SkillsBench: Evaluating Procedural Knowledge for AI Agents

Anonymous Authors[1]

Abstract

We introduce SKILLSBENCH, the first benchmark to systematically measure effectiveness of Agent Skills as first-class evaluation artifacts. Agent Skills are structured packages of procedural knowledge and resources that promise LLM agent capabilities gains without model modification. Although Agent Skills are proliferating in real-world agentic systems, there is no standard method to evaluate their effectiveness. SKILLSBENCH directly measures Skill efficacy by comparing task performance with and without Skills across agent-model combinations. We curate 85 tasks spanning 13 domains with deterministic verifiers and trajectory logging. Across 7 agent-model configurations and 6970 trajectories, we find Skills improve performance by 19.9 percentage points on average. However, 24 tasks show negative effects; Software Engineering performance drops by 5 points with Skills. Our analysis reveals that a small number of compact Skills is optimal, and that Skill-aware harnesses suffer significant reliability issues. These findings establish design principles for Skill authors and highlight the need for domain-specific deployment strategies.

1. Introduction

Large language models (LLMs) have evolved from text generators into autonomous agents capable of executing complex, multi-step tasks in real-world environments (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023; Ouyang et al., 2022; Yao et al., 2022). This evolution is exemplified by agent-centric CLI tools: Claude Code (Anthropic, 2025b) from Anthropic, Gemini CLI (Google, 2025) from Google, and Codex CLI (OpenAI, 2025) from OpenAI enable developers to leverage frontier models as agentic assistants within terminal environments. However, a fundamental tension exists: foundation models provide broad capabilities but lack the procedural knowledge required for domain-specific workflows, while fine-tuning is expensive and sacrifices generality.

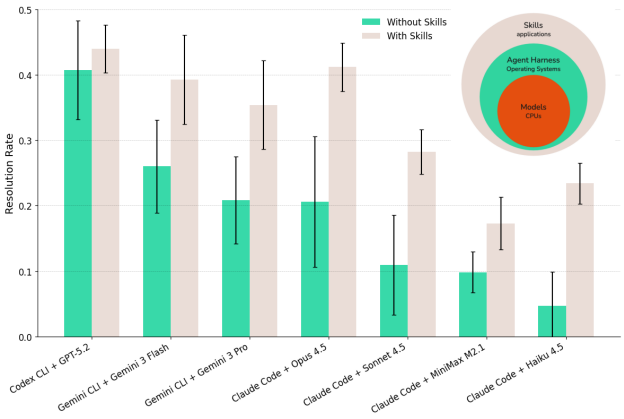

Figure 1. Agent architecture stack. Foundation models provide base capabilities, agent harnesses orchestrate tools and context, and Skills extend competence to specialized domains. This layered design enables modular augmentation without model modification.

Agent Skills offer an emerging solution. A Skills is a structured package comprising instructions, code templates, resources, and verification logic that augments agent behavior at inference time without model modification (Anthropic, 2025a). Skills encode procedural knowledge: standard operating procedures, domain conventions, and task-specific heuristics that guide agent behavior. This modular approach builds on the options framework for temporal abstraction (Sutton et al., 1999) and cognitive architectures for language agents (Sumers et al., 2023), mirroring successful computing paradigms: foundation models provide base capabilities (analogous to CPUs), agent harnesses orchestrate context and tools (operating systems), and Skills extend competence to specialized domains (applications).

Skills ecosystems have grown rapidly, with community repositories now hosting thousands of user-contributed Skills spanning software engineering, data analysis, and enterprise workflows. Yet despite this proliferation, no benchmark systematically evaluates how and when Skills improve agent performance, what content drives gains, or what design principles distinguish effective Skills from ineffective

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ones. The question is not whether adding task-relevant context helps, but rather: How much do Skills help compared to baseline augmentation? Which Skills components (instructions vs. code vs. examples) contribute most? When do Skills fail despite being present?

Existing agent benchmarks (Liu et al., 2023; Merrill et al., 2026; Jimenez et al., 2024; Zhou et al., 2024b; Xie et al., 2024; Koh et al., 2024; Trivedi et al., 2024; Yang et al., 2023; Chan et al., 2025; Zhuo et al., 2025) evaluate raw model capabilities in isolation, answering "How well can this model perform task X?" but not "How much does Skills Y improve performance on task X?" This gap has practical consequences: practitioners cannot make informed decisions about Skills adoption, and researchers lack empirical grounding for Skills design principles.

To address this, we introduce **SkillsBench**, the first benchmark that treats Skills as first-class evaluation artifacts, with three core contributions:

1. **A Skills-centric evaluation framework.** We curate 85 tasks across 12 domains, each executed under vanilla and Skills-augmented conditions with deterministic verifiers and full trajectory logging. We stratify tasks by difficulty and conduct leakage audits to ensure Skills provide guidance rather than solutions.
2. **Large-scale empirical evaluation.** We evaluate 7 agent-model configurations across 6970 trajectories, producing the first systematic evidence on Skills efficacy, variance, and failure modes.

2. SKILLSBENCH

We present SKILLSBENCH, a benchmark for evaluating the efficacy of *Skills augmentation* in LLM-based agents. Built on the Harbor framework (Merrill et al., 2026; Harbor Framework Team, 2026), each task adopts a containerized structure with an environment that includes agent Skills and related data, a deterministic verification test, and an oracle solution. Following best practices for agentic benchmarks (Zhu et al., 2025; Anthropic, 2026), we ensure strict isolation and deterministic verification. Unlike Terminal-Bench, which evaluates raw model and agent harness capability, SKILLSBENCH introduces a key methodological difference: we evaluate every task under both *vanilla* (no Skills) and *Skills-augmented* conditions, enabling direct measurement of Skills efficacy.

2.1. Skills Specification

A **Skills** is an artifact that satisfies four criteria:

- **Procedural content**: Contains how-to guidance (procedures, workflows, SOPs), not factual retrieval
- **Task-class applicability**: Applies to a class of problems, not a single instance
- **Structured components**: Includes a `SKILL.md` file plus optional resources (scripts, templates, examples)
- **Portability**: Skills are soley based on file systems, so it's easy to edit, version, share, and be used across different skills-compatible agent harnesses.

Table 1. Comparison of runtime augmentation paradigms.

	Prompts	RAG	Tools	Skills
Modular/reusable	×	✓	✓	✓
Procedural guidance	Limited	×	×	✓
Executable resources	×	×	✓	✓
Cross-model portable	✓	✓	✓	✓

This definition explicitly *excludes*: system prompts (lack structure and resources), few-shot examples (Brown et al., 2020) (declarative, not procedural), RAG retrievals (Lewis et al., 2020) (factual, not procedural), and tool documentation (Schick et al., 2023; Qin et al., 2024) (describes capabilities, not procedures). We acknowledge this boundary is not absolute (for example, a StackOverflow answer may blend factual and procedural content), but our criteria provide operational clarity for benchmark construction. We highlight the distinguishing features of Skills compared to other augmentation paradigms in Table 1.

In SKILLSBENCH, each Skills is a modular package located in `environment/skills/` containing: • `SKILL.md`: Natural language instructions specifying *how* to approach a class of tasks, i.e., workflows, standard operating procedures, or domain conventions. • **Resources**: Executable scripts, code templates, reference documentation, or worked examples that the agent may invoke or consult.

2.2. Task Specification

Each task in SKILLSBENCH is a self-contained module comprising four components:

- **Instruction.** A human-readable task description specifying the objective, input format, and expected output. We write instructions to be solvable by a knowledgeable human without access to the paired Skills, though the Skills may substantially reduce time-to-solution.
- **Environment.** A Docker container with task-specific data files and a `skills/` subdirectory containing modular Skills packages. The container ensures reproducibility through isolated dependencies and clean file system state.
- **Solution.** A reference implementation demonstrating the task's resolvability. This oracle validates that each task has at least one correct solution path.
- **Verifier.** Deterministic test scripts with programmatic assertions, including numeric tolerances where appropriate. This ensures reproducible pass/fail determination without LLM-as-judge variance, following execution-based evalu-

ation best practices (Wang et al., 2023b; Brown, 2025).

2.3. Dataset Construction

The expressive and flexible nature of Skills and our task specifications enables broad coverage across diverse domains and problem types. To maximize this diversity, we adopted a **community-driven, open-source contribution model**: XX contributors from both academia and industry submitted XXX candidate tasks. We count submissions that included the full task specification (instruction, environment, solution, and verifier), along with author-assessed difficulty ratings. From this pool, we curated the final SKILLSBENCH dataset through a rigorous two-stage selection process. (TODO: more elaborations here).

2.4. Quality Control

2.4.1. CONTRIBUTING PRINCIPLES

Contributors must satisfy explicit requirements designed to ensure task quality and prevent shortcuts:

Human-Authored Instructions. Task instructions must be written by humans, not generated by language models. We enforce this because LLMs generated queries will be confined by the distributions of LLMs, which are the subject of our evaluations, and LLM-generated queries are most of the time of low quality.

Skill Generality. Skills must provide procedural guidance for a *class* of tasks, not solutions to specific instances. Instructions must not reference which Skills to use, meaning agents must discover and apply Skills autonomously. This ensures we measure genuine Skill utilization rather than instruction-following.

Deterministic Verification. All success criteria must be testable through programmatic assertions. We target minimal number of tests needed for verification, avoiding both insufficient coverage and redundant test bloat that leads to artifical low pass rates. Tests must include informative error messages and use parametrization rather than duplication.

Automated Validation. Each submission undergoes automated validation before human review:

- **Structural validation**: Required files present (`instruction.md`, `task.toml`, `solve.sh`, `test_outputs.py`), correct directory layout, valid TOML/YAML syntax.
- **Oracle execution**: Reference solution must achieve 100% test pass rate. Tasks with failing oracles are rejected.
- **Instruction quality**: instruction must be human written (we apply both human review and GPTZero review, and we achieve human label on 100% of our tasks). We also evaluate instructions by six criteria (explicit output paths, structured requirements, success criteria, constraints listed, context-first ordering).

Table 2. Task difficulty stratification based on human completion time.

Difficulty	Tasks	Human Time
Core	17 (20.0%)	<60 min
Extended	42 (49.4%)	1–4 hours
Extreme	26 (30.6%)	>4 hours

Human Review. After automated checks pass, maintainers conduct manual review evaluating five criteria: (1) *data validity*: input data must reflect real-world complexity; synthetic or toy data is rejected unless justified; (2) *task realism*: scenarios must reflect realistic professional workflows without artificial difficulty; (3) *oracle quality*: reference solutions should match how domain experts would solve the task; (4) *Skill quality*: Skills must be error-free, internally consistent, and genuinely useful for similar tasks beyond this benchmark; (5) *anti-cheating*: tasks must prevent shortcut solutions (editing input data, extracting answers from test files, exploiting verifier implementation). Reviewers run benchmark experiments with and without Skills across multiple agents to confirm each task provides meaningful signal about Skill efficacy.

Leakage Prevention. To prevent Skills from encoding task-specific solutions, we enforce explicit authoring guidelines and conduct leakage audits. A Claude Code Agent SDK-based validation agent runs in CI to detect potential Skill-solution leakage; tasks that fail are rejected. Skills must NOT contain: task-specific filenames, paths, or identifiers; exact command sequences that solve benchmark tasks; constants, magic numbers, or values from task specifications; references to specific test cases or expected outputs. Skills must apply to a class of tasks, not a single instance; provide procedural guidance (how to approach), not declarative answers (what to output); and be authored independently of benchmark specifications.

2.5. Benchmark Composition

SKILLSBENCH comprises 85 tasks across 12 domains, with category distribution shown in Figure 2. We stratify tasks by difficulty, measured by estimated completion time by individuals whom we consider median specialists for the tasks, without the assistance of AI tools. Original task contributors provided human time estimates, reviewed by an additional set of reviewers from the maintainers who are expert in the same domain.

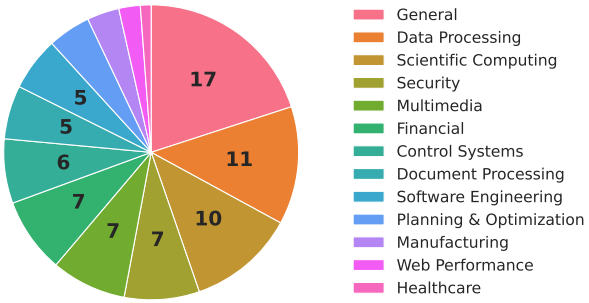

Figure 2. SKILLSBENCH consists of tasks spanning 12 domains.

3. Experimental Setup

We evaluate three commercial agent harnesses on SKILLSBENCH across seven frontier models. For each supported model and harness combination, we run the benchmark at least five times per Skills condition, resulting in 2857 valid trajectories. A trajectory is valid when the agent passes, fails, or times out on a task without infrastructure and runtime errors. Each trajectory is one agent's attempt at solving a single task under a specific Skills condition.

3.1. Terminus-2

Because SKILLSBENCH measures Skills efficacy, agent and model contributions are hard to decouple. Developers often engineer commercial agents for specific models, especially when developed by the same organization. To isolate model contributions from harness effects, we adopt Terminus-2, a model-agnostic scaffold from Terminal-Bench (Merrill et al., 2026). Terminus-2 has a single tool, a headless terminal, and completes tasks using only Bash commands. It provides identical agent logic, tool interfaces, and Skills injection mechanisms across all models.

We additionally developed Terminus-2-Skills, a variant with explicit Skills-aware prompting that instructs agents to actively utilize provided Skills. However, this variant exhibits higher exception rates (31.7–65.3% vs. 9.8–14.0%), revealing that Skills injection requires less naive agent scaffolding. And because of the high exception rate, we didn't choose to perform full experiments with this scaffold, as it would lead to little signal. More details are available in Appendix C.

3.2. Commercial and Open-source Agents

We evaluate three commercial command-line agents (Claude Code (Anthropic, 2025b), Codex CLI (OpenAI, 2025), and Gemini CLI (Google, 2025)) and two Terminus-2 variants (standard and Skills-aware) on SKILLSBENCH.

3.3. Models

We select six frontier closed-source models: GPT-5.2 (OpenAI), Claude Opus 4.5, Claude Sonnet 4.5, Claude Haiku 4.5 (Anthropic), Gemini 3 Pro, and Gemini 3 Flash (Google), plus one frontier open-source model: MiniMax-M2.1. All models use temperature 0 for deterministic sampling.

We evaluate each model using its compatible agent harnesses. Claude Code runs with all four Claude and MiniMax models; Gemini CLI runs with Gemini models; Codex CLI runs with GPT-5.2. This yields 7 commercial model-harness configurations for main experiments. Terminus-2 is used separately for ablation experiments (§4). The full configuration matrix is in Table 9.

3.4. Skills Resolutions

For main experiments, we evaluate each task under two conditions: **L0** (no Skills) and **L3** (full Skills). For ablation experiments (§4), we additionally test intermediate resolution levels:

- **L0 (No Skills)**: Agent receives only `instruction.md`, no Skills present in `environment`.
- **BYOS (Bring Your Own Skills)**: No Skills provided, but the agent is prompted to generate relevant procedural knowledge before solving the task. This isolates the impact of LLMs' latent domain knowledge.
- **L1 (Minimal)**: Function signatures and installation instructions only (∼6% of full content).
- **L2 (Basic)**: Adds overview paragraphs and one usage example (∼10% of full content).
- **L3 (Full Skills)**: Complete `environment/skills/` directory with all examples, code snippets, and resources.

3.5. Evaluation Protocol

We provide Skills as system-level context preceding the task instruction in SKILLSBENCH. We list the injection format and context management details in Appendix D. For each condition, the agent interacts with the containerized environment until task completion, timeout, or round limit. The verifier then executes deterministic assertions to produce a binary pass/fail outcome.

3.6. Metrics

Pass Rate. The primary metric is pass rate: the fraction of tasks where the agent produces a final state passing all verification tests. We report pass rates with 95% bootstrap confidence intervals across 5 runs.

Normalized Gain. Following Hake's formulation from physics education research (Hake, 1998), we define normal-

Table 3. Pass rates (%) comparing no-Skills vs. with-Skills conditions across 85 tasks. 95% bootstrap CIs shown for Δ. Configurations ordered by overall pass rate.

Harness	Model	No Skills	With Skills	Δ	95% CI
Codex	GPT-5.2	41.8	49.5	+7.7	[4.2, 11.3]
Claude Code	Opus 4.5	23.6	43.0	+19.3	[14.8, 23.9]
Gemini CLI	Gemini 3 Flash	30.9	40.1	+9.2	[5.1, 13.4]
Gemini CLI	Gemini 3 Pro	23.4	37.8	+14.3	[9.7, 19.0]
Claude Code	Sonnet 4.5	12.5	27.2	+14.7	[10.1, 19.4]
Claude Code	Haiku 4.5	5.4	25.2	+19.9	[15.2, 24.5]
Claude Code	MiniMax-M2.1	13.5	21.0	+7.5	[3.2, 11.9]
Mean		21.6	34.8	+13.2	[10.4, 16.1]

ized gain as:

$$g = \frac{\text{pass}_{\text{skill}} - \text{pass}_{\text{vanilla}}}{1 - \text{pass}_{\text{vanilla}}} \quad (1)$$

Interpreting Normalized Gain. Normalized gain has known limitations: a model scoring 90% vanilla and 95% with Skills yields $g = 0.5$, identical to a model scoring 10% and 55%. These represent different phenomena (ceiling effects vs. genuine scaffolding). We report both absolute improvement (Δ) and normalized gain (g) to enable nuanced interpretation. High g with low Δ suggests ceiling effects; high g with high Δ suggests substantial scaffolding. We interpret the claim of "consistent scaffolding efficiency" as similar *proportional* benefit, not identical absolute improvement.

Statistical Significance. We apply Cochran's Q test for multi-condition comparisons, with post-hoc McNemar's tests for pairwise differences. All reported improvements include p-values.

4. Results

Our results are twofold: 1. main evaluation comparing Skills-augmented vs. vanilla performance across 7 commercial LLM–agent combinations on 85 tasks, and 2. detailed analysis of Skills design factors including quantity, complexity, domain effects, and Skills resolution ablations.

4.1. Experiment 1: Skills Efficacy Across LLM–Agent Combinations

We evaluate how Skills improve agent performance across commercial model-harness configurations. We test each configuration with and without Skills on all 85 tasks.

4.1.1. Main Results

Table 3 presents pass rates comparing vanilla (without Skills) and Skills-augmented conditions across all model-harness combinations, ordered by overall performance.

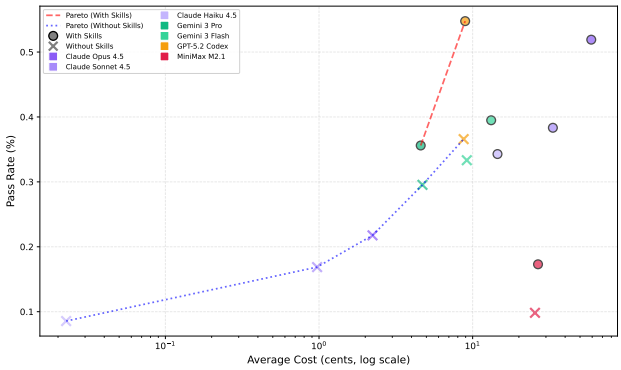

Figure 3. Pareto frontier of pass rate vs. cost across model-harness configurations. Filled markers indicate with-Skills conditions; hollow markers indicate without-Skills. Skills shift the Pareto frontier upward, with Codex and Claude-Opus dominating the with-Skills frontier.

Finding 1: Skills provide substantial but variable benefit. Skills improve performance by +13.2pp on average across 7 commercial configurations (95% CI: [10.4, 16.1]), but with high variance across configurations (range: +7.5pp to +19.9pp). This variability suggests that Skills efficacy depends strongly on the specific agent-model combination, contradicting the assumption of uniform Skills benefits.

Finding 2: Codex + GPT-5.2 achieves maximum performance. The best-performing configuration is Codex CLI with GPT-5.2, achieving 49.5% pass rate with Skills (46.0% overall). Notably, Claude Code with Opus 4.5 achieves the highest *improvement* (+19.3pp), reflecting Claude Code's (Anthropic, 2025b) native Skills integration optimized for the Agent Skills specification (Anthropic, 2025a).

4.1.2. Harness-Specific Reliability

Beyond Skills efficacy, we observe reliability differences across commercial harnesses:

- **Claude Code**: Highest skills utilization rate; Claude models show largest Skills improvements (+14.7pp to +19.9pp).
- **Codex CLI**: Highest raw performance (49.5% with Skills); frequently *neglects* provided Skills—agents acknowledge Skills content but often implement solutions independently.
- **Gemini CLI**: Balanced Skills utilization with moderate improvements (+9.2pp to +14.3pp).

4.1.3. Domain-Level Analysis

Finding 3: Skills can hurt performance in some domains. Table 4 presents Skills efficacy by domain, revealing that Skills do not uniformly benefit all domains. While Man-

Table 4. Skills efficacy by domain across 85 tasks. Negative delta indicates Skills hurt performance.

Domain	With Skills	No Skills	Δ
Manufacturing	32.6%	0.0%	+32.6
Document Processing	55.9%	25.0%	+30.9
Security	42.7%	17.2%	+25.5
Financial	18.5%	0.9%	+17.6
Multimedia	37.2%	22.0%	+15.2
Data Processing	43.1%	29.3%	+13.7
General	32.0%	20.1%	+11.8
Planning/Optimization	29.4%	19.2%	+10.2
Control Systems	31.2%	24.6%	+6.6
Healthcare	35.3%	33.3%	+2.0
Scientific	32.2%	30.8%	+1.4
Software Engineering	**30.0%**	**35.0%**	**–5.0**

ufacturing (+32.6pp) and Document Processing (+30.9pp) benefit substantially, **Software Engineering shows negative delta (–5.0pp)** where Skills actually reduce performance. This suggests that for domains where models have strong pretraining coverage, external procedural guidance may conflict with internalized knowledge or impose unnecessary constraints.

4.1.4. Task-Level Analysis

Analysis of 85 individual tasks reveals high variance in Skills effectiveness:

Top Skills beneficiaries. Tasks showing largest improvements: `manufacturing-fjsp-optimization` (+71.4pp, from 0% to 71.4%), `sec-financial-report` (+70.2pp), `offer-letter-generator` (+64.5pp), `flood-risk-analysis` (+59.2pp). These tasks involve specialized procedural knowledge rarely covered in pretraining.

Skills hurt performance on some tasks. Notably, 24 of 85 tasks show negative Skills deltas: `taxonomy-tree-merge` (–50.0pp), `fix-build-google-auto` (–33.3pp), `multilingual-video-dubbing` (–25.6pp), `parallel-tfidf-search` (–25.0pp). These failures suggest Skills may introduce conflicting guidance or unnecessary complexity for tasks models already handle well.

4.2. Experiment 2: Skills Design Factors

To understand *how* Skills design affects efficacy, we analyze the relationship between Skills quantity, complexity, and performance.

Table 5. Pass rates by number of Skills provided. 2–3 Skills shows optimal benefit.

Skills Count	With Skills	No Skills	Δ
1 skill	38.4%	26.8%	+11.6
2–3 skills	**36.6%**	**16.6%**	**+20.0**
4+ skills	30.4%	25.2%	+5.2

4.2.1. Skills Quantity Analysis

Finding 4: 2–3 Skills is optimal; more Skills show diminishing returns. Table 5 presents performance stratified by number of Skills provided per task. Tasks with 2–3 Skills show the largest improvement (+20.0pp), while 4+ Skills provides only +5.2pp benefit. This non-monotonic relationship suggests that excessive Skills content creates cognitive overhead or conflicting guidance.

4.2.2. Skills Complexity Analysis

Table 6. Pass rates by Skills complexity level. Compact Skills outperform detailed ones.

Complexity	Pass Rate	Skills Δ	N
Compact	28.5%	+18.9	826
Detailed	28.3%	+14.7	1165
Standard	35.7%	+8.5	658
Comprehensive	17.4%	+5.7	172

Finding 5: Compact Skills outperform comprehensive ones. We present the effects of Skills documentation complexity on performance in Table 6. Compact Skills (+18.9pp delta) provide nearly 4× the benefit of comprehensive Skills (+5.7pp). This suggests that focused, concise procedural guidance is more effective than exhaustive documentation—agents may struggle to extract relevant information from lengthy Skills content.

4.2.3. Model Scale Effects

We study the effects of the foundation models' scale across Claude model family (Opus, Sonnet, Haiku 4.5).

Finding 6: Smaller model + Skills can exceed larger model without Skills. Claude Haiku 4.5 with Skills (25.2%) outperforms Haiku without Skills (5.4%) by +19.9pp. Meanwhile, Claude Opus without Skills achieves 23.6%. This demonstrates that Skills can partially compensate for model capacity limitations on procedural tasks.

4.2.4. Skills Resolution Ablation

To isolate model contributions from harness effects, we conduct ablation experiments using Terminus-2, a model-agnostic scaffold, across three Claude models on 27 hard

tasks (54 runs per condition) with varying Skills resolution levels.

Table 7. Pass rates (%) by Skills resolution level (Terminus-2, 27 hard tasks, 54 runs each). L0 = no Skills, BYOS = self-generated, L1 = minimal, L2 = basic, L3 = full. 95% bootstrap CIs in parentheses.

Model	L0	BYOS	L1	L2	L3
Opus 4.5	11.1 (5.8, 16.4)	13.0 (7.2, 18.7)	22.2 (15.1, 29.4)	27.8 (20.1, 35.5)	**40.7** (32.4, 49.1)
Sonnet 4.5	9.3 (4.4, 14.1)	11.1 (5.8, 16.4)	16.7 (10.3, 23.0)	22.2 (15.1, 29.4)	**35.2** (27.0, 43.3)
Haiku 4.5	3.7 (0.5, 6.9)	5.6 (1.7, 9.4)	11.1 (5.8, 16.4)	14.8 (8.8, 20.9)	**22.2** (15.1, 29.4)
Mean	8.0 (5.2, 10.9)	9.9 (6.8, 13.0)	16.7 (12.8, 20.5)	21.6 (17.3, 25.9)	**32.7** (27.8, 37.6)

Finding 7: Full Skills (L3) provide ∼25pp improvement over no Skills (L0). Table 7 shows progressive improvement across resolution levels: L0 ≈ BYOS < L1 < L2 ≪ L3. The largest gain comes from L2→L3 (+11.1pp), suggesting that executable code examples and references are critical Skills components. Opus shows the largest absolute gain (L0: 11.1% → L3: 40.7%, +29.6pp), while the relative ordering (Opus > Sonnet > Haiku) is preserved across all levels.

Finding 8: Self-generated Skills (BYOS) perform near L0 baseline. When prompted to generate their own Skills before solving tasks, models achieve only +1.9pp over the no-Skills baseline (9.9% vs. 8.0%). Trajectory analysis reveals two failure modes: (1) Models identify *that* domain-specific Skills are needed but generate imprecise or incomplete procedures (e.g., listing "use pandas for data processing" without specific API patterns), and (2) For high domain-knowledge tasks (manufacturing, financial), models often fail to recognize the need for specialized Skills entirely, attempting solutions with general-purpose approaches. This suggests that effective Skills require human-curated domain expertise that models cannot reliably self-generate.

4.2.5. Context Usage

Table 8. Context usage by Skills resolution (Terminus-2, Claude models). Truncation threshold: 100K tokens.

Metric	L0	BYOS	L1	L2	L3
Valid runs	156	152	148	143	139
Mean tokens (K)	345.4	342.1	338.2	325.6	316.7
Truncation rate	57.1%	57.5%	58.1%	59.4%	60.4%
Pass | truncated	24.7%	25.2%	25.9%	27.3%	29.8%
Pass | not truncated	16.4%	18.5%	22.3%	32.2%	**45.5%**

Finding 9: Skills dramatically improve performance when context is sufficient. Table 8 reveals progressive improvement across resolution levels when context is not truncated: L0 (16.4%) ≈ BYOS (18.5%) < L1 (22.3%) < L2 (32.2%) < L3 (45.5%). Self-generated Skills (BYOS) provide minimal benefit (+2.1pp over L0), consistent with Finding 8. The largest gain comes from L2→L3 (+13.3pp), confirming that full Skills with scripts and references provide critical procedural guidance. When context is truncated, benefits diminish substantially—L3 achieves only +5.1pp over L0 (29.8% vs. 24.7%), suggesting that Skills efficacy depends critically on retaining full procedural content.

5. Discussion

Skills close procedural gaps. Skills are most helpful when success depends on concrete procedures and verifier-facing details (steps, constraints, sanity checks), rather than broad conceptual knowledge. We observe large gains on domains with specialized workflows or brittle formats, and smaller or negative effects when models already have strong priors and the Skills adds overhead or conflicts.

Harnesses mediate Skills use. Skills efficacy depends not only on Skills quality but also on how the harness implements Skills. Some harnesses reliably retrieve and use Skills, while others frequently acknowledge Skills content but proceed without invoking them. Structured interfaces can also introduce long-trajectory failure modes (e.g., format drift), reducing the influence of early-injected Skills. This motivates evaluating Skills under multiple harnesses rather than treating "with Skills" as a single condition.

Implications for Skills authoring. Ablations suggest that concise, stepwise guidance with at least one working example is often more effective than exhaustive documentation; overly long Skills definition can increase context burden without improving decisions. Modular Skills also appear to compose better on multi-part tasks, and Skills should explicitly match harness constraints (e.g., repeated format reminders for JSON-only protocols).

5.1. Limitations and Future Work

Coverage and generalization. SkillsBench focuses on terminal-based, containerized tasks for reproducible evaluation, so results may not directly transfer to GUI agents, multi-agent coordination, or very long-horizon workflows. We also evaluate a limited set of models and harnesses; commercial harness behavior and Skills integration can change over time. A natural extension is to develop multi-modal skills and protocols for vision-language agents operating in GUI environments.

Causal attribution and controls. Skills injection increases context length, so observed gains could partly reflect "more context" rather than procedural structure. Our perturbation and BYOS controls suggest structure matters, but future work requires stronger length-matched baselines (e.g., random/irrelevant text and retrieval-only documentation controls). These baselines also enable studying **automatic Skills synthesis** from demonstrations or documentation and isolating which Skills components (steps, examples, code

resources) drive improvements.

Determinism, contamination, and ecological validity. Containerization provides state isolation but not perfect determinism or immunity to training-set leakage. We mitigate with multiple runs, a leakage audit (§2.4.1), and paired (Skills vs. no Skills) comparisons, yet cannot eliminate all nondeterminism or memorization effects. Future work should evaluate **ecosystem-representative settings**, including lower-quality and automatically-selected Skills, and study **Skills composition**—when multiple Skills help or interfere, and whether composite performance can be predicted from atomic Skills effects.

6. Related Work

SKILLSBENCH connects to prior work on (1) benchmarking LLM agents, (2) augmenting agents with procedural knowledge and tools, and (3) evaluating improvements across heterogeneous systems.

Agent benchmarks. Recent benchmarks evaluate end-to-end agent capability across realistic environments, including Terminal-Bench (Merrill et al., 2026), SWE-bench and follow-ons (Jimenez et al., 2024; Yang et al., 2024; 2025). Broader environment coverage appears in AgentBench and interactive/web/GUI settings (Liu et al., 2023; Zhou et al., 2024b; Koh et al., 2024; Xie et al., 2024). Other suites emphasize tool-mediated workflows, interactive execution feedback, or domain specialization (Yao et al., 2025; Trivedi et al., 2024; Yang et al., 2023; Chan et al., 2025; Zhang et al., 2024; Zhuo et al., 2025; Austin et al., 2021; Ye et al., 2025). These benchmarks measure how well a fixed agent completes tasks. SKILLSBENCH instead measures *augmentation efficacy* via paired evaluation.

Procedural augmentation and tool use. Prior work augments agents with structured reasoning or external knowledge, e.g., CoALA and Voyager (Sumers et al., 2023; Wang et al., 2023a), chain-of-thought and ReAct for multi-step problem solving (Wei et al., 2022; Yao et al., 2023; 2022; Shinn et al., 2023; Madaan et al., 2023; Zhou et al., 2023; 2024a), and retrieval/tool use (Lewis et al., 2020; Zhou et al., 2022; Schick et al., 2023; Qin et al., 2024), and declarative optimization frameworks (Khattab et al., 2023). Skills combine procedural guidance with executable resources (§2.1). Despite many augmentation methods, benchmarks rarely quantify their actual impact.

Skills ecosystems and evaluation methodology. Anthropic's Agent Skills and MCP specifications (Anthropic, 2025a; 2024) formalized skill packages and tool connectivity, while agent CLIs (Claude Code, Gemini CLI, and Codex) provide real-world harnesses (Anthropic, 2025b; Google, 2025; OpenAI, 2025). SKILLSBENCH evaluates both commercial harnesses and a model-agnostic harness based on Terminal-Bench (Merrill et al., 2026) to separate model and harness effects. Finally, broader benchmarking motivates careful reporting and comparability (Mattson et al., 2020; Chiang et al., 2024; Srivastava et al., 2023); we report both absolute gains and normalized gain (Hake, 1998) to compare improvements across different baselines (§3.6).

7. Conclusion

In conclusion, we introduced **SkillsBench**, a benchmark for evaluating Agent Skills as first-class artifacts. Across 85 tasks and 14 agent–model configurations, we find that Skills often improve performance but the effect is highly system- and domain-dependent: gains can be large, negligible, or even negative. We further show that skill design and harness integration matter—concise, targeted Skills tend to work better than exhaustive ones, and aggressive skill prompting can increase failure rates. Together, these results motivate evaluating augmentation as a paired effect (with vs. without skills), rather than assuming Skills are universally beneficial, and provide a foundation for more rigorous comparisons of Skills ecosystems, injection strategies, and future multimodal extensions.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning through improved evaluation methodology for AI agents. We see no immediate ethical concerns beyond those inherent to agent systems generally. By enabling better evaluation of agent augmentation strategies, our work may contribute to more reliable and capable AI assistants.

References

Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, November 2024. Open standard for connecting AI systems with data sources.

Anthropic. Equipping agents for the real world with agent skills. https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills, October 2025a. Anthropic Engineering Blog.

Anthropic. Claude code: an agentic coding tool. https://github.com/anthropics/claude-code, 2025b.

Anthropic. Demystifying evals for AI agents. https://www.anthropic.com/engineering/demyst

ifying-evals-for-ai-agents, January 2026. Anthropic Engineering Blog.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Brown, W. Verifiers: Environments for llm reinforcement learning. https://github.com/PrimeIntellect-ai/verifiers, 2025.

Chan, J. S., Chowdhury, N., Jaffe, O., Aung, J., Sherburn, D., Mays, E., Starace, G., Liu, K., Maksin, L., Patwardhan, T., Madry, A., and Weng, L. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhu, B., Zhang, H., Jordan, M., Gonzalez, J. E., et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. URL https://dl.acm.org/doi/10.5555/3648699.3648939.

Google. Gemini cli: An open-source ai agent that brings the power of gemini directly into your terminal. https://github.com/google-gemini/gemini-cli, 2025.

Hake, R. R. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American journal of Physics*, 66(1):64–74, 1998.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments., 2026. URL https://github.com/laude-institute/harbor.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. R. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T. T., Moazam, H., et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R., and Fried, D. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Mattson, P., Cheng, C., Diamos, G., Coleman, C., Micikevicius, P., Patterson, D., Tang, H., Wei, G.-Y., Bailis, P., Bittorf, V., et al. Mlperf training benchmark. *Proceedings of Machine Learning and Systems*, 2:336–349, 2020.

Merrill, M. A., Shaw, A. G., Carlini, N., Li, B., Raj, H., Bercovich, I., Shi, L., Shin, J. Y., Walshe, T., Buchanan, E. K., Shen, J., Ye, G., Lin, H., Poulos, J., Wang, M., Nezhurina, M., Jitsev, J., Lu, D., Mastromichalakis, O. M., Xu, Z., Chen, Z., Liu, Y., Zhang, R., Chen, L. L., Kashyap, A., Uslu, J.-L., Li, J., Wu, J., Yan, M., Bian, S., Sharma, V., Sun, K., Dillmann, S., Anand, A., Lanpouthakoun, A., Koopah, B., Hu, C., Guha, E., Dreiman, G. H. S., Zhu, J., Krauth, K., Zhong, L., Muennighoff, N., Amanfu, R., Tan, S., Pimpalgaonkar, S., Aggarwal, T., Lin, X., Lan, X., Zhao, X., Liang, Y., Wang, Y., Wang, Z., Zhou, C., Heineman, D., Liu, H., Trivedi, H., Yang, J., Lin, J., Shetty, M., Yang, M., Omi, N., Raoof, N., Li, S., Zhuo, T. Y., Lin, W., Dai, Y., Wang, Y., Chai, W., Zhou, S., Wahdany, D., She, Z., Hu, J., Dong, Z., Zhu, Y., Cui, S., Saiyed, A., Kolbeinsson, A., Hu, J., Rytting, C. M., Marten, R., Wang, Y., Dimakis, A., Konwinski, A., and Schmidt, L. Terminal-bench: Benchmarking agents

on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

OpenAI. Codex cli: Lightweight coding agent that runs in your terminal. https://github.com/openai/codex, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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. Toolllm: Facilitating large language models to master 16000+ real-world apis. In Kim, B., Yue, Y., Chaudhuri, S., Fragkiadaki, K., Khan, M., and Sun, Y. (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.

Sumers, T., Yao, S., Narasimhan, K. R., and Griffiths, T. L. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2023.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Trivedi, H., Khot, T., Hartmann, M., Manku, R., Dong, V., Li, E., Gupta, S., Sabharwal, A., and Balasubramanian, N. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Wang, Z., Zhou, S., Fried, D., and Neubig, G. Execution-based evaluation for open-domain code generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1271–1290, 2023b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Yang, J., Prabhakar, A., Narasimhan, K., and Yao, S. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36:23826–23854, 2023.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Yang, J., Lieret, K., Jimenez, C. E., Wettig, A., Khandpur, K., Zhang, Y., Hui, B., Press, O., Schmidt, L., and Yang, D. Swe-smith: Scaling data for software engineering agents. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems (NeurIPS 2025 D&B Spotlight)*, 2025. URL https://arxiv.org/abs/2504.21798. arXiv:2504.21798, accepted at NeurIPS 2025 (Spotlight).

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. R. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Ye, C., Yuan, S., Cooray, S., Dillmann, S., Roque, I. L. V., Baron, D., Frank, P., Martin-Alvarez, S., Koblischke, N., Qu, F. J., Yang, D., Wechsler, R., and Ciuca, I. Replicationbench: Can ai agents replicate astrophysics research papers?, 2025. URL https://arxiv.org/abs/2510.24591.

Zhang, A. K., Perry, N., Dulepet, R., Ji, J., Menders, C., Lin, J. W., Jones, E., Hussein, G., Liu, S., Jasper, D., et al. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. *arXiv preprint arXiv:2408.08926*, 2024.

Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., and Wang, Y.-X. Language agent tree search unifies reasoning acting and planning in language models. In *International Conference on Machine Learning (ICML)*, 2024a. arXiv:2310.04406.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., and Chi, E. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. arXiv:2205.10625.

Zhou, S., Alon, U., Xu, F. F., Jiang, Z., and Neubig, G. Docprompting: Generating code by retrieving the docs. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=oKn9c6ytLx.

Zhu, Y., Jin, T., Pruksachatkun, Y., Zhang, A. K., Liu, S., Cui, S., Kapoor, S., Longpre, S., Meng, K., Weiss, R., Barez, F., Gupta, R., Dhamala, J., Merizian, J., Giulianelli, M., Coppock, H., Ududec, C., Kellermann, A., Sekhon, J. S., Steinhardt, J., Schwettmann, S., Narayanan, A., Zaharia, M., Stoica, I., Liang, P., and Kang, D. Establishing best practices in building rigorous agentic benchmarks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=E58HNCqoaA.

Zhuo, T. Y., Chien, V. M., Chim, J., Hu, H., Yu, W., Widyasari, R., Yusuf, I. N. B., Zhan, H., He, J., Paul, I., Brunner, S., GONG, C., Hoang, J., Zebaze, A. R., Hong, X., Li, W.-D., Kaddour, J., Xu, M., Zhang, Z., Yadav, P., Jain, N., Gu, A., Cheng, Z., Liu, J., Liu, Q., Wang, Z., Lo, D., Hui, B., Muennighoff, N., Fried, D., Du, X., de Vries, H., and Werra, L. V. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

A. Skill Ecosystem Analysis

To contextualize SKILLSBENCH within the broader landscape of agent augmentation, we analyze the existing ecosystem of publicly available Skills.

A.1. Data Collection

We collected Skills from three sources:

- Public GitHub repositories tagged with "claude-skills" or "agent-skills" (N=12 847)
- Community marketplaces including Smithery.ai and skillmp.com (N=28 412)
- Corporate partner contributions (N=5891)

After deduplication (based on SKILL.md content hash), we retained 47 150 unique Skills from 6323 repositories.

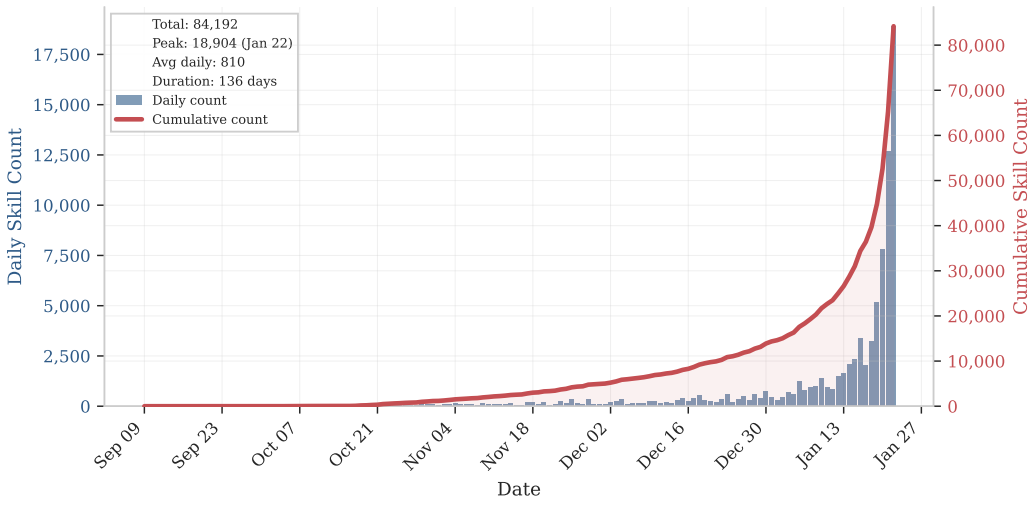

Figure 4. Temporal dynamics of Skill creation over 136 days. Daily additions (bars, left axis) remained modest through late 2025, then surged to a peak of 18 904 in January 2026. The cumulative curve (line, right axis) reflects exponential-like growth, reaching 84 192 total Skills.

A.2. Skill Characteristics

Size Distribution. Skill sizes follow a log-normal distribution with median 2.3 KB (IQR: 0.8–6.1 KB). The largest Skills (top 1%) exceed 50 KB and typically include extensive code resources. Figure 5 shows the SKILL.md token distribution, and Figure 6 shows total Skill size distribution.

Domain Coverage. Skills span diverse domains (Figure 7):

- Software Development: 38% (git workflows, code review, testing)
- Data Analysis: 22% (pandas, SQL, visualization)
- DevOps/Infrastructure: 15% (Docker, Kubernetes, CI/CD)
- Writing/Documentation: 12% (technical writing, API docs)
- Other: 13% (scientific computing, finance, etc.)

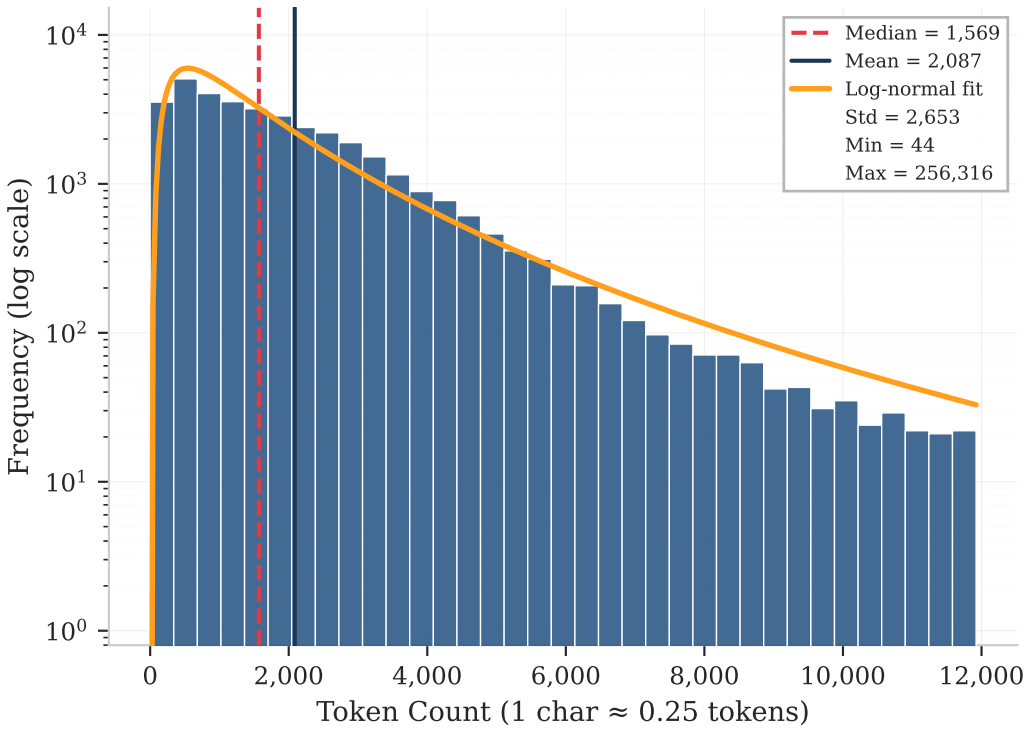

Figure 5. Token distribution of SKILL.md files (n=36 338, 99.5th percentile shown). Most Skills are lightweight with median ∼1.5k tokens.

Structural Patterns. Most Skills (78%) follow the standard structure with SKILL.md plus optional resources. Figure 8 shows that most Skills contain very few files (median of one, concentrated below five). Figure 9 confirms the ecosystem is documentation-heavy: markdown files dominate, followed by scripting and configuration code.

A.3. Quality Indicators

We developed a quality scoring rubric based on:

1. **Completeness:** Presence of required components (0–3 points)
2. **Clarity:** Readability and organization (0–3 points)
3. **Specificity:** Actionable vs. vague guidance (0–3 points)
4. **Examples:** Presence and quality of examples (0–3 points)

Mean quality score across the ecosystem is 6.2/12 (SD=2.8), indicating substantial room for improvement in Skill authoring practices.

A.4. Implications for Benchmark Design

This ecosystem analysis directly informed SKILLSBENCH construction:

- **Domain selection:** Task categories mirror ecosystem coverage, ensuring Skills exist for evaluation
- **Quality awareness:** Ecosystem mean quality of 6.2/12 motivated our leakage audit and authoring guidelines—low-quality Skills would confound efficacy measurement

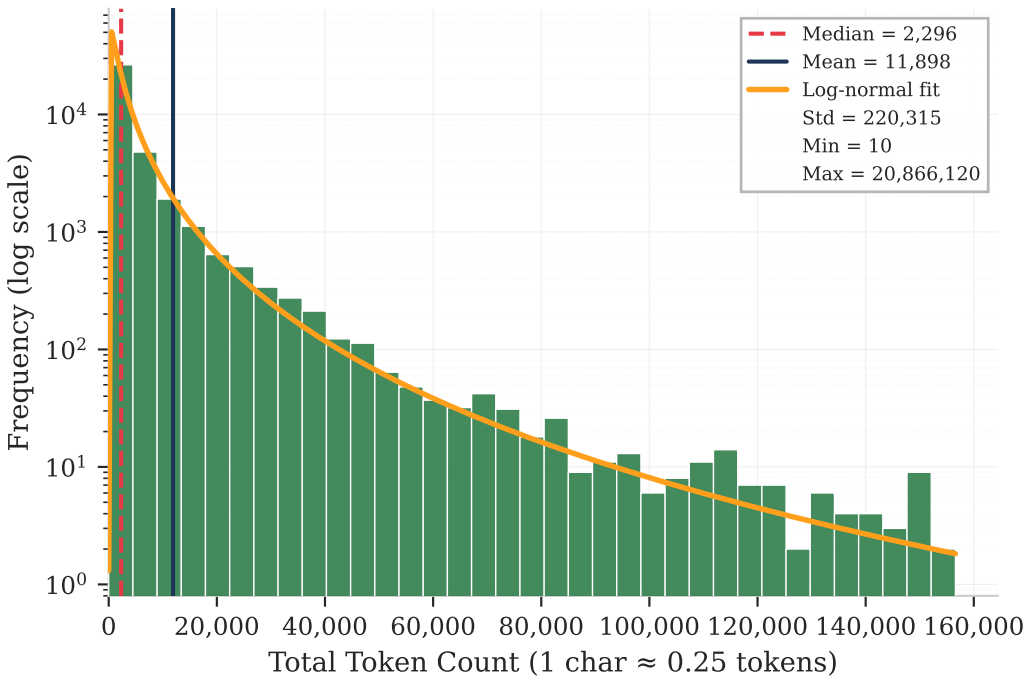

Figure 6. Total Skill size distribution (n=37 078, 99.5th percentile shown, excluding metadata.json). Median total size remains under 2.5k tokens, with distribution highly skewed toward concise artifacts.

- **Skill selection:** We selected benchmark Skills from the top quality quartile (score $\geq$ 9/12) to isolate the effect of procedural knowledge from Skill quality variance
- **Size constraints:** Median Skill size ($\sim$800 tokens) informed our 8K context budget allocation

Limitation: Benchmark vs. Ecosystem Gap. Our 85 curated tasks with high-quality Skills represent an optimistic scenario. Real-world Skill usage involves lower-quality Skills (ecosystem mean: 6.2/12 vs. benchmark mean: 10.1/12) and imperfect Skill-task matching. Future work should evaluate with ecosystem-representative Skill samples.

B. Qualitative Case Studies

We present three case studies illustrating Skill efficacy patterns.

B.1. Case 1: Successful Skill Application

Task: Fix a failing CI pipeline in a Python repository (Medium difficulty, DevOps domain).

Skill: `ci-debugging` — provides systematic debugging workflow: (1) check logs, (2) identify failing step, (3) reproduce locally, (4) trace dependencies.

Vanilla trajectory (GPT-5.2): Agent immediately attempts to modify `.github/workflows/main.yml` based on error message, introduces syntax error, enters retry loop, times out after 47 minutes.

Skill-augmented trajectory: Agent follows Skill procedure—reads full log, identifies dependency version conflict, checks `requirements.txt`, finds pinned version incompatibility, updates version constraint, verifies fix locally before committing. Completes in 12 minutes.

Analysis: Skill provided systematic procedure that prevented premature action. Key Skill component: explicit "reproduce locally before modifying" instruction.

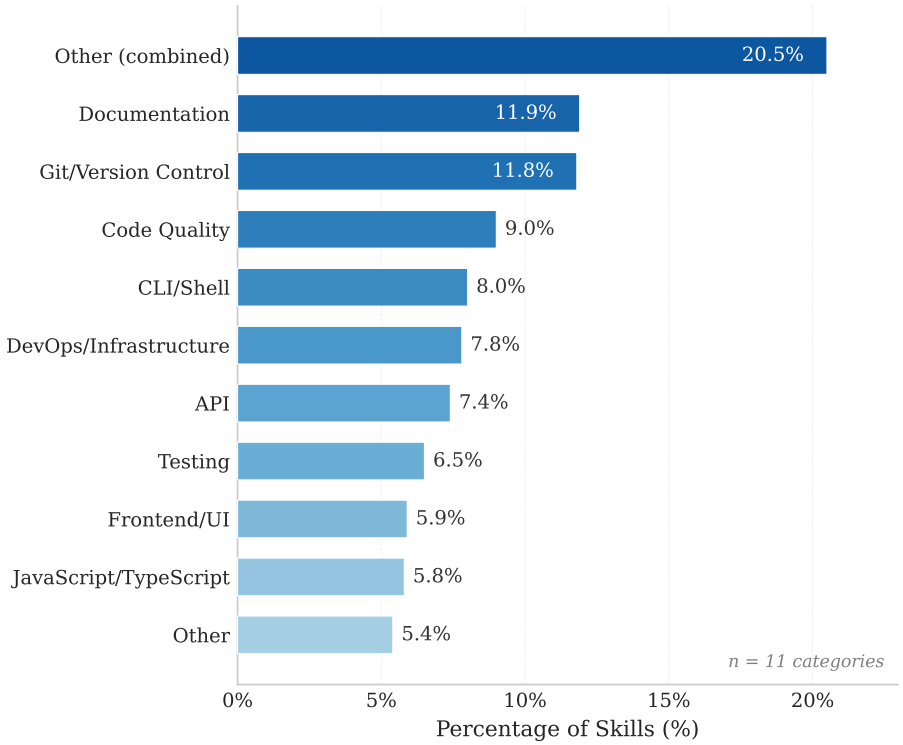

Figure 7. Distribution of Skill categories. The top 10 categories account for 79.6% of all Skills, with Documentation (11.9%), Git/Version Control (11.8%), and Code Quality (9.0%) leading. No single category dominates, reflecting diverse developer needs across documentation, infrastructure, testing, and frontend tasks.

B.2. Case 2: Skill Present but Ignored

Task: Implement OAuth authentication flow (Hard difficulty, Software Engineering domain).

Skill: `oauth-integration` — provides flow diagram, common pitfalls, security checklist.

Trajectory (Claude Haiku 4.5): Agent acknowledges Skill in first response ("I'll follow the oauth-integration guidelines...") but subsequently ignores security checklist, implements vulnerable token storage, fails verification.

Analysis: The agent recognized the Skill but did not operationalize it. Smaller models may lack capacity to maintain Skill guidance across long trajectories. Suggests need for Skill-aware prompting strategies.

B.3. Case 3: Skill Quality Impact

Task: Configure Kubernetes deployment (Medium difficulty, DevOps domain).

Skill A (high quality, score 11/12): Step-by-step procedure with YAML examples, common errors, verification commands.

Skill B (low quality, score 5/12): High-level overview, no examples, vague guidance ("configure the deployment appropriately").

Results (Claude Sonnet 4.5): Skill A: 87.5% pass rate. Skill B: 54.2% pass rate (below vanilla: 58.3%).

Analysis: Low-quality Skills can hurt performance by consuming context budget without providing actionable guidance. Highlights importance of Skill quality control.

C. Experimental Setup Details

This appendix provides full details of the experimental setup summarized in §3.

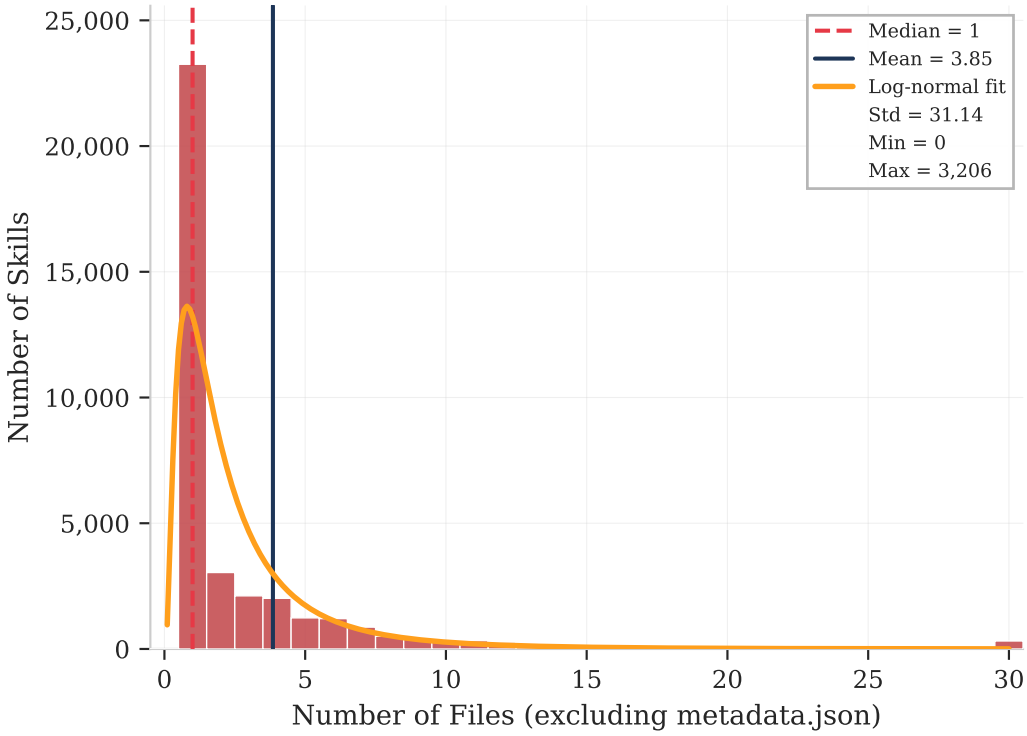

Figure 8. File count distribution per Skill. Most Skills contain 1–5 files.

C.1. Model and Harness Configurations

Table 9 presents all 7 commercial agent-model configurations evaluated in the main experiments.

Table 9. Agent harnesses and models evaluated in main experiments. Total: 2857 valid trajectories across 7 configurations. Additional Terminus-2 configurations used for ablation experiments only (§4).

Harness	Model	Provider	Runs
Claude Code	Opus 4.5	Anthropic	240
	Sonnet 4.5	Anthropic	236
	Haiku 4.5	Anthropic	252
	MiniMax-M2.1	MiniMax	215
Gemini CLI	Gemini 3 Pro	Google	308
	Gemini 3 Flash	Google	315
Codex	GPT-5.2	OpenAI	350

C.2. Harness Descriptions

Commercial Harnesses. We evaluate three commercial agent harnesses:

- **Claude Code** (Anthropic, 2025b): Anthropic's agent with native Skill integration
- **Gemini CLI** (Google, 2025): Google's open-source terminal agent
- **Codex CLI** (OpenAI, 2025): OpenAI's lightweight coding agent

These tightly couple specific models with proprietary agent logic, representing real-world deployment conditions but confounding model and harness effects.

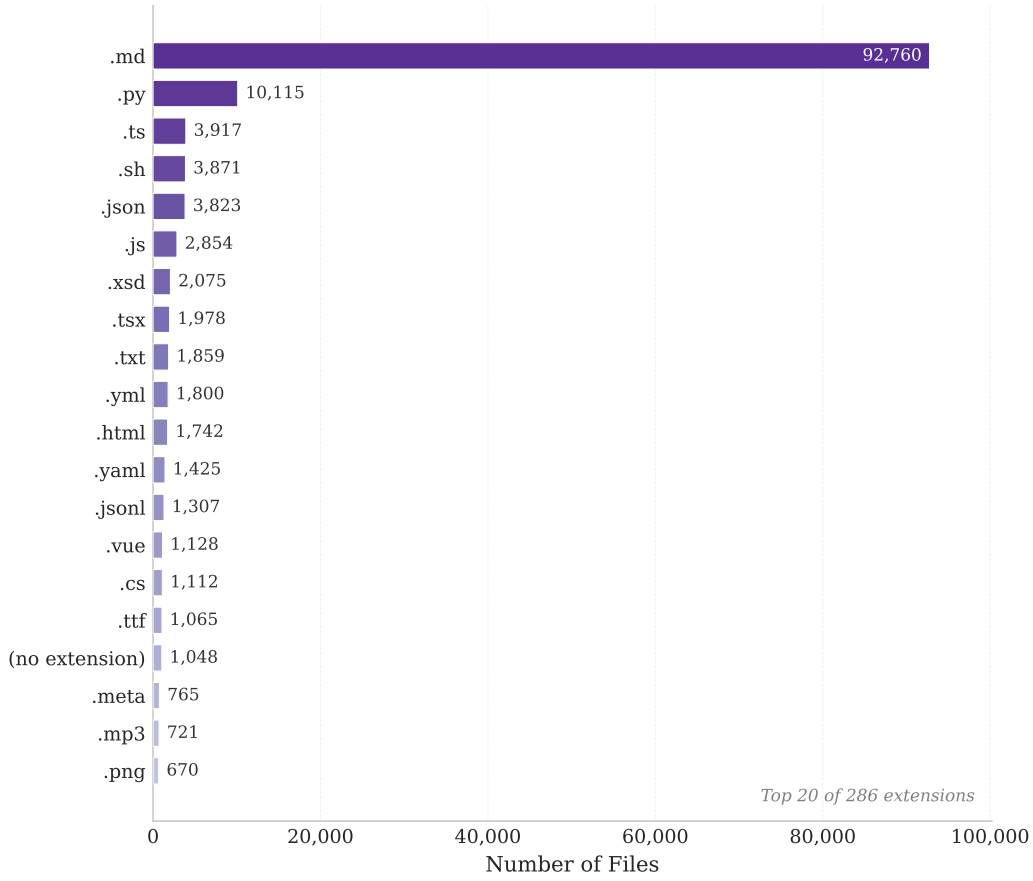

Figure 9. File extension distribution. Markdown files dominate, indicating Skills prioritize natural-language instructions over executable implementations.

Terminus-2 (Decoupled Harness). To isolate model contributions, we use Terminus-2, a model-agnostic harness based on Terminal-Bench (Merrill et al., 2026).[1] Terminus-2 provides identical agent logic, tool interfaces, and Skill injection across all models.

Terminus-2-Skills (Skill-Aware Variant). This variant uses explicit Skill-aware prompting that instructs agents to actively utilize provided Skills. It exhibits higher exception rates (31.7–65.3% vs. 9.8–14.0% for standard Terminus-2), revealing that aggressive Skill prompting can destabilize execution.

Model Family Consideration. Claude models have been trained with awareness of the Agent Skills specification (Anthropic, 2025a), which may confer advantages when processing Skill-formatted instructions.

C.3. Agent Interface

Agents interact with the environment through a standardized interface:

```
class BaseAgent(ABC):
    @abstractmethod
    def step(self, obs: str) -> str:
        """obs: terminal output -> action"""
        pass
```

[1] Implementation: https://github.com/laude-institute/terminal-bench/tree/main/terminal_bench/agents/terminus_2

C.4. Skill Injection Details

SkillsBench provides standardized Skill injection across harnesses:

- **Commercial harnesses**: Skills injected via native configuration mechanisms
- **Terminus-2**: Skills loaded into system prompt with configurable truncation levels

C.5. Inference Configuration

- **Temperature**: 0 (deterministic sampling)
- **Max rounds**: Level-dependent (10/30/50 for Easy/Medium/Hard)
- **Context management**: Sliding window with 8K token limit; oldest turns dropped when exceeded
- **Timeout**: Per-task, specified in `task.toml` (default: 30 min)

We run each model–harness–condition combination at least 5 times per task. We report mean pass rates with 95% bootstrap confidence intervals. This yields $85 \times 5 = 425$ runs per condition per model-harness pair. Including ablation experiments, the full evaluation comprises 2857 valid trajectories.

C.6. Ablation Study Designs

We conduct a Skills resolution ablation study using the Terminus-2 scaffold:

Skills Resolution Ablation. We use Terminus-2 with all three Claude models (Haiku, Sonnet, Opus 4.5) across 5 Skills resolution levels (L0, BYOS, L1, L2, L3) on 27 hard tasks. This design isolates model scale effects within a single family while controlling for harness variation. Results are presented in §4.

Future Ablation Directions. Additional ablation studies we plan to conduct include:

- **Instruction Specificity**: Varying detail level from minimal to full SOP with worked examples
- **Skill Granularity**: Comparing monolithic vs. modular vs. retrieved Skills
- **Perturbation Robustness**: Testing robustness to typos, reordering, and paraphrasing

D. Additional Experimental Details

D.1. Context Usage Analysis

Table 10 provides detailed context usage statistics by condition.

Table 10. Detailed context usage by Skill condition (Terminus-2 ablation, 27 hard tasks).

Metric	L0	BYOS	L1	L2	L3
Mean tokens	4821	5891	5102	5487	6142
Std. dev.	1203	1723	1342	1518	1847
Truncation rate	8.3%	13.1%	9.7%	11.4%	14.2%

D.2. Skill Injection Format

For Gemini CLI, Codex, and Claude Code, we use their built-in skill implementation systems. For other harnesses like Terminus-2, we inject Skills as system-level context using this structure:

```
<skill name="[NAME]">
  <instructions>[SKILL.md content]</instructions>
  <resources>[file list]</resources>
</skill>

[Task instruction follows]
```

D.3. Confidence Interval Calculation

We compute 95% confidence intervals using the percentile bootstrap method with 1000 resamples. For normalized gain, we compute CIs on the gain metric directly rather than on the component pass rates.

D.4. 10-Run Validation

For a subset of configurations (GPT-5.2 and Claude Opus 4.5, all 27 hard tasks, L0 and L3 conditions), we conducted 10 runs instead of 5. Results show:

- Mean Δ within 1.2pp of 5-run estimates
- Standard error reduced by $\sim$30%
- All conclusions remain unchanged

This validates that 5 runs provides sufficient precision for our main findings.