AUTOSCAFFOLD: AI-GUIDED SCAFFOLDING FOR REINFORCEMENT LEARNING OF LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

AI-for-AI can improve reinforcement learning (RL) through automated training configuration. We extend this paradigm to adaptive training-time scaffolding with AutoScaffold. A frozen teacher LLM analyzes a student's evolving bottlenecks and provides strategies and hints to generate informative experience when unscaffolded rollouts yield little learning signal. With minimal domain-specific information, the teacher examines training evidence, proposes class-level strategies and task-specific hints, and adjusts their use, guided by held-out validation of class-level revisions. This process requires no hand-crafted curriculum or human intervention during training. Across embodied tasks, search-based question answering, mathematical reasoning, and GPU kernel generation, AutoScaffold improves over corresponding RL baselines when evaluated without scaffolds. Gains include 7.2 percentage points on ALFWorld and 3.8 points in average question-answering exact match. On mathematical reasoning, it surpasses QuestA's human-designed scaffolding strategy with approximately one-third as many student training steps. The teacher also rediscovers intuitive strategies, including withdrawing support as the student learns. These results suggest that adaptive training assistance can convert teacher guidance into the student's own capabilities.

1 INTRODUCTION

Using AI to improve AI (AI-for-AI) has emerged as a promising approach to automating model development. Recent work shows that LLMs can optimize inference harnesses (Lee et al., 2026), develop efficient GPU kernels (Zhang et al., 2025), and assist AI research (Yamada et al., 2025). Among these directions, automating *reinforcement learning (RL) post-training* is particularly compelling because it can improve the model's own capabilities. Existing efforts have explored this direction through training configuration: for example, Chen et al. (2026) use an LLM to redesign training environments, while Karpathy (2026) and Fang et al. (2026) use LLM agents to search for better training strategies.

However, effective RL training requires more than selecting an appropriate configuration. Researchers must analyze training dynamics at the level of individual tasks, identify model's behaviors and why the model fails, then determine what support could help it learn more effectively from those tasks, like using curriculum learning, changing the mixture of data, or providing scaffold. These decisions also need to be optimized as the model's capabilities evolve: support that is useful early in training may become unnecessary or misaligned with new bottlenecks. Automating diagnosis and support design is therefore an important step toward AI-driven RL post-training.

For instance, a concrete bottleneck is the *learning cliff*: when a task exceeds the model's current capabilities, almost every sampled rollout fails (Zhang et al., 2026). Under group-relative RL with binary rewards, such all-failure groups produce zero relative advantages and provide no reward-based policy-gradient signal. A *scaffold* can supply additional training-time guidance, such as a strategy or partial solution, to make successful rollouts accessible. Existing methods demonstrate several forms of support: Scaf-GRPO introduces tiered hints when learning stagnates (Zhang et al., 2026), QuestA augments mathematical questions with partial solutions (Li et al., 2026a), and SkillRL and Skill0 use reusable skills for agentic tasks (Xia et al., 2026; Lu et al., 2026). However, these scaffolding mechanisms are often engineered on a case-by-case basis around particular task structures. A mathematical solution prefix, for example, does not directly provide the action sequencing or

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.

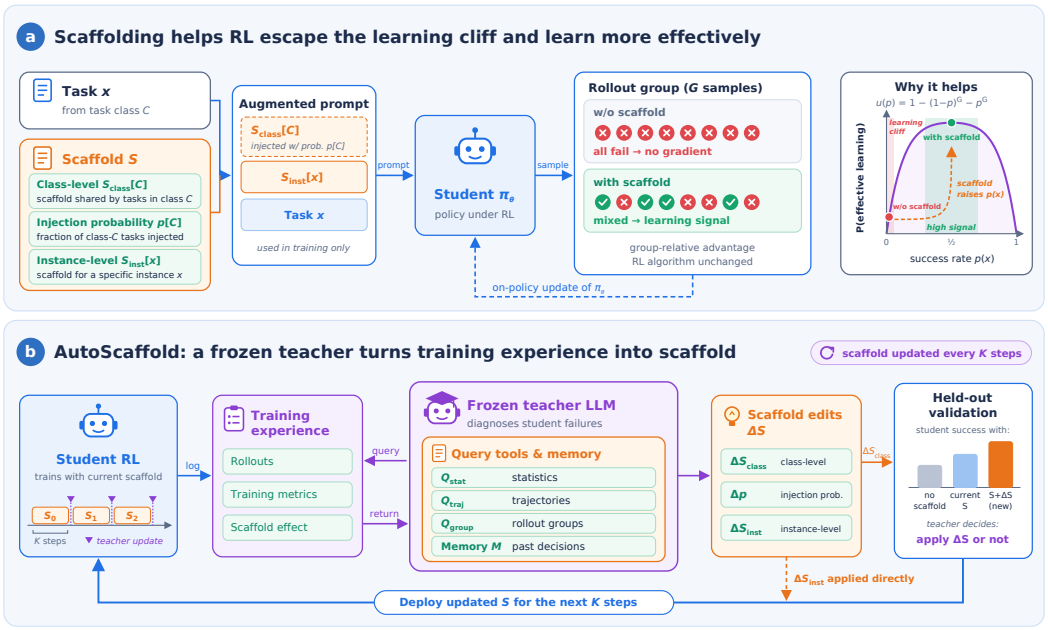


Figure 1: **Overview of AUTOSCAFFOLD. (a)** Scaffold, injected into the student's training prompts, helps student obtain training signal on hard tasks and learn effectively, supported by analysis in Appendix A. **(b)** Every K steps, a frozen teacher analyzes the training experience, proposes scaffold edits ΔS, and validate it to ensure the quality. The updated scaffold is used for the next K steps.

object-interaction strategies needed for an ALFWorld task. Extending scaffolding across domains therefore requires deciding what form of support is appropriate and how to construct it from the student's failures. Moreover, support must evolve with the student, because guidance that initially enables learning may later become redundant or encourage dependence on supplied information.

Adaptive scaffolding methods have begun to address the student's changing needs. SAGE refreshes self-generated hints but adjusts their strength through predefined rollout-based rules (Liao et al., 2026). SkillRL revises skills from sampled validation failures, but its teacher operates on a prescribed evidence sample, and proposed revisions are incorporated without a separate comparison against the existing library (Xia et al., 2026). When sampled failures are inconclusive, the teacher needs to inspect additional evidence; when several scaffold revisions are plausible, it needs feedback on which ones actually help the current student. They motivate a more autonomous scaffold-design process that can actively investigate the student's difficulties and empirically assess proposed interventions as learning progresses.

We therefore study how LLM can automate scaffold design during RL post-training, with the goal of helping the student learn more effectively and solve tasks independently at inference time. We propose AutoScaffold, an AI-for-AI framework in which a frozen teacher LLM observes an ongoing RL training run, designs training scaffold, and injects it into the student's training prompts. The approach draws on an important educational principle of scaffolding, that learner-contingent support that enables a learner to accomplish tasks beyond their current independent ability (Wood et al., 1976; Vygotsky, 1978). AutoScaffold turns this principle into an automated process that connects failure diagnosis, scaffold construction, and empirical feedback throughout training.

More specifically, in our proposed AutoScaffold (Figure 1), the teacher operates in a *closed loop*. Every K training steps, it uses a set of query tools to query training statistics, inspect individual trajectories or groups of attempts on the same task, and review previous scaffold decisions. The teacher chooses which experience to examine and uses it to diagnose failures, construct class-level strategies or instance-specific hints, and adjust class-level injection probabilities. Before deploying class-level edits, it validates the edits on held-out evaluation of the current student, comparing with existing, proposed, and no scaffolds, then decides which edits to retain, revise, or discard. Subsequent training provides evidence for the next cycle. This process automates decisions about what support to provide and when to revise it, using a shared teacher procedure across domains instead of

a separately crafted scaffold curriculum for each task. Our analysis in Appendix A further motivates this adaptation by showing how the effect of scaffolds on group-relative learning signals depends on the student's current success probabilities.

We evaluate AutoScaffold across four domains with distinct task structures and RL algorithms: ALFWorld (embodied household tasks, GiGPO), search-based question answering (GiGPO), KernelBench (GPU kernel generation, rubric-based GRPO), and mathematical reasoning (GRPO and DAPO). **All evaluations use the student without scaffolds.** AutoScaffold improves over the corresponding RL baselines across these domains and is competitive with or surpasses methods that use human-designed scaffolding procedures. On mathematical reasoning, it also surpasses QuestA with approximately one-third as many student training steps. Ablations highlight the value of interactive evidence access and scaffold validation; scaffold evolution shows how the teacher adjusts content and injection as the student learns.

Our main contributions are as follows:

- We introduce AutoScaffold, an AI-for-AI framework that automates scaffold design during RL post-training. A frozen teacher LLM diagnoses student failures, constructs support, and revises its content and injection based on live training evidence, using a shared procedure across task domains.
- We design the AutoScaffold system to include functional blocks such as interactive evidence access, decision memory, structured scaffold editing, probabilistic class- and instance-level prompt injection, and held-out validation of class-level proposals. Together, these components allow the teacher to investigate failures, intervene in training, and revise its decisions using empirical feedback. Ablations demonstrate the importance of these components for effective scaffolding.
- To guide AutoScaffold in matching support to the student's evolving capabilities, we analyze how scaffold injection affects group-relative learning signals under binary rewards (Appendix A). The results motivate AutoScaffold's use of complementary class-level and instance-level scaffolds and its periodic revision of support as the student learns.
- We evaluate AutoScaffold across four domains with distinct task structures and RL algorithms, consistently improving over the corresponding RL baselines under scaffold-free evaluation. On mathematical reasoning, AutoScaffold surpasses QuestA with approximately one-third as many student training steps. Analyses of scaffold evolution further illustrate how the teacher adjusts support to the student's changing capabilities.

2 Related Work

AI for automating model development. LLMs increasingly improve AI systems themselves, optimizing harnesses (Lee et al., 2026), writing GPU kernels (Ouyang et al., 2025; Zhang et al., 2025), and conducting parts of research (Lu et al., 2024; Yamada et al., 2025; Karpathy, 2026). For model training, LLMs have designed rewards (Ma et al., 2024), generated environments (Chen et al., 2026), tuned hyperparameters (Liu et al., 2025), and searched for RL training strategies (Fang et al., 2026), and PostTrainBench (Rank et al., 2026) evaluates end-to-end automated post-training. These methods operate on training configuration; AutoScaffold instead intervenes within a run at the level of task classes and instances, turning the student's rollouts into guidance attached to its training prompts.

Scaffolding in RL for LLMs. To address the learning cliff (Zhang et al., 2026), prior work supplies task-specific hints, such as tiered hints in Scaf-GRPO (Zhang et al., 2026), partial solutions in QuestA (Li et al., 2026a), and self-generated hints in SAGE (Liao et al., 2026), or reusable skills for agentic tasks (Xia et al., 2026; Lu et al., 2026; Lin et al., 2026). The form of guidance is typically designed by hand for one domain, and any adaptation follows predefined rules. The closest methods are SAGE, which adjusts hint strength with rollout-based rules, and SkillRL, whose teacher revises skills from a fixed sample of failures without measuring their effect. In contrast, AutoScaffold lets the teacher choose which evidence to inspect, applies one procedure across domains, and validates class-level proposals on the current student before deployment.

3 THE PROPOSED AUTOSCAFFOLD APPROACH

3.1 PRELIMINARY: SCAFFOLDS FOR INFORMATIVE RL TRAINING

We explain how scaffolds make difficult tasks informative for GRPO and introduce two complementary forms of support.

Why scaffolds are needed. GRPO (Shao et al., 2024) samples G rollouts per task x and computes policy advantages from their reward differences. Under binary rewards, all-failure groups yield zero advantages (Zhang et al., 2026; Liao et al., 2026). Raising rollout success probabilities can make otherwise uninformative tasks useful for training (Li et al., 2026a; Liao et al., 2026).

Augmenting the prompt with a scaffold S, such as a strategy, hint, or partial solution, can raise success probability from $p_\theta(x)$ to $p_\theta(x, S)$ without changing the reward. Proposition 2 in Appendix A quantifies the resulting reward contrasts: for independent binary-reward rollouts with $p = p_\theta(x, S)$,

$$\mathbb{E}[V \mid x, S] = \frac{G-1}{G} p(1-p), \qquad V = \frac{1}{G}\sum_{i=1}^{G}(R_i - \bar{R})^2, \tag{1}$$

where $\bar{R}$ is the group's mean reward. Raising a low success probability toward $\frac{1}{2}$ thus increases expected reward variation and enables more informative rollout comparisons.

Scaffold content and granularity. A *class-level* scaffold $\mathcal{S}_{\text{class}}[C]$ shares strategies across tasks in class C, as in SkillRL and Skill0 (Xia et al., 2026; Lu et al., 2026). Proposition 3 characterizes its benefit when support raises mean success probability while preserving task heterogeneity. An *instance-level* scaffold $\mathcal{S}_{\text{inst}}[x]$ supplies task-specific information, such as Scaf-GRPO's tiered hints, QuestA's solution prefixes, or SAGE's self-generated plans (Zhang et al., 2026; Li et al., 2026a; Liao et al., 2026). Proposition 4 quantifies its benefit when success probability rises from zero into $(0, 1)$.

Static scaffolds can miss emerging difficulties or persist after the student no longer needs them, leaving some tasks insufficiently supported and others unnecessarily assisted. Adjusting injection frequency alone cannot repair outdated guidance; its content, target tasks, and benefit must also be reassessed as the student learns.

3.2 AUTOMATIC SCAFFOLD DESIGN AND ADAPTATION

Our proposed AutoScaffold automates scaffold content, granularity, and use during RL training to improve the student's ability to solve tasks without assistance.

The challenges. Scaffold design requires diagnosing difficulties and adapting support to the current policy. Low success rates alone do not distinguish a missing strategy from a need for task-specific information. Moreover, Proposition 1 in Appendix A shows that, for G independent binary-reward rollouts with $p = p_\theta(x, S)$, the probability of mixed outcomes is $u(p) = 1 - (1-p)^G - p^G$. It peaks at $p = \frac{1}{2}$ and vanishes at both extremes: guidance that breaks an all-failure pattern may later yield mostly all-success groups. This motivates adapting guidance strength, while selecting its form and content requires evidence about the student's behavior and response to intervention.

The proposed design. We ground scaffold decisions in current training evidence and empirical feedback. As shown in Figure 1(b), every K student updates a frozen teacher LLM $\mathcal{T}$ investigates failures, proposes edits, and reviews their effects before deploying class-level changes. At scaffold-update cycle k, it controls $S_k = (\mathcal{S}_{\text{class},k}, \mathcal{S}_{\text{inst},k}, p_k)$, where the first two components collect the current scaffolds defined in Section 3.1, and $p_k[C]$ specifies the injection probability for class C. The teacher accesses recent experience $\mathcal{L}_k$ and memory $\mathcal{M}_k$ of prior diagnoses and decisions, and proposes changes through a structured editing interface.

Evidence gathering. The training experience in Figure 1(b) records tasks, classes, trajectories, rewards, and scaffold conditions. To investigate within its context limit, the teacher uses three tools: Q_{stat} returns class-level success rates and counts of tasks without successful rollouts; Q_{traj} retrieves trajectories filtered by class and success rate; and Q_{group} returns all G attempts on one task. Starting from summaries and representative rollouts, the teacher chooses further queries within a budget, using each response to guide its next inquiry and memory to account for earlier interventions.

Scaffold proposals. The scaffold-editing block in Figure 1(b) translates the diagnosis into a proposal $\Delta S_k = (\Delta\mathcal{S}_{\text{class},k}, \Delta\mathcal{S}_{\text{inst},k}, \Delta p_k)$ to add, revise, or remove class guidance, modify instance hints, and adjust injection within $0 \le p_k[C] \le p_{\max} < 1$. Empty edits are allowed. These controls address recurring errors through shared strategies, isolated obstacles through instance hints, and diminishing benefit through reduced injection, as warranted by the evidence.

Validation and selection. The held-out validation block in Figure 1(b) informs the teacher's choice of class-level edits for deployment. A revision that addresses observed failures may still be ineffective on other tasks in the same class. We test proposals on held-out tasks at the fixed checkpoint π_{θ_k} after the current training block, isolating the effect of scaffold changes.

Before training, we reserve held-out tasks $\mathcal{V}_C$ from training data for each class. For an edited class C, we compare current guidance $A_C^{\text{old}} = \mathcal{S}_{\text{class},k}[C]$, proposed guidance A_C^{new}, and no scaffold ($\varnothing$) on the same tasks with matched seeds and evaluation settings. Using n rollouts per task, we estimate reward as

$$\widehat{r}_{k,C}(A) = \frac{1}{n|\mathcal{V}_C|} \sum_{x\in\mathcal{V}_C} \sum_{i=1}^{n} R(x, \tau_{x,i}^{A}), \quad \tau_{x,i}^{A} \sim \pi_{\theta_k}(\cdot \mid \text{augment}(x, A)). \tag{2}$$

Under binary rewards, this is the observed success rate. Comparing new with old guidance tests whether the edit improves existing support; comparing either with no scaffold tests whether that support benefits the current student.

The teacher combines these comparisons with its diagnosis to retain, adopt, revise, remove, or partially accept the proposed edits. Immediate reward gains do not guarantee better subsequent learning; later scaffold-free evaluation assesses improvement in the student's own capabilities. Instance-level edits instead derive from verified privileged information and take effect directly, providing task-specific support without held-out comparative evaluation.

The final edits $\widehat{\Delta S}_k$ yield $S_{k+1} = \text{Apply}(S_k, \widehat{\Delta S}_k)$, where Apply executes text edits and probability changes. The diagnosis and decision enter memory; the next training block uses S_{k+1} and supplies evidence for the next investigation. Appendix B illustrates this loop.

3.3 Scaffold-Augmented Student Training

As shown in Figure 1(a), the accepted scaffolds augment student prompts and remain fixed for each block of K student updates. For task x in class $C(x)$, draw $z_x \sim \text{Bernoulli}(p_k[C(x)])$ for the rollout group. Let $\mathcal{S}^{(1)}_{\text{class},k}[C] = \mathcal{S}_{\text{class},k}[C]$ and $\mathcal{S}^{(0)}_{\text{class},k}[C] = \varnothing$. The augmented prompt is

$$\widetilde{x}_k = \text{augment}\big(x, \mathcal{S}^{(z_x)}_{\text{class},k}[C(x)], \mathcal{S}_{\text{inst},k}[x]\big), \tag{3}$$

with absent instance hints represented by $\varnothing$. All G rollouts are sampled independently from $\pi_\theta(\cdot \mid \widetilde{x}_k)$, so reward comparisons share the same support. The RL update uses the unchanged task reward and the same augmented conditioning for policy likelihoods, preserving the underlying on-policy procedure. Limiting the injection probability to $p_{\max} < 1$ gives each rollout group at least a $1 - p_{\max}$ probability of receiving no class-level guidance, following privileged information dropout (Kamienny et al., 2020). Instance hints may still be present when $z_x = 0$; scaffold-free evaluation removes both forms of support. Trajectories, rewards, and injection conditions are recorded for the next teacher investigation.

AutoScaffold thus makes scaffold design an ongoing part of RL training: the teacher chooses what evidence to inspect, what support to construct, and which class-level edits to deploy, using the student's response to guide subsequent decisions. We next evaluate whether this shared decision procedure improves scaffold-free performance across task domains.

4 Experiments

4.1 Experimental Setup

Domains and tasks. We evaluate four domains: **ALFWorld**, with six household task types (Pick, Look, Clean, Heat, Cool, Pick2) requiring planning and object interaction (Shridhar et al., 2021);

search-based QA, covering single-hop (NQ, TriviaQA, PopQA) and multi-hop questions (HotpotQA, 2Wiki, MuSiQue, Bamboogle) (Kwiatkowski et al., 2019; Joshi et al., 2017; Mallen et al., 2023; Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Press et al., 2023); **KernelBench**, with GPU kernel tasks ranging from single operators to full networks (Ouyang et al., 2025); and **math reasoning**, covering eight competition-level benchmarks (Mathematical Association of America, 2024; 2025; Harvard-MIT Mathematics Tournament, 2025; Brown University Mathematical Olympiad, 2025; Mathematical Association of America, 2023; Lightman et al., 2023; Lewkowycz et al., 2022; He et al., 2024). Appendix C provides details.

Baselines. We compare **(i) training-free methods**: ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023); **(ii) RL without scaffolds**: PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024), and GiGPO (Feng et al., 2025) for ALFWorld, Search-R1 (Jin et al., 2025), ZeroSearch (Sun et al., 2026), StepSearch (Wang et al., 2025), and GiGPO for QA, and StitchCUDA (Li et al., 2026b) for kernels; and **(iii) RL with scaffolds**: SkillRL (Xia et al., 2026), Skill0 (Lu et al., 2026), and SkillC (Lin et al., 2026) for agentic tasks, and LUFFY (Yan et al., 2025), Scaf-GRPO (Zhang et al., 2026), QuestA (Li et al., 2026a), and SAGE (Liao et al., 2026) for math.

Implementation. Across domains, the teacher is GPT-5.5 with default reasoning effort, scaffold updates occur every $K = 10$ steps, and $p_{\max} = 0.5$. We follow GiGPO's training settings for ALFWorld and QA, using Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct, respectively. KernelBench uses StitchCUDA's rubric-based RL with a Dr. Kernel cold-start student and 600 training tasks. Math follows SAGE (Qwen3-4B-Instruct with GRPO) and QuestA (Nemotron-1.5B with DAPO), using their official training datasets. Experiments use 4–8 H200 GPUs; **All evaluations remove scaffolds to measure the student's own capability.** Our results are the mean of 3 independent runs.

4.2 Main Results

ALFWorld. AutoScaffold reaches 93.9%, improving over GiGPO by 7.2 points and leading five of six task categories (Table 1). Gains are largest on Look (+19.7), Cool (+16.2), and Pick2 (+15.3), where successful exploration is difficult. The 1.5B student also surpasses SkillRL (89.9%, evaluated with scaffolds) and SkillC (90.6%), both using Qwen2.5-7B-Instruct.

Table 1: Success rates (%) on ALFWorld with Qwen2.5-1.5B-Instruct. Prompting and RL baseline results are from Feng et al. (2025); ours are averaged over 3 seeds. Gray rows are from Lin et al. (2026) and use the larger Qwen2.5-7B-Instruct (reference only). **Bold**: best among 1.5B models.

Method	Pick	Look	Clean	Heat	Cool	Pick2	Overall
Qwen2.5-1.5B-Instruct	5.9	5.5	3.3	9.7	4.2	0.0	4.1
Training-free methods							
ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8
Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8
RL baselines without scaffold							
PPO	64.8	40.5	57.1	60.6	46.4	47.4	54.4
RLOO	88.3	52.8	71.0	62.8	66.4	56.9	69.7
GRPO	85.3	53.7	84.5	78.2	59.7	53.5	72.8
GiGPO	94.4	67.5	94.8	**94.4**	79.8	76.4	86.7
RL baselines with scaffold (Qwen2.5-7B-Instruct, reference only)							
SkillRL	97.9	71.4	90.0	90.0	95.5	87.5	89.9
Skill0	100.0	84.6	91.1	84.2	88.9	55.5	85.9
SkillC	88.5	78.6	91.2	94.7	100.0	88.9	90.6
AutoScaffold (ours)	**100.0**	**87.2**	**95.1**	93.8	**96.0**	**91.7**	**93.9**

Search-based QA. AutoScaffold achieves 45.9% average exact match, 3.8 points above GiGPO, and leads six of seven benchmarks (Table 2). At 3B, it exceeds Skill0 and SkillRL by 5.1 and 7.0 points. It also surpasses 7B Skill0 (44.4%) and approaches 7B SkillRL (47.1%). Training uses only NQ and HotpotQA; gains on unseen datasets indicate broader transfer.

Mathematical reasoning. With Qwen3-4B-Instruct, AutoScaffold matches SAGE at 70.0% average accuracy (+4.2 over the base model) and exceeds GRPO by 1.3 points, while LUFFY reduces accuracy by 5.2 points (Table 3). AutoScaffold attains SAGE's performance without a manually designed hinting strategy. With Nemotron-1.5B, it reaches 62.4%, exceeding DAPO by 2.0 points,

Table 2: Exact match (%) on search-based QA with Qwen2.5-3B-Instruct, trained on NQ and HotpotQA (†: in-domain; ⋆: out-of-domain). RL baseline results are from Feng et al. (2025); SkillRL and Skill0 are from their original papers (Xia et al., 2026; Lu et al., 2026). Gray rows use the larger Qwen2.5-7B-Instruct (reference only). **Bold**: best among 3B models.

Method	**Single-Hop QA**			**Multi-Hop QA**				**Avg.**
	NQ†	TriviaQA⋆	PopQA⋆	HotpotQA†	2Wiki⋆	MuSiQue⋆	Bamboogle⋆	
RL baselines without scaffold								
R1-Instruct	27.0	53.7	19.9	23.7	29.2	7.2	29.3	27.1
Search-R1	34.1	54.5	37.8	32.4	31.9	10.3	26.4	32.5
ZeroSearch	41.4	57.4	44.8	27.4	30.0	9.8	11.1	31.7
StepSearch	–	–	–	34.5	32.0	**17.4**	34.4	–
GiGPO	42.0	59.5	42.4	36.9	37.0	12.6	64.1	42.1
RL baselines with scaffold								
SkillRL	38.6	57.6	40.3	33.6	31.1	13.3	58.1	38.9
Skill0	39.8	57.5	42.3	35.1	33.7	13.3	63.7	40.8
RL baselines with scaffold (Qwen2.5-7B-Instruct, reference only)								
SkillRL	45.9	63.3	45.9	43.2	40.3	20.2	73.8	47.1
Skill0	42.7	61.1	45.3	40.0	38.3	16.4	66.9	44.4
AUTOSCAFFOLD (ours)	**45.7**	**62.8**	**48.7**	**40.1**	**39.3**	16.5	**68.1**	**45.9**

QuestA by 1.3 points, and DeepSeek-R1-Distill-32B (58.1%; Table 4). It surpasses QuestA in 350 versus 1,100 steps, roughly one-third of the training.

Table 3: Accuracy (%) on math benchmarks with Qwen3-4B-Instruct. RL baseline results are from Liao et al. (2026). Avg.: mean over the six benchmarks. **Bold** / underline: best / second best.

Method	AIME24	AIME25	AMC23	MATH-500	Minerva	Olympiad	Avg.
Qwen3-4B-Instruct	52.1	43.1	92.2	93.6	46.1	67.7	65.8
RL baselines without scaffold							
SFT	14.4	22.1	55.2	78.9	38.1	40.2	41.5
GRPO	55.8	45.0	**95.0**	**96.0**	**50.1**	70.4	68.7
RL baselines with scaffold							
LUFFY	42.3	36.0	86.4	91.1	48.2	59.4	60.6
Scaf-GRPO	**59.8**	45.2	92.2	95.1	48.9	69.8	68.5
SAGE-LIGHT	59.2	47.1	92.2	95.1	49.5	70.5	68.9
SAGE	58.1	**52.1**	94.2	95.4	49.2	**71.2**	**70.0**
AUTOSCAFFOLD (ours)	59.0	51.3	93.4	95.6	49.5	71.1	**70.0**

Table 4: Accuracy (%) on math benchmarks with Nemotron-1.5B. DAPO and QuestA results are from Li et al. (2026a). Gray: larger models (reference only). **Bold**: best among small models.

Method	AIME24	AIME25	HMMT Feb 25	OlympiadBench	BRUMO25	Avg.
Training-free base models						
DeepSeek-R1-Distill-1.5B	28.7	22.3	12.0	52.4	31.8	29.4
Qwen3-1.7B	48.3	36.8	22.2	56.1	44.1	41.5
Nemotron-1.5B	61.8	49.5	31.6	64.6	58.2	53.1
Larger models (reference only)						
DeepSeek-R1-Distill-32B	72.6	51.8	33.0	65.0	68.0	58.1
Qwen3-8B	76.0	67.3	44.8	68.6	68.3	65.0
RL baselines without scaffold						
DAPO (2K steps)	69.5	59.8	38.9	68.1	66.0	60.4
RL baselines with scaffold						
QuestA (1.1K steps)	69.3	**60.0**	37.9	69.7	68.3	61.1
AUTOSCAFFOLD (ours)	**72.6**	58.1	**42.1**	**70.2**	**69.2**	**62.4**

GPU kernel generation. AutoScaffold improves all six KernelBench metrics over StitchCUDA (Table 5). Correctness rises from 59% to 63% on Level 1 and 50% to 54% on Level 2; Level 2 Fast_1 rises from 20% to 25%. It also improves correctness and Fast_1 on the difficult full-network tasks in Level 3.

Table 5: Pass@3 of Correctness and Fast_1 on the KernelBench test set. Fast_1: the generated kernel is faster than the PyTorch reference (Ouyang et al., 2025). **Bold** / underline: best / second best.

Method	Level 1		Level 2		Level 3	
	Correctness	Fast_1	Correctness	Fast_1	Correctness	Fast_1
Qwen3-8B-Cold-Start	51	14	43	16	<u>32</u>	8
StitchCUDA	<u>59</u>	<u>17</u>	<u>50</u>	<u>20</u>	<u>32</u>	<u>13</u>
AUTOSCAFFOLD (ours)	**63**	**19**	**54**	**25**	**33**	**14**

Across these four domains, the same framework supports four RL algorithms (GiGPO, rubric-based GRPO, GRPO, and DAPO) and five students without a manual curriculum.

4.3 ABLATION STUDY

Table 6 evaluates AutoScaffold's design choices on ALFWorld with Qwen2.5-1.5B-Instruct.

Table 6: Ablations on ALFWorld (Qwen2.5-1.5B-Instruct), success rate (%) evaluated without scaffold. Static: fixed Skill0 scaffold. $\hat{\Delta}$: gap to full AUTOSCAFFOLD.

	AUTOSCAFFOLD	GiGPO	*Teacher decision*		*Scaffold adaptation & injection*			
			w/o query tools	w/o validation	Static	K=5	K=20	$p_{\max}$=1
Success	**93.9**	86.7	85.3	79.8	78.6	90.1	90.4	86.0
$\hat{\Delta}$	–	−7.2	−8.6	−14.1	−15.3	−3.8	−3.5	−7.9

Query tools. Replacing interactive queries with a fixed sample of failed trajectories and a single teacher response, following Xia et al. (2026), reduces success to 85.3%. The limited evidence restricts diagnosis, supporting the value of interactive investigation.

Scaffold validation. Deploying every proposal without held-out validation reduces success to 79.8%. Harmful edits can affect later updates, motivating validation before use.

Static scaffolds. A fixed class-level scaffold from Skill0's published ALFWorld skill bank (Lu et al., 2026) trails AutoScaffold by 15.3 points and also underperforms the RL baseline. Fixed guidance can become redundant as the student learns and cannot address emerging failures, supporting continual scaffold revision.

Update interval K. Intervals of 5 and 20 steps yield 90.1% and 90.4%, respectively. Frequent changes may destabilize training, while infrequent changes may lag the student's needs. The default $K = 10$ balances responsiveness and diagnostic reliability, using half as many teacher interventions as $K = 5$. All three settings outperform the RL baseline.

Injection probability. Injecting scaffolds into every prompt ($p_{\max} = 1$) reduces success to 86.0%, 7.9 points below the default and slightly below GiGPO (86.7%). Always-present guidance creates a mismatch with scaffold-free evaluation and may encourage dependence. Limiting injection lets the student also practice without class-level guidance.

4.4 ANALYSIS

Dynamic adaptation. At representative scaffold updates on ALFWorld, we branch from the current student into two runs: one retains the old scaffold and one uses vanilla RL without scaffolds. After 20 steps, AutoScaffold outperforms both at every branching point (Figure 2). The old-scaffold comparison supports revising guidance as the student evolves; the vanilla-RL comparison shows that support remains useful at these checkpoints.

Teacher behavior in AutoScaffold. Figure 3 shows how the teacher's scaffold changes as the student improves on ALFWorld. Early in training (Stage 1), when most task classes rarely succeed, the teacher writes broad workflows, including general command rules and step-by-step procedures for two-object, heating, and cooling tasks. As the student begins to solve these classes (Stage 2), the teacher removes the generic class workflows and keeps only a narrow fix for a remaining heating error. In the final stage, it adds guidance for specific recurring failures, such as using the exact lamp name in look tasks and not picking up the first object again in two-object tasks. It then leaves the

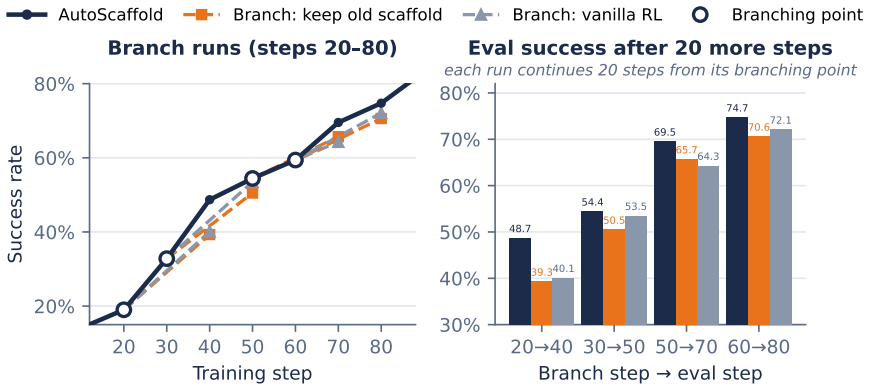


Figure 2: From each branching point, we train 20 more steps either with the old scaffold fixed (orange) or with vanilla RL (gray). AUTOSCAFFOLD outperforms both at every branching point.

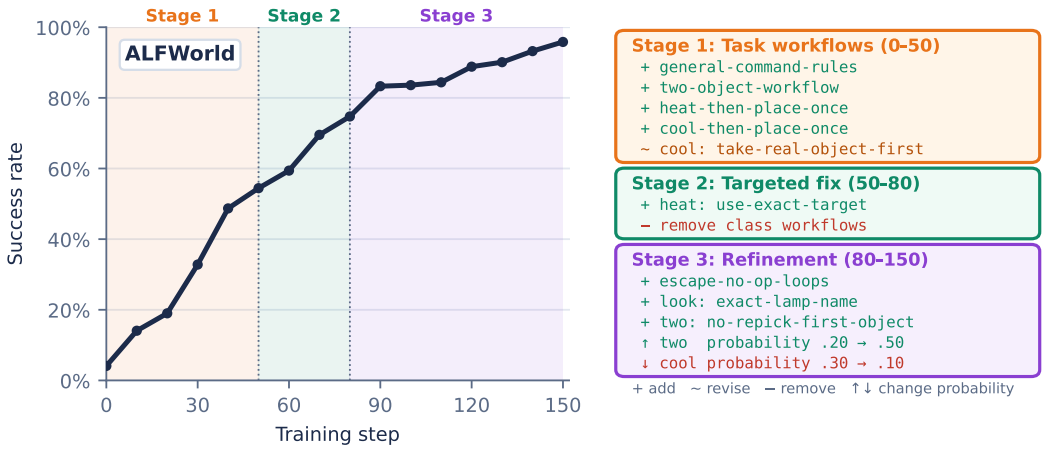


Figure 3: Scaffold is automatically optimized during training on ALFWorld. Left: success rate over training, shaded by stage. Right: scaffold edits made by the teacher in each stage (+ add, ∼ revise, − remove, ↑↓ change injection probability).

text unchanged and adjusts only injection probabilities, raising them for the still-difficult two-object class and lowering them for classes the student has largely mastered. Scaffold thus shifts from broad procedures toward targeted guidance, and its use is reduced where it is no longer needed. These adjustments are driven by the training evidence the teacher inspects rather than by a predefined schedule; Appendix B walks through representative decisions.

5 CONCLUSION

We presented AUTOSCAFFOLD, an AI-for-AI framework in which a frozen teacher LLM designs and revises training-time scaffold for a student during RL post-training. The teacher investigates the student's training experience, proposes class-level strategies and instance-level hints, adjusts how often they are used, and validates class-level edits before deployment. With a shared procedure and no domain-specific scaffold design, AUTOSCAFFOLD improves over the corresponding RL baselines in four domains under scaffold-free evaluation, and matches or outperforms methods that rely on human-designed scaffolds where such baselines exist. Ablations show that interactive evidence access, validation, and continual revision each contribute to these gains, and analysis shows the teacher moving from broad procedures to targeted guidance and reducing support as the student improves. Future work includes reducing the cost of teacher interventions and understanding when gains under scaffold transfer to unscaffolded behavior.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing. We have **not** used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We have reviewed all AI-assisted work (which only includes polishing the writing). We take responsibility for the final content of this work, including text or artifacts produced with the aid of generative AI.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL https://arxiv.org/abs/2402.14740.

Brown University Mathematical Olympiad. BRUMO 2025. Problems available at https://huggingface.co/datasets/MathArena/brumo_2025, 2025. Accessed September 2026.

Chao Chen, Chengzu Li, Zhiwei Li, Yinhong Liu, and Zhijiang Guo. From trainee to trainer: Llm-designed training environment for rl with multi-agent reasoning, 2026. URL https://arxiv.org/abs/2606.17682.

Haoyang Fang, Wei Zhu, Boran Han, Alex Zhang, Zhenyu Pan, Shuo Yang, Shuai Zhang, Jiading Gai, Peng Tang, Cuixiong Hu, Xuan Zhu, Huzefa Rangwala, George Karypis, and Bernie Wang. Llmzero: Discovering adaptive training strategies for rl post-training via llm agents, 2026. URL https://arxiv.org/abs/2606.18388.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, 2025.

Harvard-MIT Mathematics Tournament. HMMT February 2025. https://www.hmmt.org/; problems available at https://huggingface.co/datasets/MathArena/hmmt_feb_2025, 2025. Accessed September 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020. URL https://arxiv.org/abs/2011.01060.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

Pierre-Alexandre Kamienny, Kai Arulkumaran, Feryal Behbahani, Wendelin Boehmer, and Shimon Whiteson. Privileged information dropout in reinforcement learning, 2020. URL https://arxiv.org/abs/2005.09220.

Andrej Karpathy. autoresearch: AI agents running research on single-GPU nanochat training automatically. https://github.com/karpathy/autoresearch, 2026. GitHub repository.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation, 2026a. URL https://arxiv.org/abs/2507.13266.

Shiyang Li, Zijian Zhang, Winson Chen, Yuebo Luo, Mingyi Hong, and Caiwen Ding. Stitchcuda: An automated multi-agents end-to-end gpu programing framework with rubric-based agentic reinforcement learning, 2026b. URL https://arxiv.org/abs/2603.02637.

Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning, 2026. URL https://arxiv.org/abs/2602.03143.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. Skillc: Learning autonomous skill internalization in llm agents via contrastive credit assignment, 2026. URL https://arxiv.org/abs/2605.27899.

Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization, 2025. URL https://arxiv.org/abs/2402.01881.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization, 2026. URL https://arxiv.org/abs/2604.02268.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models, 2024. URL https://arxiv.org/abs/2310.12931.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories, 2023. URL https://arxiv.org/abs/2212.10511.

Mathematical Association of America. American Mathematics Competitions (AMC) 12, 2023. Problems available at https://huggingface.co/datasets/AI-MO/aimo-validation-amc, 2023. Accessed September 2026.

Mathematical Association of America. American Invitational Mathematics Examination (AIME) 2024. Problems available at https://huggingface.co/datasets/HuggingFaceH4/aime_2024, 2024. Accessed September 2026.

Mathematical Association of America. American Invitational Mathematics Examination (AIME) 2025. Problems available at https://huggingface.co/datasets/math-ai/aime25, 2025. Accessed September 2026.

Anne Ouyang, Simon Guo, Simran Arora, Alex L. Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. Kernelbench: Can llms write efficient gpu kernels?, 2025. URL https://arxiv.org/abs/2502.10517.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023. URL https://arxiv.org/abs/2210.03350.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. Posttrainbench: Can llm agents automate llm post-training?, 2026. URL https://arxiv.org/abs/2603.08640.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL https://arxiv.org/abs/2010.03768.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching, 2026. URL https://arxiv.org/abs/2505.04588.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop questions via single-hop question composition, 2022. URL https://arxiv.org/abs/2108.00573.

Lev S Vygotsky. Mind in society: The development of higher psychological processes, 1978.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization, 2025. URL https://arxiv.org/abs/2505.15107.

David Wood, Jerome S Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100, 1976.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance, 2025. URL https://arxiv.org/abs/2504.14945.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL https://arxiv.org/abs/1809.09600.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning, 2026. URL https://arxiv.org/abs/2510.19807.

Zijian Zhang, Rong Wang, Shiyang Li, Yuebo Luo, Mingyi Hong, and Caiwen Ding. Cudaforge: An agent framework with hardware feedback for cuda kernel optimization, 2025. URL https://arxiv.org/abs/2511.01884.

A Theoretical Analysis

We analyze how scaffold injection changes GRPO reward contrasts to motivate AutoScaffold. Following Liao et al. (2026), we consider binary rewards $R(x, \tau) \in \{0, 1\}$.

Setup. For a prompt x, GRPO samples a group of G rollouts $\{\tau_i\}_{i=1}^{G}$ from the current policy π_θ and computes standardized advantages

$$A_i = \frac{R_i - \bar{R}}{s + \epsilon}, \quad \bar{R} = \frac{1}{G}\sum_{i=1}^{G} R_i, \quad s^2 = \frac{1}{G}\sum_{i=1}^{G}(R_i - \bar{R})^2, \tag{4}$$

where $\epsilon \geq 0$ is a numerical stabilizer. The policy gradient is weighted by these advantages: when $A_i = 0$ for all i, the group contributes no gradient regardless of the rollout content.

Degenerate groups and the learning cliff. Under binary rewards, each rollout independently succeeds with probability $p = p_\theta(x)$. A group is *degenerate* if all rollouts share the same outcome ($\bar{R} \in \{0, 1\}$), in which case $s = 0$ and $A_i = 0$ for all i. This underlies the *learning cliff* (Zhang et al., 2026; Li et al., 2026a): when p is very small, nearly all groups are all-failure and the task contributes almost no learning signal. The probability of obtaining a non-degenerate group therefore determines whether a prompt contributes to learning at all.

Proposition 1 (Non-degenerate group probability (Liao et al., 2026))**.** *For a prompt with success probability $p \in [0, 1]$ and group size $G \geq 2$,*

$$u(p) := \Pr[s > 0] = 1 - (1 - p)^G - p^G. \tag{5}$$

u is strictly concave on $[0, 1]$ and symmetric about $p = \frac{1}{2}$, its unique maximizer, with $u(\frac{1}{2}) = 1 - 2^{1-G}$.

Proof. A group is degenerate exactly when all G rollouts fail or all succeed, which occurs with probability $(1 - p)^G + p^G$. Moreover, $u''(p) = -G(G - 1)\left[p^{G-2} + (1 - p)^{G-2}\right] < 0$ for all $p \in [0, 1]$, so u is strictly concave. Together with the symmetry $u(p) = u(1 - p)$, this implies that u has a unique maximum at $p = \frac{1}{2}$. □

When $p \ll 1$, which is the common regime for hard tasks, $u(p) \approx Gp$: the group is non-degenerate only when at least one rollout succeeds, an event that becomes rare as $p \to 0$. Conversely, when $p \approx 1$, $u(p) \approx G(1 - p)$: almost all groups are all-success, again yielding no reward contrast.

Proposition 2 (Expected within-group reward variance (Liao et al., 2026))**.** *Define the within-group reward variance $V = \frac{1}{G}\sum_{i=1}^{G}(R_i - \bar{R})^2$. Under binary rewards with success probability p,*

$$\mathbb{E}[V \mid x] = \frac{G - 1}{G}\, p(1 - p), \tag{6}$$

which is strictly concave in p and uniquely maximized at $p = \frac{1}{2}$.

Proof. For binary rewards, $V = \bar{R}(1-\bar{R})$. Since $G\bar{R} \sim \text{Binomial}(G, p)$, we have $\mathbb{E}[\bar{R}] = p$ and $\mathbb{E}[\bar{R}^2] = p^2 + p(1-p)/G$. Hence $\mathbb{E}[V \mid x] = p - p^2 - p(1-p)/G = \frac{G-1}{G}\, p(1-p)$. □

Propositions 1 and 2 show that $p = \frac{1}{2}$ maximizes both the probability of a non-degenerate group and its expected reward variance. The two quantities play different roles under GRPO. Because advantages are standardized, any non-degenerate group yields advantages of roughly unit scale ($\frac{1}{G}\sum_i A_i^2 = s^2/(s+\epsilon)^2 \approx 1$), so the primary quantity is $u(p)$, i.e., whether a prompt contributes at all. The reward variance measures signal strength when advantages are not normalized by s, and we use it below as a tractable proxy for class-level analysis. Both quantities are symmetric and concave, vanish at $p \in \{0, 1\}$, and peak at $p = \frac{1}{2}$, supporting the same qualitative conclusions.

Implications for scaffold design. Let $p_\theta(x)$ denote the student's success probability without scaffold and $p_\theta(x, S)$ the probability under scaffold S. Scaffold that raises $p_\theta(x, S)$ from near zero toward $\frac{1}{2}$ increases both the non-degenerate group probability and the expected reward variance, enabling gradient updates on tasks that would otherwise contribute almost nothing. Scaffold that is too strong instead overshoots: if $p_\theta(x, S) \approx 1$, groups become degenerate again (all-success), and the learning signal collapses despite a high success rate. Effective scaffold should therefore make success reachable while leaving meaningful work for the student, for example by providing strategies or partial hints rather than complete solutions.

Remark 1 (Scaffold should adapt to the student). The per-step signal is maximized when $p_\theta(x, S) \approx \frac{1}{2}$. Since p_θ changes as the student trains, a scaffold that is well calibrated at step t will generally overshoot at a later step $t' > t$ as the student improves, pushing $p_\theta(x, S)$ toward 1. Hence no fixed scaffold keeps every task in the high-signal regime throughout training. This statement concerns the per-step signal and does not by itself characterize the long-term learning trajectory.

Class-level scaffold and aggregate learning signal. The preceding analysis concerns a single prompt with success probability p. Per-task success rates can only be estimated from G rollouts and are therefore noisy, so we analyze scaffold at the class level, in terms of the class success rates without scaffold (s_C) and with scaffold (σ_C).

Consider a task x drawn uniformly from class C. Its success probability p_x varies across tasks, reflecting the heterogeneous difficulty of the class. We write $p_x = s_C + \epsilon_x$, where ϵ_x is a zero-mean perturbation with $\text{Var}(\epsilon_x) = v_C$. The class-average within-group reward variance is

$$\bar{V}_C := \frac{G-1}{G}\, \mathbb{E}_x\big[p_x(1-p_x)\big] = \frac{G-1}{G}\big[s_C(1-s_C) - v_C\big], \tag{7}$$

where the cross term vanishes because $\mathbb{E}[\epsilon_x] = 0$. The term $s_C(1-s_C)$ corresponds to a homogeneous class with uniform difficulty s_C, and the subtracted v_C is the reduction due to task heterogeneity (consistent with Jensen's inequality, since $p(1-p)$ is concave).

Assumption 1 (Uniform shift)**.** *Class-level scaffold shifts every task equally: $p_x^S = p_x + (\sigma_C - s_C) \in [0, 1]$ for $x \in C$, preserving variance v_C.*

Assumption 1 is an idealization. In practice, a shared strategy may help some tasks more than others and may leave the hardest tasks at zero success; we treat such tasks separately below.

Proposition 3 (Class-level scaffold benefit)**.** *For fixed v_C and $s_C \in [0, \frac{1}{2})$, $\bar{V}_C$ increases with s_C:*

$$\frac{\partial \bar{V}_C}{\partial s_C} = \frac{G-1}{G}(1 - 2s_C) > 0. \tag{8}$$

Under Assumption 1, if class-level scaffold raises the class success rate from s_C to σ_C, the change in class-average reward variance is

$$\bar{V}_C^S - \bar{V}_C = \frac{G-1}{G}\,(\sigma_C - s_C)(1 - \sigma_C - s_C), \tag{9}$$

which is positive if and only if $s_C < \sigma_C < 1 - s_C$.

Proof. The derivative follows from equation 7. Under Assumption 1, scaffolding gives mean σ_C and unchanged variance v_C, so $\bar{V}_C^S = \frac{G-1}{G}\big[\sigma_C(1-\sigma_C) - v_C\big]$. Subtracting gives

$$\bar{V}_C^S - \bar{V}_C = \frac{G-1}{G}\big[\sigma_C(1-\sigma_C) - s_C(1-s_C)\big] = \frac{G-1}{G}\,(\sigma_C - s_C)(1 - \sigma_C - s_C).$$

The first factor is positive if and only if $\sigma_C > s_C$, and the second if and only if $\sigma_C < 1 - s_C$. □

The improvement depends only on the two class-level rates s_C and σ_C. It is proportional to both the scaffold effect $(\sigma_C - s_C)$ and the factor $(1 - \sigma_C - s_C)$, and for a given s_C it is largest at $\sigma_C = \frac{1}{2}$. If scaffold is so strong that $\sigma_C \geq 1 - s_C$, the class-level signal no longer increases and can even decrease, although the success rate is higher. This formalizes why class-level scaffold should raise success rates toward, but not far beyond, $\frac{1}{2}$.

Instance-level scaffold and zero-success tasks. Class-level scaffold mainly benefits tasks that the student can already partially solve ($p_x > 0$), lifting their success rates toward the high-signal regime. For tasks where the student's success probability is zero, a shared strategy is often insufficient, and targeted support such as a partial solution or an intermediate result is needed.

Let $\mathcal{Z}_C = \{x \in C : p_\theta(x) = 0\}$ denote the set of zero-success tasks in class C, with $\alpha_C = |\mathcal{Z}_C|/|C|$. These tasks produce only all-failure groups, contributing no reward-relative gradient or within-group reward variance:

$$\bar{V}_C = \frac{G-1}{G}\,(1-\alpha_C)\,\mathbb{E}_{x\notin\mathcal{Z}_C}\big[p_x(1-p_x)\big]. \tag{10}$$

Proposition 4 (Instance-level scaffold resolves zero-success tasks)**.** *Suppose instance-level scaffold $S_{\text{inst}}[x]$ is applied to each zero-success task $x \in \mathcal{Z}_C$, raising its success probability from 0 to $p_x^S \in (0,1)$, while the remaining tasks are unchanged. The class-average variance increases by*

$$\bar{V}_C^S - \bar{V}_C = \frac{G-1}{G}\,\alpha_C\,\mathbb{E}_{x\in\mathcal{Z}_C}\big[p_x^S(1-p_x^S)\big] > 0. \tag{11}$$

Proof. Under instance-level scaffold, the class-average reward variance becomes $\bar{V}_C^S = \frac{G-1}{G}\big[(1-\alpha_C)\,\mathbb{E}_{x\notin\mathcal{Z}_C}[p_x(1-p_x)] + \alpha_C\,\mathbb{E}_{x\in\mathcal{Z}_C}[p_x^S(1-p_x^S)]\big]$. Subtracting equation 10 yields equation 11, which is positive because $p_x^S(1-p_x^S) > 0$ for $p_x^S \in (0,1)$. □

The improvement is proportional to α_C: the more zero-success tasks there are, the greater the benefit, which is particularly relevant early in training. The requirement $p_x^S < 1$ matters: a hint that reveals the complete solution makes every rollout succeed, so the group is again degenerate and contributes nothing. Propositions 3 and 4 thus formalize the complementary roles of the two scaffold levels: class-level scaffold shifts the difficulty of tasks the student can already attempt toward the high-signal regime, while instance-level scaffold expands the set of tasks that contribute to training at all. This motivates using instance-level scaffold mainly for tasks that remain at zero success under class-level scaffold, with hints that are partial rather than complete.

Scope of the analysis. The analysis above characterizes the reward contrasts available under the scaffolded prompt, i.e., the signal for updating $\pi_\theta(\cdot \mid x, S)$. It does not model how improvements under scaffold transfer to the unscaffolded policy $\pi_\theta(\cdot \mid x)$, which is what we evaluate. In particular, with injection probability q, the expected class-level signal $(1-q)\,\bar{V}_C(s_C) + q\,\bar{V}_C(\sigma_C)$ is linear in q and would favor $q = 1$ whenever scaffold helps. In practice, always-on scaffold creates a mismatch between training and evaluation and encourages reliance on privileged information (cf. the $p_{\max}$=1 ablation in Section 4.3), which motivates capping q at $p_{\max} < 1$. Characterizing transfer remains future work.

B Case Study: How the Teacher Gathers Evidence and Decides

B.1 ALFWorld

We illustrate scaffold optimization on ALFWorld using the teacher's decision log.

Gathering evidence (step 70). The teacher made five tool calls. It first called `get_stats` to obtain per-class training statistics, held-out per-class success with and without the current scaffold, and the number of all-fail groups in the current window. Based on these statistics, it then called `get_traces` to retrieve four all-fail trajectories each for the three weakest classes (Pick2, Look, and Heat). Finally, it calls the decision memory M to review its last seven decisions.

Diagnosis. From this evidence, the teacher identified a distinct failure mode for each weak class and judged which classes no longer needed support:

> *"pick_two_obj_and_place has the worst training success and 11 all-fail groups; failures often place one target object, then pick it back up or never search for/place the second object. Add a two-object hint. look_at_obj_in_light remains low with several all-fail groups; failures turn on a lamp or find the object but do not bring the object to the lamp and inspect it there. Add a concrete lamp workflow. Heat still has a useful paired gap and many bare all-fails, but injected failures show the model reheats or retrieves the object after depositing it; update the heat text to emphasize one final put and stop. Clean is high and the paired gap is small-positive, so keep but reduce p. Cool and pick_and_place need no text."*

Decision. The teacher translated the diagnosis into class-specific edits, each targeting the failure mode it had observed:

- **Pick2** (worst; 11 all-fail groups): guide finding and placing the second object; $p = 0.4$.
- **Look** (low success, several all-fail groups): add a concrete workflow, i.e., take the object, bring it to the desk lamp, `use desklamp`, then examine; $p = 0.35$.
- **Heat** (scaffold still helps, but injected rollouts reheat or retrieve the object after placing it): rewrite the existing entry to emphasize placing the object once and stopping; $p = 0.3$.
- **Clean** (high success, small scaffold benefit): retain guidance and reduce p to 0.15.
- **Cool** and **Pick** (already mastered): no text; $p = 0$.

The decision thus varies with the student's ability in each class: new, specific guidance where the student fails, a revision where existing guidance is being misused, a lower dose where the student is already strong, and no scaffold where it is not needed.

B.2 Mathematical Reasoning

In math, instance-level scaffold follows QuestA (Li et al., 2026a): the first $r\%$ of a reference solution is prepended to the problem. Besides writing class-level text, the teacher can adjust r for buckets of problems grouped by their rollout outcome (all-pass, all-fail, or mixed).

Gathering evidence (cycle 1). This is the first cycle with real training data. The teacher made four tool calls. It first called `get_stats` to obtain the outcome buckets of the current window and their hint ratios. It then called `get_traces` twice to read eight all-fail rollouts, and finally called `get_problems` to list all-pass problems with $r \in [25, 50]$.

Diagnosis. The teacher separated the problems by outcome and reasoned about each group differently:

> *"The all-fail set is only 116/1280, but it is already at the maximum hint ratio, so bucket-level raising is impossible. Traces show several all-fail rollouts contain substantial correct reasoning but lose credit through final-answer mismatch, wrong option boxing, not matching the requested quantity, or abandoning the derived conclusion. A small global final-answer audit note is likely to create successes in some zero-gradient groups without increasing hint dependence. Since 284 problems are all-pass at r=50, they should be annealed on next visit to reduce learning-to-continue-hints and test whether the skill survives with less scaffold. Mixed r=50 problems still provide gradient, so leave them unchanged for now. Keep p modest; no evidence yet justifies a large text increase."*

Decision. The teacher made adjustments in opposite directions for different groups of problems:

- **All-fail problems** (hint ratio already at its maximum; failures often due to answer-format near misses): rather than giving more of the solution, add a short class-level note with a modest $p = 0.18$:

  > *"Before boxing an answer, do a final audit: reread exactly what the problem asks for, compare it with your derived conclusion, and ensure the boxed value/option matches that quantity. If you found an option label, box the label; if asked to restore a number, box the full restored number, not just intermediate digits."*

- **All-pass problems** with $r \in [25, 50]$ (284 problems solved under the current hint): lower the hint ratio by 20 points, to test whether the student can solve them with less support and to avoid reliance on the hint.
- **Mixed-outcome problems**: leave unchanged, since they already provide learning signal.

As in ALFWorld, the decision follows the student's current ability: where more of the same support is no longer possible, the teacher targets the specific failure it observed, and where the student already succeeds, it reduces support.

C Evaluation Domains

We evaluate AutoScaffold on four domains that differ in task structure, interaction mode, and reward.

ALFWorld. ALFWorld (Shridhar et al., 2021) is a text-based household environment in which the agent completes long-horizon tasks through multi-step planning and object interaction. It defines six task types, each used as a task class for class-level scaffold: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2). We evaluate on the standard 134-game test split; an episode succeeds if the task is completed within the environment's step limit, and we report the success rate.

Search-based QA. The model answers open-domain questions by interacting with a search tool, deciding when and what to query and how to combine the retrieved passages into a final answer. We train on NQ and HotpotQA and evaluate on three single-hop datasets, NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), and four multi-hop datasets, HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). NQ and HotpotQA are in-domain; the others are out-of-domain. Exact match (EM) is used as the reward and the evaluation metric.

KernelBench. KernelBench (Ouyang et al., 2025) asks the model to write GPU kernels that reproduce a reference PyTorch implementation and run faster than it. Level 1 contains 100 relatively simple tasks involving basic operators (e.g., matrix multiplication); Level 2 contains 100 medium-difficulty tasks composed of multi-step operator combinations; Level 3 contains 50 challenging tasks involving full network architectures (e.g., AlexNet). We train on a dataset of 600 kernel generation tasks. Correctness is verified against the reference output, and we report correctness and Fast_1, the fraction of kernels that are faster than the PyTorch reference. Speed is measured on H200;

Math reasoning. The student performs multi-step reasoning and outputs a final answer, which is checked against the ground truth. We use eight competition-level benchmarks: the recent contests AIME 24 (Mathematical Association of America, 2024), AIME 25 (Mathematical Association of America, 2025), HMMT Feb 25 (Harvard-MIT Mathematics Tournament, 2025), and BRUMO 25 (Brown University Mathematical Olympiad, 2025), together with AMC 23 (Mathematical Association of America, 2023), MATH-500 (Lightman et al., 2023), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). Following the corresponding baselines, the Qwen3-4B-Instruct setting (following SAGE) reports AIME 24/25, AMC 23, MATH-500, Minerva Math, and OlympiadBench (Table 3), and the Nemotron-1.5B setting (following QuestA) reports AIME 24/25, HMMT Feb 25, OlympiadBench, and BRUMO 25 (Table 4). We use the evaluation setting in the paper of baseline.

D Teacher Cost

Cost per scaffold update. Each scaffold update consists of (i) investigation, in which the teacher queries Q_{stat}, Q_{traj}, and Q_{group} and reads its memory; (ii) proposal; and (iii) held-out validation of class-level edits, which runs student rollouts under three conditions. Teacher calls run on an external API; validation rollouts run on the training GPUs. Table 7 reports the cost per domain. Validation dominates the time per update, while the API cost of the teacher is a few dollars per run.

Table 7: Teacher cost per domain. Per-update values are means over all scaffold updates in a run.

	ALFWorld	Search QA	Math	KernelBench
Teacher wall-clock per update (min)	4	3	5	4
Validation wall-clock per update (min)	11	13	21	16
Total teacher API cost per run (USD)	2.35	3.13	5.57	2.76