LEARNING TO COACH FOR EXPERIENTIAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models can learn from experience, but raw solution trajectories are often too long and noisy to provide effective guidance. In this work, we propose **Learning to Coach (L2C)**, a framework that trains a dedicated LLM-as-a-Coach to extract actionable experiential knowledge from an actor model's previous trajectory. The actor remains frozen, while the LLM-as-a-Coach is trained to maximize a reward given by the correctness of the actor's guided response. We study two such rewards: a same-instance reward, which improves subsequent responses on the original problem, and a cross-instance reward, which elicits knowledge that transfers to other instances. Across mathematical reasoning and interactive text-games, L2C consistently outperforms self-refinement and an untrained LLM-as-a-Coach. Running experiential learning for more iterations further improves accuracy and uses additional inference compute more effectively than enlarging the actor's decoding budget. The trained LLM-as-a-Coach also transfers to out-of-distribution tasks and adapts its guidance to the actor it coaches.

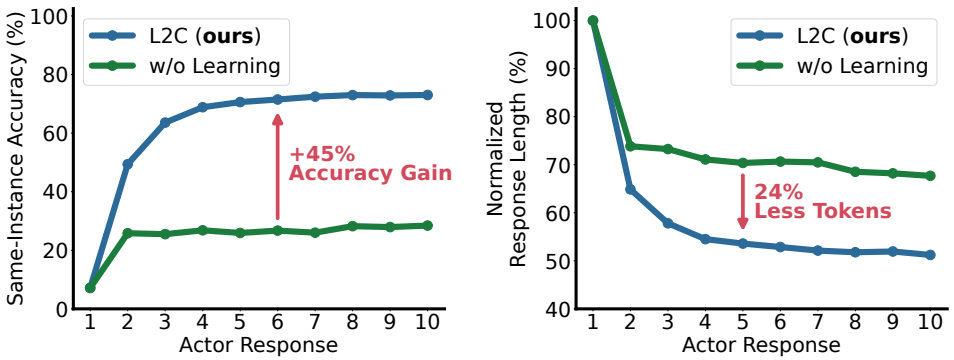


Figure 1: Through experiential learning, a trained LLM-as-a-Coach steadily improves the actor's accuracy, while an untrained coach quickly plateaus (left). Responses also become shorter (right), indicating that learned coaching enables more accurate and efficient test-time scaling.

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong capabilities in mathematical reasoning (Guo et al., 2025; Yang et al., 2025) and planning and interactive decision making (Wang et al., 2023; Park et al., 2023). Beyond solving a problem in a single pass, an LLM can often improve its answer by inspecting an earlier response, identifying errors, and trying again. Its own interaction history therefore carries valuable *experiential knowledge*: failed approaches, useful intermediate results, latent environment rules, and promising directions for subsequent responses.

However, extracting useful knowledge from a solving trajectory is itself a challenging reasoning problem. A raw trajectory can be long, redundant, or misleading, and may contain incorrect conclusions alongside useful evidence. Simply conditioning the model on its entire previous response, as in self-refinement, does not distinguish between these components (Madaan et al., 2023; Akyurek et al., 2023). As a result, the model may repeat its original mistakes, attend to irrelevant details, or even regress after an additional solve. Alternatively, one can fine-tune the solving model directly

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.

using task-level rewards, but doing so changes the model's parameters and can be expensive when the same model must retain broad, general-purpose capabilities.

This motivates a different division of labor: instead of updating the model that performs the task, we train a separate model to learn how to extract useful experience from prior trajectories, while keeping the task-performing model frozen. This separation lets the solving model retain its original parameters while delegating the interpretation of prior trajectories to a dedicated model. It also makes the extracted knowledge explicit and inspectable, rather than implicitly encoded.

We term this delegated model an *LLM-as-a-Coach*, as in (Ye et al., 2026). It reads the actor's prior trajectory and expresses, in natural language, what went wrong, which intermediate results are worth keeping, and which directions are promising for the next response. It distills the trajectory into concise, actionable experiential knowledge for the actor's subsequent response.

In this work, we propose **Learning to Coach (L2C)**, a framework that trains a dedicated LLM-as-a-Coach to extract actionable experiential knowledge from a frozen actor model's previous solving trajectory. In L2C, experiential learning proceeds as follows: the actor first makes an initial response, the LLM-as-a-Coach converts the resulting trajectory into concise experiential knowledge, and the actor makes a guided response. Rather than supervising the LLM-as-a-Coach with human-written critiques, we optimize it with reinforcement learning, using the correctness of the actor's guided response as the reward. The LLM-as-a-Coach therefore learns to produce guidance that improves the downstream behavior of the actor being coached.

We study two reward variants. The *same-instance* reward evaluates the guided response on the instance from which the experiential knowledge was extracted, encouraging the LLM-as-a-Coach to diagnose errors and preserve useful intermediate results. The *cross-instance* reward evaluates the extracted knowledge on disjoint instances, encouraging it to capture reusable task structure. L2C also supports experiential learning over multiple iterations, where the LLM-as-a-Coach repeatedly updates its experiential knowledge from the actor's latest trajectory before the actor solves again.

We evaluate L2C across two domains, mathematical reasoning and interactive text-games. Across tasks and model sizes, L2C consistently outperforms the frozen base actor, Self-Refinement, and an untrained LLM-as-a-Coach. The gains are particularly large in text-games, where interaction trajectories expose latent environment rules. Training with the cross-instance reward further produces transferable knowledge when such rules are shared across instances.

Our analysis shows that experiential learning provides an effective form of test-time scaling: performance continues to improve across iterations, while an untrained LLM-as-a-Coach quickly plateaus. On math, L2C with more iterations outperforms doubling the actor's decoding budget across all evaluated model sizes. The trained LLM-as-a-Coach also transfers to out-of-distribution evaluations, and adapts its guidance to the actor. These results show that learning to extract experiential knowledge can improve language models without updating their parameters.

2 METHOD

We propose **Learning to Coach (L2C)**, which learns from trajectories produced by a frozen actor during deployment. Once these trajectories have been collected, an LLM-as-a-Coach extracts concise experiential knowledge from the given trajectory, which is supplied back to the actor to improve its later performance. Instead of modifying the actor directly, L2C keeps the actor fixed and instead optimizes the coach to produce experiential knowledge that improves the actor's performance.

Learning to Coach Let π_{actor} be a frozen actor and π_θ the trainable LLM-as-a-Coach. Running the actor on an instance x leaves behind a trajectory (x, y), where $y \sim \pi_{\text{actor}}(\cdot \mid x)$, which may be a solution trace or a multi-turn interaction gathered during actor use. Let $\mathcal{D}$ denote the resulting distribution over collected trajectories. Given $(x, y) \sim \mathcal{D}$, the coach extracts experiential knowledge

$$e \sim \pi_\theta(\cdot \mid x, y). \tag{1}$$

To reward the experiential knowledge e, we provide it to the frozen actor on a target instance z:

$$y'_z \sim \pi_{\text{actor}}(\cdot \mid z, e), \qquad r(z, e) = \mathcal{V}(z, y'_z), \tag{2}$$

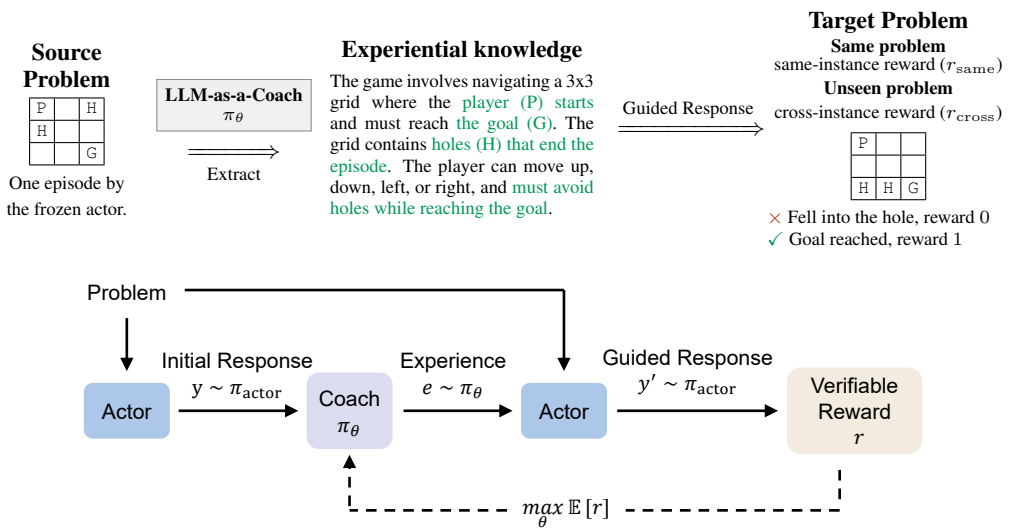


Figure 2: Overview of L2C. A frozen actor π_{actor} first responds to problems, producing trajectories; an LLM-as-a-Coach π_θ extracts experiential knowledge from each trajectory. The actor then produces a guided response conditioned on the extracted knowledge, and the verifiable correctness of that response serves as the reward. Only the coach is trained. Under the same-instance reward, the target is the source problem; under the cross-instance reward, the target is an unseen problem.

where $\mathcal{V}$ is a deterministic verifier. Thus, the coach is rewarded when its guidance leads the actor to a correct response. Accordingly, we train the coach to maximize this reward:

$$\max_\theta \quad \mathbb{E}_{(x,y)\sim\mathcal{D},\, e}\left[\, r \,\right], \tag{3}$$

where r is one of two reward variants, which differ only in which target instances the experiential knowledge is evaluated on.

The *same-instance reward* evaluates the knowledge on the instance x that y was collected on,

$$r_{\text{same}} = r(x, e), \tag{4}$$

which encourages the coach to distill instance-specific knowledge.

The *cross-instance reward* instead evaluates the knowledge on a disjoint set of target instances $\mathcal{P}$,

$$r_{\text{cross}} = \frac{1}{|\mathcal{P}|} \sum_{z\in\mathcal{P}} r(z, e), \tag{5}$$

which encourages reusable knowledge, such as shared task rules and general strategies.

For each source trajectory, we sample multiple candidate knowledge snippets, evaluate each through actor rollouts, and optimize the coach with GRPO (Shao et al., 2024).

Experiential Learning The LLM-as-a-Coach updates its experiential knowledge as new actor trajectories become available. Starting with $e^{(0)} = \emptyset$ and an unguided response $y^{(1)} \sim \pi_{\text{actor}}(\cdot \mid x)$, we iterate for $k = 1, \ldots, K-1$:

$$e^{(k)} \sim \pi_\theta(\cdot \mid x, y^{(k)}, e^{(k-1)}), \tag{6}$$

$$y^{(k+1)} \sim \pi_{\text{actor}}(\cdot \mid x, e^{(k)}). \tag{7}$$

This yields K actor responses, including the initial unguided response, and $K-1$ knowledge extractions. The updated knowledge replaces the previous version, allowing the coach to remove incorrect conclusions, retain useful discoveries, and refine its guidance over time. The $K{=}2$ setting uses one observed trajectory followed by one guided response; larger K models continued learning from accumulated deployment experience. Viewed from a meta-learning perspective (Finn et al., 2017), generating the experience from new actor trajectories forms the inner loop, while updating the coach with reinforcement learning based on the actor's downstream reward forms the outer loop. Algorithm 1 summarizes the training procedure.

Algorithm 1 Experiential Learning with L2C

Input: Collected trajectories $\mathcal{D}$; Frozen actor π_{actor}; LLM-as-a-Coach π_θ; Iterations K; Reward $r \in \{r_{\text{same}}, r_{\text{cross}}\}$
Output: Trained LLM-as-a-Coach π_θ

```
for each collected trajectory (x, y^(1)) ~ D do
    e^(0) ← ∅
    for k ← 1 to K − 1 do
        e^(k) ~ π_θ(· | x, y^(k), e^(k−1))      ▷ Extract experiential knowledge
        y^(k+1) ~ π_actor(· | x, e^(k))          ▷ Guided response
        Update π_θ to maximize E[r]              ▷ Equation (3)
    end for
end for
return π_θ
```

3 Experiments

3.1 Setup

Base Models In every setting, the actor and the LLM-as-a-Coach are initialized from the same base model. For math, we use Qwen3-1.7B, Qwen3-4B, and Qwen3-8B (Yang et al., 2025), all in thinking mode. For text-games, we use Qwen3-1.7B (thinking mode) on FrozenLake and Qwen3-4B-Instruct-2507 on Sokoban.

Datasets Our experiments cover three datasets: the math corpus DAPO-Math-17K (Yu et al., 2025a) and two interactive text-game environments, FrozenLake and Sokoban, both built on TextArena (Guertler et al., 2025). DAPO-Math-17K comprises roughly 14K English math problems with verifiable numerical answers. FrozenLake asks the agent to navigate a grid to a goal tile while steering clear of holes, whereas Sokoban requires planning a sequence of pushes that moves a box onto a target without dropping into a hole or jamming the box against a wall. Both games are presented purely as text and played out over multiple interaction turns. Full per-dataset configurations are deferred to Appendix A.2.

Training We train only the LLM-as-a-Coach using GRPO (Shao et al., 2024) with eight candidates per source trajectory and AdamW (Kingma & Ba, 2015) at a learning rate of 10^{-6}. We consider same-instance training at $K{=}2, 10$ and cross-instance training at $K{=}2$. Appendices A.1 and A.3 give the full training budgets and configurations.

Baselines All baselines share L2C's frozen actor and evaluation protocol. **(1) Base Model.** The actor responds once, without experiential knowledge. **(2) Self-Refinement.** The actor revises its answer in a second pass using its full initial trajectory, without a separate coach or extracted experiential knowledge (Madaan et al., 2023; Akyurek et al., 2023). **(3) LLM-as-a-Coach (w/o training).** The same experiential learning procedure as L2C with an untrained coach.

Evaluation Each evaluation rollout consists of an initial response, experiential knowledge extraction, and a guided response. We report two accuracies, mirroring the two rewards of Section 2.

The *same-instance accuracy* scores the guided response on the instance the experiential knowledge was extracted from (Equation (4)). At iteration k,

$$\text{Acc}_k^{\text{same}} \;=\; \mathbb{P}_{(x,y^{(1)})\sim\mathcal{D}}\Big[\, y^{(k)} \text{ is correct} \,\Big], \tag{8}$$

the fraction of instances the actor answers correctly at its k-th response.

The *cross-instance accuracy* instead applies knowledge extracted from a source instance x to the disjoint probe set $\mathcal{P}$ (Equation (5)):

$$\text{Acc}^{\text{cross}} \;=\; \mathbb{E}_{(x,y)\sim\mathcal{D},\; z\sim\mathcal{P}}\big[\, \mathcal{V}(z, y'_z) \,\big], \qquad y'_z \sim \pi_{\text{actor}}(\cdot \mid z,\, e), \tag{9}$$

measuring how well the extracted knowledge transfers to unseen instances.

Table 1: Results of L2C and baselines under the same-instance reward (Equation (4)) on text-games and math. L2C consistently enhances base models' ability to learn from prior interactions, improving guided accuracy across tasks and scales.

Model	Task	Method	Same-Instance Accuracy
Qwen3-1.7B	FrozenLake	Base Model	7.4
		Self-Refinement (Madaan et al., 2023; Akyurek et al., 2023)	36.5
		LLM-as-a-Coach (w/o training)	23.4
		LLM-as-a-Coach + L2C	**65.4**
Qwen3-4B-Ins	Sokoban	Base Model	9.1
		Self-Refinement	13.8
		LLM-as-a-Coach (w/o training)	12.8
		LLM-as-a-Coach + L2C	**26.6**
Qwen3-1.7B	Math	Base Model	61.4
		Self-Refinement	54.3
		LLM-as-a-Coach (w/o training)	66.1
		LLM-as-a-Coach + L2C	**67.8**
Qwen3-4B	Math	Base Model	67.7
		Self-Refinement	71.4
		LLM-as-a-Coach (w/o training)	74.4
		LLM-as-a-Coach + L2C	**78.6**
Qwen3-8B	Math	Base Model	75.8
		Self-Refinement	77.8
		LLM-as-a-Coach (w/o training)	81.9
		LLM-as-a-Coach + L2C	**82.9**

3.2 Results

We present L2C results on math and text-games in Table 1, reporting the same-instance accuracy $\mathrm{Acc}_2^{\mathrm{same}}$ with the LLM-as-a-Coach trained using experiential learning at $K{=}2$ under the same-instance reward (Equation (4)). In all settings the actor and LLM-as-a-Coach are initialized from the same Qwen3 model; the actor is frozen throughout, with only the LLM-as-a-Coach updated.

L2C delivers its largest gains on text-games. Unlike math, where the problem statement fully specifies the task, text-games have latent rules that must be inferred from interaction. Through training, the LLM-as-a-Coach learns to infer these rules from the actor's trajectory and distills them into higher-quality experiential knowledge; conditioned on it, the actor acts more effectively and achieves better performance.

On math, L2C consistently improves over the untrained coach across all model sizes. The LLM-as-a-Coach distills the actor's own reasoning trajectory into experiential knowledge that concentrates on its most useful steps, steering the actor toward correct solutions. L2C also outperforms Self-Refinement, underscoring the value of *coaching*: rather than re-feeding the actor its own full trajectory, the LLM-as-a-Coach distills it into a compact, actionable experiential knowledge snippet that more effectively guides the actor's response.

3.3 LLM-as-a-Coach Yields Transferable Experiential Knowledge

We train $K{=}2$ LLM-as-a-Coach with either the same-instance reward (Equation (4)) or the cross-instance reward (Equation (5)), and evaluate both on same- and cross-instance accuracy. We consider FrozenLake with Qwen3-1.7B and Sokoban with Qwen3-4B-Instruct-2507. The bare actor without experiential knowledge serves as the baseline. Appendix B.4 provides full evaluation details.

Figure 3 shows that training with the cross-instance reward yields large, transferable gains: cross-instance accuracy climbs far above the no-experience baseline, and the improvement generalizes

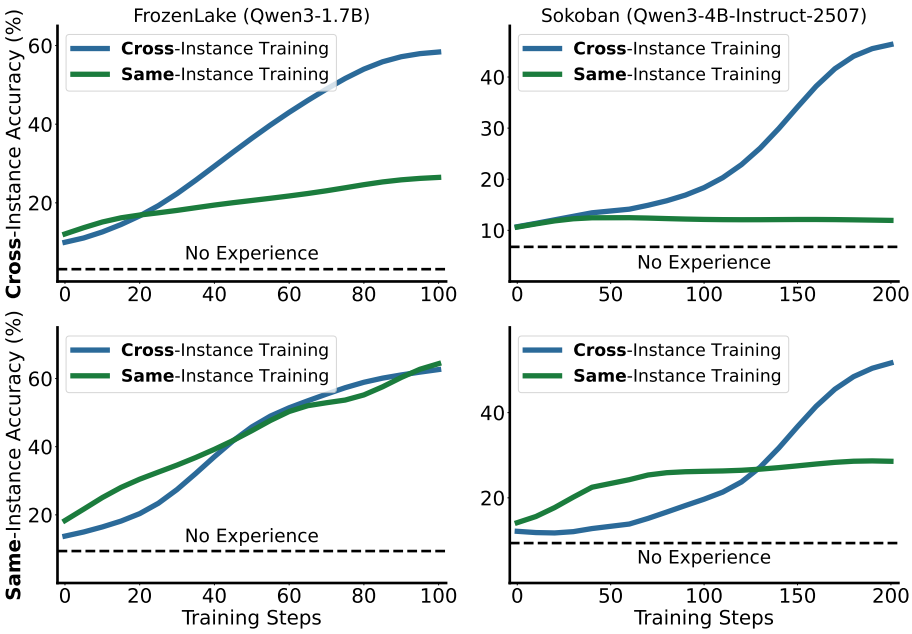


Figure 3: Same- and cross-instance accuracy of $K{=}2$ LLM-as-a-Coach trained with the same-instance reward or the cross-instance reward. Cross-instance reward training produces substantially more transferable experiential knowledge on both FrozenLake (left) and Sokoban (right).

across environments and model sizes: FrozenLake with Qwen3-1.7B and Sokoban with Qwen3-4B-Instruct-2507 alike. The cross-instance reward also transfers better than the same-instance reward: on FrozenLake the cross-instance-trained LLM-as-a-Coach reaches far higher accuracy than the same-instance-trained one, and a similar gap emerges on Sokoban. Rewarding knowledge for helping other instances thus pushes the LLM-as-a-Coach to extract environment-general rules rather than memorize instance-specific details, positioning L2C as a mechanism for distilling reusable, environment-general knowledge that generalizes to unseen instances. We also examine cross-instance transfer on DAPO math with Qwen3-8B. Appendix C presents the results and discusses how shared task structure affects the extent of transfer.

3.4 Scaling Up Experiential Learning Iterations

Section 3.2 reports the $K{=}2$ case of experiential learning; here we scale K to 10 under the same-instance reward (Equation (4)). We train the LLM-as-a-Coach under the $K{=}10$ objective (Algorithm 1) and compare it against an untrained LLM-as-a-Coach, reporting $\mathrm{Acc}_k^{\mathrm{same}}$ at every actor response $k \in \{1, \ldots, 10\}$; the two settings coincide at $k{=}1$, the initial unguided response. Figure 4 plots per-response accuracy for Qwen3-4B-Instruct-2507 on Sokoban and Qwen3-1.7B on FrozenLake, together with the actor response length and experiential knowledge length on Sokoban. Appendix D reports $\mathrm{Acc}_{10}^{\mathrm{same}}$ across all three model sizes and both tasks.

Additional experiential learning iterations improve accuracy monotonically, especially with a trained LLM-as-a-Coach. In Figure 4 (top), the trained curve rises steadily across all ten actor responses, while the untrained coach plateaus after the first guided response. L2C improves $\mathrm{Acc}_{10}^{\mathrm{same}}$ over the bare actor and untrained coach across all three model sizes and both tasks (Appendix D). Iteration is therefore the strongest axis of test-time scaling that L2C unlocks.

Accuracy rises even as the actor's responses grow shorter. Figure 4 (bottom-left) shows that the actor's responses contract monotonically with k under both coaches, so later responses cost less than the first. Progressively refined experiential knowledge thus lets the actor reach higher accuracy at a lower inference cost.

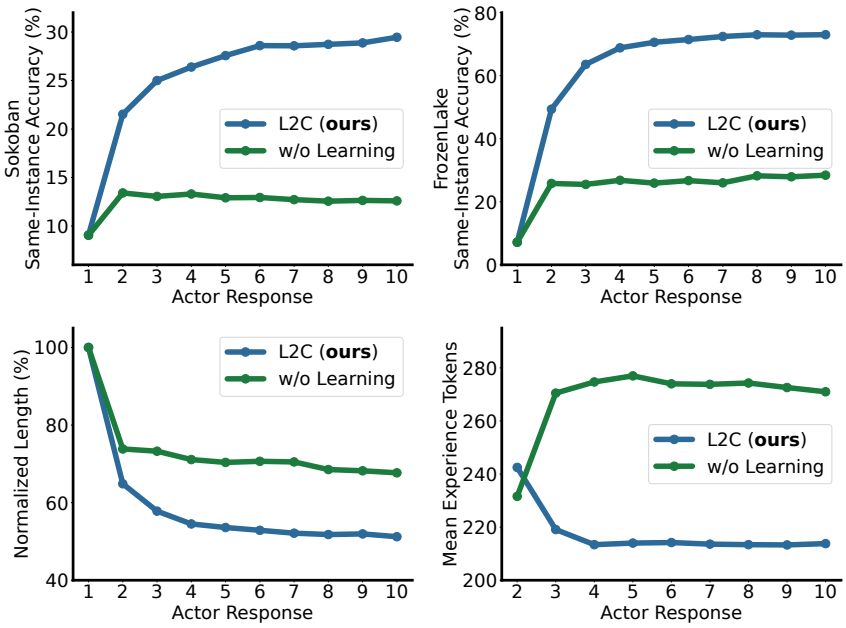


Figure 4: Experiential learning with $K{=}10$ under the same-instance reward. **Top**: same-instance accuracy on Sokoban (left) and FrozenLake (right). **Bottom**: normalized actor-response length and experiential-knowledge length on FrozenLake. The trained LLM-as-a-Coach improves accuracy across iterations while producing progressively shorter experiential knowledge and actor responses.

The trained LLM-as-a-Coach also produces shorter experiential knowledge in later responses. Figure 4 (bottom-right) shows that its output sits below the untrained baseline from $k{=}3$ onward and keeps contracting with k. Because the LLM-as-a-Coach consumes prior knowledge and the actor's fresh response, this contraction indicates that each extraction step *refines* the previous knowledge, making experiential knowledge progressively more direct and precise.

Experiential learning uses additional inference compute more effectively than simply enlarging the actor's decoding budget. Compared with a doubled 32k decoding budget on math (Table 2), a single guided response at $K{=}2$ already matches the extended-budget result for Qwen3-1.7B and surpasses it for Qwen3-4B; increasing to $K{=}10$ yields further gains across all evaluated model sizes. Iterative coaching thus converts inference compute into repeated opportunities for course correction—a form of test-time scaling that decoding-budget expansion alone does not provide. These results suggest that *where* the extra compute is spent matters as much as *how much* is spent.

Table 2: Compute scaling on DAPO with a frozen actor. "L2C, K iterations" denotes running a trained coach for K iterations. Experiential learning uses additional inference compute more effectively than doubling the decoding budget.

Model	Compute Allocation	DAPO Accuracy (%)
Qwen3-1.7B	Decoding 32k	67.9
	L2C, 2 iterations	67.8
	L2C, 10 iterations	**71.0**
Qwen3-4B	Decoding 32k	77.0
	L2C, 2 iterations	78.7
	L2C, 10 iterations	**82.6**

3.5 LLM-as-a-Coach Adapts Its Guidance to the Actor

Beyond accuracy, we ask whether the LLM-as-a-Coach tailors the experiential knowledge it extracts to the actor it coaches. We measure the per-token negative log-likelihood that the frozen actor assigns to its own guided response y' when conditioned on the experiential knowledge e,

$$\text{Negative Log-Likelihood} \;=\; -\frac{1}{|y'|}\sum_{t}\log \pi_{\text{actor}}(y'_t \mid y'_{<t},\, x,\, e)\,, \qquad y' \sim \pi_{\text{actor}}(\cdot \mid x, e), \quad (10)$$

where y' is drawn from the actor's own on-policy distribution. A lower negative log-likelihood means the guided response lies in a higher-probability region of the actor's own output distribution, indicating that e is well matched to this particular actor.

Figure 5 plots the negative log-likelihood on the DAPO test split across LLM-as-a-Coach training steps for Qwen3-1.7B and Qwen3-4B, with the untrained LLM-as-a-Coach (step 0) as the reference. L2C training consistently drives it below this baseline, and the gap widens over training.

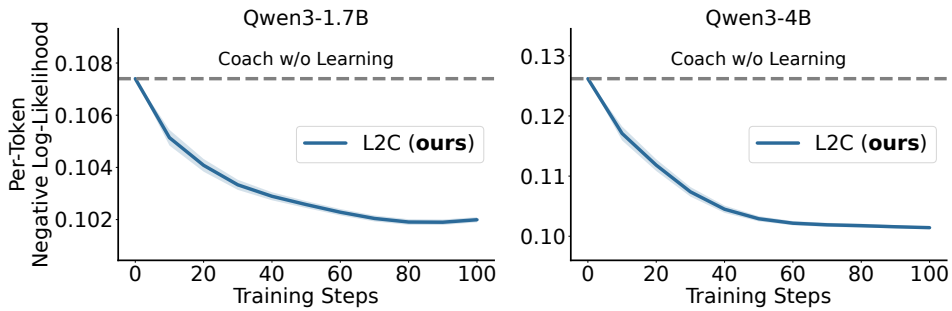


Figure 5: Per-token negative log-likelihood of the frozen actor's guided responses across LLM-as-a-Coach training steps on the DAPO test split, for Qwen3-1.7B (left) and Qwen3-4B (right). L2C training consistently reduces the negative log-likelihood below the untrained LLM-as-a-Coach baseline, indicating that the trained coach's guidance is better adapted to the actor's own policy.

We attribute this to two coupled effects. First, the trained LLM-as-a-Coach supplies clearer intermediate cues, so the actor commits more decisively to its guided response. Second, because the LLM-as-a-Coach is optimized against the actor's own on-policy rollouts, its experiential knowledge comes to elicit reasoning that stays close to the actor's familiar language and reasoning templates rather than pushing it off-distribution. This is consistent with prior work relating response likelihood to the quality and compatibility of the provided context (Wang et al., 2026).

3.6 LLM-as-a-Coach Generalizes Beyond Verifiable Tasks

We train the LLM-as-a-Coach on DAPO math with the same-instance reward and evaluate its transfer to AIME 2026, text-games, and IFEval, with DAPO as the in-distribution reference. All four benchmarks follow the same $K{=}2$ protocol as the main results—an initial response, experiential knowledge extraction, and a guided response—and are scored on the guided response: DAPO, AIME 2026, and the text-games by same-instance accuracy, and IFEval by strict instruction-following accuracy. Full evaluation protocols are provided in Appendices B.2 and B.3. L2C improves over the untrained LLM-as-a-Coach on all four benchmarks (Figure 6): harder math, interactive text-games, and instruction following, in addition to the in-distribution reference. Table 4 confirms positive transfer across model sizes. Training on verifiable math rewards therefore develops a general ability to extract and communicate actionable knowledge from prior trajectories, rather than a skill tied to the training domain.

4 Related Work

Learning from Experience Language agents can improve by reusing information acquired through prior interactions (Silver & Sutton, 2025; Zhang et al., 2025a). Existing approaches prompt agents to reflect on failures (Shinn et al., 2023; Madaan et al., 2023), retain knowledge in external memory (Zhao et al., 2024; Park et al., 2023; Sumers et al., 2023), construct reusable skill or workflow libraries (Wang et al., 2023; 2024b), or optimize prompts and programs through textual feedback (Yuksekgonul et al., 2024; Agrawal et al., 2025; Lee et al., 2025). These methods demonstrate the value of experiential knowledge, but typically rely on fixed prompting strategies, hand-designed memory operations, or heuristic extraction procedures. L2C instead treats experience extraction as a trainable decision problem: an LLM-as-a-Coach is optimized using the downstream reward of the frozen actor's guided response. Its same-instance and cross-instance rewards further distinguish knowledge that helps revisit a particular problem from knowledge that transfers across instances.

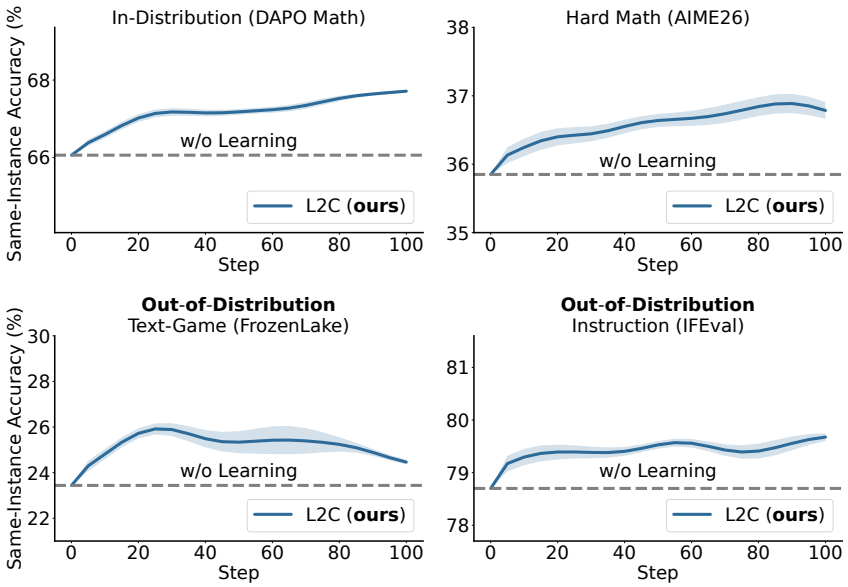


Figure 6: Performance across L2C training steps for LLM-as-a-Coach trained on DAPO math. Results use Qwen3-1.7B on DAPO (top-left), AIME 2026 (top-right), and FrozenLake (bottom-left), and Qwen3-4B on IFEval (bottom-right). Coaching ability trained with verifiable math rewards transfers to out-of-distribution reasoning, interaction, and instruction-following tasks.

Learned Critics and Teachers Auxiliary language models have been used to evaluate or improve model outputs, including AI-feedback critics (Bai et al., 2022), LLM-as-a-Judge systems (Zheng et al., 2023), process reward models (Lightman et al., 2024; Wang et al., 2024a), and models that identify errors in generated code or reasoning (McAleese et al., 2024). Teacher models can also revise instruction-tuning data (Liu et al., 2024) or provide natural-language feedback for refinement (Akyurek et al., 2023; Ye et al., 2026). Related RL approaches optimize an actor using a fixed critic (Zhang et al., 2025b; Huang et al., 2026), train a critic for refinement utility (Yu et al., 2025b), or jointly evolve the critic and actor (Li et al., 2026). Unlike standard single-round critique models, L2C extracts and refines experiential knowledge, rather than generating one-off error feedback confined to a single instance.

5 Conclusion

In this work, we introduced Learning to Coach (L2C), a framework that trains a dedicated LLM-as-a-Coach to extract actionable experiential knowledge from a frozen actor's previous trajectory. By optimizing the LLM-as-a-Coach with the reward of the actor's guided response, L2C improves downstream performance without modifying the actor. Across mathematical reasoning and interactive text-games, L2C consistently outperforms self-refinement and an untrained LLM-as-a-Coach. We further showed that the cross-instance reward induces transferable knowledge when instances share latent structure, and that scaling up experiential learning iterations provides more effective test-time scaling than enlarging the actor's decoding budget. The trained LLM-as-a-Coach also produces compact experiential knowledge and transfers to out-of-distribution tasks. Moreover, it adapts its guidance to the actor it coaches. These results establish LLM-as-a-Coach training as a modular approach for enabling language models to benefit from prior interactions while preserving the capabilities of the models being coached.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, including improving clarity, concision, and readability. We did not use these tools for retrieval or discovery, research ideation or execution, drafting substantive sections of the paper, generating synthetic datasets, or proving mathematical claims. We reviewed all AI-assisted edits to ensure that they preserved the intended meaning and accurately reflected our methods and results. We take responsibility for the final content of this work, including all text revised with the aid of generative AI.

ETHICS STATEMENT

All authors of this work have adhered to the ICLR Code of Ethics. Our experiments use publicly available benchmark datasets and simulated text-game environments, without direct involvement of human subjects or collection of personal data. This research aims to improve language models' ability to learn from prior experience in mathematical reasoning, game playing, and instruction following. We have considered potential ethical issues related to model bias, fairness, and the reliability of generated guidance. Our study is limited to controlled benchmark evaluations and does not include deployment in high-stakes settings. We encourage responsible use of our methods, compliance with the licenses of the underlying models and datasets, and appropriate safety and privacy assessments before broader deployment. We take responsibility for the scientific content and conclusions of this work.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of the results reported in this paper. To support reproducibility, Section 2 and Algorithm 1 describe the L2C framework, its reward objectives, and the experiential learning procedure. Section 3.1 identifies the base models, baselines, and evaluation metrics. Appendix A.2 describes the training datasets and text-game configurations, while Appendices A.1 and A.3 specify the training procedures, hyperparameters, rollout budgets, and training steps for each variant. Appendix A.4 provides the prompt templates for knowledge extraction and guided responses under both reward settings. Appendix B documents the evaluation sample sizes, aggregation procedures, and protocols for same-instance, cross-instance, and out-of-distribution evaluation.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *First Workshop on Foundations of Reasoning in Language Models*, 2025.

Afra Feyza Akyurek, Ekin Akyurek, Ashwin Kalyan, Peter Clark, Derry Tanti Wijaya, and Niket Tandon. RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7716–7733, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.427. URL https://aclanthology.org/2023.acl-long.427/.

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

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. Textarena. *arXiv preprint arXiv:2504.11442*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Lei Huang, Xiang Cheng, Chenxiao Zhao, Guobin Shen, Junjie Yang, Xiaocheng Feng, Yuxuan Gu, Xing Yu, and Bing Qin. Bootstrapping exploration with group-level natural language feedback in reinforcement learning. *arXiv preprint arXiv:2603.04597*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015. URL https://openreview.net/forum?id=8gmWwjFyLj.

Yoonho Lee, Joseph Boen, and Chelsea Finn. Feedback descent: Open-ended text optimization via pairwise comparison. *arXiv preprint arXiv:2511.07919*, 2025.

Zhicong Li, Lingjie Jiang, Yulan Hu, Xingchen Zeng, Yixia Li, Xiangwen Zhang, Guanhua Chen, Zheng Pan, and Xin Li. No more stale feedback: Co-evolving critics for open-world agent learning. *arXiv preprint arXiv:2601.06794*, 2026. URL https://arxiv.org/abs/2601.06794.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Yilun Liu, Shimin Tao, Xiaofeng Zhao, Ming Zhu, Wenbing Ma, Junhao Zhu, Chang Su, Yutai Hou, Miao Zhang, Min Zhang, et al. Coachlm: Automatic instruction revisions improve the data quality in llm instruction tuning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 5184–5197. IEEE, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. Llm critics help catch llm bugs. *arXiv preprint arXiv:2407.00215*, 2024.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 2025.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

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

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439. Association for Computational Linguistics, 2024a. URL https://aclanthology.org/2024.acl-long.510/.

Sai Wang, Yu Wu, and Zhongwen Xu. Cogito, ergo ludo: An agent that learns to play by reasoning and planning. *arXiv preprint arXiv:2509.25052*, 2025.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking, 2026. URL https://arxiv.org/abs/2603.10165.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Tianzhu Ye, Li Dong, Guanheng Chen, He Zhu, Xun Wu, Shaohan Huang, and Furu Wei. Llm-as-a-coach: Experiential learning for non-verifiable tasks. *arXiv preprint arXiv:2607.18110*, 2026.

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. *arXiv preprint arXiv:2503.14476*, 2025a.

Tianshu Yu, Chao Xiang, Mingchuan Yang, Pei Ke, Bosi Wen, Cunxiang Wang, Jiale Cheng, Li Zhang, Xinyu Mu, Chuxiong Sun, and Minlie Huang. Training language model to critique for better refinement. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26760–26804, Vienna, Austria, 2025b. Association for Computational Linguistics. URL https://aclanthology.org/2025.findings-acl.1373/.

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

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025a.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025b.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of NeurIPS*, 2023. URL https://neurips.cc/virtual/2023/poster/73434.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Training Details

A.1 Training Procedure

We optimize the LLM-as-a-Coach with GRPO (Shao et al., 2024) while keeping the actor frozen. For each source trajectory, we sample $n{=}8$ candidate experiential-knowledge snippets from the LLM-as-a-Coach; their rewards form one GRPO advantage-normalization group. Each candidate conditions an independent guided response from the frozen actor, which the verifier $\mathcal{V}$ scores as a binary reward: answer correctness on math, and the outcome of the guided playthrough on text-games. Actor responses are limited to $16{,}384$ tokens on math and, on text-games, to five interaction turns of at most $1{,}024$ tokens each; LLM-as-a-Coach responses are limited to $8{,}192$ tokens throughout. We optimize all LLM-as-a-Coach models with AdamW (Kingma & Ba, 2015), using a constant learning rate of 10^{-6}.

We train three variants of the LLM-as-a-Coach, each initialized from the same checkpoint as its frozen actor. **(1) Same-instance experiential learning** ($K{=}2$) maximizes the same-instance reward (Equation (4)) and is trained for 100 steps on math (Qwen3-1.7B/4B/8B), FrozenLake (Qwen3-1.7B), and Sokoban (Qwen3-4B-Instruct-2507). **(2) Same-instance experiential learning** ($K{=}10$) maximizes the same reward across ten actor responses: one initial response followed by nine guided responses. At each iteration, one of the eight trajectories $(e^{(k)}, y^{(k+1)})$ is drawn uniformly to seed the next iteration. Training runs for 100 steps on math (Qwen3-1.7B/4B/8B) and Sokoban (Qwen3-4B-Instruct-2507), and 200 steps on FrozenLake (Qwen3-1.7B). **(3) Cross-instance experiential learning** ($K{=}2$) maximizes the cross-instance reward (Equation (5)), where the probe set $\mathcal{P}$ holds eight instances for math and seven for text-games. Training runs for 100 steps on math and FrozenLake and 200 steps on Sokoban. These budgets bring the reward curves close to convergence. Full configurations are provided in Appendix A.3.

A.2 Datasets

Our training data span three task domains: mathematical problem solving and two interactive, text-only games. For mathematics, we use DAPO-Math-17K (Yu et al., 2025a), which includes roughly 14K English-language problems whose final numerical answers can be automatically verified. For interactive reasoning, we adopt the FrozenLake and Sokoban environments provided by TextArena (Guertler et al., 2025).

In FrozenLake, the agent navigates a $3{\times}3$ board containing two holes and must find a safe route to the goal. Sokoban is instantiated on a $6{\times}6$ board with a single box; the agent must push the box onto its target while avoiding holes, walls, and irreversible deadlocks. Following prior work (Wang et al., 2025), we omit some of the game instructions so that the agent must discover the missing mechanics through interaction. At every turn, TextArena converts the current board configuration into a textual observation, and the language model responds with an action, resulting in a multi-turn trajectory.

A.3 Training Configuration

We optimize the LLM-as-a-Coach with GRPO (Shao et al., 2024) while keeping the actor frozen. For each source trajectory, we sample $n{=}8$ candidate experiential-knowledge snippets from the LLM-as-a-Coach. Each candidate conditions an independent guided response by the actor, and the resulting eight rewards form one GRPO advantage-normalization group. All LLM-as-a-Coach models are optimized with AdamW (Kingma & Ba, 2015) using a constant learning rate of 10^{-6}.

Math. For math, each mini-batch contains 64 source problems for the Qwen3-1.7B and Qwen3-4B LLM-as-a-Coach models and 128 source problems for the Qwen3-8B LLM-as-a-Coach. Actor responses, both initial and guided, are limited to $16{,}384$ tokens, and LLM-as-a-Coach responses are limited to $8{,}192$ tokens. Under the same-instance reward, each candidate conditions an independent guided response on the source problem, and a deterministic answer verifier assigns a binary correctness reward.

Text-games. For text-games, each mini-batch contains 64 environment seeds. An initial or guided response is a playthrough of at most five interaction turns, with actor responses limited to $1{,}024$

tokens per turn. LLM-as-a-Coach responses are limited to 8,192 tokens. Under the same-instance reward, each candidate conditions an independent guided playthrough using the same environment seed as the initial response, and the binary win/loss outcome provides the reward.

Training variants. We train three variants of the LLM-as-a-Coach, each initialized from the same base checkpoint as its corresponding actor.

- **Experiential learning with $K=2$.** We use the same-instance reward with $K=2$, corresponding to one initial response, one extraction step, and one guided response. The main results use 100-step checkpoints: math with Qwen3-1.7B/4B/8B, FrozenLake with Qwen3-1.7B, and Sokoban with Qwen3-4B-Instruct-2507. The Sokoban run continues to 200 steps for comparison with cross-instance training.
- **Experiential learning with $K=10$.** We use the same-instance reward with $K=10$, corresponding to one initial response followed by nine guided responses. At each iteration, the LLM-as-a-Coach samples eight candidate updates to the experiential knowledge, and each candidate is evaluated through an independent guided response. One of the eight candidates is then selected uniformly, together with its corresponding actor trajectory, to provide the state for the next iteration. We train the math LLM-as-a-Coach models and the Qwen3-4B-Instruct-2507 Sokoban LLM-as-a-Coach for 100 gradient steps and the Qwen3-1.7B FrozenLake LLM-as-a-Coach for 200 gradient steps.
- **Cross-instance experiential learning.** We use $K=2$ and replace the same-instance reward with the cross-instance reward in Equation (5). Each step uses 64 source instances and a probe set disjoint from the sources. Each candidate is applied independently to every probe instance, and its reward is the mean guided accuracy over the probe set. We use eight probes for math, shared across all source groups within a step, and seven per-group probes for text-games. We train the math and FrozenLake LLM-as-a-Coach models for 100 gradient steps and the Qwen3-4B-Instruct-2507 Sokoban LLM-as-a-Coach for 200 gradient steps.

Table 3 summarizes the number of GRPO gradient steps in each setting.

Table 3: **L2C training configurations.** We report the number of GRPO optimizer steps for same-instance experiential learning ($K=2$), same-instance experiential learning ($K=10$), and cross-instance experiential learning ($K=2$). All runs use a frozen actor, eight LLM-as-a-Coach samples per source trajectory, AdamW, and a constant learning rate of 10^{-6}.

Reward	Protocol	Task (model)	Steps
Same-instance	$K=2$ (Section 3.2)	Math (Qwen3-1.7B / 4B / 8B)	100
		FrozenLake (Qwen3-1.7B)	100
		Sokoban (Qwen3-4B-Instruct-2507)	100
Same-instance	$K=10$ (Section 3.4)	Math (Qwen3-1.7B / 4B / 8B)	100
		Sokoban (Qwen3-4B-Instruct-2507)	100
		FrozenLake (Qwen3-1.7B)	200
Cross-instance	$K=2$ (Section 3.3)	Math (Qwen3-8B)	100
		FrozenLake (Qwen3-1.7B)	100
		Sokoban (Qwen3-4B-Instruct-2507)	200

A.4 Prompt Templates

L2C uses two families of prompts, matching the two rewards of Section 2: the *same-instance* prompts behind the main results (Sections 3.2 and 3.4), and the *cross-instance* prompts behind the transfer experiment (Section 3.3). We list both below.

Same-instance prompts. For experiential knowledge update on the math dataset, we use the prompt template in Figure 7.

You are solving a problem. Below are your previous attempts and the notes you took while solving. Refine the notes so your next attempt is more likely to be correct.

Problem and your latest attempt:
{LATEST_EXPERIENCE}

Your existing notes on this problem (from earlier attempts):
Notes
{PREVIOUS_EXPERIENCE}

Your task:
- Critique the latest attempt. Where did the reasoning go wrong, or what was left unverified? If the answer seems correct, what would make you more sure?
- Update the notes so they capture: (a) what you have already tried and why it did or did not work, (b) the most promising direction for the next attempt, (c) any intermediate results worth keeping (lemmas, simplifications, candidate answers with confidence).
- The notes will REPLACE the previous notes, not be appended. Keep them concise. They will be re-read at every attempt.

After reasoning step by step, output the final notes in exactly this format:

Notes
- ...
- ...

Figure 7: The prompt template for the LLM-as-a-Coach's notes update on the math dataset.

The LLM-as-a-Coach's output replaces the previous notes verbatim (no per-line item extraction).

For experiential knowledge update on text-based games, we use the prompt template in Figure 8.

You are playing a game. Below is your latest playthrough and the notes you took while playing. Refine the notes so your next attempt is more likely to win.

Interaction history (the game environment (input) and your response and action (output)):
{LATEST_EXPERIENCE}

Your existing notes on this game (from earlier attempts):
Notes
{PREVIOUS_EXPERIENCE}

Your task:
- Critique the latest playthrough. Which actions moved you toward or away from the goal, what feedback did the environment give, and where did the attempt fail?
- Update the notes so they capture: (a) confirmed facts about this map (goal location, hazards/walls, what each action does), (b) action sequences that worked or failed and why, (c) the most promising plan for the next attempt.
- The notes will REPLACE the previous notes, not be appended. Keep them concise. They will be re-read at every attempt.

After reasoning step by step, output the final notes in exactly this format:

Notes
- ...
- ...

Figure 8: The prompt template for the LLM-as-a-Coach's notes update on text-based games.

For guided responses on the same problem (or game), we embed the LLM-as-a-Coach's notes with the prompt template in Figure 9.

You have been working on the problem below. Here are your notes from previous attempts; use them to avoid past mistakes and find the correct answer.

Notes
{experience}

Solve the problem. You may continue from a promising direction in the notes or start over if needed; the notes are guidance, not constraints.

Problem:
{prompt}

Figure 9: The prompt template for a guided response with the LLM-as-a-Coach's notes.

Cross-instance prompts. The cross-instance variant rewards *transferable* experience, so its prompts extract general, high-level insight that is accumulated into a running knowledge base and then applied to *new* instances. For experience update on the math dataset we use Figure 10; on text-based games (the setting of the FrozenLake transfer experiment in Section 3.3) we use Figure 11. The accumulated experience is applied to a new instance with the solve template in Figure 12.

You are an AI language model that continuously refines its internal experience.

Here is the latest interaction (the user's question and your answer):
{LATEST_EXPERIENCE}

Here is the previous experience:
Experience
{PREVIOUS_EXPERIENCE}

Your task:
Based on the latest interaction and the previous experience, generate an additional experience for future learning.

Rules:
- The experience you generate MUST be formatted strictly as a markdown list where each item starts with "- EXPERIENCE ITEM:", one per line:
- EXPERIENCE ITEM: ...
- EXPERIENCE ITEM: ...
- EXPERIENCE ITEM: ...
- The experience you generate will be directly appended to the previous experience.
- The change should introduce a general, high-level, widely applicable insight, not a detail from the specific interaction. The updated experience must remain concise, structured, and meaningful.
- If the new insight conflicts with any previous experience item, you can describe the conflict and provide a resolution in the new item.

After careful reasoning step by step, output the final result in exactly this format:

Additional Experience:
Experience
- EXPERIENCE ITEM: ...
- EXPERIENCE ITEM: ...
- EXPERIENCE ITEM: ...

Figure 10: The cross-instance prompt template for experience extraction on the math dataset: general, transferable insight accumulated as appended experience items.

You are an AI language model that continuously refines its internal experience.
Here is the interaction history (the game environment (input) and your response and action (output)):
{LATEST_EXPERIENCE}

Here is the previous experience:
Experience
{PREVIOUS_EXPERIENCE}

Your task:
Based on the multi-round interaction history and the previous experience, generate experience for future learning. You should conduct a deep, comparative analysis to infer the game rules and the fundamental principles behind winning and losing. Using the interaction history and environment feedback, hypothesize the game rules and effective winning strategies, and organize these insights into 1–2 concise, high-level, and widely applicable experience items that help the player succeed in the game.

Rules:
- The experience you generate MUST be formatted strictly as a markdown item which starts with "- EXPERIENCE ITEM:":
- EXPERIENCE ITEM: ...
- EXPERIENCE ITEM: ...
- The experience you generate will be directly appended to the previous experience. Do not repeat the previous experience. Make sure the newly generated experience is different from the previous experience.
- Your generated experience should be possible rules, instructions or winning strategies for the game. The experience should be generally useful rather than only applicable for the current map (board).

After careful reasoning step by step, output the final result in exactly this format:

Additional Experience (Rules or Strategies):
Experience
- EXPERIENCE ITEM: ...

Figure 11: The cross-instance prompt template for experience extraction on text-based games: transferable game rules and winning strategies rather than map-specific notes.

Given previous learned experience:
Experience
{experience}

Solve the new problem and explain what part of experience you use and how you use it in the reasoning process:
{prompt}

Figure 12: The cross-instance prompt template for solving a new instance with the accumulated experience.

A.5 Examples of Learned Experience

The two reward variants produce qualitatively different experiential knowledge, mirroring their prompts. Figure 13 shows a representative *same-instance* note learned on FrozenLake (Qwen3-1.7B): it is tied to one specific map, recording exact obstacle coordinates and a memorized step-by-step solution path. Figure 14 shows a *cross-instance* experience from the same setting: it abstracts the interaction into general game rules and a reusable navigation strategy intended to apply across maps.

Notes
- **Board Layout**: 3x3 grid with player (P) at (0,0), goal (G) at (2,2), and obstacles (H) at (0,2) and (1,2). Valid moves: up (w), down (s), left (a), right (d).
- **Optimal Path**:
1. **Down (s)** from (0,0) to (1,0) → Establishes vertical path toward the goal.
2. **Down (s)** from (1,0) to (2,0) → Progresses toward the goal.

3. **Right (d)** from (2,0) to (2,1) → Advances row.
4. **Right (d)** from (2,1) to (2,2) → Directly reaches the goal (G).
- **Critical Failures**:
- Moving **up (w)** from (0,0) is invalid (out of bounds).
- Moving **left (a)** from (0,0) is invalid (out of bounds).

Figure 13: A *same-instance* experience (v5) learned on FrozenLake (Qwen3-1.7B). The note memorizes one map's layout and a concrete solution path.

Experience
- EXPERIENCE ITEM: **Game Rules and Winning Strategy**
The game involves navigating a 3x3 grid where the player (P) starts at the top-left corner (row 0, column 0) and must reach the goal (G) at the bottom-right corner (row 2, column 2). The grid contains obstacles (H) that block movement. The player can move up, down, left, or right, and must avoid obstacles while reaching the goal.
- EXPERIENCE ITEM: **Effective Winning Strategy**
The optimal strategy is to move **down** first to progress toward the goal. From the starting position (0,0), move **down** to (1,0), then **right** to (1,1), (1,2), and finally **down** to (2,2). This path avoids obstacles and ensures the shortest route to the goal. Always prioritize moving **down** to progress toward the goal, and use **right** to navigate through empty spaces.

Figure 14: A *cross-instance* experience (v4) learned on FrozenLake (Qwen3-1.7B). The experience abstracts the interaction into game rules and a reusable strategy rather than a single map's solution.

B Evaluation Details

B.1 Evaluation Configuration

For math, we evaluate on the DAPO test split (1,000 problems, n=16 rollouts per problem). For text-games, we use 500 held-out environment seeds (n=8 playthroughs per seed).

B.2 Out-of-Distribution Generalization Evaluation

For the out-of-distribution generalization results of Section 3.6 (Table 4), we evaluate on AIME 2026 with 30 problems and n=128 rollouts per problem, and on the text-games with 500 held-out environment seeds and n=8 playthroughs per seed. The in-distribution DAPO reference follows the main evaluation protocol of Appendix B.1.

Table 4: **Out-of-distribution transfer of LLM-as-a-Coach models trained on DAPO math.** Same-instance accuracy (%) after one guided response on AIME 2026 and text-game tasks. The actor remains frozen, and Δ denotes the percentage-point improvement of L2C over the untrained LLM-as-a-Coach.

Out-of-Distribution Task	**Model**	$\mathrm{Acc}_2^{\text{same}}$ **(w/o training)**	$\mathrm{Acc}_2^{\text{same}}$ **(L2C)**	Δ **(pp)**
AIME26	Qwen3-1.7B	35.9	**37.4**	+1.5
	Qwen3-8B	61.6	**63.4**	+1.8
Text-game	Qwen3-1.7B (FrozenLake)	23.4	**26.3**	+2.9
	Qwen3-4B (Sokoban)	7.6	**9.4**	+1.8

B.3 IFEval Out-of-Distribution Evaluation

For the out-of-distribution instruction-following evaluation (Section 3.6), we evaluate the trained LLM-as-a-Coach–actor system on IFEval (Zhou et al., 2023) under the same K=2 protocol used

for math and text-games: the frozen actor first answers the prompt on its own, the LLM-as-a-Coach extracts experiential knowledge from the resulting trajectory, and the actor then produces a guided response conditioned on that knowledge. Only the guided response is scored. The untrained-coach baseline is step 0 of the identical pipeline.

We evaluate on the first 250 prompts of the IFEval test split as ordered by `lm-eval-harness`. For each prompt, the LLM-as-a-Coach samples n=16 independent experiential-knowledge snippets from the actor's initial response, using temperature 0.6, top-p=0.95, top-k=20, and an 8,192-token budget, matching the training-time rollout configuration. Each snippet conditions one guided response, yielding $250 \times 16 = 4{,}000$ guided responses per checkpoint. In both actor passes the model's chat template is applied as a single user turn with thinking mode disabled, and decoding is greedy with a 1,280-token budget. Each guided response is scored by IFEval's prompt-level strict criterion (`prompt_level_strict_acc`); we report the per-prompt mean over the 16 snippets and then average over the 250 prompts. All other hyperparameters follow `lm-eval-harness` defaults.

B.4 Cross-Instance Transfer Evaluation

We evaluate cross-instance transfer using two disjoint pools drawn from the task: a source pool $\mathcal{S}$ of $|\mathcal{S}|$=64 instances and a held-out probe set $\mathcal{P}$ of $|\mathcal{P}|$=250 instances (16,000 source–probe pairs), used for FrozenLake (Qwen3-1.7B) and Sokoban (Qwen3-4B-Instruct-2507) in Section 3.3, where instances are environment seeds. For each source instance $s \in \mathcal{S}$, the actor first responds to s without experiential knowledge and the LLM-as-a-Coach compacts the resulting trajectory into experiential knowledge e_s. At each checkpoint we then report three quantities:

- **Cross-instance accuracy**: apply each e_s to every probe instance $z \in \mathcal{P}$ and average the actor's guided accuracy over all $|\mathcal{S}| \times |\mathcal{P}|$ source–probe pairs, matching the cross-instance reward of Equation (5).
- **Same-instance guided accuracy**: apply each e_s back to its own source instance s, mirroring the same-instance setting of the main results; it serves as a memorization discriminator.
- **No-experience baseline**: the actor's accuracy on the probe instances without any experiential knowledge.

Both pools are held fixed across checkpoints, and $\mathcal{P}$ is disjoint from $\mathcal{S}$ so that transfer is always measured on instances the experiential knowledge was not extracted from.

C Cross-Instance Transfer on Math

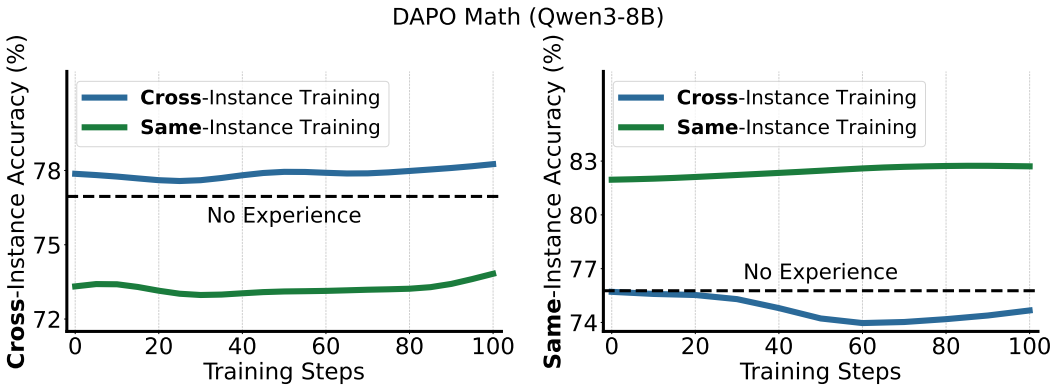


Figure 15: Cross-instance (left) and same-instance (right) accuracy on DAPO math with Qwen3-8B, comparing LLM-as-a-Coach models trained with the two reward variants. Cross-instance gains are smaller than in text-games, consistent with less experiential knowledge being shared across math problems than across instances governed by common game rules.

Experiential knowledge transfers across instances when solving one instance reveals structure that also applies to others. We study this on DAPO math with a frozen Qwen3-8B actor and a Qwen3-

8B LLM-as-a-Coach, complementing the text-game experiments of Section 3.3. Figure 15 compares same- and cross-instance training; its caption specifies the evaluation protocols used for each curve.

Cross-instance accuracy increases from 78.0% with the untrained coach (step 0) to 78.4% after 100 training steps, a change of only 0.4 percentage points. At step 100, the bare actor achieves 77.2%, so the trained coach exceeds the same-checkpoint no-experience baseline by 1.20 percentage points. These are single-run estimates, with the evaluation seed varying across checkpoints. Thus, the additional transfer associated with coach training is relatively smaller than textgame transfer, despite guided accuracy exceeding the bare actor's accuracy at the final checkpoint.

These results show positive cross-instance transfer on math, with further improvement from coach training, although the gains are smaller than those observed in text-games. This difference is consistent with the amount of experiential knowledge shared across instances. Math trajectories can reveal reusable reasoning strategies, but many intermediate results and solution steps depend on the particular problem. In text-games, latent rules such as hazards, dynamics, and valid moves remain shared across maps, making experience from one instance more broadly applicable. The results therefore suggest that the extent of transfer depends on how much reusable experiential knowledge the task provides, rather than on a strict distinction between mathematical and interactive domains.

D Experiential Learning Results at $K=10$

Table 5 reports the final-iteration accuracy $\mathrm{Acc}_{10}^{\text{same}}$ under the $K=10$ protocol for the bare actor, the untrained coach, and L2C, across all three model sizes and both tasks. L2C improves over both baselines on every combination of model size and task.

Table 5: **Accuracy under same-instance experiential learning at $K=10$.** The base-actor rows report initial-response accuracy $\mathrm{Acc}_{1}^{\text{same}}$, whereas the remaining rows report final accuracy $\mathrm{Acc}_{10}^{\text{same}}$ after nine guided responses. L2C outperforms the untrained LLM-as-a-Coach in all evaluated settings.

Model	Task	Method	Accuracy
Qwen3-1.7B	FrozenLake	Base Model	7.4
		LLM-as-a-Coach w/o training, $\mathrm{Acc}_{10}^{\text{same}}$	28.4
		L2C, $\mathrm{Acc}_{10}^{\text{same}}$	**73.0**
Qwen3-4B-Ins	Sokoban	Base Model	9.1
		LLM-as-a-Coach w/o training, $\mathrm{Acc}_{10}^{\text{same}}$	12.6
		L2C, $\mathrm{Acc}_{10}^{\text{same}}$	**29.5**
Qwen3-1.7B	Math	Base Model	61.4
		LLM-as-a-Coach w/o training, $\mathrm{Acc}_{10}^{\text{same}}$	69.9
		L2C, $\mathrm{Acc}_{10}^{\text{same}}$	**71.0**
Qwen3-4B	Math	Base Model	67.7
		LLM-as-a-Coach w/o training, $\mathrm{Acc}_{10}^{\text{same}}$	75.5
		L2C, $\mathrm{Acc}_{10}^{\text{same}}$	**82.6**
Qwen3-8B	Math	Base Model	75.8
		LLM-as-a-Coach w/o training, $\mathrm{Acc}_{10}^{\text{same}}$	82.9
		L2C, $\mathrm{Acc}_{10}^{\text{same}}$	**85.6**

E Effect of Model Size

We isolate the effect of LLM-as-a-Coach capacity by fixing the actor at Qwen3-4B and varying the LLM-as-a-Coach across Qwen3-1.7B, Qwen3-4B and Qwen3-8B, each trained for 100 GRPO steps with the same-instance reward (Equation (4)). Figure 16 reports both the $K=2$ and the $K=10$ settings on DAPO test.

L2C training consistently improves over the untrained coach across all LLM-as-a-Coach sizes in both the $K=2$ and $K=10$ settings, confirming that the training recipe is robust to LLM-as-a-Coach capacity. However, LLM-as-a-Coach size itself has a surprisingly small effect on final accuracy:

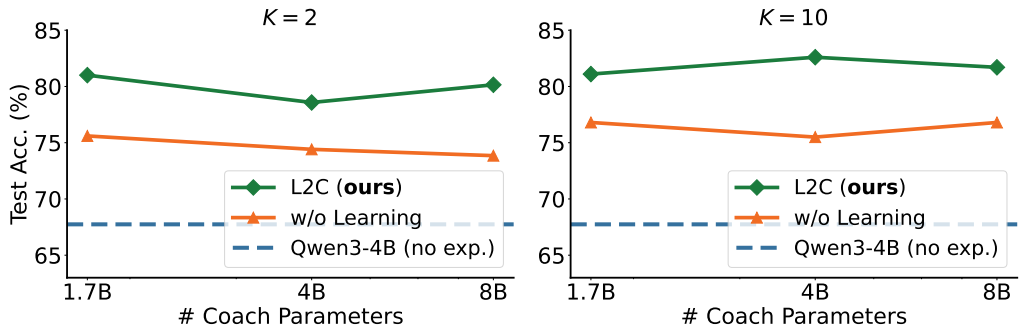


Figure 16: Effect of LLM-as-a-Coach size on guided accuracy, with the actor fixed at Qwen3-4B on DAPO test. **Left**: $K{=}2$. **Right**: $K{=}10$. L2C training consistently improves over the untrained coach across all LLM-as-a-Coach sizes, yet LLM-as-a-Coach size itself has little effect on final accuracy under both settings.

at both $\mathrm{Acc}_2^{\mathrm{same}}$ and $\mathrm{Acc}_{10}^{\mathrm{same}}$, all three trained LLM-as-a-Coach models reach comparable accuracy, and the untrained variants likewise cluster within a narrow range.

Actor capacity, by contrast, is the dominant factor. Pairing the same LLM-as-a-Coach with a stronger actor yields far larger gains than scaling the LLM-as-a-Coach itself (Table 5). This suggests that guided accuracy is primarily rate-limited by the actor's ability to leverage experiential knowledge, rather than by who produced it. The finding highlights the practical potential of using a small, lightweight model as the LLM-as-a-Coach to enhance a much larger actor at modest additional inference cost.