

```
Solve the following competition math problem. Show concise reasoning and
final answer.
Type: {problem_type}
Level: {level}
Problem:
{problem}
```

Skill-conditioned training prompt. For SST, we prepend the top-k retrieved skill blocks to the problem input, with $k = 2$ by default. The training target is the gold solution only; the model is not trained to reproduce the retrieved skills. If no skill is retrieved, the prompt includes a short fallback line indicating that the model should solve directly.

Skill-conditioned solver prompt

```
Solve the following competition math problem. Use the retrieved skills when
they fit the problem. Show concise reasoning and final answer.
{retrieved_skill_blocks}
Type: {problem_type}
Level: {level}
Problem:
{problem}
```

Fallback when no skill is retrieved

```
Relevant skills:
- None retrieved. Solve directly.
```

Retrieved skill block format. Each retrieved skill is represented as a compact procedural description. The default prompt uses the full skill format below.

Retrieved skill block

```
Relevant skill {rank}:
- Skill ID: {skill_id}
- Family: {skill_family}
- Goal: {goal}
- When to use: {applicability}
- Steps: {canonical_steps}
- Common failure modes: {failure_modes}
```

Memory-update prompt. The memory-update prompt is used during bootstrapping and online memory maintenance. Given the current problem, the model output, and the retrieved skills, the model selects one memory operation from REUSE, REVISE, CREATE, MERGE, and DROP. We require the output to be a single JSON object with no markdown or extra text. Reuse and drop target an existing skill; revise, create, and merge additionally provide a proposed skill entry.

Memory-update prompt, condensed

```
You are maintaining a model-relative skill memory for competition math.
Given the current problem, the model's reasoning and final answer, and the
retrieved skills, decide whether to reuse, revise, create, merge, or drop a
skill.

Rules:
- Prefer reuse over create.
- Prefer revise over create if the reasoning pattern is close.
- Create a new skill only when no existing skill covers the pattern.
- Use only the provided skill family IDs.
- Reuse, revise, merge, and drop must name exact existing skill IDs.
- Do not invent missing families, target IDs, or merge sources.
- Return exactly one JSON object with no markdown fences and no extra text.

Return fields:
{action, target_skill_id, proposed_skill, reason, confidence}

Families:
{families_json}

Problem:
{problem}

Model output:
{prediction}

Retrieved skills:
{retrieved_skills_json}
```

Chat template. Before vLLM inference, each text prompt is wrapped as a single user message and passed through the model tokenizer's chat template with generation enabled. This keeps the prompting format consistent across base-model evaluation, non-retrieval evaluation, and retrieval-conditioned evaluation.

F Limitations

First, our empirical scope covers only two 3B backbones and two streaming settings; whether the same retention–transfer behavior holds for larger models, longer streams, multilingual or multimodal settings, or domains with weak prior procedural representation remains open.

Second, SST depends on schema extraction and retrieval quality. Incomplete, generic, or mismatched schemas provide weak or misleading conditioning, and inaccurate memory updates may fragment related skills or merge distinct ones. More systematic stress tests on noisy, ambiguous, or adversarial inputs are needed.

Third, the theoretical analysis is a local explanatory argument, not a global guarantee. The linearized result shows schema-conditioned directions need not increase shared-parameter displacement, but this does not guarantee nonconvex optimization finds such solutions in practice.

Fourth, we do not report full multi-seed statistics with confidence intervals for all comparisons. Future work should include repeated runs and larger-scale ablations over retrieval thresholds, memory-update rules, and prompt formats.

Fifth, SST introduces additional overhead for schema extraction, retrieval, and memory updates. Scaling behavior for very large streams or high-throughput settings remains unmeasured.

Finally, SST reduces but does not eliminate forgetting, and should be viewed as improving the retention–plasticity trade-off rather than solving continual learning.

G Broader Impact

SST may benefit applications requiring stable sequential model updates by reducing unnecessary overwriting of prior capabilities without retaining large amounts of old data. The explicit schema representation also offers a procedural interface for inspecting which reasoning patterns are reused or revised during training, which may assist in diagnosing failures and understanding how updates affect model behavior.

At the same time, more effective continual adaptation could amplify both beneficial and harmful uses, including harmful automation or unsafe agent behavior if deployed without safeguards. Learned schemas may also preserve and transfer biases or unsafe procedures encoded in training data, and should be evaluated not only for accuracy and forgetting but also for fairness, robustness, and safety in the intended deployment context.

Users should not over-interpret the explicit schema structure as a formal guarantee of model behavior: schemas are conditioning context for training, not certificates of internal reasoning. Any deployment should include independent evaluation, red-teaming, and monitoring for regressions. Future releases of code or schema memories should document data sources, licenses, intended use, and known limitations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the paper's core contribution, Skill Schema Transport (SST), and delimit the setting to continual domain adaptation for language models under streaming math and agent-task updates. The numerical claims are phrased relative to the sequential post-training baselines and match the reported experiments in Section 5 and Table 2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the work, with the limitations provided in the appendix. This satisfies the requirement that the authors acknowledge limitations of their approach and clarify the scope of the results.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: Appendix B states the assumptions for the local linearized SST analysis, presents Theorem 1 and Corollaries 1–2, and provides the corresponding proofs. The appendix also states the scope of the result, making clear that it is a local first-order bound on shared-parameter displacement and not a global convergence guarantee for arbitrary optimization.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The paper describes the SST algorithm, schema extraction and memory-update procedure, update streams, evaluated backbones, baselines, metrics, and evaluation protocol in Sections 4–5. Appendix D reports the main SFT, RL, evaluation, replay, regularization, and compute settings, and Appendix E provides the prompt formats used for raw solving, skill-conditioned solving, and memory updates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No].

   Justification: The paper uses existing public benchmarks and publicly available model backbones, and it describes the experimental setup and evaluation protocol. However, the implementation code, processed stream construction, exact run scripts, and any generated schema-memory artifacts are not currently released.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Appendix D provides the main SFT/RL hyperparameters, decoding and vLLM evaluation settings, replay and regularization baseline details, and compute setup; Appendix E provides the prompt templates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No].

Justification: The paper reports accuracy-based results and ablations but does not include error bars, confidence intervals, or formal statistical significance tests. The experiments cover multiple streams, backbones, and order permutations, but a full multi-seed statistical analysis is not included.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [No].

Justification: Appendix D reports the typical resource setup for the main SFT and evaluation jobs, including a single A100 GPU, 8 CPU cores, 256GB host memory, and a 72-hour wall-clock limit. The current version does not yet break down runtime, memory use, or total GPU-hours for each individual experiment, baseline, and stream setting.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes].

Justification: The research studies continual domain adaptation for language models using existing benchmark datasets and does not involve human-subject experiments, collection

of private user data, or deployment of systems that directly affect individuals. The paper frames SST as a forgetting-mitigation method, reports retention and utility metrics, and states the assumptions and scope of the theoretical and empirical claims.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The work may have positive impacts by making continual adaptation more reliable, reducing catastrophic forgetting, and helping models reuse procedural knowledge across domains without relying on replay of old data. Potential negative impacts include propagating incorrect or biased skill schemas, making unwanted behaviors more persistent through procedural reuse, or enabling more capable domain-specialized agents if applied in unsafe settings; these risks motivate careful evaluation of schema quality, retention, and downstream use.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release new pretrained language models, image generators, high-risk scraped datasets, or deployed agent systems. The work proposes an SST training framework and evaluates it on existing benchmark datasets, so safeguards for releasing high-risk models or datasets are not applicable.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The paper cites the main datasets, model backbones, software tools, and prior methods used in the experiments, but the current version does not yet provide an exhaustive asset table listing the license or terms of use for every dataset, model, and software package. We do not redistribute these assets as part of this submission.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not introduce new datasets, benchmarks, model checkpoints, or other standalone assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The paper does not conduct crowdsourcing experiments or human-subject studies. All experiments are performed using existing benchmark datasets and model evaluations.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A].

Justification: The paper does not involve human-subject studies or crowdsourcing experiments. Therefore, IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes].

Justification: SST uses an instruction-tuned language model as a core methodological component for extracting candidate skill schemas and proposing schema-memory updates. This usage is part of the research method itself rather than a writing aid, since the generated schemas affect the schema-conditioned training data and the resulting empirical performance. he relevant implementation details are documented in Appendix D, and the prompt formats are documented in Appendix E.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.