D-SKILL: DISTILLING TEACHER KNOWLEDGE THROUGH AGENT SKILLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier language-model agents excel at complex tasks, but high inference costs make transferring their capabilities to smaller models essential. Traditional knowledge distillation achieves this through parameter updates, requiring access to the student's weights and additional training compute while encoding the transferred knowledge implicitly in the parameters. We introduce D-Skill, a framework that treats the teacher as a black box and distills its behavioral advantage into an interpretable natural-language skill by analyzing paired rollouts and identifying behavioral divergences and agreements. The resulting lessons are consolidated offline, with any compatible skill-update backend, into a skill document that is provided to the student at inference time. D-Skill constructs the skill entirely offline from fixed trajectories, without labels, further environment interaction, or weight updates after rollout collection. Across SpreadsheetBench-Verified, ALFWorld, and chess puzzles spanning multiple teacher–student pairs, D-Skill closes 82%–95% of the teacher–student gap without touching model weights. After a single offline pass over the paired traces, D-Skill outperforms one-epoch SFT baselines and remains competitive with multi-epoch SFT. Under the same model family, the resulting student also incurs 5.7×–8.3× lower inference cost per task than the teacher. Our results demonstrate that natural-language skill distillation offers a performant, interpretable, and efficient alternative to weight-based capability transfer.

1 INTRODUCTION

Frontier agents can reliably solve complex agentic tasks (OpenAI, 2026b; Anthropic, 2026a; Jimenez et al., 2024; Yao et al., 2024; Xie et al., 2024; Merrill et al., 2026), but their high inference costs (Chen et al., 2023b) make it desirable to transfer their capabilities to cheaper student models (Hinton et al., 2015; Hsieh et al., 2023; Zeng et al., 2024; Chen et al., 2023a; 2024; Kang et al., 2026). Classical knowledge distillation performs this transfer by updating the student's parameters using supervision such as the teacher's output distribution (Hinton et al., 2015; Kim & Rush, 2016; Gu et al., 2023), complete teacher-generated trajectories (Ho et al., 2023; Hsieh et al., 2023; Zeng et al., 2024; Chen et al., 2023a; 2024; Kang et al., 2026), or teacher evaluations of student-generated samples (Agarwal et al., 2024; Bai et al., 2022; Lee et al., 2023). Despite differences in supervision, these approaches impose several constraints: weight access is required, excluding many proprietary API-served models; backpropagation adds substantial computational cost; and sufficient training data is needed for the student to generalize. Moreover, the transferred knowledge is encoded in the model's parameters, making it difficult to inspect, revise, or correct after training (Meng et al., 2022; Yao et al., 2023).

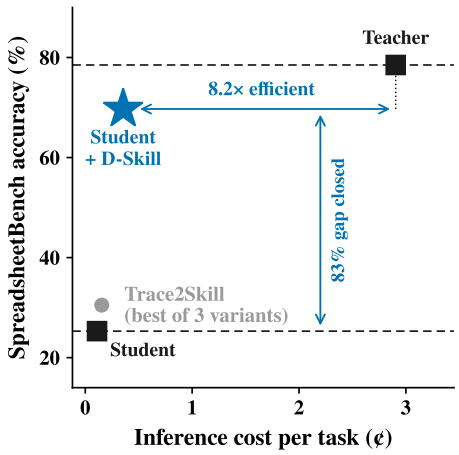


Figure 1: **D-Skill closes the gap at a fraction of its cost.** D-Skill closes 84% of the teacher-student gap at 8.3× lower inference cost on SpreadsheetBench-Verified. Teacher: GPT-5.6 Terra; student: GPT-5.6 Luna.

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.

Recent work has shown that *skills*, natural-language artifacts encoding reusable task knowledge, strategies, and procedures, can substantially improve agent performance without modifying model weights (Anthropic, 2025a; Ni et al., 2026; Yang et al., 2026; Zhang et al., 2026). Because skills remain external to the model, they are portable across deployments and can be inspected or edited after construction (Anthropic, 2025a). Existing skill-generation methods are generally not organized around the teacher–student relationship central to distillation. Teacher-driven methods compile stronger-model trajectories or teacher-generated modules without conditioning the transfer on the target student's behavior (Ni et al., 2026; Fu et al., 2024; Qiu et al., 2025). Student-centered methods instead learn from the target agent's own experience, including contrasts between successful and failed rollouts, without a matched stronger-agent reference (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024; Suzgun et al., 2026; Zhang et al., 2026; Yang et al., 2026; Ma et al., 2026). D-Skill instead constructs a student-targeted skill by comparing teacher and student trajectories collected on the same tasks. We ask whether their behavioral differences can be distilled, in a single offline pass over fixed traces, into an external textual update.

We propose **D-Skill**, a method for distilling model capabilities into a skill by comparing paired teacher and student rollouts. For each unlabeled task, we obtain one rollout from each model, either through fresh execution or from previously saved traces. A difference analyst extracts lessons from both divergences and agreements between the two behaviors. Divergences identify states in which the student behaves differently from the teacher and pair the student's behavior with the teacher's alternative, while agreements identify shared strategies that should be preserved. These lessons are consolidated into a single skill, which is loaded into the student's context at inference time. Once the rollouts have been collected, D-Skill requires no answer keys, labels, held-out evaluation, or further environment interaction. The teacher and student contribute only through their fixed traces: neither is queried or fine-tuned during analysis and consolidation, and no model weights are modified.

We evaluate D-Skill on SpreadsheetBench Verified, ALFWorld and chess puzzles using different teacher-student pairs with different model families. As presented in section 4.1, across all three benchmarks and different teacher-student settings, D-Skill closes 83.4% to 89.3% of the teacher–student performance gap on SpreadsheetBench Verified, 84.6% to 94.6% on ALFWorld, and 82.2% to 87.7% on chess puzzles. This transfer also yields substantial inference-cost savings: in the GPT-5.6 Terra–Luna setting, the D-Skill student costs $8.3\times$ less per task than the teacher on SpreadsheetBench Verified and $5.7\times$ less per episode on ALFWorld (see section 5.3). We further compare D-Skill with supervised fine-tuning and find that it provides stronger transfer after a single pass over the teacher corpus while attaining performance comparable to multi-epoch fine-tuning (Section 5.2). These results demonstrate that textual skill distillation can match weight-based transfer in the evaluated setting while producing an artifact that remains readable, editable, and portable. In summary, this paper makes the following contributions:

- We formulate *skill distillation* as an alternative to weight-based knowledge distillation, encoding transferred capabilities in an external, interpretable artifact that requires neither weight access nor gradient-based training.
- We introduce **D-Skill**, a simple yet effective method that distills paired teacher and student rollouts by identifying their behavioral divergences and agreements and consolidating the resulting lessons into a reusable skill. After rollout collection, the method requires no labels, held-out evaluation, or further environment interaction.
- We demonstrate across three benchmarks and multiple teacher–student pairs that D-Skill closes most of the performance gap. In addition, it also outperforms the evaluated skill-learning baselines while retaining substantially lower inference cost than the teacher.

2 Related Work

Knowledge distillation. Knowledge distillation trains a compact student to match a stronger teacher's predictions (Bucilă et al., 2006; Hinton et al., 2015), extended to sequence-level imitation (Kim & Rush, 2016), compressed language models (Sanh et al., 2019), and on-policy variants that supervise the student's own rollouts to close the train–inference mismatch (Gu et al., 2023; Agarwal et al., 2024; Xu et al., 2025; Zhao et al., 2026). Throughout, two assumptions persist: the transfer *medium* is parameters (gradient updates, so closed-weight students are out of reach), and the signal is distributional (logits or token-level divergences, requiring white-box teacher access and

a shared tokenizer). D-Skill relaxes both: knowledge is retained as an explicit textual skill consumed in context by black-box students, and the signal is the behavioral contrast between teacher and student trajectories on shared tasks.

Skill and memory-augmented agents. A complementary line of work equips agents with persistent memories or skills—external artifacts carried across tasks and consulted at inference time. Reflexion stores reflections on past failures (Shinn et al., 2023), Voyager builds libraries of executable skills through exploration (Wang et al., 2023), and subsequent work consolidates trajectories into procedural memory (Wang et al., 2024; Fang et al., 2026; Li et al., 2026). SkillGen contrastively induces and validates skills from an agent's successful and failed rollouts (Ma et al., 2026). Teacher-driven approaches instead transfer stronger capabilities: Trace2Skill compiles stronger-model trajectories into textual skills (Ni et al., 2026), while AgentDistill packages teacher-generated procedures as reusable MCP modules (Qiu et al., 2025). D-Skill instead constructs a student-targeted skill by directly comparing teacher and student trajectories. D-Skill addresses this gap by contrasting teacher and student trajectories, producing guidance targeted to their behavioral differences.

Automatic textual optimization. A third line improves a text artifact through iterative textual updates: APE and OPRO search over instructions with LLM proposers (Zhou et al., 2022; Yang et al., 2024), Promptbreeder evolves prompts self-referentially (Fernando et al., 2023), DSPy compiles multi-stage pipelines by tuning their prompts against a validation metric (Khattab et al., 2023), and TextGrad backpropagates natural-language critiques as "textual gradients" (Yuksekgonul et al., 2024). The same machinery has recently been applied beyond instructions: GEPA evolves prompts by reflecting on execution traces (Agrawal et al., 2026), and SkillOpt optimizes a portable skill file through gated patch-style edits validated on held-out tasks (Yang et al., 2026), one instance of a growing family of self-evolving skill methods (Jiang et al., 2026). These methods treat text as an optimizable artifact, typically using task evaluation to select or refine updates. D-Skill follows the same broad paradigm but focuses on teacher–student distillation: it derives textual updates from behavioral contrasts between a stronger teacher and a target student on matched tasks. D-Skill thus extends automatic textual optimization to capability transfer while remaining compatible with different skill-update backends.

3 METHOD

Figure 2 summarizes D-Skill, which consists of three stages: rollout collection, difference analysis, and skill update. No stage computes a gradient or requires access to the parameters of either model. Collecting fresh rollouts requires the teacher and student to interact with the task environment; when saved traces are available, this interaction can instead have occurred during an earlier execution. In either case, all subsequent analysis and skill construction operate offline on fixed traces.

3.1 PROBLEM SETUP

Let $\mathcal{T} = \{t_1, \ldots, t_n\}$ be a set of unlabeled tasks sampled from a task distribution $\mathcal{D}$, and let π_S and π_T denote a student and a stronger teacher policy operating in the same task and action space. Given paired trajectories $\{(\tau_S(t), \tau_T(t)) : t \in \mathcal{T}\}$, our objective is to construct a natural-language skill s such that the skill-conditioned student π_S^s improves over the original student without modifying its parameters. If $J(\pi)$ denotes expected task performance, we quantify the transfer by the fraction of the teacher–student gap closed:

$$\mathrm{GapClosed}(s) = \frac{J(\pi_S^s) - J(\pi_S)}{J(\pi_T) - J(\pi_S)}.$$

3.2 ROLLOUT COLLECTION

For each task $t \in \mathcal{T}$, we collect one trajectory from each model, $\tau_S(t)$ and $\tau_T(t)$, recording the actions issued and the observations returned. The paired trajectories are the unit of supervision; no later stage analyzes either trajectory in isolation. The two policies attempt each task independently, with neither conditioned on the other's behavior. When trajectories are collected specifically for D-Skill, both policies interact with the task environment through ordinary execution; alternatively,

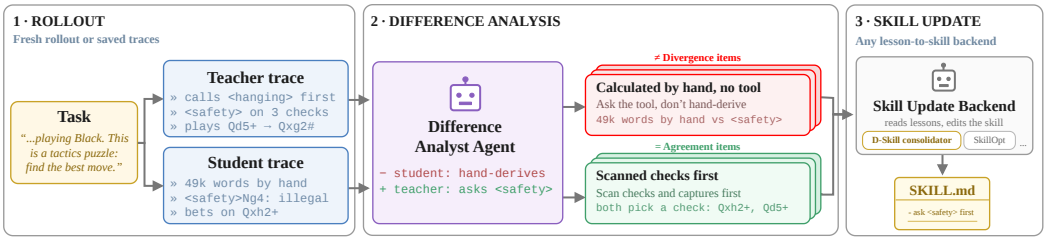


Figure 2: **Ground-truth-free skill distillation from paired rollout traces.** (1) **Rollout:** Paired teacher and student traces are obtained from fresh rollouts or saved executions. (2) **Difference analysis:** A Difference Analyst Agent compares the fixed traces without answer keys, labels, or further environment interaction, extracting divergence and agreement items. (3) **Skill update:** A Consolidation Agent converts the lessons into patches, merges them, and applies the update to `SKILL.md`.

trajectories may be retrieved from logs of prior executions, allowing distillation to begin directly from saved traces. The two trajectories need only correspond to the same task and use a shared action interface. Once the trajectories have been collected, D-Skill operates fully offline: neither π_S nor π_T is invoked again, and skill construction requires no further environment access, ground-truth labels, rewards, or evaluation feedback.

3.3 Difference Analysis

In the second stage, a difference analyst agent $\mathcal{A}$ converts each paired student and teacher trace into a set of structured, reusable lessons. The analyst can inspect the actions and observations recorded in the traces, but it has no access to answer keys, labels, or additional environment interaction.

Rather than asking the analyst to compare the trajectories freely, we require it to examine each pair along a predefined set of behavioral dimensions. These dimensions cover, for example, what information each agent gathered before acting, how it selected and executed actions, how it responded to errors or unproductive intermediate observations, and how it verified the final result. This shared schema makes lessons comparable across task pairs and discourages the analyst from emphasizing superficial differences in naming, phrasing, or reasoning style. Each lesson must cite concrete evidence from the paired traces and express a task-general principle rather than an instance-specific correction. Details about the prompt for the analyst can be found in appendix B.

For each trajectory pair, the analyst produces at most three lessons, ranked by their anticipated usefulness on future tasks, and assigns each to one of two classes: a *divergence lesson* identifies a behavioral difference between the two agents, specifying the student's behavior, the teacher's alternative, and a reusable instruction intended to address the gap; an *agreement lesson* identifies a behavior shared by both agents that is judged worth preserving in the distilled skill.

3.4 Skill Update

To construct the final skill document, we use a modified version of the Trace2Skill consolidation backend (Ni et al., 2026) as the D-Skill consolidator. We retain its map–reduce–apply procedure but adapt its input to consume the structured divergence and agreement lessons produced by our difference-analysis stage. The complete lesson pool is shuffled and partitioned into k chunks, which are processed sequentially to update the skill document. For each chunk, a consolidation agent $\mathcal{C}$ performs three phases. In the map phase, it independently converts each lesson into a candidate patch to the current skill, such as appending to an existing section, replacing existing content, or creating a new section. In the reduce phase, it merges the patches by removing duplicates, resolving conflicts, and generalizing related rules. Finally, in the apply phase, the merged patch is deterministically incorporated into the current skill before the next chunk is processed.

We use this adapted Trace2Skill backend throughout our experiments. D-Skill's output from the preceding stage is a general pool of structured lessons, so other compatible skill-updating backends could be substituted without changing the rollout-collection or difference-analysis stages, more details can be found in section 4.2.

Table 1: Main results. Each block reports the bare student (floor), the student with the distilled D-Skill skill, and the teacher; %GC is the fraction of the teacher–student gap closed.

(a) SpreadsheetBench-Verified (SSB-V) and ALFWorld

		SSB-V		ALFWorld	
Teacher	Student config	Score	%GC	Score	%GC
DeepSeek-V4-Pro-0423 + Skill	Qwen3.6-35B-A3B	28.0±1.4	—	59.0±0.7	—
	+ D-Skill	**73.2**±3.0	**88.2%**	**84.2**±3.3	**85.7%**
	GPT-5.6 Luna	25.3±1.1	—	55.2±4.2	—
	+ D-Skill	**72.8**±2.1	**88.0%**	**83.3**±3.5	**84.6%**
	Teacher	79.3±1.8	100%	88.4±0.5	100%
GPT-5.6 Terra + Skill	Qwen3.6-35B-A3B	28.0±1.4	—	59.0±0.7	—
	+ D-Skill	**73.1**±3.4	**89.3%**	**83.1**±1.4	**93.8%**
	GPT-5.6 Luna	25.3±1.1	—	55.2±4.2	—
	+ D-Skill	**69.7**±0.8	**83.4%**	**83.1**±3.2	**94.6%**
	Teacher	78.5±0.7	100%	84.7±1.1	100%

(b) Chess puzzles

Teacher	Student config	Score	Est. ELO	%GC
Sonnet 5	Haiku 4.5	41.0±1.4	1050±26	—
	+ D-Skill	**53.5**±1.0	**1306**±21	**82.2%**
	Teacher	56.2±3.2	1383±79	100%
Haiku 4.5	Qwen3.6-35B-A3B	24.8±0.4	663±74	—
	+ D-Skill	**39.0**±3.5	**979**±72	**87.7%**
	Teacher	41.0±1.4	1050±26	100%

4 EXPERIMENTS

We evaluate D-Skill across three agentic domains and multiple teacher–student configurations. For each task and teacher–student pair, we collect paired rollouts on the same unlabeled tasks, apply the difference-analysis procedure from Section 3, and consolidate the resulting lessons into a skill document. At evaluation time, this skill is equipped by the student without modifying its parameters.

Datasets and Benchmarks. We evaluate D-Skill in three domains with distinct interaction and evaluation requirements. **SpreadsheetBench Verified (SSB-V)** (Ma et al., 2024; Shortcut Research Team, 2025) evaluates agents on spreadsheet-editing tasks. We divide its 400 tasks using a fixed shuffled split: 200 tasks form the distillation set for rollout collection, and the remaining 200 form a held-out evaluation set on which we report accuracy. **ALFWorld** (Shridhar et al., 2020) evaluates embodied decision-making in text-based household environments. We collect teacher and student traces on 57 games from its training split and report success rate on the standard 134-game test split. **Chess** evaluates tactical reasoning in an agentic interaction setting. We evaluate on a fixed set of 200 puzzles from the public Lichess puzzle database and collect distillation traces on 100 additional puzzles that are disjoint from the evaluation set. We apply a strict solve criterion: a puzzle is counted as correct only when the agent successfully executes the complete tactical sequence. A more detailed setup for the chess puzzle environment can be found in Appendix A. Across all three domains, the distillation tasks provide only task descriptions and interaction traces; no correctness labels or reference solutions are supplied to the difference-analysis or skill-update stages.

Models and conditions. For SpreadsheetBench-Verified and ALFWorld, we consider two black-box teacher configurations: DeepSeek-V4-Pro-0423 (DeepSeek-AI, 2026) and GPT-5.6 Terra (OpenAI, 2026a), each equipped with an existing skill solely to strengthen the teacher and create a meaningful teacher–student capability gap. Before the teacher trajectories are passed to the difference analyst, we remove the teacher's reasoning and all skill- or system-prompt-related content; the an-

alyst receives only the task description and the teacher's externally observable action–observation sequence, and thus, the skills equipped by the teachers are never visible in the whole pipeline. D-Skill therefore treats the teacher as a black-box policy and has no access to the internal mechanism that produces its behavior. From the perspective of the distillation pipeline, equipping the teacher with a skill is operationally equivalent to selecting a stronger black-box teacher, as only the resulting behavior is exposed. The students are Qwen3.6-35B-A3B (Qwen Team, 2026) and GPT-5.6 Luna (OpenAI, 2026a), representing different model families. For each teacher–student pair, we report three conditions: the student baseline, the same student augmented with the distilled skill, and the teacher. Both the analyst and consolidation agents are DeepSeek-V4-Pro-0423. For chess puzzles, where the performance gaps among the GPT-series models are comparatively small, we instead consider two teacher–student pairs: (a) Sonnet-5 (Anthropic, 2026b) as the teacher and Haiku-4.5 (Anthropic, 2025b) as the student, and (b) Haiku-4.5 as the teacher and Qwen3.6-35B-A3B as the student. Sonnet-5 serves as both the analyst and consolidation agent for the chess experiments.

Metrics. SSB-V accuracy, ALFWorld success rate, and chess solve rate are reported on a 0–100 scale. We also report the fraction of the corresponding teacher–student gap closed, as defined in Section 3. For chess, solve performance is converted to an estimated human-scale tactical rating using the logistic fit described in Appendix A. Student and teacher results are mean±standard deviation over two evaluation passes; D-Skill results are mean±standard deviation over six evaluations, comprising two passes for each of three independently distilled skills.

4.1 Main results

Table 1 reports the student's baseline performance, its performance with the distilled D-Skill documents, and the performance of the teacher that generated the source traces. Across the two agentic benchmarks in Table 1a, D-Skill closes 83–95% of the teacher–student gap in all eight teacher–student–task configurations without updating the student's weights. With DeepSeek-V4-Pro as the teacher, Qwen3.6-35B-A3B improves from 28.0 to 73.2±3.0 on SpreadsheetBench-Verified, while GPT-5.6 Luna improves from 25.3 to 72.8±2.1; both close 88% of the gap to the teacher's score of 79.3. On ALFWorld, the same students improve from 59.0 and 55.2 to 84.2±3.3 and 83.3±3.5, closing 85–86% of the gap. Replacing DeepSeek-V4-Pro with GPT-5.6 Terra, a teacher from a different vendor with a different trace style, yields similar results: Qwen and Luna reach 73.1±3.4 and 69.7±0.8 on SpreadsheetBench-Verified, corresponding to 89% and 83% gap closure, and both reach 83.1 on ALFWorld, corresponding to 94–95% gap closure.

Table 1b extends the evaluation to chess puzzles, a tool-mediated, non-language domain with a smaller teacher–student gap. Distilling Sonnet 5 into Haiku 4.5 raises the student's score from 41.0 to 53.5, an 12.5-point improvement equivalent to approximately 256 Elo and 82.2% gap closure. Distilling Haiku 4.5 into Qwen3.6-35B-A3B raises the score from 24.8 to 39.0, a 14.2-point improvement equivalent to 316 Elo and 88% gap closure. These results show that D-Skill generalizes beyond agentic tasks to reasoning-intensive domains such as chess, where success requires strategic planning and sequential decision-making. Representative distilled skills are logged in appendix D.

4.2 Ablations

Teacher-student contrast improves transfer. Table 2 tests whether D-Skill's gains can be explained by either side of the paired data alone. Across all three conditions, we hold fixed the 200-task distillation set, analyst and consolidation models, lesson format, and consolidation procedure, varying only which traces are available to the analyst. The student-only condition allows the analyst to diagnose weaknesses from the student's behavior without access to a stronger policy, whereas the teacher-only condition extracts general guidance from teacher behavior without observing the target student's failures. Both one-sided variants improve substantially over the student floor, but neither accounts for the full D-Skill

Table 2: **Teacher-student contrast improves performances.** Results average 3 independent skills, each evaluated twice.

	Qwen3.6 35B-A3B		GPT-5.6 Luna	
Method	Score	%GC	Score	%GC
Student trace only	47.7±6.7	38.3%	49.8±8.8	45.4%
Teacher trace only	56.3±6.1	55.2%	63.5±5.4	70.7%
D-Skill	**73.2**±3.0	**88.2%**	**72.8**±2.1	**88.0%**

gain. For Qwen3.6-35B-A3B, the student-only and teacher-only variants reach 47.7 and 56.3, respectively, compared with 73.2 for D-Skill; for GPT-5.6 Luna, they reach 49.8 and 63.5, compared with 72.8. Full paired contrast therefore closes 88% of the teacher–student gap for both students, versus 38–45% for student-only analysis and 55–71% for teacher-only analysis. These results indicate that a strong analyst can derive useful guidance from either trajectory in isolation, while directly contrasting matched teacher and student behavior yields larger and more consistent gains by targeting student-specific gaps.

Analyst and consolidation agents. D-Skill uses an LLM in two places: the *analyst*, which contrasts teacher and student traces and extracts candidate lessons, and the *consolidation agent*, which merges those lessons into the skill document. Table 3 replaces each stage with the student model itself (Qwen3.6-35B-A3B) on chess, where Haiku 4.5 is the teacher. Keeping a strong analyst and letting the student consolidate costs about five points (39.0 to 34.3), and the resulting document still closes 59% of the teacher–student gap. Replacing the analyst is far more damaging: with a Qwen analyst the document closes only 39–48% of the gap regardless of who consolidates. The student analyst produces agreement and divergence items that contain fewer confirmed lessons and noisier gap explanations, and the student consolidation agent fails to compress useful information in the items, resulting in a much longer skill document. The practical implication is that a teacher-grade model is needed only for the analysis stage, which runs once per training batch, and the merging can be delegated to a cheaper model without collapsing the method.

Table 3: **A good analyst is needed.** Chess accuracy of the Qwen student with the skill produced by each analyst-consolidation pair. Results average 3 independent skills, each evaluated twice.

	Consolidation	
Analyst	Sonnet	Qwen
Sonnet	**39.0**±3.5 **(88%)**	34.3±3.4 (59%)
Qwen	31.1±4.5 (39%)	32.6±3.9 (48%)

Other skill backends. D-Skill separates difference analysis from skill consolidation: its analyst produces structured lessons rather than a completed document, allowing those lessons to be consumed by any skill-update backend that accepts textual edits. We test this modularity with SkillOpt (Yang et al., 2026). SkillOpt ordinarily rolls out the student, reflects on unsuccessful trajectories, merges and ranks proposed edits under a fixed budget, and accepts an edit only when it improves held-out validation accuracy. We modify only the input to its reflection step: instead of receiving student trajectories, the SkillOpt analyst receives the D-Skill lessons associated with each minibatch. All downstream stages, including merging, ranking, and validation gating, remain unchanged.

Table 4: **D-Skill is compatible with an alternative skill-update backend.** Results average three independently distilled skills, each evaluated twice. The SkillOpt-backed variant retains SkillOpt's held-out validation gate.

Method	Score	%GC
Student (Qwen3.6-35B-A3B)	24.8±0.4	—
SkillOpt	29.6±1.4	29.7%
D-Skill w/ SkillOpt backend	**36.9**±5.7	**74.9%**
D-Skill	**39.0**±3.5	**87.7%**

Table 4 reports chess performance with Qwen3.6-35B-A3B on different skill generation framework and update backend. At the same rollout budget of 1 epoch, the lesson-conditioned backend reaches 36.9 ± 5.7, outperforming SkillOpt with its original trajectory-reflection analyst by 7.3 points and performing within the uncertainty of D-Skill's native consolidator. Thus, the resulting lessons also improve a backend originally designed to learn from self-generated failures.

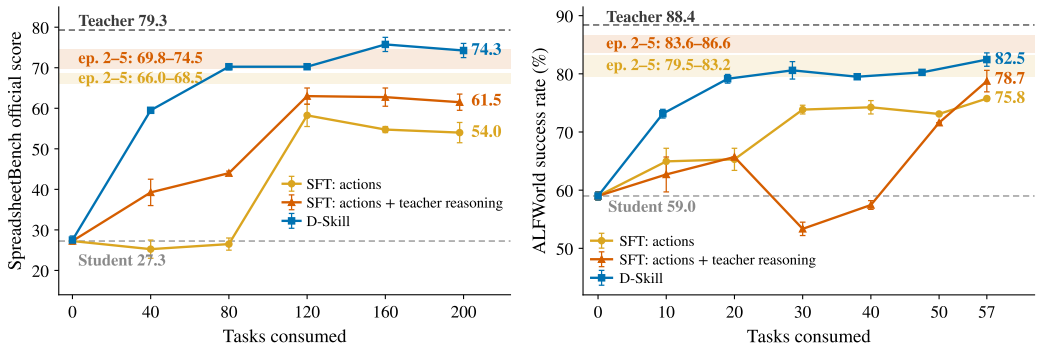
5 Analysis

5.1 Comparing with Other Skill Learning Pipelines

Table 5 compares D-Skill with Trace2Skill (Ni et al., 2026), an existing method that distills a skill document from execution traces. The published Trace2Skill configuration improves upon the skill-free student but closes only 8–13% of the student–teacher gap. To strengthen the baseline, we additionally construct two variants that progressively provide the resources available to D-Skill: first replacing the student author with the teacher model, and then replacing the student traces with the

Table 5: SpreadsheetBench-Verified: D-Skill vs. Trace2Skill baselines. All numbers are means over six evaluations, comprising two passes for each of three independently distilled skills.

Method	Qwen3.6-35B-A3B		GPT-5.6 Luna	
	Score	%GC	Score	%GC
Student	28.0±1.4	—	25.3±1.1	—
Teacher (DS-V4-Pro + Skill)	79.3±1.8	100%	79.3±1.8	100%
Trace2Skill (student trace, student-authored)	34.6±4.4	12.8%	29.4±0.5	7.6%
Trace2Skill (teacher trace, teacher-authored)	39.5±3.3	22.4%	39.2±2.7	25.7%
Trace2Skill (student trace, teacher-authored)	44.3±3.1	31.8%	45.7±5.7	37.7%
D-Skill	**73.2**±3.0	**88.2%**	**72.8**±2.1	**88.0%**



(a) SpreadsheetBench Verified

(b) ALFWorld

Figure 3: **One-pass D-Skill outperforms one-epoch SFT and remains competitive with its multi-epoch performance.** Both methods use Qwen3.6-35B-A3B as the student and the same teacher-trajectory corpus on (a) SpreadsheetBench Verified and (b) ALFWorld. The x-axis reports the number of teacher tasks processed, and the shaded bands show SFT performance over epochs 2-5.

teacher's own traces, with both resulting skills applied to the student. The strongest variant, teacher-authored documents constructed from student traces, closes 32–38% of the gap. Substituting teacher traces does not improve performance further and instead reduces the scores from 44.3 to 39.5 for Qwen3.6-35B-A3B and from 45.7 to 39.2 for GPT-5.6 Luna. In contrast, D-Skill reaches scores of 73.2 and 72.8, closing approximately 88% of the gap for both students.

5.2 Comparing with Weight Updates

A natural alternative use of the teacher corpus is to fine-tune the student on it. We therefore compare D-Skill with supervised fine-tuning (SFT) using exactly the same DeepSeek-V4-Pro-0423 teacher trajectories. We remove the teacher's skill and system prompt so that both methods receive the same teacher action–observation content. We fine-tune Qwen3.6-35B-A3B with LoRA (Hu et al., 2021) through the Tinker API (Thinking Machines Lab, 2025) on SpreadsheetBench-Verified and ALFWorld for five epochs under two variants: teacher actions alone and teacher actions together with teacher reasoning. We save checkpoints at several points during the first epoch and at every epoch boundary, evaluating each checkpoint twice. For D-Skill, we evaluate the document after the corresponding consolidation chunk and plot one lineage per benchmark, as independently distilled lineages achieve similar final accuracy. Figure 3 plots performance against the number of unique teacher tasks processed during the first pass over the corpus; the shaded bands show the range attained by SFT over epochs two through five, where the training plateaus. The exact setup of the SFT training can be found in appendix C.

D-Skill reaches multi-epoch SFT performance after a single offline pass over the paired traces. On SpreadsheetBench Verified, D-Skill reaches 74.3 after one pass, compared with 54.0 for action-only SFT and 61.5 for reasoning-augmented SFT; after four additional epochs, the two SFT variants reach 66.0–68.5 and 69.8–74.5, respectively. On ALFWorld, D-Skill reaches 82.5 after one pass,

compared with 75.8 and 78.7 for the two SFT variants, which rise to 79.5–83.2 and 83.6–86.6 over epochs two through five. Thus, using the same teacher-trajectory corpus, D-Skill provides substantially stronger one-pass transfer and remains competitive with multi-epoch SFT without gradient updates or access to the student's weights.

5.3 Cost Analysis

Figure 4 compares inference cost per task at list prices for models from OpenAI. D-Skill approaches teacher performance at a fraction of the inference cost: GPT-5.6 Luna with the distilled skill is 8.3× cheaper than the teacher on SpreadsheetBench and 5.7× cheaper on ALFWorld, costing only 0.35 cents per task for SpreadsheetBench and 2.0 cents per episode for ALFWorld, respectively. At these costs, D-Skill closes 83.4% of the teacher–student gap on SpreadsheetBench and 94.6% on ALFWorld. However, D-Skill is slightly more expensive than the bare student, as the additional cost comes primarily from reading the skill document and, in some cases, from the extra interaction steps the document induces while solving the task. Overall, D-Skill recovers most of the teacher's performance while retaining a substantial inference-cost advantage.

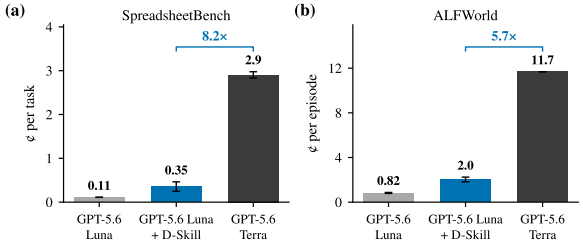


Figure 4: **Inference cost per task at model list prices** on (a) SpreadsheetBench Verified and (b) ALFWorld. Costs are reported in cents episode.

Distillation adds a one-time cost that grows linearly with the number of training tasks, since the analyst makes one call per teacher–student trace pair and the consolidator processes the resulting lessons in fixed-size chunks. It costs about 2.3 cents per training task on SpreadsheetBench and 2.6 cents on ALFWorld, i.e., $4.66 for our 200 SpreadsheetBench tasks and $1.51 for our 57 ALFWorld tasks.

Table 6: **One-time distillation cost** GPT-5.6 Terra→Luna, billed at list prices. Per task is per training task.

	Tasks	Total	Per task
SSB-V	200	$4.66	2.3¢
ALFWorld	57	$1.51	2.6¢

6 Limitations

D-Skill has three main limitations. First, it requires a capable analyst at distillation time: replacing the analyst with the weaker student substantially reduces the fraction of the gap closed (Table 3), although this is a one-time rather than per-inference cost. Second, loading the distilled document increases inference cost relative to the bare student by adding input tokens and, in some cases, inducing additional interaction steps, even though the resulting agent remains substantially cheaper than the teacher. Third, D-Skill can transfer only capabilities that can be expressed as instructions and executed by the student. It can improve procedures such as planning, verification, and tool use, but cannot supply fundamental capabilities the student lacks, such as deeper calculation, perception, or sufficient underlying model capacity.

7 Conclusion

We introduced D-Skill, a method that distills the behavioral gap between a black-box teacher and student into a natural-language skill by contrasting paired rollouts on the same tasks. Once the trajectories are collected, D-Skill operates fully offline, requiring no further policy queries, environment interaction, labels, rewards, evaluation feedback, or weight updates. Across SpreadsheetBench Verified, ALFWorld, and chess puzzles, D-Skill closes 82–95% of the teacher–student performance gap across multiple model families while retaining a substantial inference-cost advantage over the teacher. D-Skill also provides stronger one-pass transfer and remains competitive with multi-epoch SFT when using the same teacher-trajectory corpus, without gradient updates or access to the student's weights. Together, these results demonstrate that natural-language skill distillation is a practical, interpretable, and inference-efficient alternative to weight-based capability transfer in the evaluated domains.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, and interpreting results. We have not used generative AI tools for Generating synthetic data sets, helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, and cleaning and reformatting dataset, and formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and assisting with translation are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, creating or editing software code, drafting parts of a research paper, summarizing or analyzing existing literature, editing a research paper to improve readability, identifying relevant literature, and formating references. We have reviewed all AI-assisted work. We checked LLM-generated research ideas for potential plagiarism through a manual literature survey, LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

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. Engineering blog, October 16, 2025.

Anthropic. System card: Claude Haiku 4.5. https://www.anthropic.com/claude-haiku-4-5-system-card, October 2025b. October 2025.

Anthropic. Introducing Claude Fable 5.1 and Claude Mythos 5.1. Anthropic, September 2026a. URL https://www.anthropic.com/claude-fable-and-mythos-5-1. Accessed September 20, 2026.

Anthropic. System card: Claude Sonnet 5. https://www.anthropic.com/claude-sonnet-5-system-card, June 2026b. June 30, 2026.

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

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 535–541, 2006.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023a.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023b.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9354–9366, 2024.

DeepSeek-AI. DeepSeek-V4 technical documentation. https://fe-static.deepseek.com/chat/transparency/deepseek-V4-model-card-EN.pdf, April 2026. Model card, April 27, 2026.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Publishing, New York, 1978.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. 2023.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. volume 37, pp. 119919–119948, 2024.

Mark E. Glickman. Example of the Glicko-2 system. http://www.glicko.net/glicko/glicko2.pdf, 2012. Boston University.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models. 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 14852–14882, 2023.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the association for computational linguistics: ACL 2023*, pp. 8003–8017, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 2021.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling llm agent into small models with retrieval and code tools. volume 38, pp. 106501–106538, 2026.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. 2023.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. 2023.

Dacheng Li, Shiyi Cao, Fangzhou Zhao, Shuo Yuan, Sumanth Hegde, Joseph E Gonzalez, and Ion Stoica. Rlem: Reinforcement learning for external memory. In *COLM 2026 Workshop on Efficient Reasoning*, 2026.

Lichess. Lichess open database: Puzzles. https://database.lichess.org/#puzzles, 2026. CC0 puzzle database; snapshot downloaded 2026.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. Skillgen: Verified inference-time agent skill synthesis, 2026.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. volume 37, pp. 94871–94908, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. volume 35, pp. 17359–17372, 2022.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition. https://openai.com/index/gpt-5-6/, July 2026a. Model release, July 2026.

OpenAI. GPT-6 Astra: A new generation of intelligence. OpenAI, September 2026b. URL https://openai.com/index/gpt-6-astra/. Accessed September 20, 2026.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, et al. Agentdistill: Training-free agent distillation with generalizable mcp boxes, 2025.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all. https://qwen.ai/blog?id=qwen3.6-35b-a3b, April 2026. Model release, April 16, 2026.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. volume 36, pp. 8634–8652, 2023.

Shortcut Research Team. SpreadsheetBench Verified: A curated evaluation set. https://shortcut.ai/blog/posts/spreadsheetbench-verified, December 2025. 400 human-validated tasks curated with the original SpreadsheetBench authors; released December 2, 2025.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. 2020.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. pp. 7080–7106, 2026.

Thinking Machines Lab. Announcing Tinker. https://thinkingmachines.ai/news/announcing-tinker/, October 2025. Fine-tuning API for open-weight models; released October 1, 2025.

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

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Wenda Xu, Rujun Han, Zifeng Wang, Long Le, Dhruv Madeka, Lei Li, William Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *International Conference on Learning Representations*, volume 2025, pp. 64616–64646, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10222–10240, 2023.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. pp. 3053–3077, 2024.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. 2026:86069–86100, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. 2022.

A CHESS PUZZLE ENVIRONMENT SETUP

Table 7: Analysis tools available to the chess agent. Every tool is a static computation on a copy of the board; nothing is searched or evaluated.

Tool	Input and output
`<hanging></hanging>`	No input. Lists every piece that is attacked and has no defender, for both sides.
`<safety>m1, ...</safety>`	Up to five candidate moves. For each: whether it gives check (checkmate and stalemate named explicitly), the number of legal replies, every forcing reply (checks and captures with the captured piece named), attackers vs. defenders of the destination square, and the agent's pieces left undefended afterwards.
`<threats></threats>`	No input. Null-move audit: the opponent's forcing moves if the agent passed, and any immediate mate threats.
`<board>move</board>`	One move. The board after the agent's move only, with material, undefended pieces, and check/checkmate status.
`<simulate>move, r1, ...</simulate>`	One candidate and up to five opponent replies, each simulated independently from the position after the candidate: resulting board, material, undefended pieces, and whether the agent is in check.
`<line>m1, r1, m2, ...</line>`	A complete line of two to six plies supplied by the agent (own moves and opponent replies alternating): the final board, material, undefended pieces, and check/mate status.

Task. Each episode is a single Lichess tactics puzzle drawn from the public Lichess puzzle database (Lichess, 2026). A puzzle specifies a starting position (FEN), a scripted opponent move that creates the tactic, and a unique solution line of one or two solver moves; a two-move puzzle also scripts the opponent's forced reply after the first solver move. The agent plays the solver side and must produce every solver move in order. Following Lichess grading, a move is correct if it matches the solution or delivers checkmate; the first incorrect move ends the puzzle. A puzzle counts as solved only if all solver moves are correct; The agent never receives correctness feedback: after a correct move it simply sees the next position, and after an incorrect one the episode ends.

Puzzle sets. We filter the database to puzzles with popularity ≥ 100, at least 1,000 plays, rating deviation ≤ 80, and a solution of exactly one or two solver moves. The **evaluation set** has 200 puzzles, ratings ranging from 649–1987; 179 two-move and 21 one-move puzzles. The **distillation set** has 100 puzzles disjoint from the evaluation set, ratings ranging from 612–2357; 78 two-move and 22 one-move. Teacher and student traces used for distillation are collected on all 100 distillation puzzles. Baselines that need a validation split (SkillOpt) divide the same 100 puzzles into 60 training and 40 validation puzzles.

Observation. At every solver move the agent receives a fresh user message containing the FEN, an ASCII board (uppercase = White, rank 8 at the top), the side to move, a side-relative material count (Q=9, R=5, B=N=3, P=1), the last six moves labeled "you" / "opponent", the halfmove clock, the complete list of legal moves in SAN, the agent's optional plan note from its previous move, and the sentence "This is a tactics puzzle: there is a clearly best move in this position. Find it." In a

two-move puzzle the second observation additionally notes that earlier tool results referred to the previous position.

Actions and tool protocol. The system prompt instructs the agent to reply each turn with either exactly one tool tag or its committed move wrapped as `<action>Nf3</action>` (SAN or UCI), never both. Per solver move the agent may issue up to $K = 10$ tool calls; an exact repeat of an earlier query in the same move is served from a cache at no cost, and every other call, including malformed or illegal ones, costs one unit. When five or fewer calls remain, the tool result carries a short low-budget notice, and when the budget is exhausted the agent is told to commit. A reply that contains neither a tool tag nor an action tag triggers a corrective message, at most twice per move, asking for one tool tag or a move; a third tagless reply is scored as no move. A committed move outside the legal list is likewise scored as incorrect.

The six tools (Table 7) are implemented in-process with `python-chess` on a copy of the live board. They only report static facts about positions the agent specifies; there is no engine, search, or evaluation anywhere in the environment. Multi-argument tools accept at most five candidate moves (`safety`), five opponent replies (`simulate`), or six plies (`line`); `threats` is unavailable while the agent is in check.

Estimated Elo. Lichess rates puzzles on the same Glicko-2 scale as players (Glickman, 2012; Lichess, 2026), so a puzzle rated r is one that a player rated r solves about half the time. For each evaluation pass we fit a logistic model of solve probability against puzzle rating, $P(\text{solved} \mid r) = \sigma\big(\alpha + \beta\,\frac{r-1200}{400}\big)$, to the 200 evaluation puzzles by maximum likelihood, and report the rating at which the fitted probability crosses one half, $\widehat{R} = 1200 - 400\,\alpha/\beta$. The parameterization mirrors the Elo expected-score formula (Elo, 1978) with a free slope. Each document's Elo is the mean over its two passes; the table reports mean $\pm$ s.d. over distillation replicates for +D-Skill rows and over passes for floors and teachers.

B Difference Analyst Prompt

The difference analyst uses one system prompt template for all three domains (below). Only the slots differ across domains: `<<ENVIRONMENT>>` describes the environment and its action protocol; `<<COMPARISON>>` lists example behavioral dimensions to compare (e.g., inspection, search, tool use, recovery, completion discipline); `<<UNIT>>` and `<<EVIDENCE>>` name the trace unit to quote (steps, or turns and tool calls); `<<CHECK>>` and `<<GENERALIZE>>` specify what a runtime check and a generalizable instruction look like in the domain. In chess, `<<JUDGE>>` additionally forbids asserting that any specific move was correct, and "agent" reads "player". For SpreadsheetBench-Verified, `<<COMPARISON>>` also asks the analyst to diff the two agents' produced answer ranges and trace each difference back to the decision that caused it, without stating which value is correct.

The user message contains the task instruction, the expert trace, and the student trace. For SpreadsheetBench-Verified it also includes the graded answer range and the cell contents each agent left in that range (capped at 4,000 characters). Neither reveals the gold values. Each trace is truncated to its head (first 10,000 characters) and tail, capped at 35,000 characters in total, so the analyst sees the task statement, the early inspection turns, and the end of every episode.

```
<<ENVIRONMENT>>

- The EXPERT trace comes from a much stronger agent whose
working style the student should learn to imitate.
- The STUDENT trace comes from the weaker agent you are coaching
.

You have NO answer key and NO correctness information for either
 trace, and you must not invent any. Do not judge whether either
 agent solved the task<<JUDGE>>. Your job is purely behavioral:
identify how the expert's way of working differs from the
student's, and convert the most consequential differences into
```

```
reusable lessons that would make the student's future traces
resemble the expert's.

<<COMPARISON>>

Anchor every observation in specific <<UNIT>> visible in the
traces -- quote or paraphrase the concrete <<EVIDENCE>>. Never
speculate about <<UNIT>> that are not shown.

Write AT MOST 3 items total, ordered by expected impact on the
student's future behavior:
- "Failure Memory Item" -- a behavioral gap: what the student
did, what the expert did instead, and ONE concrete reusable
instruction.
- at most 1 "Success Memory Item" -- a behavior the student
already shares with the expert and should keep.

Every item's Content must additionally include one concrete
check the student can <<CHECK>>.

Every instruction must generalize across <<GENERALIZE>>.

Output format (exactly this structure, failure items first):

# Failure Memory Item 1
## Title
<short name>
## Description
<the observed student-vs-expert difference, with trace evidence>
## Content
<the single reusable instruction the student should follow>

# Success Memory Item 1
## Title
<short name>
## Description
<the shared behavior, with trace evidence>
## Content
<the instruction to keep doing it>
```

C SUPERVISED FINE-TUNING BASELINES

The SFT curves in Figure 3 fine-tune the same student on the same teacher episodes that D-Skill distils, and score every checkpoint with the same harness and scorer as the rest of the paper.

LoRA rank 16 through the Tinker API (Thinking Machines Lab, 2025); peak learning rate 10^{-4}, 5% warm-up, linear decay; five epochs; loss on assistant tokens only. Training is ground-truth-free: all teacher episodes are used, failures included, with no correctness labels. The teacher's system prompt (which carries its skill package) is replaced by the student's harness prompt, so training inputs match evaluation inputs. Two target variants: *actions only*, and *teacher reasoning + actions* with the reasoning in the student's `<think>` channel. Checkpoints are saved inside epoch 1 after a fixed number of training episodes (the x-axis of Figure 3) and at every epoch end, served in bf16 with thinking enabled (temperature 1.0, top-p 0.95, top-k 20), and scored as the mean of two passes.

Table 8: SFT within epoch 1

	Targets	episodes consumed					
		40	80	120	160	200	
SSB	actions	25.2±3.2	26.5±2.1	58.2±3.9	54.8±1.1	54.0±3.5	
	reasoning + actions	39.2±4.6	44.0±0.7	63.0±2.8	62.8±3.2	61.5±2.8	
		10	20	30	40	50	57
ALFWorld	actions	65.0±3.2	65.3±2.7	73.8±1.1	74.2±1.6	73.1±0.0	75.8±0.5
	reasoning + actions	62.7±4.2	65.7±0.0	53.4±1.6	57.5±1.1	71.6±0.0	78.8±2.6

Table 9: SFT across epochs

	Targets	epoch 2	epoch 3	epoch 4	epoch 5
SSB	actions	68.0±2.1	66.0±4.9	68.5±2.1	66.5±0.7
	reasoning + actions	69.8±0.4	74.5±3.5	74.2±2.5	74.5±2.1
ALFWorld	actions	79.5±1.6	82.4±0.5	82.0±1.1	83.2±0.6
	reasoning + actions	83.5±1.1	86.2±1.6	85.8±1.1	86.6±2.1

D Distilled Skill Documents

For each benchmark we show excerpts of the highest-scoring D-Skill document with Qwen3.6-35B-A3B as the student. By a fixed rule, each excerpt is the document's opening section and the first three rules of its *Common Pitfalls* section, rendered from Markdown; [...] marks omitted text. The complete documents are in the supplementary material.

SpreadsheetBench-Verified

Core Workflow

1. Inspect the entire workbook: load and print all non-empty cells with coordinates, values, types, and number formats.

Always follow a two-stage loop: inspect the entire workbook before any modification, and verify the output after saving by reloading the workbook from disk (do not rely on in-memory state). This habit catches misplacements and provides a clear picture of the result.

- [!] CRITICAL: Never create a new workbook with `openpyxl.Workbook()`. Always load the existing file with `openpyxl.load_workbook(spreadsheet_path)` and modify it in place. After saving, reload the output and print the exact answer range (as stated in `answer_position`) to confirm all values are plain data (no formula strings) and that all original sheet names are still present. Only call `TASK_COMPLETE` after this check passes.

After this inspection, you have all the information needed to write the solution; do not inspect again unless you encounter a specific issue during implementation. Proceed directly to writing the script.

2. After 2–3 focused inspection steps, write a Python script that produces the required output and saves it to the `output_path` specified in the task. If you have inspected the workbook for more than 3 turns without writing any code, or if you receive a system warning about a missing output file, stop and immediately write the output script, even if you are unsure of some details. Verify the file exists with `assert os.path.exists(OUTPUT_PATH)`, then reload it with `data_only=True` and run the verification checks (see Verification). Never continue exploring without producing the deliverable.

- Always use the exact absolute file paths provided in the task instruction (`spreadsheet_path` and `output_path`). Never change directory or guess paths; pass them directly to Python scripts.

• If the task allows solving without VBA, always use Python/openpyxl; never write VBA macros. This keeps the solution simple, auditable, and easy to verify.

[...]

Common Pitfalls

• Writing a formula string (e.g., `=VLOOKUP(...)`) instead of the computed result. The grader expects the evaluated value, not the formula text. Always compute programmatically and write the result.

• **String matching**: Always check `isinstance(value, str)` before performing string operations. Normalize both the lookup value and cell text by stripping whitespace and case-folding (e.g., `.strip().casefold()` or `.strip().upper()` depending on need). Apply the same normalization when comparing sheet names, header labels, or any string identifiers from different sources. Never convert arbitrary cell types to strings with `str(...)` for text matching; this can cause false positives or missed deletions.

• **Number formats**: After writing any value that should preserve the source's format (e.g., date, currency, percentage, or any value copied from another cell), explicitly copy `.number_format` from the source cell (`target.number_format = source.number_format`). Default formats can cause incorrect display.

[...]

ALFWORLD

Core Approach

Follow the canonical task flow:

1. Locate and take the target object.
2. Navigate to the appropriate processing station.
3. Use the specific interaction verb that appears in the admissible action list.
4. Navigate to the goal receptacle.
5. Place the object there.

Follow the natural subgoal order implied by the instruction: acquire → transform → deliver.

Multi-object rule: When the task instruction contains multiple object nouns (e.g., "examine the bowl with the desklamp"), treat every noun as a required object that must be found and used. Do not treat any as a mere locator. Before taking any action, list all object nouns and check off each as you find them. If a required object is missing, search for it before interacting with the others.

Act on current observations and admissible actions immediately. Execute a direct, linear path toward the goal state. Before choosing any action, ask: "Does my current observation already give me enough information to take a productive step toward the goal, or am I only verifying what I already know?" If the former, take that step directly; if the latter, skip it.

Trust admissible actions as ground truth. When you arrive at a location and the admissible actions list contains an interaction with your target object (e.g., `take mug 1 from diningtable 1`), select it without re-verifying from memory or prior observations.

Before acting, scan the admissible action list and match your next step to an available action verb. Do not assume an action is available—verify it appears in the list.

Task Completion

When at a target surface and holding the object to be placed there, scan the admissible actions for a `move ... to ...` command that names both the held object and the destination you are facing. Select that exact action immediately—do not check inventory, look around, or deliberate. Execute it without delay if the goal requires placement there.

After successfully placing the object at the goal surface, stop acting. The task is complete. Do not reopen containers, check inventory, or retrieve items.

After executing a `move` or `put` action that places the target object at the goal destination, read the observation for confirmation (e.g., "You move the X to the Y"). Once confirmed, stop acting immediately. Do not retrieve the object, reopen containers, or restart any part of the task sequence. Runtime check: "Did my last observation confirm the target object is now in the target destination?" If yes, the task is complete.

[...]

Common Pitfalls

• **DO NOT** use `inventory` or `look` as a search action when the target is not in hand and not on the current surface. Navigate to the next unvisited candidate surface instead. `look` is only for reorientation when no plausible `go to` destination exists; if any `go to` action leads to an unvisited surface, select it — never substitute `look` for navigation.

• **DO NOT** revisit surfaces you have already examined and found empty. Track visited locations and skip them.

• **DO NOT** repeat the same action (e.g., `examine`, `look`) on the same surface or object if the previous attempt returned no new information. If an observation does not advance progress toward the goal, immediately switch to a different admissible action: go to an unexplored location or use a different interaction verb. Runtime check: After any observation, ask: "Did this reveal new information that moves me closer to the goal?" If not, pick an action you have not already tried on this object or surface.

[...]

CHESS PUZZLES

When NOT to Analyze: Fast Paths and Stopping Rules

Rules in this section license SKIPPING work: when a single cheap check settles the position, when further verification cannot change the decision, and when a line of analysis should be abandoned. Every entry must name (a) the exact trigger that makes the shortcut safe and (b) the condition under which the shortcut must NOT be taken. A rule without both parts does not belong here.

• Trigger: the mating/tactical idea's supporting facts (king can't escape, no capture or block available) have already been stated once and confirmed. Shortcut: stop re-verifying and commit to the move. Do NOT take this shortcut if the first check was itself based on a hand-parsed FEN that hasn't been cross-checked against the printed board/legal-move list.

[...]

Common Pitfalls

• Restating the same candidate-move analysis in prose a second time with no new tool result. Ask "did I already write this exact conclusion in a prior turn?" — if yes, emit a tool call or <action> instead of another paragraph.

• After a tool call discards or confirms a candidate, the very next turn must contain another tool call or an <action> tag — never a turn of pure manual analysis. If any sentence or Q&A pair (e.g. "If X, Y plays Z? No.") appears twice in your own output, that is an infinite loop: stop mid-thought and issue a tool/action tag immediately, do not finish the paragraph first.

• Committing to a move that your own analysis already concluded loses material/"is bad" against its critical reply, or whose every self-calculated line was tagged bad, equal, or drawing. Once your own calculation (with or without a tool) produces a concrete refutation of a candidate, that candidate is dead — do not loop back and re-examine it again hoping for a different conclusion. Either move to a genuinely different candidate and verify it, or re-run <simulate>/<safety>

on the exact refuted line to check if the refutation still holds; never restate a plan that contradicts your own just-derived refutation.

[. . .]