# WHICH SKILL TO DISTILL? SGUID: SELECTING A COMPACT SKILL BANK FOR MODEL–SKILL CO-EVOLUTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Skills, reusable procedural guidance added at inference, can substantially improve LLM downstream performance (Li et al., 2026). Prior work retrieves skills from a bank by semantic relevance, then uses them as inference-time patches or for model distillation. The individual utility of each skill, however, is largely neglected. We first show that, in on-policy distillation where skill-conditioned policies serve as teachers, fewer than $25\%$ of retrieved skills provide useful distillation signals. We then propose SGUID, a method for selecting a compact subset of skills for distillation. SGUID retains a skill only if it consistently yields effective learning signals during training. The selected skills are then distilled to produce a better model. Our results show that *not all* skills are worth distilling. Across four models from the Olmo and Qwen families, distilling 6 selected skills matches or exceeds full-bank distillation in mean avg@12 on three of the four models, and on all four after a second round that distills 3 newly selected skills, while the full banks are up to $11\times$ larger. Importantly, SGUID supports stable model–skill co-evolution: after a distillation round, a new candidate bank is curated from the updated model's rollouts, and SGUID selects which skills to internalize next. In the second round, this loop selects 3 new skills and improves `Qwen3-8B` from 64.3% to 66.3%. The selection step is essential for stability: on `Qwen3-4B`, naively updating the model with unfiltered skills degrades performance, including a $0.3$ percentage point drop on HMMT25, whereas SGUID improves HMMT25 by $0.5$ points after the first round and $1.1$ points after the second. These results identify skill selection as the key mechanism for stable model–skill co-evolution.[1]

## 1 INTRODUCTION

LLMs have demonstrated strong potential to advance challenging real-world domains, varying among reasoning (Liu et al., 2024; Georgiev et al., 2025; Balunovic et al., 2026; Dekoninck et al., 2026; Labiad et al., 2026), coding (Jimenez et al., 2024; Yang et al., 2024; Merrill et al., 2026; Ivison et al., 2026; Yang et al., 2026a), and multi-turn interactions (Barres et al., 2025; Lyu et al., 2026), and more (Phan et al., 2025; Shao et al., 2025). Skills, reusable pieces of knowledge, serve as an intermediate layer between LLMs and downstream tasks. They assist the LLMs by augmenting the prompt with cognitive shortcuts (Didolkar et al., 2025), task procedures, common mistake patterns, or structured packages of code and resources (Anthropic, 2026; Li et al., 2026; Shi et al., 2026). Such task-specific knowledge can be curated from human expertise, the model's own successes and failures, or a mixture of both (Ni et al., 2026; Yang et al., 2026b; Tang et al., 2026), and can be retrieved on request by task relevance (Li et al., 2025; Xia et al., 2026). Skills are shown to be powerful tools that complement model strengths, bridging general-purpose capabilities and downstream expertise (Li et al., 2026; Xie et al., 2026; Didolkar et al., 2025; Ouyang et al., 2026; Xia et al., 2026; Ni et al., 2026; Yang et al., 2026b).

Though skills patch model behavior, the improvements are often averaged over a bank of skills. Yet the *individual utility* is largely understudied. It is unclear if such gains are driven by all skills or only *a fraction of* them. In SWE tasks, for example, Han et al. (2026) find that only 7 out of 49

[1] We plan to release our code upon paper acceptance.



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.

coding skills yield meaningful improvements, while many others have little effect or even degrade performance. This raises two coupled challenges for jointly improving the models and skill banks: how to develop and select non-trivial skills that enter the bank, and how to internalize skill-induced capabilities into the model.

Prior work addresses parts of this picture. Systems that maintain a skill bank select rather than accumulate, keeping only the skills whose usefulness is validated rather than assumed from relevance (Fu et al., 2026; Shi et al., 2026). Meanwhile, on-policy self-distillation has been used to internalize skill-induced behavior into model weights, producing stronger LLMs that no longer require skill patches at inference (Huang et al., 2026a; Wang et al., 2026a). However, it remains open how to identify the skills that are worth distilling, and whether internalizing those can support stable model–skill co-evolution.

To this end, we study skill selection and co-evolution via on-policy self-distillation, where skills serve as privileged context for teacher policies. We first find that across three Qwen models of varying sizes, fewer than $25\%$ of skills provide active learning signals. This suggests that most retrieved skills are not effective teachers for the current student. We then propose SGUID (skill **S**election **GUI**ded by training **D**ynamics), a method that builds a compact skill bank by observing the training dynamics throughout the distillation. In detail, SGUID takes on one distillation run with the whole skill bank, keeping the retrieved skill that produces a consistent and verifier-aligned distillation signal from early to late training. Those skills then form the compact bank used to update the student policy later. The updated model in turn samples rollouts from which a new candidate bank is curated. This whole procedure allows the skill bank and the model to evolve together.

We evaluate SGUID on four models (`Olmo-3-7B`, `Qwen3-1.7B`, `Qwen3-4B`, and `Qwen3-8B`), with AIME24, AIME25, and HMMT25 benchmarks. SGUID distills only 6 skills, yet matches or exceeds full-bank distillation in mean avg@12 on three of the four models, while the full banks are up to $11\times$ larger. On Qwen3-4B, using SGUID instead of full-bank distillation improves performance from $64.2\%$ to $65.1\%$. Using SGUID to distill 3 skills for a second round yields further gains: `Qwen3-8B`'s performance rises from $64.3\%$ to $66.3\%$. Ablations further show that the improvements come from selecting useful and persistent skills rather than merely compressing the bank. Moreover, SGUID is robust to the number of selected skills: compact banks with 4–12 skills remain competitive with, and sometimes outperform, the full bank. We also discuss whether SGUID can be applied over an early training window instead of a full run, which lowers the cost of skill selection.

Our contributions are summarized as follows:

- We propose SGUID, a training-dynamics-based method for selecting compact skill banks for on-policy self-distillation. With only 6 skills, SGUID matches or exceeds full-bank distillation in mean avg@12 on three of the four models, while the full banks are up to $11\times$ larger.
- We show that selected skills enable model–skill co-evolution. Using SGUID for a second round improves `Qwen3-8B` mean avg@12 from $64.3\%$ to $66.3\%$, while co-evolution without skill selection hurts HMMT25 performance.
- We validate the selection signal through ablations on bank size, skill ranking, and the training-dynamics window. Compact banks with 4–12 skills remain competitive, top-ranked skills by our selection signals are most reliable, and training statistics from a 25-step run provide a promising but scale-dependent proxy for standard skill selection with SGUID.

## 2 Background

### 2.1 Problem Setting

We study skill-guided self-distillation for verifiable tasks. Suppose we have a problem $x$ and a reference solution $y^\star$. A language model with parameters $\theta$ defines an autoregressive policy $p_\theta(y \mid x) = \prod_{n=1}^{|y|} p_\theta(y_n \mid x, y_{<n})$. A skill bank $B_\theta$ can be curated from the current policy $\theta$'s rollouts,

verified by the provided ground truth $y^\star$. Informally, for $\hat{y} \sim p_\theta(\cdot \mid x)$,

$$B_\theta = \mathrm{Curate}(\{(x, y^\star, \hat{y}) : \hat{y} \sim p_\theta(\cdot \mid x)\}) . \tag{1}$$

Each skill $s \in B_\theta$ is a compact natural-language procedure, heuristic, or failure-mode description. Skills from the bank are retrieved to serve as training-time context for the teacher (Huang et al., 2026a; Wang et al., 2026a), or as inference-time patches (Yang et al., 2026b; Ni et al., 2026).

### 2.2 On-Policy Self-Distillation

On-Policy Self-Distillation (OPSD) turns privileged information into a dense training signal (Zhao et al., 2026; Hübotter et al., 2026). Given $(x, y^\star)$, the teacher $P_T$, initialized from $\theta$, additionally conditions on $y^\star$, to judge the student $P_S$. This is calculated over responses sampled from the student $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_{|\hat{y}|}) \sim P_S(\cdot \mid x)$. Teacher and student therefore share same parameters and differ only in what they condition on. The relation holds at initialization; during training the teacher is either frozen at that copy or kept synchronized with the student, and we report which choice each method uses in Table 5.

$$\begin{aligned} P_S(y_n \mid x, \hat{y}_{<n}) &= p_\theta(y_n \mid x, \hat{y}_{<n}), \\ P_T(y_n \mid x, y^\star, \hat{y}_{<n}) &= p_\theta(y_n \mid x, y^\star, \hat{y}_{<n}). \end{aligned} \tag{2}$$

OPSD then minimizes the average next-token divergence (e.g., KL) between these two distributions over positions:

$$\mathcal{L}_{\mathrm{OPSD}}(\theta) = \mathbb{E}_{(x,y^\star)} \mathbb{E}_{\hat{y} \sim P_S(\cdot|x)} \left[ D(P_T \| P_S)(\hat{y} \mid x, y^\star) \right] . \tag{3}$$

The divergence $D$ can be further written in terms of the teacher–student log-probability gap. For a vocabulary token $v$, let

$$\Delta_{T,S}(v; x, y^\star, \hat{y}_{<n}) := \log P_T(v \mid x, y^\star, \hat{y}_{<n}) - \log P_S(v \mid x, \hat{y}_{<n}). \tag{4}$$

Then

$$D(P_T \| P_S)(\hat{y} \mid x, y^\star) = \frac{1}{|\hat{y}|} \sum_{n=1}^{|\hat{y}|} \mathbb{E}_{v \sim q_n} \left[ \Delta_{T,S}(v; x, y^\star, \hat{y}_{<n}) \right] , \tag{5}$$

where $q_n$ is a distribution over tokens at position $n$. [2]

### 2.3 On-Policy Self-Distillation with Skills

Researchers find that skill-guided teachers $p_\theta(\cdot \mid x, s)$ provide better learning signals than answer-guided ones $p_\theta(\cdot \mid x, y^\star)$ (Wang et al., 2026a; Huang et al., 2026a). Here, a skill $s$ is retrieved based on the semantic distance between the problem $x$ and skill descriptions.

Both policies score the student rollout. For each sampled token $\hat{y}_n$, we can compute the teacher–student log-probability gap in equation 4, with the skill $s$ instead of $y^\star$: $\Delta_n(s; \hat{y}) := \Delta_{T,S}(\hat{y}_n; x, s, \hat{y}_{<n})$. A positive gap means that the skill-conditioned teacher favors the sampled token more than the student. The plain support score

$$a(s, \hat{y}) = \frac{1}{|\hat{y}|} \sum_{n=1}^{|\hat{y}|} \Delta_n(s; \hat{y}) \tag{6}$$

summarizes the teacher's stance on the rollout. In practice, SGSD (Huang et al., 2026a) uses a robust estimate $\tilde{a}(s, \hat{y})$, obtained by masking irrelevant tokens and clipping extreme gaps before averaging.

The reference solution is used to verify the student's rollout in order to reverse the sign of the teacher signal when it disagrees with the verifier. Let $r(x, y^\star, \hat{y}) = +1$ if the verifier returns correct, and $-1$ otherwise. We write the resulting verifier-aligned teacher sign as Verify:

$$\mathrm{Verify}(s, \hat{y}, y^\star) = \mathrm{sign}(r(x, y^\star, \hat{y})\, \tilde{a}(s, \hat{y})) . \tag{7}$$

[2] For the KL divergence $D_{\mathrm{KL}}(P_T \| P_S)$, $q_n = P_T(\cdot \mid x, y^\star, \hat{y}_{<n})$.

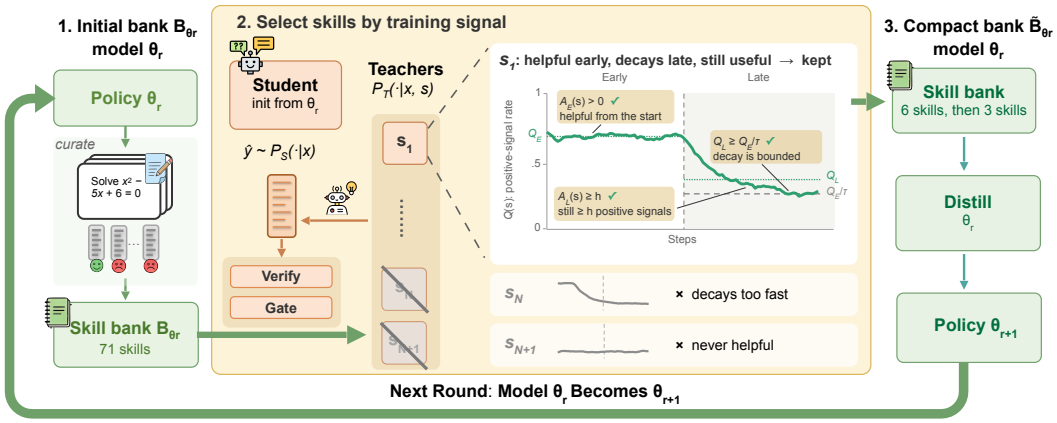


Figure 1: Overview of SGUID at co-evolution round $r$. **(1) Curate:** the current policy $\theta_r$ rolls out on training problems, the rollouts are checked against the reference solutions $y^\star$ to curate a candidate skill bank $B_{\theta_r}$ (the bank size shown is that of `Qwen3-8B`). **(2) Select skills by training signal:** the student $P_S(\cdot \mid x)$, initialized from $\theta_r$, generates a rollout $\hat{y}$ that is scored by privileged teachers $P_T(\cdot \mid x, s_i)$, each conditioned on a retrieved skill $s_i$. Skills that contribute early, still produce at least $h$ positive events late, and whose contribution rate decreases by at most a factor of $\tau$ are kept; skills whose signal decays too fast, stays at zero, or that are never retrieved are discarded. **(3) Internalize:** restarting from $\theta_r$, the compact bank $\widetilde{B}_{\theta_r}$ with $|\widetilde{B}_{\theta_r}| = K \ll |B_{\theta_r}|$ is distilled into the model, producing $\theta_{r+1}$, which becomes the policy for the next round.

A gate then filters out weak signals whose support-score magnitude $|\tilde{a}(s, \hat{y})|$ is at most a confidence threshold $\kappa > 0$:

$$\mathrm{Gate}_\kappa(\tilde{a}(s, \hat{y})) = \mathbf{1}[|\tilde{a}(s, \hat{y})| > \kappa], \tag{8}$$

Using the same teacher–student divergence $D$ as in equation 3, now with the teacher conditioned on $s$, the single-skill form of the skill-guided self-distillation objective is

$$\mathcal{L}_{\mathrm{SGSD}}(\theta) = \mathbb{E}_{(x,y^\star)}\mathbb{E}_{\hat{y}\sim P_S(\cdot|x)}\left[\mathrm{Verify}(s, \hat{y}, y^\star)\,\mathrm{Gate}_\kappa(\tilde{a}(s, \hat{y}))D(P_T\|P_S)(\hat{y} \mid x, s)\right]. \tag{9}$$

This formulation generalizes the original SGSD objective (Huang et al., 2026a), which uses a particular bounded teacher–student discrepancy.

## 3 Method

Figure 1 summarizes SGUID as a three-stage model–skill co-evolution loop. Given a policy $\theta_r$ and its candidate skill bank $B_{\theta_r}$, our method selects a compact subset $\widetilde{B}_{\theta_r} \subseteq B_{\theta_r}$, retaining skills that consistently provide verifier-aligned teacher signals across training. We then distill this compact bank to obtain an updated policy $\theta_{r+1}$.

**Motivation** Retrieval surfaces potentially relevant skills, but it does not guarantee an active learning signal on a student rollout. Figure 2 shows that, among 6,400 logged retrievals per model, only 21.2% for Qwen3-1.7B, 19.7% for Qwen3-4B, and 6.4% for Qwen3-8B are actually used for distillation, while the rest are zeroed out by the gate. This motivates us to ask whether *a compact bank suffices*, including only persistently active skills. We hypothesize that persistently useful skills should be identified using both semantic relevance and the model's own training dynamics.

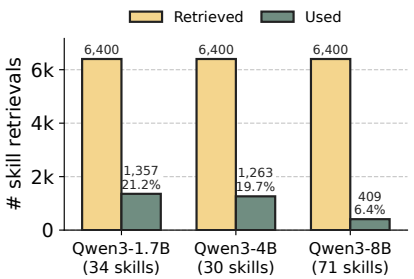


Figure 2: Skill retrieval does not guarantee active teacher signals. $< 25\%$ of skills are active.

**Skill Selection** To identify skills that provide useful and persistent signals, we collect statistics during a training

run with the full candidate skill bank $B_{\theta_r}$ of the current policy $\theta_r$. Let $\mathcal{X} = \{(x_i, y_i^\star)\}_{i=1}^N$ denote the training data. For each $(x, y^\star) \in \mathcal{X}$, the student samples a rollout $\hat{y} \sim P_S(\cdot \mid x)$. For each skill $s \in B_{\theta_r}$, we define its retrieval indicator as $I_{\text{ret}}(s, x) := \mathbf{1}[s \text{ is retrieved for } x]$. When $I_{\text{ret}}(s, x) = 1$, we compute its signed training coefficient as

$$c(s; x, y^\star, \hat{y}) = \text{Verify}(s, \hat{y}, y^\star)\ \text{Gate}_\kappa(\tilde{a}(s, \hat{y}))\,. \tag{10}$$

We anticipate that a retrieved skill $s$ provides a useful contribution on an example when $c(s; x, y^\star, \hat{y}) > 0$. A positive coefficient indicates that the skill-conditioned teacher provides an active signal aligned with the verifier.

We split the distillation into early and late phases, $p \in \{\text{E}, \text{L}\}$. For each skill $s$, let $N_p(s)$ denote its number of retrievals in phase $p$, and let $A_p(s)$ count the retrievals that produce a positive, verifier-aligned coefficient $c > 0$. Whenever $N_p(s) > 0$, its conditional contribution rate is $Q_p(s) = A_p(s)/N_p(s)$.

We retain skills that contribute in the early phase, produce at least $h$ positive signals in the late phase, and whose contribution rate decreases by at most a factor of $\tau > 1$:

$$\mathcal{F}_{\theta_r} = \left\{ s \in B_{\theta_r} : A_{\text{E}}(s) > 0,\ A_{\text{L}}(s) \geq h,\ Q_{\text{L}}(s) \geq \frac{Q_{\text{E}}(s)}{\tau} \right\}. \tag{11}$$

Positive contribution counts imply positive retrieval counts, so both rates are well defined for every skill in $\mathcal{F}_{\theta_r}$. The late-phase count threshold avoids decisions based on sparse events, while the rate constraint removes skills whose useful signal fades during training. Finally, we select the $K$ candidates with the largest $A_{\text{L}}(s)$, breaking ties by $Q_{\text{L}}(s)$, to obtain $\widetilde{B}_{\theta_r}$.

**Skill Internalization and Evolution** After selecting $\widetilde{B}_{\theta_r}$, we restart from the same policy $\theta_r$ and train using only the compact bank. Let $\text{Distill}(\theta, B)$ denote the parameters obtained by running the skill-conditioned gated distillation procedure from $\theta$ with bank $B$; the compact-bank update is

$$\theta_{r+1} = \text{Distill}(\theta_r, \widetilde{B}_{\theta_r}). \tag{12}$$

To continue the co-evolution, we repeat the same procedure with the updated policy $\theta_{r+1}$. Specifically, we sample and verify rollouts from $\theta_{r+1}$ and curate a new candidate bank $B_{\theta_{r+1}}$ using equation 1. Starting from $\theta_{r+1}$, we apply the same selection criterion based on training dynamics to obtain a new compact bank $\widetilde{B}_{\theta_{r+1}}$, and distill only the newly selected skills:

$$\theta_{r+2} = \text{Distill}(\theta_{r+1}, \widetilde{B}_{\theta_{r+1}}). \tag{13}$$

# 4 Experiments

## 4.1 Experimental Setup

**Models and Datasets** We train four models from two families, `Olmo-3-7B`, `Qwen3-1.7B`, `Qwen3-4B`, and `Qwen3-8B`, on DAPO-Math-17K (Yu et al., 2026), following Wang et al. (2026b) and Huang et al. (2026a), and evaluate them on three math reasoning benchmarks: AIME24, AIME25, and HMMT25.

**Baselines** We compare against two types of methods: (1) **Learning w/ correct answers only**: GRPO (Shao et al., 2024) compares the student's answer against the ground-truth answer to obtain a scalar reward. OPSD (Zhao et al., 2026) conditions the teacher policy on the ground truth to provide supervision for the student. (2) **Learning w/ correct answers and skills**: SGSD (Huang et al., 2026a) conditions the teacher on a retrieved skill from a large bank to judge the student outputs. The teacher's supervision signal is then verified with the ground-truth answer. Following their setup, we train the models for 200 steps and report the checkpoint with the best avg@12.

**Implementation Details** We follow Huang et al. (2026a) to build the initial skill bank. The candidate skills are generated by the student's own base model rather than by a stronger external model.

Table 1: Main results across three math reasoning benchmarks. We report the avg@12 accuracy for the best checkpoint according to the average performance on AIME24, AIME25, and HMMT25 across 200 training steps. **Bold** marks the best and underline the second best. # Distilled Skills is the number of skills the teacher can retrieve from; for SGSD we follow the original banks in Huang et al. (2026a). R1 and R2 denote the first and second co-evolution rounds of SGUID (ours). The skills selected for each model and round are listed in Appendix B.

| Model | Method | # Distilled Skills | AIME24 | AIME25 | HMMT25 | Mean |
|---|---|---|---|---|---|---|
| Olmo-3-7B | Base | – | 53.1 | 39.4 | **26.9** | 39.8 |
| | GRPO | – | 53.6 | 42.5 | 25.8 | 40.6 |
| | OPSD | – | 42.5 | 37.5 | 20.8 | 33.6 |
| | SGSD | 38 | 54.4 | 41.1 | 24.4 | 40.0 |
| | SGUID, R1 | 6 | **55.0** | 42.2 | 25.6 | 40.9 |
| | SGUID, R2 | 3 | **55.0** | **42.8** | 26.1 | **41.3** |
| Qwen3-1.7B | Base | – | 50.6 | 35.6 | 23.1 | 36.4 |
| | GRPO | – | 49.2 | 36.9 | 24.7 | 36.9 |
| | OPSD | – | 56.4 | 41.7 | 27.5 | 41.9 |
| | SGSD | 34 | 55.6 | **46.1** | 27.2 | **43.0** |
| | SGUID, R1 | 6 | 55.3 | 43.1 | 28.6 | 42.3 |
| | SGUID, R2 | 3 | **56.9** | 43.3 | **28.9** | **43.0** |
| Qwen3-4B | Base | – | 73.9 | 65.8 | 46.7 | 62.1 |
| | GRPO | – | 76.1 | 66.7 | 45.3 | 62.7 |
| | OPSD | – | **77.8** | 68.9 | 46.1 | 64.3 |
| | SGSD | 30 | 75.6 | 70.3 | 46.7 | 64.2 |
| | SGUID, R1 | 6 | 76.7 | **71.4** | 47.2 | 65.1 |
| | SGUID, R2 | 3 | 76.9 | 70.8 | **47.8** | **65.2** |
| Qwen3-8B | Base | – | 77.5 | 67.2 | 43.9 | 62.9 |
| | GRPO | – | 76.9 | 68.1 | 45.3 | 63.4 |
| | OPSD | – | 78.1 | 69.7 | **48.1** | 65.3 |
| | SGSD | 71 | 77.2 | 67.2 | 47.2 | 63.9 |
| | SGUID, R1 | 6 | 75.3 | 71.9 | 45.6 | 64.3 |
| | SGUID, R2 | 3 | **78.3** | **73.9** | 46.7 | **66.3** |

During training, the thinking mode is disabled for the student and enabled for the teacher, following common practice (Zhao et al., 2026; Huang et al., 2026a); at evaluation time, the thinking mode is enabled. All experiments are conducted on 4 H200 GPUs with LoRA (Hu et al., 2021). The bank size is fixed in advance and shared by every model: we retain $K = 6$ skills in round 1 and $K = 3$ in round 2, motivated by training statistics showing that fewer skills continue to provide persistent signals after the first round of skill internalization. $K$ is never tuned per model, per benchmark, or against test performance, and the selection thresholds are set from training-time statistics alone; the ablation over $K$ in Table 3 is a post-hoc sensitivity check on a single model rather than the procedure used to choose the default. More experimental details are in Appendix A.

### 4.2 Main findings

**A selected skill bank matches or outperforms the full bank.** Table 1 reports the main comparison on AIME24, AIME25, and HMMT25. GRPO yields only modest improvements over the base models, likely because trajectory-level rewards provide sparse supervision. On the Qwen3 models, OPSD and SGSD provide denser token-level supervision and yield larger gains than GRPO. Notably, SGUID achieves comparable or better performance while distilling a much smaller skill bank. In round 1, SGUID (R1) conditions the teacher on only 6 selected skills and already outperforms full-bank distillation (SGSD) across all three benchmarks on both `Olmo-3-7B` and `Qwen3-4B`; compared with SGSD, the mean scores improve from 40.0% to 40.9% and from 64.2% to 65.1%, respectively. On `Qwen3-4B`, SGUID (R1) also improves the base model mean from 62.1% to 65.1% and outperforms OPSD by 0.8 points. The advantage becomes clearer in the second round

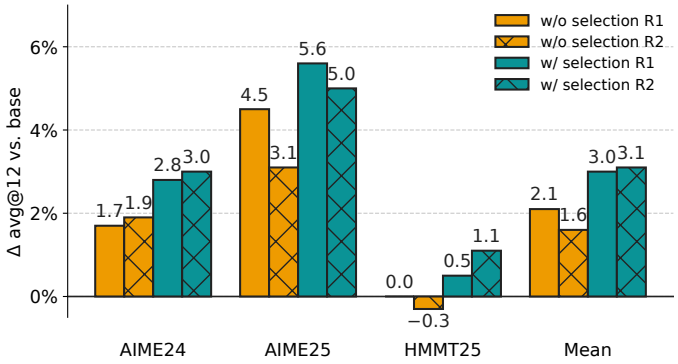


Figure 3: Two rounds of model–skill co-evolution on `Qwen3-4B`. Without selection (distilling the full bank), the mean gain shrinks from round one to round two; in contrast, SGUID (w/ selection) improves the model steadily over rounds.

Table 2: `Qwen3-1.7B` and `Qwen3-4B` avg@12 when the compact bank is selected using training statistics from the full 200-step run (SGUID (full)) or only the first 25 steps (SGUID (25 steps)). The 25-step proxy matches the full run on 1.7B but is less reliable on 4B.

| Model | Method | AIME24 | AIME25 | HMMT25 | Mean |
|---|---|---|---|---|---|
| Qwen3-1.7B | Base | 50.6 | 35.6 | 23.1 | 36.4 |
| | full, 200 steps | 55.3 | 43.1 | **28.6** | 42.3 |
| | 25 steps | **56.1** | **43.9** | 27.5 | **42.5** |
| Qwen3-4B | Base | 73.9 | 65.8 | 46.7 | 62.1 |
| | full, 200 steps | **76.7** | **71.4** | **47.2** | **65.1** |
| | 25 steps | 72.5 | 70.8 | 46.9 | 63.4 |

of co-evolution: with only 3 skills, SGUID (R2) obtains the best mean score on `Olmo-3-7B`, `Qwen3-4B`, and `Qwen3-8B`, and matches SGSD on `Qwen3-1.7B`. These results suggest that distilling a small set of well-selected skills can outperform distilling the full retrieved bank of 30–71 skills, which contains up to 11× more skills than the selected 6-skill bank.

**`Qwen3-8B` benefits most from the second co-evolution round.** SGSD outperforms OPSD on `Qwen3-1.7B` (43.0% versus 41.9%), indicating that retrieved skills compensate for a teacher that is still weak when given the answer alone. However, the gap closes on `Qwen3-4B` and reverses on `Qwen3-8B`, where SGSD retains only a 1.0 point gain over the base model and falls behind answer-only OPSD by 1.4 points, consistent with the observation in Huang et al. (2026a). Once the answer-conditioned teacher is strong enough, skills retrieved from a large unfiltered bank appear to act as noise rather than guidance. In contrast, SGUID does not suffer from this reversal: SGUID (R2) outperforms OPSD at every scale. On `Qwen3-8B`, the second co-evolution round selects 3 new skills from the candidate bank curated from the updated model's rollouts and improves the mean from 64.3% to 66.3%, surpassing OPSD by 1.0 points. This suggests that the degradation stems from *which* skills the teacher is conditioned on rather than from skill conditioning itself.

## 5 Ablation Studies and Discussion

**Skill selection is necessary for stable co-evolution.** We isolate the effect of skill selection by comparing co-evolution with the full skill bank against co-evolution with the selected bank (Figure 3). With the full bank, the mean avg@12 improvement over the base model drops from 2.1 points in round 1 to 1.6 points in round 2, mainly because the gain decreases on AIME25 from 4.5 to 3.1 points and turns negative on HMMT25 by 0.3 points. In contrast, the selected bank gives stronger and more stable gains, improving the base model by 3.0 points in round 1 and 3.1 points

in round 2. This suggests that skill selection is not only a compression step: it stabilizes model–skill co-evolution by filtering stale, redundant, or conflicting skills before they are reused in later distillation rounds.

Table 3: How compact should the bank be? `Qwen3-4B` avg@12 as the number of retained skills $K$ varies. We use $K = 6$ by default in the first round. The last row keeps the same budget but ranks skills by $A_{\mathrm{L}}(s)$ alone, dropping the persistence constraint $Q_{\mathrm{L}}(s) \geq Q_{\mathrm{E}}(s)/\tau$.

| Method | AIME24 | AIME25 | HMMT25 | Mean |
|---|---|---|---|---|
| Base | 73.9 | 65.8 | 46.7 | 62.1 |
| OPSD | **77.8** | 68.9 | 46.1 | 64.3 |
| SGSD | 75.6 | 70.3 | 46.7 | 64.2 |
| $K = 2$ | 76.9 | 69.4 | 43.6 | 63.3 |
| $K = 4$ | 75.6 | 70.8 | 46.9 | 64.4 |
| $K = 6$ | 76.7 | **71.4** | 47.2 | **65.1** |
| $K = 12$ | 75.8 | 68.9 | **49.2** | 64.6 |
| $K = 6$, no persistence | 73.3 | 67.8 | 45.8 | 62.3 |

**SGUID is robust to the choice of bank size.** We ablate the number of retained skills $K$ on `Qwen3-4B` (Table 3). Banks of $K = 4$, 6, and 12 all match or improve over SGSD while using far fewer skills, and the default $K = 6$ is best on average, improving over the base model by 3.0 points. Only the very small bank $K = 2$ falls below SGSD, dropping on HMMT25, which suggests that too few skills lose useful coverage. Selection therefore does not require precise tuning of $K$, although an extremely small bank can hurt robustness. The persistence constraint, in contrast, is not optional: ranking by late-phase positive count alone and dropping the decay bound admits two skills whose contribution rate collapses between the early and late phase, and the mean falls to 62.3%, 2.8 points below SGUID and barely above the base model. Selecting skills that are merely productive early is therefore not enough; they must keep contributing as the student improves.

**Top-ranked skills give the most reliable gains.** We next fix $K = 6$ and vary which 6 skills are retained on `Qwen3-1.7B` (Table 4), ranking the filtered candidates by their late-phase positive count $A_{\mathrm{L}}(s)$ and breaking ties by $Q_{\mathrm{L}}(s)$. The top-ranked bank is the best of the three variants on the mean and on AIME25 and HMMT25, coming within 0.7 points of full-bank SGSD while conditioning the teacher on 6 skills instead of 34. Random and bottom-ranked banks are harder to tell apart and neither matches the top-ranked bank on the mean, so the gain comes from *which* skills are kept rather than from a smaller bank alone.

Table 4: `Qwen3-1.7B` avg@12 with $K = 6$ held fixed, varying only which 6 skills are retained: the top-ranked, random (over 3 seeds for skill selection), or bottom-ranked (worst) 6 according to our selection signal. All skill selection baselines are trained for 100 steps. Bold marks the best value among the three $K = 6$ variants; Base and SGSD are reference rows.

| Method | AIME24 | AIME25 | HMMT25 | Mean |
|---|---|---|---|---|
| Base | 50.6 | 35.6 | 23.1 | 36.4 |
| SGSD | 55.6 | 46.1 | 27.2 | 43.0 |
| top-6 | 55.3 | **43.1** | **28.6** | **42.3** |
| random-6 | 54.1 $\pm$0.8 | 42.6 $\pm$1.2 | 26.9 $\pm$0.9 | 41.2 $\pm$0.9 |
| worst-6 | **55.6** | 42.8 | 26.1 | 41.5 |

**Can we identify which skills to distill at lower cost?** By default, SGUID selects skills using training statistics collected over a full 200-step distillation run with the candidate bank. Table 2 asks whether this can instead be approximated from the first 25 training steps. This approximation works well for `Qwen3-1.7B`, where the 25-step selected skills slightly improve the mean score over the full-run bank (42.5% versus 42.3%). However, it is less reliable for `Qwen3-4B`: the 25-step bank remains above the base model on average but trails the full-run bank by 1.7 points (63.4% versus 65.1%). These results suggest that early training statistics contain useful information, but that reliably predicting the compact skill bank from a much earlier training stage remains an open direction for future work.

## 6 Related Work

**Developing and maintaining the skill bank.** External skill banks can be built through expert curation, community repositories, or agent experience. Although curated skills often improve agent performance, their effects vary across tasks and models, and compact, focused skills can outperform larger bundles (Li et al., 2026). Accordingly, recent methods improve bank quality through large-scale retrieval (Li et al., 2025), trajectory consolidation (Ni et al., 2026), validation-guided editing (Yang et al., 2026b; Alzubi et al., 2026), persistent knowledge accumulation (Tang et al., 2026), or joint experience–skill memory (Jiang et al., 2026). These methods primarily optimize external artifacts for inference with a fixed model. SGUID instead closes the loop through the weights: the selected skills are internalized by distillation, and the updated model curates the next candidate bank, so selection is what carries improvement across rounds.

**Skill internalization and model–skill co-evolution.** On-policy distillation trains on student-generated trajectories with dense supervision signals from the teacher (Agarwal et al., 2024); self-distillation constructs a teacher from the same student policy, where the teacher is allowed to condition on extra privileged information such as ground-truth answers, solutions or environmental feedback (Zhao et al., 2026; Hübotter et al., 2026). Recent work uses natural-language skills as additional privileged context for the teacher: Skill-SD distills trajectory-derived skills through a synchronized teacher, while SGSD validates each skill-conditioned teaching signal against verifier outcomes (Wang et al., 2026a; Huang et al., 2026a). Complementary systems jointly evolve skill selection, utilization, and generation, or couple the skill bank with policy optimization, tools, persistent knowledge, verifiers, or self-play curricula (Shi et al., 2026; Zhang et al., 2026b; Wei et al., 2026; Zhang et al., 2026a; Tang et al., 2026; Huang et al., 2026b). SGUID builds on SGSD but shifts the focus from event-level distillation to bank-level curation: it aggregates verifier-aligned utility over training, selects a compact subset for a focused restart, and uses the updated model to seed the next skill bank.

## 7 Conclusion

Retrieval decides which skills a teacher *sees*, not which ones actually teach. We find that fewer than 25% of semantically retrieved skills produce active, verifier-aligned distillation signals, motivating SGUID for compact-bank selection. SGUID filters skills for persistent verifier-aligned contributions and then ranks the retained candidates by their late-phase positive contribution counts. Across four models, distilling 6 selected skills matches or exceeds full-bank distillation in mean avg@12 on three of the four models, while a second round with 3 newly selected skills does so on all four and improves `Qwen3-8B` from 64.3% to 66.3%. Selection also stabilizes co-evolution: on `Qwen3-4B`, selected banks preserve gains across rounds ($+3.0 \rightarrow +3.1$ points), whereas unfiltered co-evolution decays ($+2.1 \rightarrow +1.6$ points) and drops below the base model on HMMT25. These results show that skill selection is not merely bank compression, but a condition for stable model–skill co-evolution.

## 8 Limitations

We only test on single-turn math reasoning tasks. However, skills and distillation with skills via SFT have proven successful on various multi-turn agentic tasks (Ouyang et al., 2026; Xia et al., 2026; Li et al., 2026; Ni et al., 2026; Yang et al., 2026b), and we therefore expect that our proposed model–skill co-evolution with skill selection can be generalized to multi-turn agentic tasks. We also focus on whether skills can serve as useful teachers in the training process, and we do not discuss whether such skills can serve as a useful inference-time patch. Huang et al. (2026a) find that augmenting base models with skills used in distillation could harm model performance at inference; we hypothesize that this is due to a train-test mismatch, during training, the student has thinking disabled and a maximum output length of 1,024 tokens. Wu et al. (2026), however, find that conditioning on the math skills benefits math reasoning performance at inference time. We leave the co-evolution of skill sets that improve both training and inference to future work.

### AI USE STATEMENT

In this work, the authors only use generative AI to polish the text while the initial draft is written by humans. All AI-generated content is carefully verified by the authors. All research questions, experiments, and conclusions are developed and checked by the authors, without relying on AI assistants.

### REPRODUCIBILITY STATEMENT

All models (`Olmo-3-7B`, `Qwen3-1.7B`, `Qwen3-4B`, and `Qwen3-8B`), the training set (DAPO-Math-17K), and the evaluation benchmarks (AIME24, AIME25, and HMMT25) are publicly available. The SGUID selection rule is fully specified by Equations 10–11 in Section 3, with the corresponding thresholds ($h$, $\tau$, the early/late split, and $K$) listed in Table 5. Training hyperparameters for all baselines and for SGUID, including the LoRA configuration, batch sizes, and hardware, are given in Appendix A; the evaluation protocol (avg@12 and thinking mode) is described in Section 4.1 and Appendix A. The compact skill banks selected in each round for every model are listed verbatim in Appendix B, so that the distillation stage can be reproduced without re-running selection. We will release the source code, the skill-bank curation prompts, and the full candidate banks upon acceptance.

## 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.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Anthropic. A complete guide to building skills for Claude. `https://claude.com/blog/complete-guide-to-building-skills-for-claude`, January 2026.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Advances in Neural Information Processing Systems*, 38, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. Tau2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*, 2025.

Dayuan Fu, Mohan Jiang, Tongyu Wang, Dian Yang, Jiarui Hu, Liming Liu, Jinlong Hou, and Pengfei Liu. davinci-kernel: Co-evolving skill selection, summarization, and utilization via rl for gpu kernel optimization. *arXiv preprint arXiv:2606.16497*, 2026.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale. *arXiv preprint arXiv:2511.02864*, 2025.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering? *arXiv preprint arXiv:2603.15401*, 2026.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Jiazhen Huang, Xiao Chen, Xiao Luo, Yong Dai, Senkang Hu, and Yuzhi Zhao. Skill-conditioned gated self-distillation for LLM reasoning. *arXiv preprint arXiv:2605.28791*, 2026a.

Siyuan Huang, Pengyu Cheng, Haotian Liu, Tao Chen, Yihao Liu, Jingwei Ni, Shijie Zhou, Ziyi Yang, Gangwei Jiang, Mengyu Zhou, et al. Skill Self-Play: Pushing the frontier of LLM capability with co-evolving skills. *arXiv preprint arXiv:2607.22529*, 2026b.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *arXiv preprint arXiv:2606.23321*, 2026.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R Fung. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 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.

Ismail Labiad, Matthieu Kowalski, Marc Schoenauer, Rémi Munos, and Julia Kempe. Beyond repeated sampling: Learning search policies for llm reasoning. 2026. URL `https://api.semanticscholar.org/CorpusID:292227599`.

Fangzhou Li, Pagkratios Tagkopoulos, and Ilias Tagkopoulos. Skillflow: Scalable and efficient agent skill retrieval system. *arXiv preprint arXiv:2504.06188*, 2025.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Yuanjie Lyu, Chengyu Wang, Haonan Zheng, Yuanhao Yue, Junbing Yan, Ming Wang, and Jun Huang. Agenticqwen: Training small agentic language models with dual data flywheels for industrial-scale tool use. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pp. 535–551, 2026.

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.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Yijia Shao, Humishka Zope, Yucheng Jiang, Jiaxin Pei, David Nguyen, Erik Brynjolfsson, and Diyi Yang. Future of work with ai agents: Auditing automation and augmentation potential across the us workforce. *arXiv preprint arXiv:2506.06576*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026.

Liyan Tang, Cyrus Rashtchian, Chun-Sung Ferng, Andrew Tomkins, Da-Cheng Juan, and Tu Vu. WikiSkill: Compiling agent experience into persistent knowledge for skill evolution. *arXiv preprint arXiv:2608.27454*, 2026.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents. *arXiv preprint arXiv:2604.10674*, 2026a.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486, 2026b.

Yangbo Wei, Zhen Huang, Shaoqiang Lu, Junhong Qian, Qifan Wang, Chen Wu, and Lei He. SkillSmith: Co-evolving skills and tools for self-improving agent systems. *arXiv preprint arXiv:2606.01314*, 2026.

Di Wu, Devendra Singh Sachan, Wen-tau Yih, and Mingda Chen. Procedural knowledge at scale improves reasoning. *arXiv preprint arXiv:2604.01348*, 2026.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Yiqing Xie, Emmy Liu, Gaokai Zhang, Nachiket Kotalwar, Shubham Gandhi, Sathwik Acharya, Xingyao Wang, Carolyn Rose, Graham Neubig, and Daniel Fried. Hybrid-gym: Training coding agents to generalize across tasks. *arXiv preprint arXiv:2602.16819*, 2026.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, Rui Hou, Gabriel Synnaeve, Diyi Yang, and Ofir Press. Programbench: Can language models rebuild programs from scratch?, 2026a. URL `https://arxiv.org/abs/2605.03546`.

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*, 2026b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026a.

Zhiwei Zhang, Yudi Lin, Nikki Lijing Kuang, Linlin Wu, Xiaomin Li, Songtao Liu, and Fenglong Ma. Co-evolving skill generation and policy optimization. *arXiv preprint arXiv:2606.08755*, 2026b.

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.

# A EXPERIMENTAL DETAILS

Table 5: Training hyperparameters for each method. LoRA is applied to `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj` and `down_proj` in all runs. Dashes mark settings that do not apply.

| Hyperparameter | GRPO | OPSD | SGSD | SGUID |
|---|---|---|---|---|
| Learning rate | $5 \times 10^{-6}$ | $5 \times 10^{-6}$ | $5 \times 10^{-6}$ | $5 \times 10^{-6}$ |
| Completion length | 16000 | 1024 | 1024 | 1024 |
| Prompt length | 2048 | 20000 | 20000 | 20000 |
| Temperature | 1.2 | 1.1 | 1.1 | 1.1 |
| Top-$p$ / top-$k$ | – | 0.95 / 20 | 0.95 / 20 | 0.95 / 20 |
| Generations per prompt | 8 | 1 | 1 | 1 |
| Teacher count per prompt | – | 1 | 8 | 6 (R1) / 3 (R2) |
| Teacher update | – | fixed | live (synchronized) | live (synchronized) |
| Skill retriever | – | – | `Qwen3-Embedding-0.6B` (cosine) | |
| Gate threshold $\kappa$ | – | – | 0.05 | 0.05 |
| Support clip $\delta$ | – | – | 3.0 | 3.0 |
| Early/late split step | – | – | – | 75 / 100 (R1) |
| Min late-phase positives $h$ | – | – | – | 10 (R1) |
| Max rate decay $\tau$ | – | – | – | 5 (R1) |
| Retained skills $K$ | – | – | – | 6 (R1) / 3 (R2) |
| Per-device batch | 2 | 4 | 1 | 1 |
| Gradient accumulation | 4 | 2 | 4 | 4 |
| Effective batch | 32 | 32 | 16 | 16 |
| Training steps | 200 | 200 | 200 | 200 |
| Max grad norm | – | 0.1 | 0.1 | 0.1 |
| LoRA rank $r$ | 64 | 64 | 64 | 64 |
| LoRA $\alpha$ | 128 | 128 | 128 | 128 |
| LoRA dropout | 0.05 | 0.05 | 0.05 | 0.05 |

**Skill selection settings.** Skills are retrieved by cosine similarity between the problem statement and the skill description. The compact-bank distillation run retrieves the $K$ selected skills and disables dynamic update.

## B COMPACT SKILL BANKS

Tables 6–9 list the compact skill banks that SGUID selects for each model, in both co-evolution rounds. Each round curates its own candidate bank from the rollouts of the current policy, and skill identifiers are assigned within that bank. Identifiers are therefore local to a round: the same identifier in the round-1 and round-2 blocks of a table refers to two different skills.

Table 6: Compact skill banks selected for `Qwen3-1.7B`.

| ID | Title | Principle |
|---|---|---|
| *Round 1 ($K = 6$)* | | |
| `gen_007` | Solve triangle inequalities | Ensure sequences maintain triangle inequality conditions by enforcing each term $<$ sum of two preceding terms. |
| `gen_017` | Balanced parity counting | When the number of even and odd elements are equal, the minimum number of swaps required is half the total number of elements. |
| `gen_019` | Count with constraints | Calculate the number of valid configurations under specified constraints using combinatorial methods or factor pair analysis. |
| `gen_021` | Integer factor pair analysis | Identify valid integer factor pairs for area or perimeter constraints to determine dimensions. |
| `gen_023` | Count under constraints with digit analysis | Calculate the number of integers within a range that meet specific conditions, including multiples and digit patterns. |
| `gen_025` | Pythagorean triple generation | Generate primitive Pythagorean triples using integer parameters $m > n$ and verify area-to-perimeter ratios. |
| *Round 2 ($K = 3$)* | | |
| `gen_007` | Use Modular Arithmetic | Leverage mathematical structures (e.g., prime factorizations, digit sums) and modular arithmetic to narrow candidate sets and validate constraints. |
| `gen_009` | Compute 3D Distances | Calculate distances from points to lines in 3D using vector cross products: $\|a \times b\|/\|b\|$. |
| `gen_026` | Number-Theoretic Analysis | Apply modular arithmetic, prime factorization, or parity checks to evaluate divisibility, congruence, or primality properties. |

Table 7: Compact skill banks selected for `Qwen3-4B`.

| ID | Title | Principle |
|---|---|---|
| *Round 1 ($K = 6$)* | | |
| gen_002 | Simplify Expressions | Decompose complex expressions or ratios into simpler components using algebraic techniques, factoring, or number theory to reduce complexity. |
| gen_001 | Translate Constraints to Algebra | Convert geometric, number-theoretic, or combinatorial conditions into algebraic equations or expressions to isolate variables or derive relationships. |
| gen_006 | Analyze Symmetries | Adjust counts or configurations by dividing total possibilities by symmetry group order to account for equivalent arrangements. |
| gen_010 | Constraint Optimization | Maximize/minimize expressions or narrow down cases by analyzing variable relationships, parity, or integer properties under constraints. |
| gen_007 | Validate Solutions | Test candidate solutions against original constraints or edge cases to ensure consistency across scenarios. |
| gen_018 | Angle and Ratio Analysis | Apply trigonometric identities or vector proportionality to resolve relationships between angles, directions, or ratios. |
| *Round 2 ($K = 3$)* | | |
| gen_006 | Number Theory and Optimization | Apply properties of primes, factorials, or modular arithmetic to simplify constraints or optimize number constructions. |
| gen_011 | Invariant and Recursive Reasoning | Identify invariants, thresholds, or recursive patterns to determine bounds, stability, or long-term behavior in sequences or systems. |
| gen_001 | Constraint and Probability Analysis | Translate constraints into equations or logical conditions, and apply combinatorial or probabilistic reasoning to evaluate outcomes under structured scenarios. |

Table 8: Compact skill banks selected for `Qwen3-8B`.

| ID | Title | Principle |
|---|---|---|
| *Round 1 ($K = 6$)* | | |
| gen_045 | Geometric Constraint Modeling | Formulate algebraic equations from geometric relationships and properties. |
| gen_009 | Number Theory | Apply divisibility rules, modular arithmetic, and prime factorization to analyze number properties and solve equations. |
| gen_046 | Tangency Relationship Derivation | Use geometric perpendicularity properties to derive solution relationships. |
| gen_024 | Combinatorial Counting | Systematically count configurations by iterating through constraints or categorizing arrangements. |
| gen_039 | Integer Scaling Optimization | Minimize target quantities by testing minimal integer scaling parameters with common factors. |
| gen_042 | Domain Constraint Application | Limit analysis to feasible domains using function bounds and variable constraints. |
| *Round 2 ($K = 3$)* | | |
| gen_004 | Case Analysis | Systematically count solutions by bounding search spaces or splitting into cases based on constraints. |
| gen_076 | Combinatorial Counting | Count configurations using permutations, combinations, or recursive decomposition of constraints. |
| gen_039 | Combinatorial Analysis | Systematically count configurations, permutations, and combinations with constraints using enumeration or algebraic techniques. |

Table 9: Compact skill banks selected for `Olmo-3-7B`.

| ID | Title | Principle |
|---|---|---|
| *Round 1 ($K = 6$)* | | |
| gen_027 | Diophantine Analysis with Parameterization | Analyze integer solutions by fixing parameters and applying Diophantine conditions. |
| gen_010 | Translate Verbal or Qualitative Problems to Mathematics | Convert non-technical or narrative problem statements into mathematical models or equations. |
| gen_002 | Simplify Expressions Using Algebra and Number Theory | Apply algebraic and number-theoretic techniques to reduce expression complexity. |
| gen_007 | Evaluate and Sum Series | Apply closed-form formulas to compute or analyze arithmetic, geometric, or power series. |
| gen_022 | Geometric Mean and Right Triangle Properties | Apply geometric mean properties of inscribed squares in right triangles to find areas or lengths. |
| gen_014 | Use Modular Arithmetic for Simplification and Validation | Simplify calculations using modular arithmetic for divisibility and large integers. |
| *Round 2 ($K = 3$)* | | |
| gen_114 | Factorize and enumerate integer solutions | Use factorization of equations (e.g., quadratics) to enumerate valid integer parameters. |
| gen_009 | Planar triangulation and edge counting | For a convex $n$-vertex polygon, the maximum number of internal triangles in a triangulation is $T = 2n - 4$. |
| gen_076 | Apply divisibility rules to transform sum constraints | Convert divisibility requirements into sum conditions (e.g., digit sums for divisibility by 3 or 9). |