SKILLHELIX: COEVOLVING AGENT SKILLS VIA DECISION AND REASONING SYNERGY

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent skills are natural-language documents that guide a model through a class of tasks. The model can improve these skills by examining its failures and rewriting the instructions. Such loops assign credit to entire rewrites. They reveal whether a rewrite helped, but not which edits contributed. Harmful edits are therefore accepted alongside helpful ones, and the optimizer skill guiding the rewrites cannot learn which kinds of edits work. Assigning credit to individual edits requires counterfactual evaluations, with up to 2^n runs for n edits. Yet accuracy on the small batch used in each round rarely distinguishes one edit combination from another. We obtain this credit by pairing the black-box reasoning model with a small open-weight decision model. The reasoning model's outcomes ground the signal in its own behavior, identifying the questions an edit should fix and those it must preserve. The decision model provides a dense signal through forward passes that measure how each edit combination changes the likelihood of the gold answers on these questions. SKILLHELIX uses this signal in two ways. It selects edits for the task skill and refines the wording of their most influential span, retaining the change only if validation accuracy improves. It also assigns a Shapley value to every proposed edit and uses these credits to revise the optimizer skill, allowing the two skills to coevolve. With GPT-5.5 as the reasoning model, SKILLHELIX outperforms all baselines on six benchmarks, improving accuracy by an average of 3.8 points over the strongest baseline and by 12.9 points on LiveMath. Ablations show that edit-level decisions and coevolution both contribute to these gains.

1 INTRODUCTION

Agent skills, natural-language documents that describe how to approach a class of tasks, have become a lightweight way to adapt black-box language models without changing their weights (Zhang et al., 2025; Wang et al., 2025b; Zhao et al., 2024). Because a skill is text, the model can improve it itself: it attempts a batch of tasks, reads the trajectories it failed, rewrites the skill, and judges the rewrite by its effect on accuracy (Yuksekgonul et al., 2025; Agrawal et al., 2026; Zhang et al., 2026; Yang et al., 2026). The instructions that steer these rewrites, which we call the optimizer skill, can in turn be revised by the same model (Fernando et al., 2024; Zelikman et al., 2024).

This loop learns from *rewrite-level credit*: it observes whether a rewrite helped, but not which of its edits did. A rewrite typically makes several edits, each a separate hypothesis about why the model failed, yet the loop accepts or rejects them together on the strength of one accuracy measurement, much as outcome supervision assigns a single signal to an entire solution (Lightman et al., 2024). Rewrite-level credit has two consequences. For the task skill, harmful edits enter together with helpful ones, and helpful edits are discarded together with harmful ones (Figure 1). For the optimizer skill, the feedback that could improve it never arrives: it learns whether a rewrite succeeded, not which kinds of edits it should propose. Where finer-grained credit is kept, it is assigned by the model's own reflection on which parts helped (Zhang et al., 2026), and language models struggle to correct themselves without external feedback (Huang et al., 2024).

Edit-level credit requires counterfactual comparisons. Knowing what an edit contributed means comparing the skill with and without it, with and without the other edits (Shapley, 1953). With a black-box model, every such comparison is a separate evaluation run, so crediting n edits exactly

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

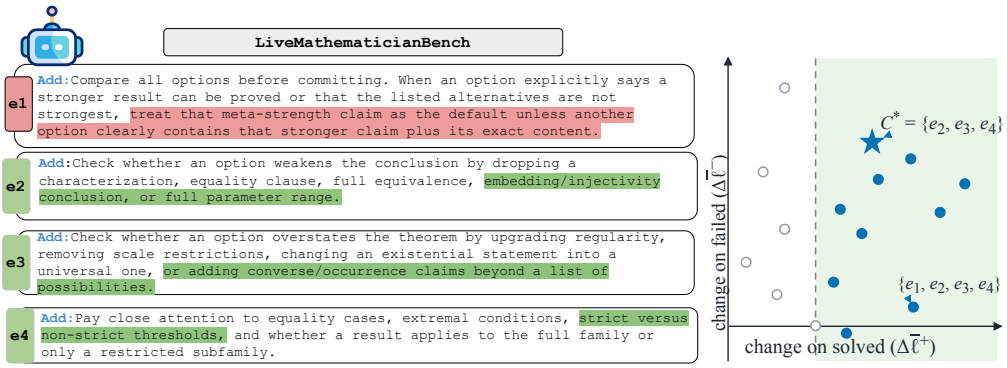

Figure 1: A real round on LiveMathematicianBench. Left: the four edits the optimizer model proposed from the reasoning model's failures. Right: every combination of these edits, scored by the decision model by how much it helps the questions the reasoning model failed (vertical) and how it affects the questions the reasoning model already solved (horizontal). Combinations that hurt the solved questions are discarded (hollow), and the best remaining one (star) keeps e2, e3, and e4 but drops e1. Accepting all four edits together would let e1 in and help the failed questions much less.

takes up to 2^n runs in every round. Existing loops therefore settle for the one comparison they can afford: the whole rewrite against the current skill.

Our key observation is that edit-level credit has two halves, knowing where to look and measuring what an edit does there, and that execution feedback and white-box scoring can supply them. The reasoning model's executions reveal failures, and an optimizer model, guided by the optimizer skill, proposes edits from those failures, but evaluating every edit combination through task execution remains costly. A small open-weight model, which we call the decision model, need not match the reasoning model on the task; because its token probabilities are readable, it can measure, with one forward pass per question, how any combination of edits changes the likelihood of the gold answer. Together they supply what each lacks. The reasoning model's outcomes tell the decision model where to look: the questions it failed, which an edit should fix, and the questions it solved, which an edit must not break. The decision model's measurements provide edit-level feedback to the optimizer. The resulting signal is on-policy, since it is measured on the failures and successes of the current skill, and dense, since every question contributes a graded measurement. The decision model does not have to solve the task; it only has to register whether an edit moves it toward the answers the reasoning model missed, and every change it recommends is still confirmed by the reasoning model's own accuracy.

We build on this pairing in SKILLHELIX, which coevolves the task skill and the optimizer skill (Figure 2). In each round, the decision model scores every combination of the proposed edits, keeps the one that most raises the gold-answer likelihood on failed questions without lowering it on solved ones, and performs span evolution on its most influential span; the result replaces the task skill only if validation accuracy improves. The same scores credit every proposed edit with its Shapley value (Shapley, 1953), including the edits that were not selected, and this credit, together with the validation outcome, is used to revise the optimizer skill. The two skills have to evolve together because the value of an edit depends on the skill it is applied to: as the task skill improves, the remaining failures change, and an optimizer skill suited to early failures keeps proposing edits for failures that no longer exist. Each skill is updated using information produced by the other. The optimizer proposes skill updates, while the reasoning model supplies task outcomes and validates the resulting task skill.

We evaluate SKILLHELIX on six benchmarks with five reasoning backbones, comparing skill construction and iterative optimization methods. The experiments report task scores, track gate performance during optimization, and compare pairings of optimizer and reasoner models. Component and edit-pool ablations examine optimizer-skill updates, span evolution, and white-box edit selection. With GPT-5.5 as the reasoning model, SKILLHELIX outperforms all baselines, and the ablations show that both white-box edit selection and optimizer-skill updates contribute to its gains. Our contributions are as follows:

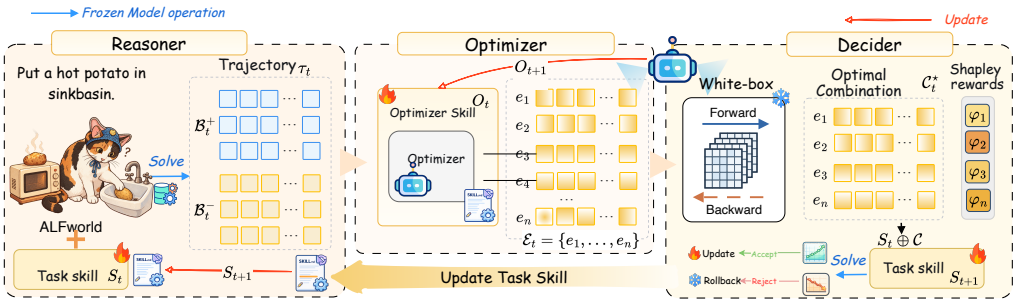

Figure 2: SKILLHELIX. The task skill S_t and the optimizer skill O_t evolve as two coupled strands. The reasoning model $\mathcal{M}$ answers training questions with S_t, and the optimizer model $\mathcal{M}_{\text{opt}}$, guided by O_t, proposes edits from these failures. The decision model p_θ scores every combination of edits on the questions $\mathcal{M}$ failed and solved. These scores select and refine the change to S_t, which is kept only if validation accuracy improves, and they credit each edit in the feedback used to revise O_t.

- We identify rewrite-level credit as a limitation shared by reflective skill optimizers and show how it affects both the task skill and the optimizer skill.
- We introduce SKILLHELIX, which obtains edit-level credit from an open-weight decision model conditioned on the reasoning model's outcomes, and uses it both to select and refine the edits applied to the task skill and to coevolve the optimizer skill.
- We evaluate SKILLHELIX across six benchmarks and five reasoning backbones, and examine its components, edit-pool size, and pairings of optimizer and reasoner models.

2 METHOD

SKILLHELIX optimizes two skills together: a **task skill**, which a black-box reasoning model uses to solve questions, and an **optimizer skill**, which guides an optimizer model in proposing edits to the task skill. Both updates depend on the same missing information, namely which of the proposed edits actually help. The optimizer model can propose edits from the reasoning model's failures, but the only measurement available is end-to-end accuracy, which is costly to obtain and credits a set of edits as a whole. SKILLHELIX obtains edit-level information from a small open-weight decision model that scores any combination of edits on the reasoning model's own failures and successes. The same scores decide how the task skill changes and credit each edit to guide how the optimizer skill changes (Figure 2).

2.1 PROBLEM FORMULATION

A skill S is a natural-language document that provides task-specific guidance to the reasoning model $\mathcal{M}$ and is supplied together with the input. We denote by $\mathcal{M}(\cdot\,; S)$ the model conditioned on skill S. We treat $\mathcal{M}$ as a black-box model, with access only to its textual outputs. Consider pairs of questions and answers $(x, y^\star)$ drawn from a task distribution $\mathcal{D}$, where $y^\star$ denotes the ground-truth answer. Our objective is to construct a task-specific skill S that maximizes the model's prediction accuracy over $\mathcal{D}$.

$$J(S) = \mathbb{E}_{(x,y^\star)\sim\mathcal{D}}\left[\mathbb{I}\{\mathcal{M}(x; S) = y^\star\}\right], \tag{1}$$

which we estimate on a validation set $\mathcal{D}_{\text{val}}$ during optimization and evaluate on a held-out test set. We denote the optimizer model by $\mathcal{M}_{\text{opt}}$, which proposes edits and iteratively refines the optimizer skill.

Skills are improved through edits. At round t, the reasoning model $\mathcal{M}$ answers a batch of training questions $\mathcal{B}_t$ conditioned on the current task skill S_t, partitioning the batch into solved questions $\mathcal{B}_t^+$ and failed questions $\mathcal{B}_t^-$. Conditioned on the optimizer skill O_t, the optimizer model $\mathcal{M}_{\text{opt}}$ takes as input S_t together with the resulting trajectories τ_t and proposes a set of edits $\mathcal{E}_t = \{e_1, \ldots, e_n\}$. Each edit adds, deletes, or replaces a span of S_t and identifies the failures it is intended to address.

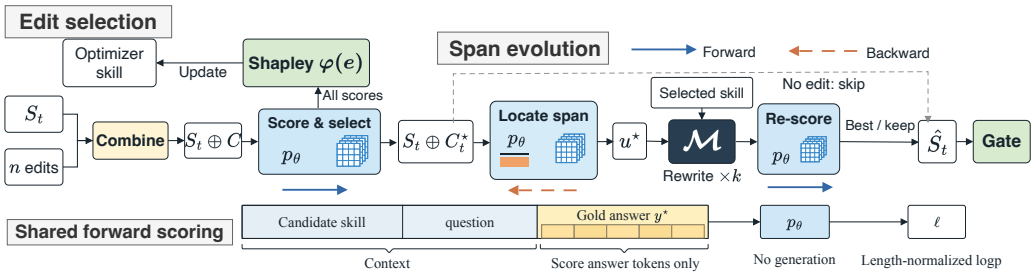

Figure 3: The frozen white-box decision model in SKILLHELIX. Teacher-forced gold-answer likelihoods on black-box successes and failures define proxy scores for edit combinations, supporting combination selection and Shapley attribution.

For any subset $\mathcal{C} \subseteq \mathcal{E}_t$, we denote by $S_t \oplus \mathcal{C}$ the skill obtained by applying the edits in $\mathcal{C}$ to S_t. In all main experiments, we use GPT-5.5 as $\mathcal{M}_{\text{opt}}$ across reasoning backbones; see Appendix A for details.

The optimizer skill induces a distribution over edits, whose value depends on the task skill:

$$Q(O; S) = \mathbb{E}_{\mathcal{E} \sim \mathcal{M}_{\text{opt}}(S, \tau; O)} \Big[\max_{\mathcal{C} \subseteq \mathcal{E}} J(S \oplus \mathcal{C}) \Big] - J(S), \tag{2}$$

where τ denotes the trajectories of $\mathcal{M}(\cdot, ; S)$ on a training batch. Since Q depends on S, the optimal optimizer skill changes as the task skill improves: later rounds exhibit different failures and require different corrective edits. The two skills must therefore be optimized jointly. However, neither objective can be evaluated at the required granularity under black-box access. Evaluating J over all 2^n combinations requires 2^n validation runs per round, each of which credits an entire combination. Revising the optimizer skill requires identifying the contribution of individual edits.

2.2 A DECISION SIGNAL FROM THE REASONING MODEL'S OUTCOMES

We need a signal that can be computed for every combination of edits, that assigns credit to individual edits, and that reflects where $\mathcal{M}$ fails. The reasoning model supplies the last property: its outcomes on $\mathcal{B}_t$ mark the questions an edit should fix and the questions it must not break. An open-weight decision model p_θ, whose token probabilities we can read, supplies the first two. The resulting signal is *on-policy*, since it is measured on the questions that the current task skill fails and solves, and *dense*, since the gold-answer likelihood registers an edit's effect even when no answer flips. Accuracy on a small batch, by contrast, rarely separates one combination of edits from another. For a skill S and a question x, p_θ reads S and x in the prompt format of $\mathcal{M}$, and we teacher-force the gold answer at the answer position:

$$\ell(S; x) = \frac{1}{|y^\star|} \sum_{j=1}^{|y^\star|} \log p_\theta\big(y_j^\star \mid S,\, x,\, y_{<j}^\star\big). \tag{3}$$

Let $\bar{\ell}^-(S)$ and $\bar{\ell}^+(S)$ denote the mean of ℓ over $\mathcal{B}_t^-$ and $\mathcal{B}_t^+$, respectively. The decision value of a combination and the set of admissible combinations are

$$r(\mathcal{C}) = \bar{\ell}^-(S_t \oplus \mathcal{C}) - \bar{\ell}^-(S_t), \qquad \mathcal{A}_t = \big\{\mathcal{C} \subseteq \mathcal{E}_t :\ \bar{\ell}^+(S_t \oplus \mathcal{C}) \geq \bar{\ell}^+(S_t)\big\}. \tag{4}$$

r measures how far a combination moves p_θ toward the gold answers of the questions $\mathcal{M}$ failed; admissibility excludes combinations that move it away from the answers $\mathcal{M}$ already gets right. Scoring one combination takes one forward pass per question and no call to $\mathcal{M}$, so all 2^n combinations can be scored in every round.

Figure 3 illustrates how the frozen decision model supplies this scoring signal.

The decision model is a proxy. We assume that changes which raise its gold likelihood on $\mathcal{M}$'s failures tend to raise $\mathcal{M}$'s accuracy. The reasoning model's validation accuracy determines task-skill acceptance (§2.3).

Algorithm 1 One round of SKILLHELIX

Answer $\mathcal{B}_t$ with $\mathcal{M}(\cdot\,; S_t)$; split into $\mathcal{B}_t^+$, $\mathcal{B}_t^-$
Propose $\mathcal{E}_t \sim \mathcal{M}_{\text{opt}}(S_t, \tau_t;\, O_t)$
Score all $\mathcal{C} \subseteq \mathcal{E}_t$ with p_θ (Eq. 4); compute ϕ (Eq. 6)
$\mathcal{C}_t^\star \leftarrow \arg\max_{\mathcal{C}\in\mathcal{A}_t} r(\mathcal{C})$
if $\mathcal{C}_t^\star \neq \emptyset$ **then**
    Rewrite the span with the largest g (Eq. 5); keep the best admissible alternative
    $S_{t+1} \leftarrow \hat{S}_t$ if its accuracy on $\mathcal{D}_{\text{val}}$ exceeds that of S_t, else S_t
else
    $S_{t+1} \leftarrow S_t$
end if
$O_{t+1} \leftarrow \mathcal{M}_{\text{opt}}(O_t, \mathcal{F}_{t-w+1:t})$

2.3 EVOLVING THE TASK SKILL

The task skill takes the admissible combination with the highest decision value, $\mathcal{C}_t^\star = \arg\max_{\mathcal{C}\in\mathcal{A}_t} r(\mathcal{C})$. Searching all combinations, rather than keeping the edits that help individually, matters because edits interact: an edit may help only together with another, or cancel another's gain. The empty combination is admissible with $r = 0$, so a round can leave S_t unchanged.

Span evolution. The selected edits determine what to change, but their wording comes from a single proposal. We refine it with the same signal at a finer granularity. To decide which span to rephrase, we attribute $\bar{\ell}^-$ to the spans that $\mathcal{C}_t^\star$ introduces:

$$g(u) = \sum_{i\in u} \nabla_{\mathbf{h}_i} \bar{\ell}^-(S_t \oplus \mathcal{C}_t^\star)^\top \big(\mathbf{h}_i - \mathbf{h}_{\text{sp}}\big), \tag{5}$$

where $\mathbf{h}_i$ is the input embedding of token i and $\mathbf{h}_{\text{sp}}$ that of a space token. $g(u)$ is a one-step integrated-gradients estimate (Sundararajan et al., 2017) of how much $\bar{\ell}^-$ would drop if span u were blanked out. The span with the largest g carries most of the fix, so its wording has the most leverage. $\mathcal{M}_{\text{opt}}$ writes k alternatives for this span given the full skill, and the decision model keeps the alternative with the highest $\bar{\ell}^-$ among those that do not lower $\bar{\ell}^+$, retaining the original span unless an alternative is strictly better. The gradient only nominates where to rewrite; the forward evaluation decides what is kept, so a poor nomination costs computation but cannot lower either likelihood.

Because p_θ is only a proxy, the reasoning model makes the final call: the next task skill is $S_{t+1} = \hat{S}_t$ if its accuracy on $\mathcal{D}_{\text{val}}$ exceeds that of S_t, and $S_{t+1} = S_t$ otherwise. When $\mathcal{C}_t^\star = \emptyset$, $S_{t+1} = S_t$ and the gate is skipped.

2.4 EVOLVING THE OPTIMIZER SKILL

Improving the optimizer skill requires knowing which of its edits helped. The accuracy change of a round cannot provide this: it credits the applied combination as a whole, and edits outside that combination are never evaluated by $\mathcal{M}$. The decision values of all 2^n combinations can. Because they are dense, an edit's contribution registers even if it flips no answer, whereas accuracy on a small batch would give most edits zero credit. We credit each edit with its Shapley value (Shapley, 1953) under r,

$$\phi(e) = \sum_{\mathcal{C}\subseteq\mathcal{E}_t\setminus\{e\}} \frac{|\mathcal{C}|!\,(n-|\mathcal{C}|-1)!}{n!} \big[r(\mathcal{C}\cup\{e\}) - r(\mathcal{C})\big], \tag{6}$$

which averages the marginal effect of e over all orders in which the edits could be added. An edit that helps only in combination with another shares that gain, the credits sum to $r(\mathcal{E}_t)$, and every proposed edit receives credit, including those that were not selected. Selection decides what enters the task skill; credit tells the optimizer how each of its proposals fared.

After the gate, $\mathcal{M}_{\text{opt}}$ revises the optimizer skill from a record $\mathcal{F}_t$ of the round, which contains each edit with its evidence and credit, the refined span before and after rewriting with the change in $\bar{\ell}^-$,

and the gate outcome:

$$O_{t+1} = \mathcal{M}_{\text{opt}}\big(O_t,\ \mathcal{F}_{t-w+1:t}\big), \tag{7}$$

where w is the number of recent per-round feedback records supplied to the optimizer. The update follows fixed instructions. SKILLHELIX does not estimate Q in Eq. (2) directly; it supplies the part of Q that an optimizer skill can act on, namely how much each kind of edit contributed. These credits measure each edit's average marginal contribution across combinations under the likelihood objective. Together with the gate outcome, they provide feedback for revising the optimizer skill.

The two updates are coupled: the task skill uses edits proposed under O_t, and the optimizer skill is revised using their effects on S_t. O_t has no separate gate; §3.3 compares SKILLHELIX with a fixed optimizer skill.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

We evaluate SKILLHELIX on six benchmarks spanning question answering, mathematical reasoning, and task execution, following the benchmark and split protocol of SkillOpt (Yang et al., 2026). The five reasoning backbones are GPT-5.5, GPT-5.4, GPT-5.4-mini, GPT-5.4-nano, and Qwen3.6-35B-A3B. GPT-5.5 serves as the optimizer across all reasoning backbones, and Qwen3.5-9B is the frozen white-box decision model. Benchmark details, comparison methods, and exact training/gate/test sizes are provided in Appendix A and Table 3.

3.2 MAIN RESULTS

Table 1 reports the per-model scores, and Figure 4 provides a radar-chart view of the comparisons. SKILLHELIX exceeds the strongest listed alternative in 28 of the 30 combinations of models and benchmarks and ties it in one. On LiveMath, gains over the strongest alternative range from 1.9 to 12.9 score points, including 67.7 versus 54.8 with GPT-5.5. The exception is OfficeQA with GPT-5.4-nano: human-written skills reach 46.5, compared with 25.6 for SKILLHELIX and 16.3 without skills. With GPT-5.5, the radar comparison shows gains over GEPA on all six benchmarks, with the largest difference on LiveMath (24.5 points).

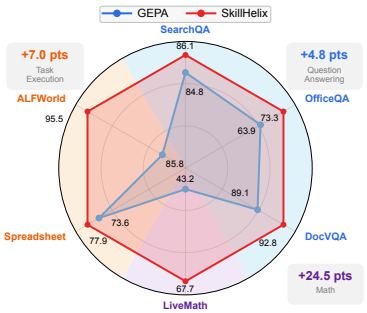

Figure 4: GEPA and SKILLHELIX with GPT-5.5.

Compared with No skill, SKILLHELIX increased the score in all 30 settings. Gains varied across backbones and benchmarks. On DocVQA, GPT-5.4-nano rose from 30.8 to 81.8 (+51.0 points), whereas GPT-5.5 rose from 78.8 to 92.8 (+14.0 points). GPT-5.5 retained the higher final score despite the smaller increase.

Optimization dynamics. Figure 5 shows higher final gate validation scores for SKILLHELIX than SkillOpt in all four settings. The final margins were 7.50 points on Spreadsheet and 4.17 points on OfficeQA for both reasoning models.

3.3 ABLATION STUDIES

Table 2 compares component ablations and edit-pool sizes with GPT-5.5. Removing span evolution or applying all edits reduced scores on all six benchmarks. Freezing the optimizer skill reduced five scores, while OfficeQA increased from 73.3 to 73.8. The default $n = 4$ outperformed $n = 2$ on all six benchmarks and $n = 6$ on four, with ties on DocVQA and ALFWorld.

3.4 OPTIMIZER AND REASONER PAIRINGS

Figure 6 compares pairings of optimizer and reasoner models separately from the main comparison. Rows and columns identify the optimizer and reasoner models, respectively. The heatmap anno-

Table 1: Results across six benchmarks. Δ denotes the absolute score change from *No skill* within the same model. Bold and underlined scores mark the highest and second-highest distinct values for each model and benchmark, including ties. Green/red indicate increases/decreases.

	Question Answering						Math		Task Execution			
	SearchQA		OfficeQA		DocVQA		LiveMath		Spreadsheet		ALFWorld	
Skill source	Score	Δ	Score	Δ	Score	Δ	Score	Δ	Score	Δ	Score	Δ
GPT-5.5												
No skill	77.7	—	33.1	—	78.8	—	37.6	—	41.8	—	83.6	—
Human skill	81.8	↑4.1	66.9	↑33.8	90.1	↑11.3	38.4	↑0.8	72.9	↑31.1	91.8	↑8.2
LLM skill	80.9	↑3.2	51.7	↑18.6	89.6	↑10.8	40.0	↑2.4	43.2	↑1.4	93.3	↑9.7
Trace2Skill	82.4	↑4.7	65.7	↑32.6	90.6	↑11.8	52.0	↑14.4	49.6	↑7.8	87.3	↑3.7
TextGrad	81.4	↑3.7	42.0	↑8.9	87.2	↑8.4	49.2	↑11.6	41.1	↓0.7	82.8	↓0.8
GEPA	84.8	↑7.1	63.9	↑30.8	89.1	↑10.3	43.2	↑5.6	73.6	↑31.8	85.8	↑2.2
SkillOpt	80.4	↑2.7	72.1	↑39.0	91.2	↑12.4	54.8	↑17.2	76.4	↑34.6	**95.5**	↑11.9
SkillHelix	**86.1**	↑8.4	**73.3**	↑40.2	**92.8**	↑14.0	**67.7**	↑30.1	**77.9**	↑36.1	**95.5**	↑11.9
GPT-5.4												
No skill	76.9	—	50.0	—	77.6	—	36.8	—	41.4	—	75.4	—
Human skill	80.1	↑3.2	59.3	↑9.3	88.5	↑10.9	41.6	↑4.8	59.3	↑17.9	74.6	↓0.8
LLM skill	79.4	↑2.5	20.4	↓29.6	90.4	↑12.8	36.8	±0.0	46.1	↑4.7	83.6	↑8.2
Trace2Skill	81.2	↑4.3	51.2	↑1.2	89.3	↑11.7	40.0	↑3.2	53.2	↑11.8	73.1	↓2.3
TextGrad	82.5	↑5.6	54.7	↑4.7	85.3	↑7.7	36.6	↓0.2	38.6	↓2.8	78.4	↑3.0
GEPA	82.4	↑5.5	60.4	↑10.4	88.3	↑10.7	41.6	↑4.8	61.1	↑19.7	74.6	↓0.8
SkillOpt	78.5	↑1.6	62.8	↑12.8	90.4	↑12.8	44.0	↑7.2	61.4	↑20.0	91.0	↑15.6
SkillHelix	**84.1**	↑7.2	**66.8**	↑16.8	**91.2**	↑13.6	**51.6**	↑14.8	**66.1**	↑24.7	**92.5**	↑17.1
GPT-5.4-mini												
No skill	75.9	—	22.1	—	71.4	—	14.7	—	36.1	—	73.1	—
Human skill	77.2	↑1.3	45.9	↑23.8	85.0	↑13.6	28.8	↑14.1	42.9	↑6.8	56.7	↓16.4
LLM skill	76.1	↑0.2	36.6	↑14.5	86.4	↑15.0	28.0	↑13.3	36.8	↑0.7	65.7	↓7.4
Trace2Skill	78.6	↑2.7	20.9	↓1.2	88.5	↑17.1	32.8	↑18.1	40.7	↑4.6	82.8	↑9.7
TextGrad	77.5	↑1.6	30.0	↑7.9	84.0	↑12.6	27.2	↑12.5	38.2	↑2.1	70.9	↓2.2
GEPA	79.4	↑3.5	45.3	↑23.2	83.7	↑12.3	27.2	↑12.5	42.5	↑6.4	81.3	↑8.2
SkillOpt	80.2	↑4.3	48.8	↑26.7	90.9	↑19.5	32.8	↑18.1	47.5	↑11.4	85.8	↑12.7
SkillHelix	**83.2**	↑7.3	**52.9**	↑30.8	**91.4**	↑20.0	**41.9**	↑27.2	**48.6**	↑12.5	**86.6**	↑13.5
GPT-5.4-nano												
No skill	55.8	—	16.3	—	30.8	—	23.2	—	23.5	—	34.3	—
Human skill	66.1	↑10.3	**46.5**	↑30.2	73.5	↑42.7	16.8	↓6.4	41.8	↑18.3	29.9	↓4.4
LLM skill	62.4	↑6.6	12.8	↓3.5	71.7	↑40.9	20.0	↓3.2	33.6	↑10.1	65.7	↑31.4
Trace2Skill	69.9	↑14.1	16.3	±0.0	77.3	↑46.5	25.6	↑2.4	35.4	↑11.9	58.2	↑23.9
TextGrad	73.4	↑17.6	38.7	↑22.4	68.3	↑37.5	20.8	↓2.4	32.5	↑9.0	42.5	↑8.2
GEPA	73.2	↑17.4	45.0	↑28.7	71.2	↑40.4	24.8	↑1.6	37.2	↑13.7	68.7	↑34.4
SkillOpt	74.8	↑19.0	12.2	↓4.1	80.2	↑49.4	27.2	↑4.0	42.5	↑19.0	69.4	↑35.1
SkillHelix	**78.2**	↑22.4	25.6	↑9.3	**81.8**	↑51.0	**33.1**	↑9.9	**45.7**	↑22.2	**70.9**	↑36.6
Qwen3.6-35B-A3B												
No skill	72.7	—	45.9	—	87.6	—	31.2	—	38.2	—	59.7	—
Human skill	74.1	↑1.4	41.9	↓4.0	87.4	↓0.2	29.6	↓1.6	44.3	↑6.1	44.8	↓14.9
LLM skill	72.6	↓0.1	46.5	↑0.6	88.4	↑0.8	24.8	↓6.4	42.9	↑4.7	60.4	↑0.7
Trace2Skill	75.4	↑2.7	32.0	↓13.9	90.4	↑2.8	29.6	↓1.6	33.2	↓5.0	70.9	↑11.2
TextGrad	76.4	↑3.7	33.7	↓12.2	84.5	↓3.1	7.2	↓24.0	22.9	↓15.3	67.9	↑8.2
GEPA	75.8	↑3.1	43.6	↓2.3	88.0	↑0.4	31.2	±0.0	45.4	↑7.2	73.9	↑14.2
SkillOpt	80.3	↑7.6	47.1	↑1.2	91.4	↑3.8	41.6	↑10.4	47.5	↑9.3	82.1	↑22.4
SkillHelix	**80.9**	↑8.2	**47.7**	↑1.8	**91.7**	↑4.1	**43.5**	↑12.3	**47.9**	↑9.7	**83.6**	↑23.9

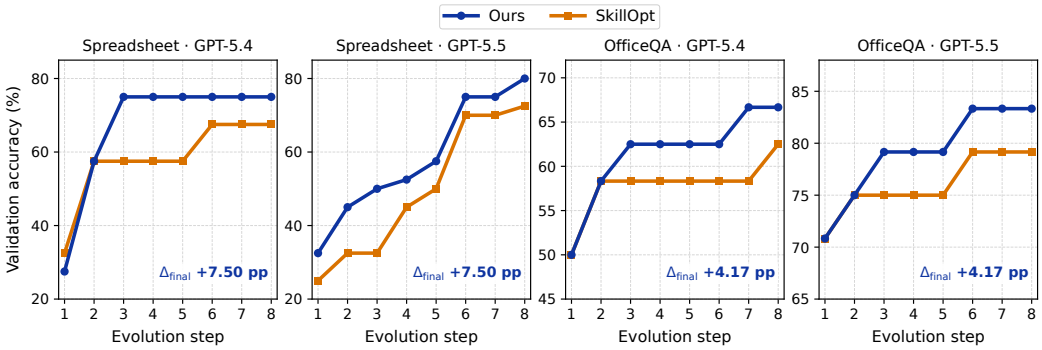

Figure 5: Gate validation scores (%) of SKILLHELIX (Ours) and SkillOpt over eight evolution steps.

Table 2: Ablations with GPT-5.5. Full is SKILLHELIX with edit-pool size $n = 4$. All edits applied bypasses white-box selection. Bold and underlined scores mark the highest and second-highest distinct values in each benchmark, including ties.

	Question Answering			Math	Task Execution	
Variant	**SearchQA**	**OfficeQA**	**DocVQA**	**LiveMath**	**Spreadsheet**	**ALFWorld**
GPT-5.5						
Frozen optimizer skill	83.3	**73.8**	<u>90.9</u>	61.3	74.3	93.3
w/o span evolution	<u>84.1</u>	71.5	89.8	<u>62.9</u>	75.4	<u>94.8</u>
All edits applied	80.6	68.6	89.6	58.1	75.0	89.6
$n = 2$	80.9	71.5	89.3	59.7	75.7	88.8
$n = 6$	82.3	72.7	**92.8**	62.1	<u>76.4</u>	**95.5**
Full ($n = 4$)	**86.1**	<u>73.3</u>	**92.8**	**67.7**	**77.9**	**95.5**

tations report score differences relative to the corresponding column's same-model diagonal. The GPT-5.5 optimizer matched or exceeded that diagonal in every plotted setting. For GPT-5.4-nano on LiveMath, the score increased from 25.8 with the matching optimizer to 33.1 with GPT-5.5.

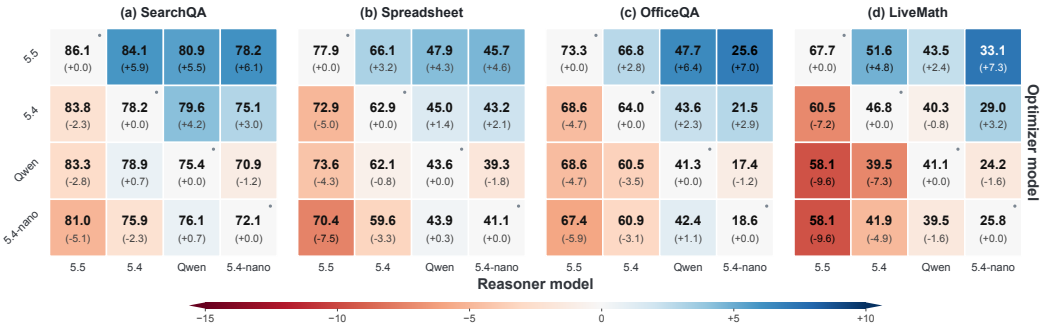

Figure 6: Scores for different pairings of optimizer and reasoner models. Parenthesized differences are relative to the same-model diagonal within each reasoner column.

4 RELATED WORK

Skill and prompt optimization. Agents reuse experience through executable skills and workflow memories (Wang et al., 2024; 2025b), or consolidate feedback into reusable insights (Zhao et al.,

2024; Shinn et al., 2023). Prompt optimizers propose revisions from scored histories (Yang et al., 2024), textual feedback (Pryzant et al., 2023; Yuksekgonul et al., 2025), and reflective evolution. GEPA combines module-level feedback, minibatch screening, and instance-level Pareto selection (Agrawal et al., 2026). Fine-grained editing also predates our work: GrIPS scores instructions assembled from phrase edits (Prasad et al., 2023), while ACE maintains model-assigned bullet utility tags informed by task feedback (Zhang et al., 2026). MIPRO models joint instruction and demonstration choices across modules with a surrogate learned from end-to-end program evaluations (Opsahl-Ong et al., 2024). SkillOpt ranks bounded skill edits by model-estimated utility and validates the resulting skill (Yang et al., 2026). SKILLHELIX retains reflective proposals but evaluates their coalitions with a frozen decision model, assigning Shapley credit to individual edits under a likelihood surrogate rather than relying on a model-assigned utility label.

Optimizing the optimizer. Optimization can also change how candidates are proposed. PE2 designs more effective optimizer instructions (Ye et al., 2024), while Promptbreeder evolves mutation prompts (Fernando et al., 2024) and STOP and Gödel Agent revise improver code or self-modification logic (Zelikman et al., 2024; Yin et al., 2025). ADAS searches agent designs through a fixed meta-agent loop (Hu et al., 2025). SkillOpt updates optimizer-side memory from epoch-level trajectory comparisons (Yang et al., 2026). Our optimizer skill instead receives coalition-based edit attribution together with the validation outcome. Recomputing these contributions on the current task skill's failures couples the updates of the two skills.

Credit assignment and attribution. Credit assignment connects outcomes to individual decisions (Minsky, 1961). Process supervision supplies finer feedback on intermediate reasoning steps (Lightman et al., 2024). SHAP and Integrated Gradients attribute predictions to input features (Lundberg & Lee, 2017; Sundararajan et al., 2017), Data Shapley and its in-run variant value training examples (Ghorbani & Zou, 2019; Wang et al., 2025a), and ContextCite attributes generated statements to context sources (Cohen-Wang et al., 2024). Here the attributed objects are skill edits, and their contributions guide revisions to the optimizer skill. For small sets of n edits, scoring all 2^n coalitions by teacher-forced evaluation yields exact Shapley values for our likelihood surrogate, not for black-box accuracy.

White-box assistance for black-box models. InstructZero generates instructions with an open model, whereas INSTINCT learns a performance predictor over its representations (Chen et al., 2024; Lin et al., 2024). BBox-Adapter reranks generated continuations, and proxy-tuning adjusts decoding logits (Sun et al., 2024; Liu et al., 2024). Likelihood-based prompt selection includes APE's desired-output likelihood and SPELL's prompt perplexity (Zhou et al., 2023; Gonen et al., 2023). Our frozen decision model scores edit coalitions against gold answers on black-box failures, subject to preserving likelihood on currently correct cases. It requires neither a separately trained performance predictor nor modified black-box decoding. Even prompt-format preferences can differ across models (Sclar et al., 2024), so a task-skill update is accepted only after its black-box validation accuracy improves.

5 CONCLUSION

We presented SKILLHELIX, which coevolves task and optimizer skills through a shared white-box decision signal. The decision model evaluates combinations of proposed edits, using the reasoning model's successes and failures to define the scoring objective. Coalition scores guide edit selection, and Shapley values computed from these scores credit individual proposals for revising the optimizer skill. Span evolution adjusts the wording of selected changes, while black-box validation determines whether the updated task skill is kept.

Across six benchmarks and five reasoning backbones, SkillHelix exceeded the strongest listed alternative in 28 of 30 settings and tied it in one. The component ablations quantify how optimizer-skill updates, span evolution, and edit selection affected the final scores. This connects the feedback used to improve a current skill with the instructions that generate future edits, while keeping all model parameters frozen.

AI USE STATEMENT

We used generative AI tools solely to improve the wording and readability of the manuscript, and not for other research tasks. All revisions were reviewed by the authors, who take full responsibility for the final content.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. InstructZero: Efficient Instruction Optimization for Black-Box Large Language Models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6503–6518. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chen24e.html.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Mądry. ContextCite: Attributing Model Generation to Context. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95764–95807. Curran Associates, Inc., 2024. doi: 10.52202/079017-3035. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/adbea136219b64db96a9941e4249a857-Abstract-Conference.html.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-Referential Self-Improvement via Prompt Evolution. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Amirata Ghorbani and James Zou. Data Shapley: Equitable Valuation of Data for Machine Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.html.

Hila Gonen, Srini Iyer, Terra Blevins, Noah Smith, and Luke Zettlemoyer. Demystifying prompts in language models via perplexity estimation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10136–10148, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.679. URL https://aclanthology.org/2023.findings-emnlp.679/.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large Language Models Cannot Self-Correct Reasoning Yet. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32808–32824, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Use Your INSTINCT: INSTruction optimization for LLMs usIng Neural bandits Coupled with Transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 30317–30345. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lin24r.html.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning Language Models by Proxy. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=dribhnhm1i.

Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, January 1961. ISSN 2162-6634. doi: 10.1109/jrproc.1961.287775. URL https://doi.org/10.1109/JRPROC.1961.287775.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3845–3864, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.277. URL https://aclanthology.org/2023.eacl-main.277/.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RIu5lyNXjT.

Lloyd S. Shapley. A value for n-person games. In Harold William Kuhn and Albert William Tucker (eds.), *Contributions to the Theory of Games, Volume II*, volume 28 of *Annals of Mathematics Studies*, pp. 307–317. Princeton University Press, 1953. URL https://ndlsearch.ndl.go.jp/books/R100000136-I1571417124183342080.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson,

K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Haotian Sun, Yuchen Zhuang, Wei Wei, Chao Zhang, and Bo Dai. BBox-Adapter: Lightweight Adapting for Black-Box Large Language Models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47280–47304. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/sun24p.html.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3319–3328. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sundararajan17a.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Jiachen T. Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data Shapley in One Training Run. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=HD6bWcj87Y.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3339f19c5fcee3ad74502947a32be9e6-Abstract-Conference.html.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills, 2026. URL https://arxiv.org/abs/2605.23904.

Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 355–385, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.21. URL https://aclanthology.org/2024.findings-acl.21/.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel Agent: A Self-Referential Agent Framework for Recursively Self-Improvement. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1354. URL https://aclanthology.org/2025.acl-long.1354/.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*,

639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=46Zgqo4QIU.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. Anthropic Engineering, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eC4ygDs02R.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models are Human-Level Prompt Engineers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=92gvk82DE-.

A EXPERIMENTAL DETAILS

A.1 BENCHMARKS AND DATA SPLITS

We evaluate on the six-benchmark suite used by SkillOpt (Yang et al., 2026). SearchQA, OfficeQA, and DocVQA cover question answering; LiveMathematicianBench (LiveMath) covers mathematical reasoning; SpreadsheetBench and ALFWorld cover task execution. Following the data-splitting protocol of SkillOpt, we use separate training, gate, and test sets, with the exact sizes given in Table 3. Training instances provide trajectories and feedback for skill optimization. The gate set is the validation set used to accept or reject candidate task-skill updates, and the held-out test set is used for final evaluation. Model parameters remain frozen throughout skill optimization.

Table 3: Dataset partitions used in our experiments. Gate denotes the validation split for accepting task-skill updates. Counts refer to task instances, including environment tasks for ALFWorld.

Dataset	Train	Gate	Test	Total
SearchQA	400	200	1,400	2,000
LiveMath	35	18	124	177
DocVQA	107	53	374	534
OfficeQA	50	24	172	246
SpreadsheetBench	80	40	280	400
ALFWorld	39	18	134	191

A.2 MODELS AND COMPARISON METHODS

The reasoning backbones are GPT-5.5, GPT-5.4, GPT-5.4-mini, GPT-5.4-nano, and Qwen3.6-35B-A3B. GPT-5.5 is the optimizer model across all five reasoning backbones. Qwen3.5-9B serves as the frozen white-box decision model p_θ. It supplies the teacher-forced reference-target likelihoods and input gradients used for edit evaluation and span evolution, as described in Section 2.

We compare SKILLHELIX with no skill, human-written skills, one-shot LLM skills, Trace2Skill, TextGrad (Yuksekgonul et al., 2025), GEPA (Agrawal et al., 2026), and SkillOpt (Yang et al., 2026). Comparisons are reported separately for each reasoning backbone and benchmark in Table 1, using the benchmark-specific task score.

A.3 OPTIMIZATION SETTINGS AND REFERENCE TRAJECTORIES

We adopt the four-epoch optimization schedule of SkillOpt (Yang et al., 2026). We use an edit-pool size of $n = 4$, yielding $2^4 = 16$ candidate subsets before admissibility filtering. For the selected span, we generate $k = 8$ alternative rewrites. The optimizer skill is updated using feedback from the latest three rounds ($w = 3$).

For ALFWorld, we use successful GPT-5.5 trajectories as reference targets for white-box scoring.