From Loading Skills to Knowing Them: Scaffolding-Removed Fine-Tuning for Agent Skill Internalization

Anonymous submission

Abstract

Loading skills at run time has become a common paradigm for strengthening an agent's problem solving: the agent pulls a full instruction document into context and follows its methodology. However, this benefit reaches only a minority of users, as most do not invoke skills in practice. We argue that generalizable skills, which encode procedural knowledge for recurring tasks such as code review and debugging, should not remain optional external add-ons but become part of the model's own capability. Internalizing this knowledge into the model makes the corresponding capabilities directly available to all users, without requiring explicit skill loading. To this end, we present **SkillCore**, a data pipeline and supervised fine-tuning recipe that internalizes such skills. Specifically, we transform real agent trajectories by removing the skill-loading scaffolding while preserving the task context and subsequent actions, thereby supervising direct execution without external skill content. Because this objective can also suppress appropriate calls to task-specific skills that provide genuinely external information, the recipe additionally mixes in a small proportion of skill-native trajectories to keep that calling behavior intact. On a held-out skill-free evaluation, SkillCore successfully internalizes generalizable skills in both Qwen3-8B and Qwen2.5-72B, improving performance over base models by 54.4% and 24.8%, respectively. Besides, it preserves skill invocation behavior on a separate task-specific-skill probe, without degrading other aspects of the model's performance, as measured by SWE-bench Verified.

1 Introduction

Large language models (LLMs) are increasingly deployed as agents that reason, plan, and act across tasks in domains ranging from software engineering to scientific discovery (Yao et al. 2023; Wang et al. 2024b). Their effectiveness depends not only on the underlying model but also on runtime scaffolds that supply task-relevant knowledge and procedures (Anthropic 2024). A prominent example is the skill: a reusable package of procedural guidance that is loaded into the agent's context on demand, enabling the agent to follow a specialized workflow without additional parameter updates (Anthropic 2025). Such reusable skill libraries can substantially improve task performance.

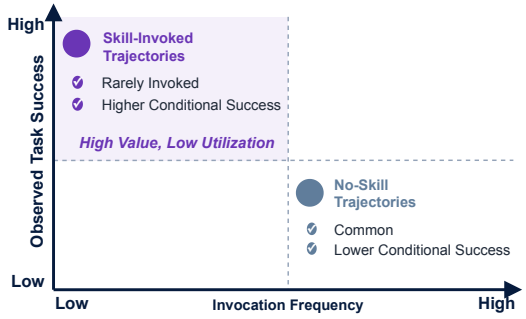


Figure 1: Runtime skills exhibit a high-value, low-utilization gap: skill-invoked trajectories achieve higher conditional task success, yet skills are rarely invoked, preventing their benefits from being broadly accessible by default. This gap motivates us to internalize skills whose procedural knowledge generalizes across tasks.

Despite these benefits, runtime skills help only when they are actually invoked. In our collected agent trajectories, only 5.4% of distinct sessions contain a successful skill load followed by subsequent agent actions, indicating that effective skill use remains sparse in actual human-agent interactions. Consequently, the procedural capabilities supplied by skills remain unavailable by default. This gap motivates skill internalization: making the reusable procedural capabilities supplied by runtime skills intrinsic to the model, so that an agent can perform the corresponding tasks even when the skill content is not supplied. The key question then becomes which skill-provided capabilities should be internalized and which should remain externally accessible.

To determine what should be internalized, we distinguish between *generalizable* and *task-specific* skills. Generalizable skills encode reusable procedures, *e.g.*, systematic debugging, code review, and table extraction, whose value extends beyond a particular system. Task-specific skills instead provide private APIs, system-bound instructions, or task-specific information that the model cannot reconstruct from its weights. An effective internalization approach must there-

This is an anonymized submission for review purposes only. Distribution, citation, or public sharing of this manuscript is strictly prohibited. Copyright and publication details will appear in the final version if accepted.

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

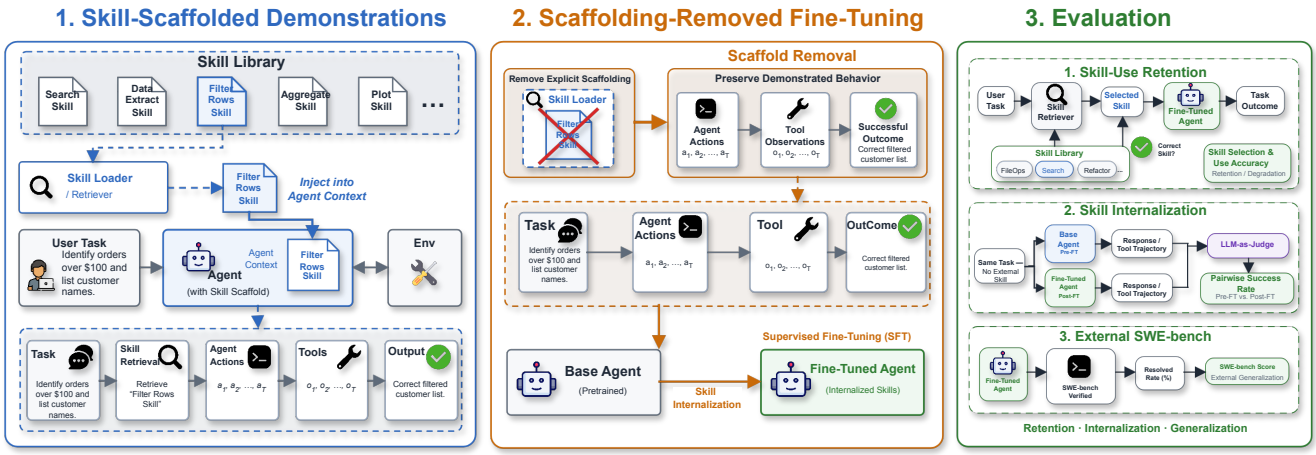


Figure 2: **Skill internalization via scaffolding-removed fine-tuning. (1) Skill-Scaffolded Demonstrations:** given a user task, a skill retriever selects and injects the relevant skill (e.g., *Filter Rows Skill*) into the agent's context, which then acts in the environment to produce a successful trajectory of actions $a_{1:T}$ and observations $o_{1:T}$. **(2) Scaffolding-Removed Fine-Tuning:** we strip the skill scaffold but keep the demonstrated behavior, yielding task-only trajectories used to fine-tune a base agent via SFT ($\mathcal{L}_{\text{SFT}} = -\sum_t \log p_\theta(a_t \mid \text{task}, \text{history})$), distilling the skill into its parameters. **(3) Evaluation:** we measure *skill-use retention* (correct skill selection when the retriever is available), *skill internalization* (base vs. fine-tuned agent on the same task without external skills, judged by an LLM), and *external generalization* (resolved rate on SWE-bench Verified).

fore reduce unnecessary dependence on generalizable skills while preserving the model's ability to invoke task-specific skills when external information is genuinely required.

In this work, we propose SkillCore, a framework that integrates a trajectory construction pipeline with a tailored supervised fine-tuning recipe to internalize generalizable skills while preserving access to genuinely external information. Specifically, for generalizable skills, the data pipeline applies skill-scaffold removal: it removes the skill invocation, returned document, and associated loading artifacts while preserving the task context and the agent's subsequent actions. This transformation pairs the preserved actions with a trajectory that excludes the skill-relevant context, providing direct supervision for executing the demonstrated procedure without that runtime scaffold. Training on these trajectories internalizes the skill's procedure into the model's weights, so it can reproduce the demonstrated behavior without the skill. By contrast, for task-specific skills, the training recipe retains a small proportion of skill-native trajectories with their invocation scaffolding intact, preserving supervision for seeking external information when it is genuinely required.

The evaluation spans three complementary dimensions: internalization of generalizable skills, retention of task-specific skill invocation, and preservation of general capabilities. First, a held-out skill-free evaluation assesses whether SkillCore improves model performance when generalizable skill content is unavailable. Under this setting, SkillCore improves the response quality of Qwen3-8B and Qwen2.5-72B over their corresponding based models by 54.4% and 24.8%, respectively. Second, on a separate evaluation suite constructed from task-specific-skill trajectories, SkillCore preserves the models' ability to invoke task-specific skills at base-model levels. Third, the resulting models maintain or improve general coding performance on SWE-bench Verified (Jimenez et al. 2024; OpenAI 2024). Additional ablations show that retaining generalizable-skill scaffolding increases runtime dependence, whereas omitting skill-native supervision suppresses appropriate task-specific calls.

Contributions.

- We formulate skill internalization as transferring reusable procedural capabilities from runtime skill scaffolds into the model while preserving the ability to access task-specific external information when needed.
- We introduce SkillCore, a framework that integrates a faithful trajectory construction pipeline with a tailored supervised fine-tuning recipe for skill internalization.
- On Qwen3-8B and Qwen2.5-72B, SkillCore improves response quality in the skill-free setting by 54.4% and 24.8%, respectively, while preserving task-specific skill invocation capabilities and maintaining or improving general coding performance on SWE-bench Verified.

2 Related Work

2.1 Agent Harness and Runtime Skill

An agent's capabilities depend not only on its underlying model but also on its harness, *i.e.*, the surrounding system of prompts, memory, tools, retrieval mechanisms, and control flow (Sumers et al. 2024; Anthropic 2024). Representative harness components include reasoning–action loops such as ReAct (Yao et al. 2023) and learned access to external tools and APIs (Schick et al. 2023; Qin et al. 2024), which complement the model's parametric knowledge with information, actions, or procedural structure at inference time.

Agent skills provide a procedural form of runtime augmentation by packaging instructions, scripts, and resources

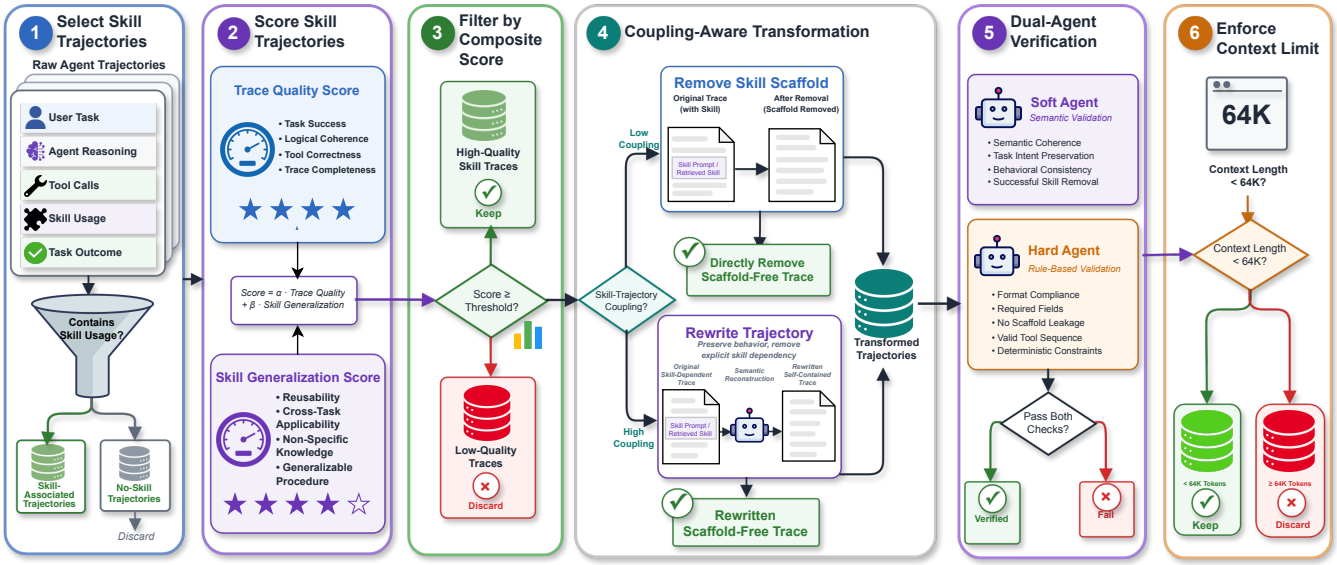


Figure 3: The trajectory construction pipeline. We select skill-using trajectories, score them for quality and generalizability, and keep the high-quality ones with generalizable skills. Each is then transformed to remove the skill scaffold, validated for structural and semantic faithfulness, and filtered by context length to form the scaffold-removed training corpus.

into discoverable directories whose contents are progressively loaded when relevant (Anthropic 2025). Skills need not be manually created: Voyager automatically generates executable code through open-ended exploration and stores successful programs in a skill library (Wang et al. 2024a), while Agent Workflow Memory induces reusable workflows from prior task experience and selectively supplies them to guide subsequent generations (Wang et al. 2025). More recent approaches automatically expand such skill libraries through environment exploration (Zheng et al. 2025), distill successful traces into callable procedural structures (Guo et al. 2026), or recursively co-evolve a retrieved skill bank with the policy (Xia et al. 2026). Despite their different representations, these methods retain the resulting procedures as external artifacts invoked at inference time. SkillCore studies a complementary boundary: internalizing generalizable procedures as intrinsic model capabilities while preserving runtime access to task-specific external information.

2.2 Context Distillation and Skill Internalization

Context distillation transfers behavior induced by privileged runtime context into model parameters. Prior work distills fixed prompts, instructions, demonstrations, and explanations so that models can reproduce the corresponding behavior with reduced context (Askell et al. 2021; Choi et al. 2023; Snell, Klein, and Zhong 2022). Related work distills outputs produced by more expensive inference-time reasoning into direct model responses (Yu et al. 2024). Despite demonstrating effective context-to-parameter transfer, these approaches primarily target single-turn text generation rather than multi-turn agent trajectories involving tool use and environment feedback.

Recent work extends this paradigm to agent trajectories and skills. On-Policy Context Distillation trains a student against a context-conditioned teacher on trajectories generated by the current policy (Ye et al. 2026). Skill-SD summarizes completed on-policy trajectories into training-only skill guidance (Wang et al. 2026), SKILL0 progressively withdraws skill context during agentic reinforcement learning according to its on-policy helpfulness (Lu et al. 2026), and SIRI mines and validates skills from current-policy rollouts before distilling beneficial skill-guided actions into the plain policy (He et al. 2026). All these approaches instantiate skill internalization through newly sampled interactive behavior. LatentSkill offers an offline alternative by compiling textual skills into modular LoRA adapters (Hu et al. 2022) that remain separately loadable at inference time (Yu et al. 2026). SkillCore instead constructs supervision offline from previously logged real-agent trajectories: it removes generalizable-skill scaffolding while preserving the recorded subsequent actions, then applies supervised fine-tuning. It therefore requires neither fresh policy rollouts nor an executable task environment during corpus construction or training, although its source trajectories are originally collected from real agent interactions. In this sense, SkillCore is environment-free and it internalizes procedures directly in a shared model rather than representing each skill as an inference-time adapter.

3 Methodology

3.1 Overview

Let $\tau = (m_1, \ldots, m_T)$ denote an agent trajectory comprising system, user, assistant, and tool messages. For trajectories involving generalizable skills, we remove the skill scaffold while preserving the demonstrated actions and observations, then fine-tune the model to execute the procedure directly. By

contrast, for task-specific skills that provide external information or interfaces and should remain callable at inference time, we retain skill-native supervision.

Building on this distinction, the remainder of this section describes how we construct scaffold-stripped and skill-native trajectories, validate the transformed data, jointly train on the two supervision sources, and evaluate the resulting model. Figure 2 provides an overview of the complete framework.

3.2 Data Selection and Scaffold Removal

Skill-Use Filtering. We collect $3{,}352{,}057$ real-world agent trajectories and filter for those that invoke skills through genuine `use_skill` calls in their structured messages. We consider a call effective only when it specifies a non-empty skill name, is linked to a successful result, and is followed by subsequent agent actions. Only $108{,}514$ trajectories (3.2%) pass this filter, confirming that skill-assisted interactions are rare in real-world usage. Because annotating and transforming the entire filtered pool is costly, we select a subset while preserving skill coverage. A conversation may be recorded as multiple nested or diverging snapshots; we retain its most complete eligible snapshot and add a branch only when it introduces an otherwise uncovered successfully loaded skill. This produces $44{,}960$ retained snapshots from $40{,}023$ distinct conversations for subsequent refinement.

Quality and Generalizability. For each filtered trajectory τ, let $\mathcal{C}_{\text{eff}}(\tau)$ be its set of effective skill call instances. We use Claude Opus 4.8 (Anthropic 2026) to assign an overall trajectory-level quality score $q(\tau) \in \{1, \ldots, 5\}$ and, for each $g \in \mathcal{C}_{\text{eff}}(\tau)$, a generalizability score $\text{gen}(g) \in \{1, \ldots, 5\}$. Higher values indicate better task execution and broader reusability, respectively; a generalizability score of 1 is task-specific, whereas 5 is broadly reusable. The judge also returns two Boolean labels: $\text{followed}(g)$ indicates whether later actions follow the loaded skill, and $\text{taskobj}(g)$ indicates whether the skill itself is being manipulated.

Let $d(\tau)$ indicate that the task is completed or ends because of collection truncation, $e(\tau)$ that no repeated error loop occurs, and $u(\tau)$ that the user expresses no dissatisfaction. We define the trajectory-quality gate and the set of generalizable calls targeted for scaffold removal as

$$\begin{aligned} Q(\tau) &= \mathbb{I}[q(\tau) \geq 4 \wedge d(\tau) \wedge e(\tau) \wedge u(\tau)]\,, \\ \mathcal{G}(\tau) &= \{g \in \mathcal{C}_{\text{eff}}(\tau) : \text{gen}(g) \geq 4, \\ &\qquad \text{followed}(g),\ \neg\,\text{taskobj}(g)\}. \end{aligned} \tag{1}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. Thus, $Q(\tau) = 1$ means that τ passes the trajectory-quality gate, while $\mathcal{G}(\tau)$ contains the calls selected for scaffold removal. We transform τ only when $Q(\tau) = 1$ and $\mathcal{G}(\tau) \neq \varnothing$; high-quality trajectories with $\mathcal{G}(\tau) = \varnothing$ remain intact as skill-native supervision.

Coupling-Aware Scaffold Transformation. Before transformation, we render each eligible trajectory into a compact, indexed representation that fits the model context. For both quality assessment and scaffold transformation, long system and tool fields are truncated while preserving their informative boundaries. We then use Claude Opus 4.8 to produce structured edits according to how tightly the skill scaffold is coupled with the surrounding task context. When scaffold traces are localized, the model identifies target call IDs and verbatim text spans for deterministic removal. For more tightly coupled cases, the framework supports rewriting only a local window around the skill call while keeping the remainder of the trajectory unchanged. Both routes remove the target invocation, linked skill document, active routing or scaffold-maintenance language, and subsequent accesses to the removed skill's internal files, while retaining the system and tool definitions, non-target skill interactions, and substantive downstream task behavior.

Validation and Assembly. We retain a transformed trajectory only after it passes two complementary validation stages. First, a deterministic structural gate verifies that every targeted skill invocation and its linked result have been removed, all non-target skill calls are retained, and the edit introduces neither missing results for previously paired calls nor orphan tool results. It additionally requires a byte-identical system message and a non-empty user request. The available tool definitions remain unchanged, allowing the model to invoke external skills when needed. These checks ensure that scaffold removal does not corrupt the agent interaction protocol, but they cannot determine whether an edit is semantically complete or overly aggressive.

We therefore perform a second, semantic review of each structurally valid transformation using Claude Opus 4.8. It checks six properties: removal of active scaffold traces, preservation of the original task intent, preservation of substantive action language, coherence between adjacent assistant messages, absence of later access to the removed skill's internal files, and continuity of the overall discourse and task structure. A trajectory is retained only if all six checks pass, yielding $10{,}037$ validated scaffold-removed trajectories. Figure 3 summarizes this scaffold-removal pipeline.

From this validated pool, we build the internalization evaluation suite of 348 tasks spanning 274 skills, maximizing skill diversity. It includes both skills represented in the training corpus and held-out skills absent from it, allowing us to test whether improvements reflect procedural transfer rather than memorization alone. We separately construct a task-specific skill-retention suite of 228 tasks spanning 188 skills from high-quality trajectories in which a task-specific skill is successfully loaded and followed, retaining the prefix immediately before the recorded skill call. Both evaluation suites undergo extensive manual review to verify trajectory quality, skill labels, and prefix validity.

3.3 Training

Before training, we serialize each trajectory using the Hermes agent template implemented in SWIFT (Zhao et al. 2025) and retain only records whose message content and tool definitions fit within a $64{,}000$-token proxy computed with the Qwen2.5-72B tokenizer. Because the proxy excludes chat-template control tokens, the trainer keeps left truncation as a fallback. We then mix the scaffold-stripped corpus with skill-native trajectories that preserve task-specific skill calls and their loaded content. By default, we mix in skill-native data at 25% of the scaffold-stripped corpus. This mixture

Model	Training	Task-specific routing (228)			Skill-content-free continuation (348)			Coding
		Call↑	Over↓	Corr@	Win↑	ΔQuality↑	Residual↓	SWE-b↑
Qwen3-8B	Start (base)	50.0	50.0	89.5	–	–	16.7	4.2
	Scaffold-retaining	–	–	–	58.9	+0.42	44.5	5.2
	Internalization only	13.2	86.8	**93.3**	**82.8**	**+1.23**	**2.6**	4.4
	SkillCore (mix25)	52.6	46.5	–	75.9	+1.00	19.5	6.2
Qwen2.5-72B	Start (base)	65.4	28.5	98.7	–	–	33.6	43.4
	Scaffold-retaining	–	–	–	35.1	−0.37	59.5	45.6
	Internalization only	7.0	89.9	**100.0**	**62.4**	**+0.45**	**0.9**	44.8
	SkillCore (mix25)	65.8	30.3	–	54.9	+0.22	30.2	45.0

Table 1: Main results at same-checkpoint operating points. *Scaffold-retaining* is the epoch-matched control that keeps the demonstrated calls. Routing metrics use the 228 task-specific prefixes; Win/ΔQuality/Residual use the 348 continuation tasks; SWE-b is auxiliary. Mix25 is the call-parity operating point (full sweep in Table 2).

jointly teaches direct execution for internalized skills and external routing when a skill should remain callable; Section 4 studies alternative mixture ratios.

We supervise only model-generated tokens. For serialized tokens $x_{1:N}$, let $a_t = 1$ for assistant language and structured tool-call tokens and $a_t = 0$ for system, user, and tool-result tokens. The latter remain visible as context but do not contribute to the loss:

$$\mathcal{L}(\theta) = -\frac{1}{\sum_{t=1}^{N} a_t} \sum_{t=1}^{N} a_t \log p_\theta(x_t \mid x_{<t}). \qquad (2)$$

We select Qwen3-8B (Qwen Team 2025) and a stronger Qwen2.5-72B model (Qwen Team 2024) that has undergone post-training on agent trajectories as our base models for fine-tuning. The Appendix details the full data-refinement funnel and the training hyperparameters for both models.

3.4 Evaluation

Generalizable Skill Internalization. This evaluation suite tests whether the model has internalized a generalizable skill's workflow, carrying out the task well on its own when the skill is no longer provided. From a shared pre-call prefix, we remove the target skill call and its returned document. The starting and fine-tuned checkpoints continue from the identical prefix for up to three assistant rounds; when a round invokes a tool, we replay the aligned source observation instead of executing it, so the protocol measures controlled continuation rather than closed-loop success. We evaluate each continuation along two axes. The first is the residual `use_skill` invocation rate (lower is better), which directly measures whether the model still depends on the removed skill. The second is the overall trajectory quality and task completion, which we assess with an LLM-as-a-judge protocol (Zheng et al. 2023) using Claude Opus 4.8; it produces a pairwise preference between the two models' continuations, together with a 1–5 score for each (higher is better).

Task-Specific Skill Retention. This suite tests whether internalization erodes appropriate routing to skills that must stay external, such as those providing proprietary interfaces or private information. It contains 228 pre-call prefixes spanning 188 task-specific skills whose recorded next action invokes a skill. With the skill interface available, each model generates a single next action. We report three metrics that together characterize this routing decision. The *call rate* is the fraction of tasks on which the model invokes `use_skill`, so a value close to the starting checkpoint indicates that the willingness to route is preserved. *Over-internalization* is the complementary failure, where the model instead substitutes another tool or answers directly; a large increase over the starting checkpoint signals negative transfer, *i.e.*, the model wrongly tries to handle an external skill on its own. Conditioned on the model actually calling, the *correct-call rate* measures whether it selects the right skill, which separates a willingness-to-route failure from a skill-selection failure. The call and over-internalization rates use the full 228 tasks as denominator, with empty generations counted as non-calls. The suite thus isolates one-step routing propensity rather than post-loading task success.

General Coding Capability. This check verifies that internalization does not degrade broad coding ability. We report the SWE-bench Verified (Jimenez et al. 2024; OpenAI 2024) score as an auxiliary diagnostic. The Appendix details the two skill-focused suites and their limitations.

4 Experiments

4.1 Overview

In this section, we conduct comprehensive experiments to demonstrate the effectiveness of SkillCore with Qwen3-8B and Qwen2.5-72B on the held-out skill-free continuation suite for generalizable skill internalization and the task-specific routing suite for skill invocation retention. We organize the study around four questions that trace the recipe end to end: whether scaffold-removed supervision truly internalizes a generalizable skill's workflow, so the model carries out the task well on its own when the skill is absent (RQ1); whether the scaffold removal itself, rather than mere exposure to the agent trajectories, is the source of the gains in continuation quality and task completion (RQ2); what routing failure pure internalization introduces on skills that must stay external (RQ3); and whether adding a small proportion of skill-native supervision recovers that routing without sacrificing internalization (RQ4). We then report secondary checks on data curation and general coding ability.

4.2 RQ1: Does Scaffold-Removed Supervision Internalize Skills?

We first train on scaffold-removed trajectories alone, without any skill-native data (*internalization-only* training). As shown in the *Internalization only* rows of Table 1, this training sharply reduces runtime skill loading while improving task completion quality on the skill-free continuation evaluation suite. For 8B, residual skill invocation falls from 16.7% to 2.6%, the fine-tuned continuation wins 82.8% of pairwise comparisons, and mean quality increases by 1.23 points. The 72B counterpart reduces residual skill invocation from 33.6% to 0.9%, with 62.4% wins and a 0.45-point quality gain. These paired results support direct procedural transfer under the controlled-continuation protocol.

4.3 RQ2: Is Scaffold Removal Important?

The *Scaffold-retaining* rows of Table 1 give the epoch-matched control that keeps the demonstrated calls instead of stripping them. Retaining the calls yields substantially higher residual invocation (44.5% and 59.5%) and lower win rates (58.9% and 35.1%) than the internalization-only runs that strip them (2.6%/0.9% residual; 82.8%/62.4% wins), and its paired quality change is small or negative ($+0.42$ for 8B, -0.37 for 72B) while the auxiliary SWE-bench score is essentially unchanged, so the difference is specific to skill loading rather than general coding ability. Thus, simply fine-tuning on skill-augmented demonstrations can reinforce runtime dependence, whereas removing the scaffold encourages direct execution. Because retaining the document changes sequence length and therefore post-filter corpus membership, this ablation supports the role of removal in the evaluated configurations but is not a perfectly matched causal estimate.

4.4 RQ3: What Routing Cost Does Pure Internalization Impose?

The benefit on generalizable skills comes with a clear failure mode (Table 1, routing columns). On the 228 task-specific prefixes, the call rate drops from 50.0% to 13.2% for 8B and from 65.4% to 7.0% for 72B. The suppressed calls are replaced by over-internalization: instead of invoking the expected skill, the model substitutes another tool or a plain-text answer, and this over-internalization rate rises from 50.0% to 86.8% for 8B and from 28.5% to 89.9% for 72B. When the models do call, however, the recorded skill name is usually correct (correct-given-called 93.3% and 100%), so the dominant observed error is whether to invoke rather than which skill to select. Because the probe contains one decision and no post-call execution, it diagnoses routing propensity rather than complete skill-use success.

4.5 RQ4: Can Skill-Native Data Recover Routing?

Adding skill-native supervision moves both models along a routing–internalization frontier (Table 2). As the native-to-core ratio grows from mix10 to mix50, the task-specific call rate increases monotonically ($44.3\% \rightarrow 52.6\% \rightarrow 66.7\%$ for 8B and $60.1\% \rightarrow 65.8\% \rightarrow 73.2\%$ for 72B) and over-internalization falls in step, but residual invocation on the skill-content-free suite rises together ($12.4\% \rightarrow 27.0\%$ and $21.6\% \rightarrow 32.2\%$) and pairwise win rates drift downward. No single mixture simultaneously maximizes routing recovery and internalization; the choice is an operating point rather than a dominant configuration. We report mix25 because it restores call rates closest to the starting checkpoints (8B: 52.6% vs. 50.0%; 72B: 65.8% vs. 65.4%) while retaining positive pairwise wins and quality gains (Table 1). This call-parity criterion is fixed before reading the continuation metrics, but it is a reporting convention, not a validation-selected optimum.

Metric	No mix	mix10	**mix25**	mix50
Qwen3-8B				
Call↑	13.2%	44.3%	52.6%	66.7%
Over↓	86.8%	55.3%	46.5%	32.9%
Win↑	82.8%	75.9%	75.9%	74.7%
ΔQ↑	+1.23	+0.95	+1.00	+0.91
Residual↓	2.6%	12.4%	19.5%	27.0%
SWE-b↑	4.4%	4.8%	6.2%	8.6%
Qwen2.5-72B				
Call↑	7.0%	60.1%	65.8%	73.2%
Over↓	89.9%	36.4%	30.3%	24.1%
Win↑	62.4%	60.1%	54.9%	54.9%
ΔQ↑	+0.45	+0.45	+0.22	+0.23
Residual↓	0.9%	21.6%	30.2%	32.2%
SWE-b↑	44.8%	44.6%	45.0%	46.8%

Table 2: Full routing–internalization trade-off across skill-native mixture ratios. Call/Over use the 228 routing tasks; Win/ΔQ/Residual use the 348 continuation tasks; SWE-b is auxiliary. Mix25 (bold) is the call-parity operating point.

4.6 Secondary Capability and Data Checks

Corpus variants do not show a monotonic data-scaling trend. The curated $2{,}369$-trajectory 8B core and its $5{,}049$-trajectory expansion reach comparable selected continuation results (82.8% vs. 82.2% win at their reporting epochs), and neither the left-truncated variant nor the less-curated 72B builds improve the observed ceiling (Table 3). This suggests that transformation quality, rather than raw corpus size, drives the effect in these settings. As a capability guardrail, SWE-bench Verified stays at or slightly above the starting checkpoint for both models; we treat it only as an auxiliary check because its complete execution protocol is not available in the archived artifacts. Per-epoch training dynamics appear in the Appendix.

5 Analysis and Limitations

Transfer Beyond Post-Filter Training Skills. The 348 evaluation tasks expand to 381 (sample, skill) pairs because some trajectories contain multiple targeted skills. For the 8B internalization-only checkpoint, 152 pairs spanning 120 skills have zero frequency in the actual length-filtered training corpus. Despite this, the zero-frequency group attains 2.0% residual invocation, 82.2% pairwise wins, and mean fine-tuned quality 3.47. Figure 4 shows that gains persist across frequency bins, supporting transfer beyond skill identities represented after filtering. This is not a globally unseen-

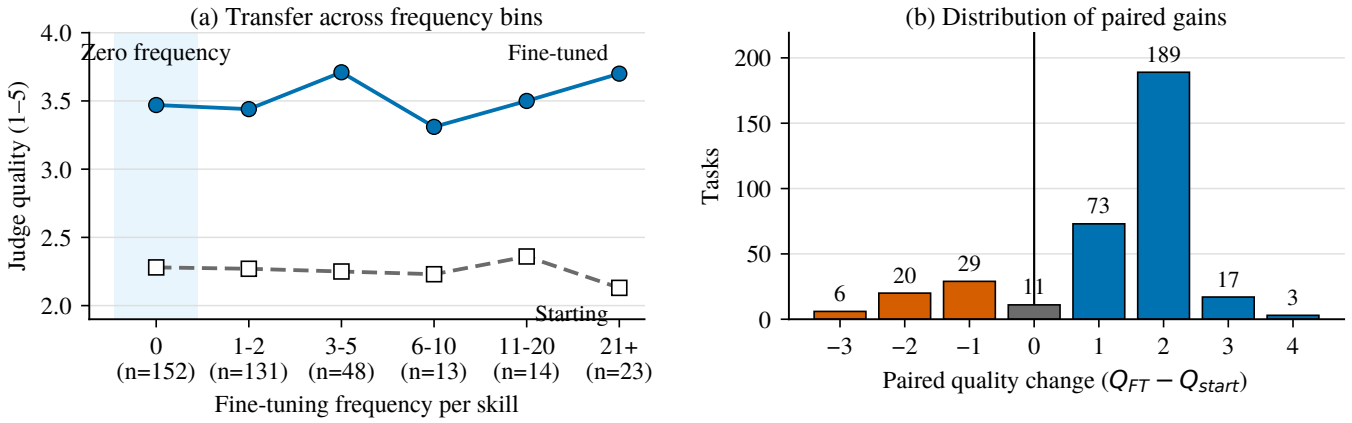


Figure 4: Transfer and paired quality analysis for the selected 8B internalization-only checkpoint. (a) Judge quality by post-filter training frequency; labels give the number of evaluated (sample, skill) pairs. (b) Distribution of per-task quality changes. The zero-frequency result is observational rather than a globally unseen-skill evaluation.

Corpus	Records	Resid.↓	Win↑	Q_{ft}
Qwen3-8B				
DS-2	2,369	2.6%	82.8%	3.49
DS-1	4,607	5.5%	81.3%	3.45
Curated	5,049	6.0%	82.2%	3.49
Qwen2.5-72B				
DS-2	2,369	0.9%	62.4%	3.30
data10k	∼10,000	8.9%	66.7%	3.49
Full data	all	12.1%	64.7%	3.39

Table 3: Corpus comparison at the selected reporting checkpoints. The model settings differ in family and initialization and are interpreted independently.

skill claim: candidate examples existed before filtering, frequency may covary with difficulty, and the high-frequency bins are small.

Paired Quality Gains. The aggregate improvement is broad rather than driven by a few outliers. On the 348 skill-content-free tasks, the 8B starting checkpoint has mode 2 (65.8%), whereas the fine-tuned checkpoint has mode 4 (67.2%). The mean paired change is +1.23, and 189 tasks improve by exactly two points. The paired distribution in Figure 4 also exposes failures: 55 tasks decrease, 11 are unchanged, and 282 improve. Thus, the average gain reflects a systematic shift while leaving a meaningful tail for error analysis.

Evaluation Validity and Failure Modes. The evidence is deliberately scoped. Results use one training seed, one deterministic generation per checkpoint, and one Claude Opus 4.8 judgment per task, without confidence intervals or an independent validation split. Table 4 summarizes the available judge diagnostics: winner labels agree with the scalar quality ordering on 97.9–98.2% of examples and 67.5% of verdicts have a quality gap of at least two points, but repeated judges, seeds, and confidence intervals are not reported. Logged-

Diagnostic	Reported value
Winner–quality ordering agreement	97.9–98.2%
Verdicts with $\lvert\Delta Q\rvert \geq 2$	67.5%
Equal-quality decisions / swap audit	2.0% / n=50

Table 4: Available judge diagnostics and uncertainty scope.

observation replay can diverge from a generated action, and the routing suite tests one decision rather than post-loading task success. Length filtering changes corpus membership between scaffold-stripped and retaining conditions, weakening strict causal interpretation. A 50-case position-swap audit finds that order sensitivity concentrates near small quality gaps, but it does not replace independent judges or human evaluation. We also lack a session-level overlap audit, and the two model settings differ in family and initialization. We therefore treat the reported values as point estimates on the logged-task distribution.

6 Conclusion

Motivated by the high-value but low-utilization gap of runtime skills, we argue that generalizable skills should be internalized into the model while task-specific skills that supply genuinely external information remain callable, and present **SkillCore**, which removes generalizable-skill scaffolds while preserving the task description and subsequent actions and mixes in a small proportion of skill-native trajectories to keep task-specific skill invocation intact. SkillCore improves the skill-free response quality of Qwen3-8B and Qwen2.5-72B over their base models by 54.4% and 24.8%, respectively, while preserving task-specific skill invocation and general coding ability. By turning sparsely invoked skills into durable model competence, SkillCore shows that capabilities delivered through runtime scaffolds can instead be made intrinsic and available to all users by default.

References

Anthropic. 2024. Building Effective Agents. https://www.anthropic.com/engineering/building-effective-agents.

Anthropic. 2025. Equipping Agents for the Real World with Agent Skills. https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Anthropic. 2026. Introducing Claude Opus 4.8. https://www.anthropic.com/news/claude-opus-4-8.

Askell, A.; Bai, Y.; Chen, A.; Drain, D.; Ganguli, D.; Henighan, T.; Jones, A.; Joseph, N.; Mann, B.; DasSarma, N.; Elhage, N.; Hatfield-Dodds, Z.; Hernandez, D.; Kernion, J.; Ndousse, K.; Olsson, C.; Amodei, D.; Brown, T.; Clark, J.; McCandlish, S.; Olah, C.; and Kaplan, J. 2021. A General Language Assistant as a Laboratory for Alignment. *arXiv preprint arXiv:2112.00861*.

Choi, E.; Jo, Y.; Jang, J.; Jang, J.; and Seo, M. 2023. Fixed Input Parameterization for Efficient Prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, 8428–8441.

Guo, Z.; Qi, D.; Gu, H.; Cheng, P.; and Xiong, Y. 2026. SKILL-DISCO: Distilling and Compiling Agent Traces into Reusable Procedural Skills. *arXiv preprint arXiv:2606.26669*.

He, Z.; Li, Y.; Huang, F.; Chen, T.; Chen, S.; Li, X.; Yu, M. H.; Liu, X.; Wei, L.; Pan, L.; Zeng, K.; and Cai, X. 2026. SIRI: Self-Internalizing Reinforcement Learning with Intrinsic Skills for LLM Agent Training. *arXiv preprint arXiv:2606.02355*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.

Jimenez, C. E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; and Narasimhan, K. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *International Conference on Learning Representations (ICLR)*.

Lu, Z.; Yao, Z.; Wu, J.; Han, C.; Gu, Q.; Cai, X.; Lu, W.; Xiao, J.; Zhuang, Y.; and Shen, Y. 2026. SKILL0: In-Context Agentic Reinforcement Learning for Skill Internalization. *arXiv preprint arXiv:2604.02268*.

OpenAI. 2024. Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; Zhao, S.; Hong, L.; Tian, R.; Xie, R.; Zhou, J.; Gerstein, M.; Li, D.; Liu, Z.; and Sun, M. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *International Conference on Learning Representations (ICLR)*.

Qwen Team. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Qwen Team. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *Advances in neural information processing systems*, 36: 68539–68551.

Snell, C.; Klein, D.; and Zhong, R. 2022. Learning by Distilling Context. *arXiv preprint arXiv:2209.15189*.

Sumers, T. R.; Yao, S.; Narasimhan, K.; and Griffiths, T. L. 2024. Cognitive Architectures for Language Agents. *Transactions on Machine Learning Research*, 2024.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2024a. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*.

Wang, H.; Wang, G.; Xiao, H.; Zhou, Y.; Pan, Y.; Wang, J.; Xu, K.; Wen, Y.; Ruan, X.; Chen, X.; and Qi, H. 2026. Skill-SD: Skill-Conditioned Self-Distillation for Multi-turn LLM Agents. *arXiv preprint arXiv:2604.10674*.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; Zhao, W. X.; Wei, Z.; and Wen, J. 2024b. A Survey on Large Language Model Based Autonomous Agents. *Frontiers of Computer Science*, 18(6): 186345.

Wang, Z. Z.; Mao, J.; Fried, D.; and Neubig, G. 2025. Agent Workflow Memory. In *International Conference on Machine Learning*, 63897–63911. PMLR.

Xia, P.; Chen, J.; Wang, H.; Liu, J.; Zeng, K.; Wang, Y.; Han, S.; Zhou, Y.; Zhao, X.; Chen, H.; Zheng, Z.; Xie, C.; and Yao, H. 2026. SkillRL: Evolving Agents via Recursive Skill-Augmented Reinforcement Learning. *arXiv preprint arXiv:2602.08234*.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.

Ye, T.; Dong, L.; Wu, X.; Huang, S.; and Wei, F. 2026. On-Policy Context Distillation for Language Models. *arXiv preprint arXiv:2602.12275*.

Yu, A.; Zhou, C.; Xu, T.; Guo, Z.; Shan, R.; Fu, Z.; Wang, J.; Liu, W.; Yu, Y.; Zhang, W.; and Lin, J. 2026. LatentSkill: From In-Context Textual Skills to In-Weight Latent Skills for LLM Agents. *arXiv preprint arXiv:2606.06087*.

Yu, P.; Xu, J.; Weston, J.; and Kulikov, I. 2024. Distilling System 2 into System 1. *arXiv preprint arXiv:2407.06023*.

Zhao, Y.; Huang, J.; Hu, J.; Wang, X.; Mao, Y.; Zhang, D.; Zhang, H.; Jiang, Z.; Wu, Z.; Ai, B.; Wang, A.; Zhou, W.; and Chen, Y. 2025. SWIFT: A Scalable lightWeight Infrastructure for Fine-Tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 29733–29735.

Zheng, B.; Fatemi, M. Y.; Jin, X.; Wang, Z. Z.; Gandhi, A.; Song, Y.; Gu, Y.; Srinivasa, J.; Liu, G.; Neubig, G.; and Su, Y. 2025. SkillWeaver: Web Agents Can Self-Improve by Discovering and Honing Skills. *arXiv preprint arXiv:2504.07079*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E. P.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*.