SEMANTIC PROJECTION FOR CONTINUAL SELF-EVOLUTION OF LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents increasingly rely on persistent natural-language skills to adapt beyond their frozen model parameters. When a shared skill is repeatedly revised from a non-stationary, heterogeneous task stream, however, improvements for new tasks can overwrite procedures needed for earlier ones. In continual learning, Orthogonal Gradient Descent (OGD) addresses analogous interference by projecting a new-task gradient onto a subspace that locally preserves prior predictions. Natural-language skill revisions, however, have neither gradients nor a canonical vector space in which such a projection can be performed. We introduce *Semantic-Scope Projected Evolution* (SSPE), which transfers the functional principle of gradient projection from parameter space to behavior space. SSPE treats an unconstrained skill revision as a proposed update, identifies acquired capabilities with which it may interfere, and uses the observed gains and regressions to construct a compatible revision rather than merely rejecting the update. This enables one shared skill to evolve across latent and recurring task contexts without exposing semantic domain identities to the evolution model. Across controlled synthetic streams and heterogeneous real-agent benchmarks, SSPE improves final cross-domain competence and mitigates forgetting relative to strong skill-evolution baselines. The evolved skill also retains the strongest average performance after transfer to a different executor model. These results establish semantic projection as a promising principle for stable and adaptive self-evolution of language agents.

1 INTRODUCTION

Language-model agents increasingly combine reasoning with tools, environments, and multi-step action (Yao et al., 2023; Zhou et al., 2024), yet most are deployed with fixed behavior. A complementary line of work asks agents to improve from their own trajectories. Verbal reflection, self-feedback, and distilled experience can repair later attempts (Shinn et al., 2023; Madaan et al., 2023; Zhao et al., 2024), while episodic memories and reusable skills support behavior over longer horizons (Park et al., 2023; Wang et al., 2024). More recent agents organize this experience into adaptive memories, reasoning strategies, context playbooks, or explicit skills (Packer et al., 2023; Suzgun et al., 2025; Xu et al., 2025; Ouyang et al., 2025; Zhang et al., 2025; Yang et al., 2026b). Natural-language skills are a particularly useful evolution target. They encode procedural knowledge in an inspectable artifact, can be revised without changing the foundation model, and can be reused across tasks and even executor models.

The central challenge changes when one shared skill evolves through a heterogeneous stream. Existing textual optimizers can diagnose traces, aggregate recurring feedback, validate edits, prune redundant instructions, or maintain optimizer-like state (Yuksekgonul et al., 2025; Wang et al., 2026; Yang et al., 2026a; Mi et al., 2026). These mechanisms make individual updates more systematic, but most are developed primarily within one task distribution. In deployment, an agent may move among spreadsheet manipulation, retrieval, function calling, and document understanding. A revision that repairs the current failures can overwrite, contradict, or deprioritize procedures needed later when an earlier capability returns. StreamBench and AgentStream show why evolution must be studied over stateful streams rather than independent examples, and that its benefit depends strongly on the model, update mechanism, and stream organization (Wu et al., 2024; Wei et al., 2025; Yan et al., 2026). We call the resulting failure *procedural catastrophic forgetting*. Current-domain val-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

idation may reject a locally poor edit, but it cannot expose regressions on absent capabilities or explain how to preserve a useful new behavior while repairing the damaged one.

This tension parallels continual learning, where a learner must remain plastic on new tasks without erasing earlier competence (Parisi et al., 2019; De Lange et al., 2022). Prior work regularizes important parameters, replays representative experience, allocates task-specific capacity, or constrains new gradients using historical information (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Rolnick et al., 2019; Buzzega et al., 2020; Rusu et al., 2016; Serra et al., 2018; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019). Orthogonal Gradient Descent (OGD) is especially suggestive. It removes from a new-task gradient the components aligned with directions to which previous predictions are sensitive, thereby retaining as much useful progress as possible without changing protected behavior (Farajtabar et al., 2020). A textual skill, however, has no canonical gradient, subtraction, or behaviorally meaningful inner product. Embedding-space orthogonality would not imply that two instructions induce compatible agent behavior. The relevant invariant must therefore be transferred at the level of observed behavior rather than text geometry.

We introduce *Semantic-Scope Projected Evolution* (SSPE), a method for continual skill evolution over heterogeneous task streams. SSPE begins with an unconstrained revision driven by the current trajectories, predicts which acquired capabilities the revision may affect, and selectively audits representative historical behavior. If the proposal causes interference, the model receives both its current gains and the concrete historical regressions, then iteratively rewrites the proposal to satisfy the observed constraints or chooses no update. This turns verification from a terminal accept-or-reject gate into constructive feedback for semantic projection. Our study formalizes this streaming setting, examines the role of historical capability evidence through controlled ablations, and evaluates final cross-domain competence, forgetting, and transfer of evolved skills across executor models. Across controlled and real heterogeneous streams, the results support using observed historical conflicts as constructive feedback for repairing an update, rather than using historical evaluation only as an accept–reject gate.

2 Related Work

Persistent experience and self-evolving agents. Language agents can learn from interaction by storing reflections, demonstrations, episodes, or reusable procedures (Shinn et al., 2023; Zhao et al., 2024; Park et al., 2023; Wang et al., 2024; Packer et al., 2023). Recent memory systems organize experience into adaptive notes, linked memories, and retrieved reasoning strategies (Suzgun et al., 2025; Xu et al., 2025; Ouyang et al., 2025; Wei et al., 2025). Surveys consequently treat parameters, prompts, memory, tools, workflows, and agent architecture as distinct evolution targets (Gao et al., 2026). SSPE focuses on a shared procedural skill and on interference introduced when that artifact is repeatedly revised across capabilities.

Textual optimization and skill evolution. Prompt optimizers search, score, or iteratively refine instructions using model feedback (Zhou et al., 2023; Yang et al., 2024; Pryzant et al., 2023; Fernando et al., 2024; Khattab et al., 2024; Agrawal et al., 2025). TextGrad generalizes this view by propagating natural-language feedback through compound systems (Yuksekgonul et al., 2025). Skill-oriented methods add structured diagnosis and semantic momentum, validation-gated edits, backward pruning, adaptive optimizer state, or verified procedural units (Wang et al., 2026; Yang et al., 2026a; Mi et al., 2026). They improve how updates are generated or selected. Our concern is complementary: repairing an update after its useful current behavior and its historical interference have both been observed.

Streaming evaluation and continual learning. StreamBench, Evo-Memory, and AgentStream evaluate agents through stateful feedback sequences rather than isolated examples (Wu et al., 2024; Wei et al., 2025; Yan et al., 2026). Continual learning addresses related interference through regularization (Li & Hoiem, 2016; Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018), replay (Rebuffi et al., 2017; Rolnick et al., 2019; Buzzega et al., 2020), architectural isolation (Rusu et al., 2016; Serra et al., 2018; Mallya & Lazebnik, 2018), and gradient constraints or projections (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Farajtabar et al., 2020; Saha et al., 2021; Riemer et al., 2019). SSPE does not project a numerical gradient. It transfers the same stability–

plasticity principle by predicting relevant historical capabilities, measuring behavioral regressions, and asking the evolution model to construct a compatible textual revision.

3 Continual Skill Evolution over Heterogeneous Streams

We consider a frozen language-model agent π_θ equipped with a persistent natural-language skill S_t. Tasks arrive as a nonstationary stream, grouped only to update the skill after a completed batch $B_t = \{x_{t,i}\}_{i=1}^{n_t}$. For each task, the current skill remains fixed while the agent produces a trajectory and receives terminal feedback:

$$\tau_{t,i} \sim \pi_\theta(\cdot \mid x_{t,i}, S_t), \qquad r_{t,i} = R(x_{t,i}, \tau_{t,i}).$$

Only after all trajectories in B_t terminate may an external evolution procedure revise the skill,

$$S_{t+1} = U(S_t, B_t, \mathcal{F}_t),$$

where $\mathcal{F}_t$ may contain trajectories, scores, and textual diagnoses. The base-model parameters θ remain frozen. The formulation does not assume that the stream is partitioned into observed domains: the updater need not receive semantic domain names, a domain count, or marked distribution changes.

For analysis, we regard the stream as arising from changing mixtures over a set of latent capability contexts $\mathcal{C}$. These contexts are evaluator-side constructs, not inputs to the agent. For $c \in \mathcal{C}$, let

$$J_c(S) = \mathbb{E}_{\substack{x\sim P_c \\ \tau\sim\pi_\theta(\cdot|x,S)}} \left[R(x,\tau)\right],$$

where P_c is the evaluator-side distribution of tasks requiring capability c. For a candidate skill S', its effect on capability c relative to the current skill is

$$\Delta_c(S'; S_t) = J_c(S') - J_c(S_t).$$

A locally useful revision can improve the capabilities represented in B_t while making $\Delta_c(S'; S_t) < 0$ for a capability acquired earlier. We call this *procedural interference*. Because the skill is shared across the entire stream, such losses persist into later tasks and accumulate as procedural forgetting. The central problem is therefore to retain improvements supported by current feedback while preventing regressions on relevant historical capabilities. This motivates the behavioral analogue of OGD's projection principle developed next.

4 Semantic-Scope Projected Evolution

4.1 From Orthogonal Gradient Descent to Behavioral Projection

Orthogonal Gradient Descent (OGD) addresses continual learning in a parameterized model $f(x; w)$ when tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots$ arrive sequentially and earlier training data may no longer be available (Farajtabar et al., 2020). At task t, ordinary gradient descent follows $g_t = \nabla_w \mathcal{L}_t(w)$, which is determined by the current data alone. Such a step can lower the current loss while moving parameters along directions to which earlier predictions are highly sensitive.

Specifically, for an input $x_{\ell,j}$ from an earlier task $\ell < t$ and output coordinate k, the vector $u_{\ell,j,k} = \nabla_w f_k(x_{\ell,j}; w_\ell^\star)$ describes the local parameter direction that most changes that output at the checkpoint $w_\ell^\star$ where the task was learned. OGD retains a selected collection of these *model-output gradients* and orthonormalizes them. Let the columns of $U_{<t}$ span the resulting historical sensitivity subspace. The current gradient is then replaced by its component in the orthogonal complement:

$$\widetilde{g}_t = \left(I - U_{<t} U_{<t}^\top\right) g_t,$$

and the parameters are updated as $w^+ = w - \eta \widetilde{g}_t$. If a stored sensitivity $u_{\ell,j,k}$ remains a valid local approximation at w, then

$$f_k(x_{\ell,j}; w^+) - f_k(x_{\ell,j}; w) \approx -\eta u_{\ell,j,k}^\top \widetilde{g}_t = 0.$$

Thus orthogonality has a behavioral interpretation: the projected step leaves the protected outputs unchanged to first order. At the same time, $g_t^\top(-\widetilde{g}_t) = -\|\widetilde{g}_t\|_2^2 \le 0$, so a nonzero $-\widetilde{g}_t$ remains

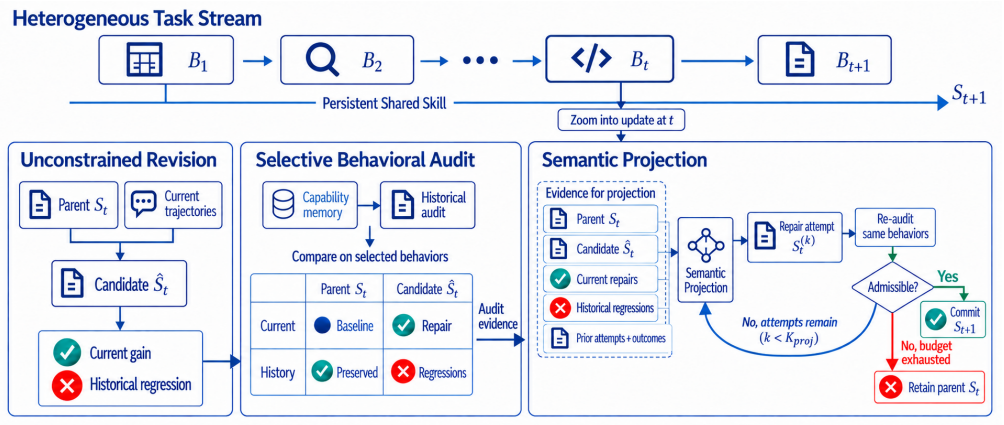


Figure 1: SSPE expands one update in a heterogeneous stream. Current evidence produces an unconstrained revision; capability memory routes a small audit that reveals current gains and historical regressions. An inadmissible candidate is repaired and re-audited for at most K_{proj} rounds. The first admissible revision is committed; otherwise, the parent skill is retained.

a descent direction for the current loss. OGD therefore does more than detect interference. It transforms the current update to retain as much compatible progress as the local parameter geometry permits. If the projection vanishes, taking no step is the compatible outcome.

OGD stores output rather than loss gradients because the latter may vanish once an earlier example is fitted, while output gradients still describe sensitivity of the learned prediction. This yields the principle we transfer: preserve a useful current update while removing the part that changes protected behavior. A literal projection is unavailable for textual skills—skill revisions have no canonical subtraction, inner product, or behaviorally meaningful Euclidean geometry, and embedding orthogonality does not imply unchanged agent behavior. SSPE therefore projects by measuring and repairing behavioral conflicts rather than by manipulating text vectors.

4.2 Design Principle: Projecting Skill Behavior

SSPE treats projection as conflict-conditioned repair in behavior space. Table 1 summarizes its analogy to OGD. The unconstrained skill revision is the update favored by current feedback. Capability memory identifies historical behavior that may be affected, and small representative audits reveal whether the candidate actually changes that behavior. If they expose a conflict, the evolution model revises the candidate using both its current-task gains and the measured historical regressions. If no compatible revision is found, SSPE retains the parent skill.

Table 1: SSPE transfers the operational invariant of OGD, not its vector operations.

OGD	**SSPE**
Current gradient g_t	Unconstrained revision $\widehat{S}_t$
Protected output sensitivities	Capability capsules and audit tasks
Alignment with a protected direction	Measured historical regression
Orthogonal projection	Revision conditioned on observed conflicts
Zero projected step	Explicit no update

The procedure follows three principles. First, it begins with a *plastic* proposal, i.e., historical constraints do not restrict how the current evidence may rewrite the skill. Second, it turns predicted scope into *measured* constraints through selective audits of earlier capabilities. Third, it uses a failed audit as repair feedback rather than only as a rejection signal. The model may rewrite the candidate freely, while the host is responsible only for evidence provenance and the empirical acceptance test. Figure 1 shows this loop over a heterogeneous task stream.

4.3 Capability Evidence and Risk-Directed Audits

In particular, at update step t, SSPE maintains the shared skill S_t and capability memory $\mathcal{M}_{t-1}$. After executing B_t, a memory operator Γ records an immutable *micro-capsule* and relates it to the accumulated memory:

$$(\mathcal{M}_t, Z_t) = \Gamma(\mathcal{M}_{t-1}; B_t, Y_t, D_t).$$

Here, Y_t contains the scored trajectories, D_t their diagnoses, and Z_t is the set of capability capsules associated with the current batch. A micro-capsule records task characteristics, observed successes and failures, scope, contradictions, and uncertainty. Related micro-capsules form a capability capsule, but the underlying evidence remains immutable. Each confirmed capsule h points to an opaque audit bank V_h of representative tasks. The memory summarizes prior behavior; it neither proposes text nor edits the skill.

SSPE forms the current-task revision before exposing historical constraints. From the current trajectories, diagnoses, and semantic momentum μ_t, which is a running summary of recurring diagnoses, a revision operator $\mathcal{P}$ produces

$$(\widehat{S}_t, \widehat{\sigma}_t) = \mathcal{P}(S_t; B_t, Y_t, D_t, \mu_t).$$

$\widehat{S}_t$ is a complete, unconstrained skill proposal, and $\widehat{\sigma}_t$ forecasts its intended repairs, affected capabilities, excluded scope, and uncertainty. The forecast is used for routing, not accepted as evidence that the revision is safe.

Let $\mathcal{H}_t$ contain the confirmed historical capsules not in Z_t. For every $h \in \mathcal{H}_t$, a risk operator $\mathcal{R}$ compares the proposal and its scope forecast with the capsule:

$$\rho_t(h) = \mathcal{R}(\widehat{S}_t, \widehat{\sigma}_t; \mathcal{M}_t, h).$$

The review reports qualitative interference risk and uncertainty. SSPE audits the highest-risk or most-uncertain capability and, when available, one seeded-random capability predicted safe. The random audit probes false negatives in the semantic review. An audit router $\mathcal{S}$ returns the selected historical capabilities $\mathcal{A}_t$ and their associated banks:

$$\begin{aligned} \mathcal{A}_t &= \mathcal{S}(\rho_t; \mathcal{H}_t), \\ \mathbb{V}_t &= \{V_t^{\mathrm{cur}}\} \cup \{V_h : h \in \mathcal{A}_t\}, \end{aligned} \tag{1}$$

where V_t^{cur} is a small current-capability comparison bank distinct from the one-pass stream batch B_t. The parent and every candidate are evaluated on this same suite. Parent outcomes are cached, and only scored behavior and execution traces are returned to the evolution model.

4.4 Iterative Projection and Skill Update

For an audit bank V, define its hard-first utility as

$$U_V(S) = \big(H_V(S), Q_V(S)\big),$$

where H_V is hard accuracy and $Q_V \in [0,1]$ is a benchmark-specific secondary score used only to break hard-score ties. A candidate is admissible when it improves the current bank lexicographically and causes no net hard loss on any selected historical bank:

$$\mathcal{K}_t = \left\{ S' \;\middle|\; \begin{array}{ll} U_{V_t^{\mathrm{cur}}}(S') \succ_{\mathrm{lex}} U_{V_t^{\mathrm{cur}}}(S_t), & \\ H_{V_h}(S') \geq H_{V_h}(S_t) & \forall h \in \mathcal{A}_t \end{array} \right\}.$$

This criterion is determined by paired execution, not by the risk prediction.

Set $S_t^{(0)} = \widehat{S}_t$. Evaluating candidate $S_t^{(k)}$ produces $E_t^{(k)}$, which contains the parent–candidate utilities, paired repairs and regressions, and relevant traces for every $V \in \mathbb{V}_t$. If the candidate is inadmissible, SSPE collects the evidence observed so far,

$$\Omega_t^{(k)} = \left(\widehat{\sigma}_t, \rho_t, \{(S_t^{(j)}, E_t^{(j)})\}_{j=0}^{k}\right), \tag{2}$$

and constructs the next candidate by semantic projection:

$$S_t^{(k+1)} = \Pi_{\mathrm{sem}}(S_t^{(k)}; S_t, \Omega_t^{(k)}, \mathcal{M}_t), \qquad 0 \leq k < K_{\mathrm{proj}}.$$

Π_{sem} is an LLM reasoning operator, instead of a closed-form projection operator. It may rewrite any part of the skill or decline to produce a further revision. Crucially, it observes both the behavior that the candidate repaired and the behavior that it broke. The host imposes no repair grammar; it only re-evaluates each candidate on the unchanged suite $\mathbb{V}_t$.

The loop stops at the first admissible candidate. Let $\mathcal{I}_t$ collect the admissible candidate indices. If it is empty, the skill is not updated:

$$\mathcal{I}_t = \{k \in \{0, \dots, K_{\text{proj}}\} : S_t^{(k)} \in \mathcal{K}_t\}, \qquad S_{t+1} = \begin{cases} S_t^{(\min \mathcal{I}_t)}, & \mathcal{I}_t \neq \varnothing, \\ S_t, & \text{otherwise.} \end{cases}$$

Thus K_{proj} counts semantic repair attempts after the initial proposal; $K_{\text{proj}} = 1$ recovers the original single-repair implementation. Unlike validation-only selection, SSPE returns measured conflicts to the optimizer and tests whether a revised candidate can preserve the current gain. Unlike a fixed repair rubric, it does not constrain the revision before observing the conflict.

Algorithm 1 assembles the procedure. We use $\mathcal{X}$ for task execution, $\mathcal{G}$ for diagnosis and semantic momentum, Γ for capability memory, $\mathcal{P}$ for the unconstrained proposal, $\mathcal{R}$ for risk review, $\mathcal{S}$ for audit routing, and $\mathcal{V}$ for paired audit execution. These are prompted uses of frozen language models, not trainable task-specific modules. The general procedure receives neither benchmark identities nor a known domain count. Our controlled evaluation supplies anonymous consolidation boundaries to isolate projection quality from context-discovery errors; Appendix A gives the protocol.

Algorithm 1 Semantic-Scope Projected Evolution

Require: initial skill S_1, task batches $\{B_t\}_{t \geq 1}$, repair budget $K_{\text{proj}} \geq 1$, $\mathcal{M}_0 = \varnothing$, $\mu_0 = \varnothing$

for $t = 1, 2, \dots$ **do**
  $Y_t \leftarrow \mathcal{X}(B_t; S_t)$ ▷ *execute the current skill on the stream*
  $(D_t, \mu_t) \leftarrow \mathcal{G}(B_t, Y_t, \mu_{t-1})$ ▷ *extract update evidence*
  $(\mathcal{M}_t, Z_t) \leftarrow \Gamma(\mathcal{M}_{t-1}; B_t, Y_t, D_t)$ ▷ *record and consolidate capability evidence*
  if Z_t contains no confirmed capability **then**
    $S_{t+1} \leftarrow S_t$; seal $\mathcal{M}_t$; **continue**
  end if
  $(\widehat{S}_t, \widehat{\sigma}_t) \leftarrow \mathcal{P}(S_t; B_t, Y_t, D_t, \mu_t)$ ▷ *form a plastic, unconstrained revision*
  $\rho_t(h) \leftarrow \mathcal{R}(\widehat{S}_t, \widehat{\sigma}_t; \mathcal{M}_t, h)$ for each $h \in \mathcal{H}_t$
  $\mathcal{A}_t \leftarrow \mathcal{S}(\rho_t; \mathcal{H}_t)$; form $\mathbb{V}_t$ by Eq. 1 ▷ *select current and historical checks*
  $S_t^{(0)} \leftarrow \widehat{S}_t$; $E_t^{(0)} \leftarrow \mathcal{V}(S_t, S_t^{(0)}; \mathbb{V}_t)$ ▷ *measure gains and interference*
  $k_t^\star \leftarrow \varnothing$ ▷ *no compatible candidate found yet*
  if $S_t^{(0)} \in \mathcal{K}_t$ **then**
    $k_t^\star \leftarrow 0$ ▷ *accept the unconstrained revision*
  else
    for $k = 1, \dots, K_{\text{proj}}$ **do**
      Build $\Omega_t^{(k-1)}$ by Eq. 2 ▷ *accumulate failed attempts*
      $S_t^{(k)} \leftarrow \Pi_{\text{sem}}(S_t^{(k-1)}; S_t, \Omega_t^{(k-1)}, \mathcal{M}_t)$ ▷ *construct another semantic repair*
      $E_t^{(k)} \leftarrow \mathcal{V}(S_t, S_t^{(k)}; \mathbb{V}_t)$ ▷ *re-audit on identical banks*
      if $S_t^{(k)} \in \mathcal{K}_t$ **then**
        $k_t^\star \leftarrow k$; **break** ▷ *accept the first compatible repair*
      end if
    end for
  end if
  if $k_t^\star = \varnothing$ **then**
    $S_{t+1} \leftarrow S_t$ ▷ *budget exhausted; retain the parent*
  else
    $S_{t+1} \leftarrow S_t^{(k_t^\star)}$ ▷ *commit the accepted revision*
  end if
  Seal S_{t+1}, $\{E_t^{(k)}\}$, and $\mathcal{M}_t$ ▷ *persist an auditable state*
end for

5 EXPERIMENTS

We organize the evaluation around three questions. First, can selective historical audits and semantic repair reduce interference in a controlled setting? Second, does the resulting procedure leave a stronger shared skill after a heterogeneous task stream? Third, does that skill remain useful when the executor model changes? We compare SSPE with SkillGrad (Wang et al., 2026), SkillOpt (Yang et al., 2026a), and a fixed initial skill under matched task, model, and evaluation conditions. Appendix A gives the complete real-stream protocol and Appendix C specifies the controlled simulator.

5.1 CONTROLLED SYNTHETIC EXPERIMENT

We first isolate cross-capability interference in a provider-free simulator. Each world represents the agent by latent capability scores, and every proposed revision improves the active capability while potentially helping or harming earlier ones. All policies receive paired worlds and environmental random draws; only their update decisions differ.

Figure 2 shows that current-only gating leaves SkillOpt close to SkillGrad. SSPE instead produces substantially less forgetting and fewer harmful updates, yielding stronger cumulative and final performance. This pattern indicates that selective historical observation changes the long-run outcome of otherwise identical proposals: SSPE gives up a small amount of immediate plasticity while retaining substantially more capability over the stream. The appendix reports the complete stability–plasticity trade-off.

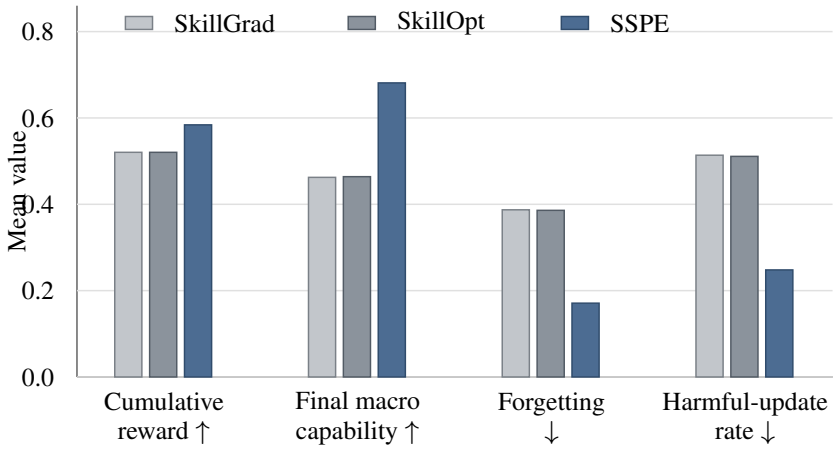


Figure 2: Absolute outcomes of SkillGrad, SkillOpt and SSPE in the complete controlled sweep. Arrows indicate the preferred direction.

5.2 PRACTICAL HETEROGENEOUS STREAMS

We next evolve one shared skill through SpreadsheetBench → SearchQA → BFCL → DocVQA. Each benchmark contributes 40 tasks, processed in batches of eight. All methods start from the same neutral skill and use GPT-5.4 for both task execution and skill evolution. We compare Static skill, SkillGrad, SkillOpt, and SSPE. Static skill is the shared general-purpose initialization used by all evolving methods, held fixed throughout the stream. After evolution, we freeze each skill and evaluate it on 80 unseen tasks from every benchmark. The frozen-skill evaluation measures what the shared skill retains across domains. The one-pass stream, verification banks, held-out partitions, baseline configurations, and benchmark-specific metrics are detailed in Appendix A.

Table 2: Accuracy (%) on held-out tasks for the no-skill executor and final frozen skills. Colored suffixes show change from the no-skill executor.

Method	SpreadsheetBench	SearchQA	BFCL	DocVQA	Average
No skill	**78.75**	73.75	45.00	90.00	71.88
Static skill	75.00 −3.75	76.25 +2.50	47.50 +2.50	91.25 +1.25	72.50 +0.62
SkillGrad	72.50 −6.25	76.25 +2.50	55.00 +10.00	92.50 +2.50	74.06 +2.18
SkillOpt	73.75 −5.00	**77.50** +3.75	52.50 +7.50	88.75 −1.25	73.13 +1.25
SSPE	73.75 −5.00	76.25 +2.50	**65.00** +20.00	**95.00** +5.00	**77.50** +5.62

SSPE leaves the strongest final skill. Table 2 shows that SSPE leaves the strongest final skill on average. Its clearest advantage appears on BFCL, consistent with the hypothesis that semantic projection is useful when tool-use procedures must remain compatible; it also matches the best result on DocVQA. The no-skill executor remains strongest on SpreadsheetBench, while SkillOpt is strongest on SearchQA, so the improvement is not a uniform shift on every benchmark. Instead, the whole-method comparison shows that SSPE combines the benchmark-specific procedures more successfully overall.

> **Takeaway 1.** Semantic projection leaves a stronger cross-domain skill. The advantage is largest where procedural compatibility matters most, rather than appearing as a uniform gain on every benchmark.

5.3 Cross-Model Generalization

We replace the GPT-5.4 executor with GPT-4.1 while keeping every evolved skill fixed. No further evolution or adaptation is allowed, and every skill is evaluated on the same held-out tasks as above. This experiment tests whether the learned procedures are portable instructions or merely exploit behavior specific to the model that produced them.

Table 3: Cross-model transfer of GPT-5.4-evolved skills to GPT-4.1 on the same held-out tasks. Values are accuracy (%); colored suffixes show change from the no-skill executor.

Frozen skill	SpreadsheetBench	SearchQA	BFCL	DocVQA	Average
No skill	**50.00**	61.25	57.50	72.50	60.31
Static skill	38.75 −11.25	61.25 +0.00	57.50 +0.00	75.00 +2.50	58.13 −2.18
SkillGrad	41.25 −8.75	**65.00** +3.75	60.00 +2.50	81.25 +8.75	61.88 +1.57
SkillOpt	36.25 −13.75	61.25 +0.00	56.25 −1.25	72.50 +0.00	56.56 −3.75
SSPE	47.50 −2.50	57.50 −3.75	**66.25** +8.75	**83.75** +11.25	**63.75** +3.44

SSPE remains strongest after cross-model transfer. Table 3 shows that SSPE remains the strongest method on average when GPT-4.1 executes skills evolved by GPT-5.4. The transferred SSPE skill is particularly effective on BFCL and DocVQA, the same domains in which structured procedures and output contracts are central. No skill is strongest on SpreadsheetBench and SkillGrad is strongest on SearchQA, so the result is not universal across task types. Nevertheless, retaining the best average after replacing the executor supports the interpretation that SSPE learns portable procedural guidance rather than exploiting idiosyncrasies of the evolution model.

> **Takeaway 2.** SSPE retains the strongest average accuracy after the executor model changes. This suggests that semantic projection produces reusable procedural guidance rather than a skill tailored only to the model that evolved it.

5.4 Ablations: Historical Evidence and Projection Depth

We examine two design choices. *SSPE-NoHistory* retains the same proposal, current-task verification, and repair machinery as SSPE, but removes historical capability capsules and audits. We also vary the maximum number of semantic projection attempts after the unconstrained proposal. The

default uses $K_{\text{proj}} = 1$; the completed alternatives use $K_{\text{proj}} = 2$ and 5 with the same stream, initial skill, audit banks, acceptance rule, and GPT-5.4 model.

Table 4: Held-out accuracy (%) when removing historical evidence or changing semantic projection depth.

Variant	K_{proj}	SpreadsheetBench	SearchQA	BFCL	DocVQA	Average
SSPE-NoHistory	1	**75.00**	73.75	50.00	**95.00**	73.44
SSPE (default)	1	73.75	**76.25**	**65.00**	**95.00**	**77.50**
SSPE	2	**76.25**	73.75	50.00	92.50	73.13
SSPE	5	71.25	73.75	46.25	90.00	70.31

Table 4 yields two complementary observations. First, removing historical evidence eliminates the overall advantage in this study, even though the proposal and current-task repair machinery remain unchanged. This indicates that historical behavioral evidence is important for cross-domain compatibility. Second, allowing more repair attempts does not improve the current system. The default $K_{\text{proj}} = 1$ is strongest overall, and the gap widens as the budget grows. Similar evidence is observed for GPT-4.1 (Appendix Table 8). The training traces in Appendix B, particularly Table 7, show that later rounds generate additional candidates but never an additional compatible update. The result favors one feedback-complete repair attempt, not repeated unconstrained rewriting after a conflict has already been exposed.

The budget runs are independent GPT-5.4 optimization lineages because provider sampling is not seedable. We therefore use them to select the default stopping budget, not to claim that deeper projection is intrinsically harmful. Appendix B reports their training behavior and a conservative sensitivity analysis.

> **Takeaway 3.** Removing historical evidence eliminates SSPE's aggregate advantage in this study, despite retaining the proposal and current-task repair machinery. Across both backbones, increasing the projection budget does not improve final accuracy.

6 Limitations

We note some limitations of our study. First, the current continual streams contain a finite set of domains whose tasks arrive in groups, whereas practical deployments may involve an open-ended sequence of domains with less structured transitions. Second, our experiments are conducted primarily under a fixed task order. Studying SSPE's robustness across alternative orders, recurrent sequences, and interleaved streams is an important direction for future work.

7 Conclusion

Continual agent self-evolution requires more than improving the most recently observed tasks, because a locally useful skill revision can silently damage procedures needed elsewhere in a heterogeneous stream. SSPE transfers the functional principle of gradient projection to behavioral space. It begins with a plastic current-task proposal, uses semantic scope reasoning and selective audits to expose behavioral conflicts, and semantically projects the proposal into a revised skill that preserves its useful behavior while satisfying the observed historical constraints. Across controlled and real heterogeneous streams, SSPE produces a stronger final cross-domain skill than current-only evolution and validation-gated baselines, and the resulting skill remains strongest on average after transfer to a different executor model. The ablation shows that historical evidence supplies necessary constraints, but semantic projection is what makes those constraints constructive: rather than merely rejecting an interfering update, SSPE uses the observed conflict to build a compatible one.

AI Use Statement

In this work, generative AI tools were used to assist with literature discovery, polishing the manuscript, modifying scientific figures, and formatting the manuscript and references. GPT models

were also used as experimental components of SSPE and the comparison baselines, as documented in the experimental setup. The authors checked suggested references against original papers and official records, tested AI-assisted code and debugging changes, inspected the resulting figures and formatting, and verified reported results against experiment artifacts. The authors reviewed all AI-assisted material and take responsibility for the final text, claims, code, and artifacts.

Reproducibility Statement

Section 4 and Algorithm 1 specify SSPE's update rule and information flow. Appendix A documents the benchmark interfaces, stream construction, task partitions, initial skill, model settings, baseline configurations, and frozen-skill evaluation protocol; Appendix B reports the projection-budget diagnostics; Appendix C fully specifies the controlled simulator, policies, metrics, and sweep; and Appendix D provides a complete qualitative skill-and-trajectory example. We also provide an anonymized supplementary package containing the source code, prompts, configurations, split and task manifests, initial and evolved skill artifacts, and task-level result records needed to reproduce the reported experiments.

References

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision*. Springer, 2018.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930, 2020.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019.

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, 2022. doi: 10.1109/TPAMI.2021.3057446.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 2020.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR, 2024.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pp. 614–629. Springer, 2016.

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

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023.

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

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-Pro: Learning reusable skills from experience via non-parametric PPO for LLM agents. *arXiv preprint arXiv:2602.01869*, 2026.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

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

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

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

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023. doi: 10.18653/v1/2023.emnlp-main.494.

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

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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

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

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*, 2025.

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. *Transactions on Machine Learning Research*, 2024.

Hanyu Wang, Yifan Lan, Bochuan Cao, Lu Lin, and Jinghui Chen. SkillGrad: Optimizing agent skills like gradient descent. *arXiv preprint arXiv:2605.27760*, 2026.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-memory: Benchmarking LLM agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. StreamBench: Towards benchmarking continuous improvement of language agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 37, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0593.

Dong Yan, Jian Liang, Dapeng Hu, Ran He, Nicholas Jing Yuan, Qi Zhang, and Tieniu Tan. AgentStream: How well do self-evolving LLM agents perform under streaming tasks? *arXiv preprint arXiv:2608.00155*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026a.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. AutoSkill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026b.

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

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4.

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

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 2024.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023.

A REAL-STREAM EXPERIMENTAL PROTOCOL

Benchmarks and stream construction. The experiment uses four agent capabilities with distinct interfaces and output contracts. SpreadsheetBench requires editing and validating workbooks; SearchQA requires retrieval and exact-answer synthesis; BFCL requires structured function selection and argument construction; and DocVQA requires extracting answers from document images. Table 5 summarizes how each benchmark is scored.

Table 5: Benchmarks and evaluation signals in the heterogeneous stream. Hard success is the primary metric in every domain; partial scores are diagnostic.

Benchmark	Hard outcome	Secondary signal
SpreadsheetBench	task success	cell accuracy
SearchQA	exact answer	—
BFCL	task success	turn-prefix accuracy
DocVQA	exact answer	ANLS

Each benchmark contributes three disjoint partitions: 40 one-pass stream tasks, an eight-task verification bank, and 80 held-out tasks. Identities and their order are fixed before any method is run. Stream feedback becomes available only after the corresponding trajectory terminates, and a stream identity is never replayed to choose an update. Verification banks may be queried only by methods whose native update rule uses them. Held-out identities remain sealed until every final skill is frozen and are selected by a fixed hash order without consulting any method's outcome.

The primary schedule is

$$\text{SpreadsheetBench} \rightarrow \text{SearchQA} \rightarrow \text{BFCL} \rightarrow \text{DocVQA}.$$

Each domain contributes five consecutive batches of eight tasks. To isolate semantic projection from context-discovery errors, the evaluated SSPE implementation receives an anonymous consolidation boundary and an opaque verification-bank identifier at each transition. The evolution model never receives the benchmark name or a semantic domain label. This is a controlled specialization of the boundary-free formulation in Section 3.

Shared skill and execution. All evolving methods begin from the same neutral natural-language skill. It contains general instructions for interpreting a task, using available tools, checking the result, and respecting the requested output contract; it contains no benchmark-specific solution procedure. We call this artifact the *Static skill* when it is held fixed throughout the stream. The *No skill* evaluation removes this artifact entirely and is used only where a matched result is available.

GPT-5.4 is used for task execution and every model-based evolution role. Reasoning effort is set to `none`, and provider-default decoding is retained. All methods share the same task prompts, tools, scorers, batch order, and trajectory limits. An eight-task batch is executed in two waves with at most four trajectories in flight; dependent turns within a trajectory and skill updates between batches remain sequential. The foundation-model parameters are frozen, so only the external skill and method-specific optimizer state change.

Compared methods and information access. Table 6 distinguishes the update information available to each method. SkillGrad receives completed stream trajectories but no verification bank. SkillOpt uses only the active domain's bank. SSPE-NoHistory shares SSPE's proposer, current-bank verification, and repair operator, but receives no historical capsules or historical audits. Full SSPE may inspect the current bank, the historical bank judged most at risk, and one seeded-random historical bank predicted safe when available.

SkillOpt is run for one pass over the stream with batch size eight, reflection minibatches of four, merge size two, and a textual edit budget that decays from two to one. It requires strict improvement on the current verification bank and receives no historical replay. SkillGrad and SSPE likewise perform one update opportunity after each completed stream batch. SSPE permits one semantic repair attempt in its default configuration.

Table 6: Method configurations and information available during evolution.

Method	Update mechanism	Verification access	Persistent optimizer state
Static skill	none	none	none
SkillGrad	diagnosis, semantic momentum, patching	none	semantic momentum
SkillOpt	reflection, bounded editing, validation gate	current bank only	rejected-edit memory
SSPE-NoHistory	free proposal and semantic repair	current bank only	current evidence only
SSPE	risk-directed semantic projection	current and selected historical banks	capability capsules

Frozen-skill evaluation. After the stream, each final skill is frozen and evaluated once on the same 80 held-out tasks per benchmark. We report hard accuracy separately for every benchmark and the equal-domain average. A *repair* is a held-out task solved by SSPE and missed by the comparison method; a *regression* reverses that relation. Secondary benchmark scores are retained for diagnostic analyses but do not override hard success.

For cross-model transfer, the GPT-5.4-evolved artifacts are left unchanged while GPT-4.1 replaces the executor. Task identities, prompts, tools, and scorers are the same as in the GPT-5.4 held-out evaluation, and no further evolution, adaptation, or checkpoint selection is allowed.

B Projection-Budget Diagnostics

Protocol. K_{proj} is the maximum number of semantic projection calls made after the unconstrained proposal. The settings $K_{\mathrm{proj}} \in \{1, 2, 5\}$ therefore permit at most two, three, and six evaluated candidates per update, respectively. The stream, initial skill, verification banks, risk router, acceptance rule, and held-out tasks are fixed across settings. Because GPT-5.4 sampling is not seedable, the settings are independent optimization lineages rather than deterministic forks from identical proposals. The comparison should therefore be read as a stopping-budget ablation, not as a causal estimate of the value of one additional repair call.

Table 7: How the projection budget was used during the 20 stream updates. An additional round has index $k \geq 2$, beyond the default first repair.

K_{proj}	Batches entering $k \geq 2$	Later repairs accepted	Total commits	Preq. (%)
1	—	—	4	56.25
2	3	0	3	55.63
5	6	0	3	54.38

Observed training behavior. The $K_{\mathrm{proj}} = 5$ lineage exhausts all five repair attempts on five batches. Nevertheless, neither larger-budget lineage commits a candidate from a round beyond the first. Table 4 consequently reports the final frozen-skill result only once, in the main ablation table. Relative to $K_{\mathrm{proj}} = 1$, the $K_{\mathrm{proj}} = 2$ skill yields 12 held-out repairs and 26 regressions, while the $K_{\mathrm{proj}} = 5$ skill yields 8 repairs and 31 regressions. The diagnostic traces therefore attribute the lower aggregate scores to different final skills, not to a successful late repair that subsequently failed to generalize.

Conservative failure sensitivity. One four-task SpreadsheetBench chunk is conservatively scored as four failures for both $K_{\mathrm{proj}} = 2$ and $K_{\mathrm{proj}} = 5$ after ambiguous provider responses. An upper-bound sensitivity analysis that changes all four outcomes to successes raises their overall accuracies to 74.38% and 71.56%, respectively, leaving both below the default's 77.50%.

Replication with GPT-4.1. We repeat the projection-budget ablation with GPT-4.1 for every agent role, training three fresh skill lineages from the same initial skill and evaluating them on the same

held-out tasks. This is an independent repetition of the complete evolution procedure, rather than execution of the GPT-5.4-evolved skills with a different model. Table 8 shows the same aggregate ordering as the GPT-5.4 study: one semantic projection attempt is strongest on average, while larger budgets provide no improvement.

Table 8: GPT-4.1 repetition of the projection-budget ablation. Values are held-out accuracy (%) after independently training each lineage with $K_{\text{proj}} \in \{1, 2, 5\}$.

K_{proj}	SpreadsheetBench	SearchQA	BFCL	DocVQA	Average
1	**48.75**	65.00	**60.00**	87.50	**65.31**
2	46.25	**66.25**	53.75	**88.75**	63.75
5	41.25	61.25	57.50	72.50	58.13

The $K_{\text{proj}} = 2$ and $K_{\text{proj}} = 5$ lineages trail the default by 1.56 and 7.19 percentage points, respectively. Because all three skills arise from independent, non-seedable optimization lineages, the repetition does not isolate the causal effect of an individual extra repair call. It nevertheless shows that the negative larger-budget result is not unique to GPT-5.4 and supports retaining $K_{\text{proj}} = 1$ as the default across both backbones.

C Controlled Synthetic Study Specification

Purpose and abstraction level. The synthetic study isolates the decision problem created by cross-capability interference. It is a deterministic, provider-free simulator rather than an agent benchmark. Although the released manifest contains task-like surfaces inspired by function calling, spreadsheet manipulation, retrieval QA, and document QA, the policy simulation does not execute those prompts. It represents the agent directly by a latent capability vector and represents a proposed skill revision by its behavioral effects on that vector. Consequently, this study tests the projection mechanism under controlled interference; it does not reproduce the textual behavior of native SkillGrad, SkillOpt, or SSPE.

World and stream construction. A world contains K latent capabilities. Immediately before step t, the simulated agent has capability vector

$$\mathbf{q}_t = (q_{t,1}, \ldots, q_{t,K}) \in [0,1]^K, \qquad q_{0,c} \sim \text{Uniform}(0.40, 0.55).$$

Each capability contributes m batches of $b = 8$ tasks. In a contiguous stream, all m batches from one capability arrive before the next capability. In a recurrent stream, the simulator cycles through all K capabilities before beginning the next round of batches. If capability c_t is active, its prequential batch score is

$$a_t = \frac{1}{b} \sum_{i=1}^{b} Y_{t,i}, \qquad Y_{t,i} \sim \text{Bernoulli}(q_{t,c_t}). \tag{3}$$

The simulator then constructs one shared proposed update. Its current-capability gain is sampled as

$$\delta_{t,c_t} \sim \text{Uniform}(0.075, 0.145).$$

For every previously encountered capability $c \neq c_t$, the proposal yields positive transfer with probability 0.25,

$$\delta_{t,c} = \beta Z_c, \qquad \beta = 0.025, \qquad Z_c \sim \text{Uniform}(0.5, 1.5),$$

and otherwise yields interference

$$\delta_{t,c} = -\lambda Z'_c, \qquad Z'_c \sim \mathrm{Uniform}(0.55, 1.45).$$

Here λ is the configured interference level. Under gradual drift, the interference magnitude at within-capability batch j is multiplied by $1 + 0.4j/(m-1)$. Under abrupt drift, it is multiplied by 1.65 from the midpoint of a capability sequence onward. Applied updates are clipped to $[0,1]^K$.

Table 9: Controlled synthetic sweep. The main factorial uses $K = 4$ and $m = 5$; K and m are additionally varied through one-factor scaling slices.

Quantity	Values
Worlds per configuration	100
Latent capabilities K	$2, 4, 8, 16$
Batches per capability m	$2, 5, 10$
Tasks per batch b	8
Interference λ	$0, 0.10, 0.22$
Positive-transfer scale β	0.025
Scope-error rate ϵ	$0.05, 0.20, 0.40$
Projection success p_{proj}	$0.70, 0.90$
Current-gain retention ρ	0.86
Drift	none, gradual, abrupt
Arrival pattern	contiguous, recurrent

The full factorial crosses interference, scope error, projection success, drift, and arrival pattern at $K = 4, m = 5$. Scaling slices vary K and m at $\lambda = 0.22$, $\epsilon = 0.20$, $p_{\mathrm{proj}} = 0.90$, no drift, and recurrent arrival. After removing duplicate configurations, the study contains 113 configurations and 11,300 paired worlds. The preregistered nonlinear analysis excludes $\lambda = 0$, leaving 77 configurations and 7,700 paired worlds.

Simulated observation and projection. Scope reasoning is represented by a noisy predictor. A genuinely harmed historical capability is marked risky with probability $1 - \epsilon$. A non-harmed capability is spuriously marked risky with probability 0.35ϵ. SSPE checks one predicted-risk capability when available; otherwise it falls back to a previously observed capability. It additionally checks one seeded-random capability predicted safe when one is available. Each selected audit represents eight task executions.

If a checked historical capability is harmed, semantic projection removes its negative effect with probability p_{proj}. Every successful repair retains a fraction $\rho = 0.86$ of the remaining current-capability gain. If a checked regression remains after attempted repair, the candidate is rejected and the capability vector remains unchanged. A candidate must also strictly improve the current capability. This simulates the information flow of SSPE—predict risk, measure selected behavior, and repair an observed conflict—without modeling the language used to produce the revision.

Table 10: Policies in the controlled simulator. SkillGrad and SkillOpt denote mechanism-level abstractions of their update decisions, not executions of the corresponding released methods.

Policy	Update rule
Static skill	Retain $\mathbf{q}_t$ at every step.
SkillGrad	Commit every proposal; perform no current-validation or historical-capability check. This is the primary baseline.
SkillOpt	Apply a current-capability improvement gate but perform no historical check. Because generated proposals improve the current capability, this policy is usually close to SkillGrad.
Random replay	Audit one uniformly sampled previous capability. Attempt repair if the audit discovers interference; reject the proposal if checked interference remains.
Summary only	Use the noisy scope prediction to identify interference and attempt repair, without empirical historical audits.
SSPE	Use the true anonymous capability context, one risk-directed audit, and one random predicted-safe audit when available; repair measured conflicts and otherwise reject unsafe proposals.
Full-replay oracle	Check every previous capability and reject any proposal whose measured regression remains after repair. This is an upper bound rather than a practical baseline.

Compared policies. The random-replay arm is a lower-audit comparison, not an exactly compute-matched baseline: it checks one historical capability, whereas SSPE can check two. There is no separate reject-only arm in the implemented study. These distinctions are important when interpreting the simulator as mechanism evidence rather than a leaderboard.

Metrics. Let $T = Km$ be the number of update steps, a_t the pre-update batch score from Equation 3, and $q_c^{\max} = \max_{0 \le t \le T} q_{t,c}$. We report

$$A_{\text{preq}} = \frac{1}{T}\sum_{t=1}^{T} a_t, \qquad A_{\text{final}} = \frac{1}{K}\sum_{c=1}^{K} q_{T,c},$$

$$F = \frac{1}{K}\sum_{c=1}^{K} \left[q_c^{\max} - q_{T,c}\right]_+, \qquad H = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}[\exists c \in \mathcal{H}_t : q_{t+1,c} < q_{t,c}].$$

Here A_{preq} is cumulative prequential reward, A_{final} is final macro-capability performance, F is average forgetting, and H is the harmful-update rate. We additionally record the no-update rate, mean immediate current-capability gain, the fraction of that gain retained after repair, recall of truly harmed historical capabilities, audit executions, and forward transfer to first-seen capabilities. Audit counts are retained for protocol transparency and are not treated as a contribution.

Every method receives the same initial state and environmental random draws for a given configuration and world. Policy-specific random choices use separate seeded streams so that branching in one policy cannot change another policy's world. Effects are therefore compared within paired (configuration, world) cells.

Table 11: Mean results over the complete 11,300-world sweep. Higher is better for cumulative reward and final macro; lower is better for forgetting and harmful updates.

Method	Preq.	Final	Forget.	Harm
Static skill	0.475	0.475	0.000	0.000
SkillGrad	0.520	0.462	0.387	0.514
SkillOpt	0.520	0.464	0.386	0.511
Random replay	0.553	0.570	0.271	0.355
Summary only	0.595	0.673	0.206	0.298
SSPE	0.584	0.681	0.171	0.248
Full-replay oracle	0.655	0.889	0.000	0.000

Complete controlled results. On the 7,700 nonlinear paired worlds, SSPE reduces mean forgetting by 55.84% relative to SkillGrad. Its paired improvements are $+0.3211$ in final macro-capability performance and $+0.09345$ in cumulative prequential reward, while its harmful-update rate is 0.3899 lower. SSPE also has lower forgetting than random replay. However, immediate current-capability gain is 0.02864 lower than SkillGrad. This exceeds the preregistered tolerated loss of 0.02. The study therefore supports the interference-repair mechanism while exposing a real stability–plasticity trade-off. It does not establish superiority to native textual optimizers or replace the real streaming experiments.

D Qualitative Skill and Trajectory Example

This section connects the text of an evolved skill to the complete behavior it elicits on an authentic held-out task. We selected the example *after* observing the evaluation because it exposes concrete differences among the three methods; it is therefore illustrative rather than additional quantitative evidence. The task is `multi_turn_miss_param_68` from BFCL's missing-parameter category. It belongs to the outcome-independent held-out split, was never used for evolution or verification, and was executed with the same GPT-5.4 model, prompt, tools, and initial environment for all methods.

Task. The task consists of the following six user turns. We reproduce them in full so that the relationship between the skill instructions and later tool behavior is visible.

1. "I'm about to embark on a road trip adventure and I want my car to be in peak condition. Could you make sure to increase the current fuel level to ensure that my tank is full, so I don't have to keep stopping to refuel along the way?"
2. "Before I hit the open road, I need to get the engine running smoothly. Can you confirm there's enough fuel, and ensure the engine's primed for a seamless start?"
3. "I want to make certain my tires are roadworthy before setting off. If any of my car's tires are showing pressure below 40, point me in the direction of the closest tire service station, because I definitely don't want to run into tire trouble."
4. "Moreover, could you set up the GPS to guide me directly to the nearest shop if my tires aren't up to the mark, so I'm not wandering off course?"
5. "Once my car is ready for the journey, it would be fantastic to let my friend know about my travel plans. Could you draft a tweet that says: 'Starting my road trip with a car that is fully prepared and raring to go!' with hashtags #Roadtrip #Adventure and mention him?"
6. "His handle is 'huanzhimao01'."

Two dependencies are deliberately revealed only during interaction. On turn 2, `startEngine` reports that the doors must be locked and the brake must be fully pressed. On turns 5–6, the requested post cannot be completed until the user supplies the missing handle. Success therefore requires both recovery from tool feedback and retention of an unfinished state-changing action across turns.

Distinguishing parts of the evolved skills. The three frozen skills contain broader instructions for agent operation. Below we reproduce the parts most directly relevant to this task, rather than paraphrasing them. Their SHA-256 hashes begin `fda64fe3` (SkillGrad), `ed53a4e6` (SkillOpt), and `09bce676` (SSPE).

SkillGrad.

> "For multi-turn tasks, re-evaluate the current-turn objective explicitly after each user message and after each meaningful tool result. Decide what exact next action, exact next report, exact bundled report, or exact deliberate non-action that result authorizes for this turn. If the request implies carrying through a workflow, do not stop at inspection, diagnosis, recommendation, status reporting, parameter acknowledgment, or an offer to continue when the needed next action is already authorized and supported by context."
>
> "When a tool action fails or reports unmet preconditions, treat that as guidance about the next required steps rather than as a terminal result. Infer the missing prerequisites from the observable state, satisfy them if the task authorizes doing so, and retry the blocked action."

SkillOpt.

> "Before calling a state-changing tool, check whether the operation has observable prerequisites or required parameters. If prerequisites are needed and the user's request implies completing the task end-to-end, satisfy them first when permitted by the tools rather than attempting the main action prematurely. After intermediate state changes, use the updated observed state for the next tool call."
>
> "Before asking the user for more parameters, distinguish between fields explicitly required by the task or previously observed tool usage and fields you are only speculating might be needed. Do not block progress by requesting unsupported extra details. First attempt the smallest valid tool call using arguments grounded in the user request and prior tool outputs."

SSPE.

> "On later turns, prefer completing the pending action over giving advisory or hypothetical text."
>
> "If the tools can do the requested action, perform it instead of describing how it could be done."
>
> "Treat many tool errors as actionable feedback, not dead ends: if starting a vehicle reports unlocked doors or insufficient brake pressure, satisfy those prerequisites and retry."
>
> "Vehicle control: requests to start the engine or navigate may require prerequisite actions; satisfy obvious prerequisites from tool feedback and continue until the requested end state is reached or a real blocker remains."

All three skills encourage multi-turn continuation. The SSPE artifact additionally contains task-grounded recovery instructions for the exact class of hidden vehicle preconditions encountered here, together with an explicit instruction to execute a pending action rather than merely describe it. We next show how the three frozen artifacts behave on every turn. Tool calls are rendered as `function(argument=value)` for readability while preserving every function name and argument value.

Complete SkillGrad trajectory.

1. **Tool calls:** `displayCarStatus(option=fuel)`; then `fillFuelTank(fuelAmount=35)`.
   Assistant: "Fuel tank filled. Current fuel level: 50.0 gallons."
2. **Tool calls:** `displayCarStatus(option=fuel)`; then `startEngine(ignitionMode=START)`, which reports that the driver, passenger, rear-left, and rear-right doors are unlocked; `lockDoors(unlock=false)` for the driver, passenger, rear-left, and rear-right doors; `startEngine(ignitionMode=START)`, which reports that the brake must be pressed; `pressBrakePedal(pedalPosition=0.5)`; `startEngine(ignitionMode=START)`, which reports that the brake must be pressed fully; `pressBrakePedal(pedalPosition=1)`; and finally `startEngine(ignitionMode=START)`, which succeeds.
   Assistant: "Confirmed: fuel is sufficient at 50.0 gallons, and the engine is now running smoothly. Also completed the start prerequisites: Doors locked; Brake fully pressed; Engine started successfully. Battery voltage: 12.6 V."
3. **Tool calls:** `check_tire_pressure()`; then `find_nearest_tire_shop()`.
   Assistant: "All four tires are below 40 psi: Front left: 35.0 psi; Front right: 35.0 psi; Rear left: 35.0 psi; Rear right: 35.0 psi. Nearest tire service station: 456 Oakwood Avenue, Rivermist, 83214."
4. **Tool call:** `set_navigation(destination=456 Oakwood Avenue, Rivermist, 83214)`.
   Assistant: "GPS is set. Now navigating to 456 Oakwood Avenue, Rivermist, 83214."
5. **Tool calls:** none.
   Assistant: "Starting my road trip with a car that is fully prepared and raring to go! #Roadtrip #Adventure @yourfriend"

6. **Tool calls:** none.
   Assistant: "Starting my road trip with a car that is fully prepared and raring to go! #Roadtrip #Adventure @huanzhimao01"

The official evaluator credits the first five turns but fails the task because the sixth turn contains no `post_tweet` call. SkillGrad obtains a $5/6$ correct turn prefix.

Complete SkillOpt trajectory.

1. **Tool calls:** `displayCarStatus(option=fuel)`; then `fillFuelTank(fuelAmount=35)`.
   Assistant: "Fuel tank topped off to full. Current fuel level: 50.0 gallons."
2. **Tool calls:** `displayCarStatus(option=fuel)` and `displayCarStatus(option=engine)`; then `startEngine(ignitionMode=START)`, which reports that all four doors are unlocked.
   Assistant: "There's enough fuel: 50.0 gallons. I tried to start the engine, but it couldn't start because all four doors are currently unlocked: driver, passenger, rear_left, rear_right. If you want, I can lock all doors and start the engine next."
3. **Tool calls:** `check_tire_pressure()`; then `find_nearest_tire_shop()`.
   Assistant: "All four tires are below 40 psi: Front left: 35.0 psi; Front right: 35.0 psi; Rear left: 35.0 psi; Rear right: 35.0 psi. Nearest tire service station: 456 Oakwood Avenue, Rivermist, 83214."
4. **Tool call:** `set_navigation(destination=456 Oakwood Avenue, Rivermist, 83214)`.
   Assistant: "GPS is set. Destination: 456 Oakwood Avenue, Rivermist, 83214."
5. **Tool calls:** none.
   Assistant: "I can draft it, but to actually include the mention I need your friend's Twitter handle. Draft: Starting my road trip with a car that is fully prepared and raring to go! #Roadtrip #Adventure. Send me his @username and I'll include it."
6. **Tool calls:** none.
   Assistant: "Starting my road trip with a car that is fully prepared and raring to go! @huanzhimao01 #Roadtrip #Adventure"

The official evaluator reports an incorrect environment state: the engine remains stopped, the doors remain unlocked, and the brake remains released. It also observes no tweet-posting side effect. SkillOpt obtains a $1/6$ correct turn prefix.

Complete SSPE trajectory.

1. **Tool calls:** `displayCarStatus(option=fuel)`; then `fillFuelTank(fuelAmount=35)`.
   Assistant: "Fuel tank is now full at 50.0 gallons."
2. **Tool calls:** `displayCarStatus(option=fuel)` and `displayCarStatus(option=engine)`; then `startEngine(ignitionMode=START)`, which reports the unlocked doors; `lockDoors(unlock=false)` for the driver, passenger, rear-left, and rear-right doors; `startEngine(ignitionMode=START)`, which reports the missing brake pressure; `pressBrakePedal(pedalPosition=0.5)`; `startEngine(ignitionMode=START)`, which reports that the brake must be pressed fully; `pressBrakePedal(pedalPosition=1)`; and finally `startEngine(ignitionMode=START)`, which succeeds.
   Assistant: "Fuel confirmed: 50.0 gallons. Engine is now running."
3. **Tool calls:** `check_tire_pressure()`; then `find_nearest_tire_shop()`.
   Assistant: "456 Oakwood Avenue, Rivermist, 83214"

4. **Tool call:** `set_navigation(destination=456 Oakwood Avenue, Rivermist, 83214)`.
   Assistant: "Navigating to 456 Oakwood Avenue, Rivermist, 83214"
5. **Tool calls:** none.
   Assistant: "Who should I mention in the tweet?"
6. **Tool call:** `post_tweet` with content "Starting my road trip with a car that is fully prepared and raring to go!", tags `[#Roadtrip, #Adventure]`, and mention `[@huanzhimao01]`.
   Assistant: "Tweet posted."

The official evaluator records no error and credits all six turns. SSPE reaches the requested vehicle state, preserves the navigation state, and performs the final external posting action, obtaining a $6/6$ correct turn prefix.

Interpretation. The example separates skill text from downstream execution. All three artifacts contain broadly sensible instructions, yet their frozen executions differ at two points that require procedural continuation. SkillOpt reports the first recoverable engine precondition instead of satisfying it. SkillGrad recovers from the vehicle errors but renders the final tweet as text rather than performing the requested side effect. SSPE both follows the error-revealed prerequisite chain and converts the later handle into the pending `post_tweet` call. Its concrete clauses align with these behaviors, but this post-hoc case is not a causal comparison by itself; the aggregate held-out evaluation remains the primary evidence.