LEARNING TO TEACH, LEARNING TO PLAY: DUAL REINFORCEMENT LEARNING WITH ACTIVE TRAJECTORY INQUIRY

Anonymous authors
Paper under double-blind review

ABSTRACT

Foundation models can guide reinforcement learning (RL) agents, but useful guidance depends on understanding how a situation arose and where it can still be changed. In dynamic games, the information needed for this judgment may be scattered across a trajectory and become apparent only during investigation, making it difficult to specify a sufficient context in advance. We introduce TrajTeach, a dual reinforcement learning framework that trains a foundation-model teacher to investigate and improve an RL student's behavior. We develop TrajQL, a structured trajectory-query language through which the teacher locates relevant events, state changes, and objects, follows up on its findings, and decides whether, where, and how to intervene. Return gains over the student's continuation train the teacher's inquiry and intervention policy, while the resulting experience trains the student. Trajectory analyses across six games illustrate how inquiry connects observed problems to actionable corrections, including tracing delayed failures to earlier decisions, recovering missing task prerequisites, and revising action timing. Only the student is deployed, preserving its original inference cost.

1 INTRODUCTION

Reinforcement learning (RL) provides a general framework for learning sequential decision-making policies through interaction with an environment (Sutton & Barto, 2018). Modern deep RL agents can execute actions at high frequency and low inference cost, and have achieved strong performance across games, robotics, and control (Mnih et al., 2015; Haarnoja et al., 2018; Badia et al., 2020a). Yet this capability often comes at the cost of extensive trial-and-error, particularly when agents learn from scratch with little task-specific prior knowledge. When rewards are sparse or delayed, useful behaviors are difficult to discover, and even beneficial actions may reveal their value only after many steps (Pathak et al., 2017; Burda et al., 2019; Badia et al., 2020b). As a result, an agent can spend a substantial fraction of its interaction budget on unproductive or repetitive exploration (Ecoffet et al., 2021).

By contrast, large language models (LLMs) and vision-language models (VLMs) acquire broad knowledge and reasoning capabilities through large-scale pretraining (Brown et al., 2020; Alayrac et al., 2022). This knowledge offers a source of guidance that does not have to be acquired solely through task-specific trial-and-error (Du et al., 2023; Yan et al., 2025). Researchers have explored several ways to bring these capabilities into agent learning, including suggesting actions and goals (Du et al., 2023; Zhou et al., 2024), evaluating behavior, and shaping rewards (Klissarov et al., 2024; Sancaktar et al., 2025). These approaches use the pretrained model as a teacher whose guidance helps the RL agent learn more effectively. The underlying expectation is that a model with broader knowledge can recognize promising courses of action that the student has not yet discovered, helping it both learn useful behavior and improve existing strategies (Bougie & Watanabe, 2025; Yan et al., 2025).

But are pretrained models always reliable teachers? Games are dynamic and temporally dependent: understanding what an agent should do often requires knowing how the current situation arose and how earlier actions played out (Hu et al., 2025). A failed jump, for example, may only make sense in light of the preceding approach and the platform's movement. By the time a failure becomes

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.

apparent, the decision that could have prevented it may already lie in the past. Yet existing methods typically expect the teacher to provide guidance from a preassembled context (Zhou et al., 2024; Wan et al., 2025; Choudhury & Sodhi, 2025; Lyu et al., 2026). This implicitly assumes that the supplied information is sufficient for the judgment at hand. In games, however, the information needed to understand a decision depends on how the situation unfolds: relevant events may be separated by many steps, and their significance may only become apparent when later outcomes are examined. It is therefore difficult to anticipate which parts of an agent's experience will be needed for each judgment, limiting what a teacher can diagnose and correct from a context assembled in advance.

Humans approach such situations differently. What they revisit can change as their understanding of the problem develops. A player reviewing a failed jump may first revisit the approach, then check the platform's earlier movement after finding that the takeoff position alone does not explain the failure. Each discovery can revise the player's explanation and suggest what to examine next. Human decision making thus combines prior knowledge with an active process of recalling, checking, and updating an understanding of the situation (Momennejad et al., 2018; Eldar et al., 2020). Moreover, this process itself improves with experience: when a diagnosis repeatedly leads to poor outcomes, the player may reconsider which details deserve attention, while useful observations become cues for future judgment (Mattar & Daw, 2018). Both what to examine and how to act on what is learned can therefore be refined through interaction and feedback.

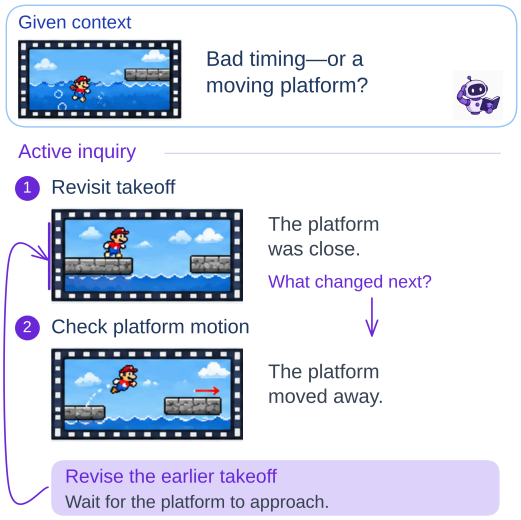


Figure 1: **Understanding before guiding.** Each discovery guides the next query, helping the teacher trace a failed jump back to an earlier decision to revise.

Motivated by this perspective, we introduce **TrajTeach**, a framework in which a foundation-model teacher learns to investigate an RL student's experience and provide targeted guidance. We design TrajQL, a structured trajectory-query language, to let the teacher retrieve relevant observations and pursue further questions before deciding whether, where, and how to intervene. The teacher's inquiry and guidance are themselves learned through environment feedback. Specifically, a dual reinforcement learning scheme uses return gains over the student's continuation from the same state to train the teacher, while the resulting intervention transitions train the student.[1]

Our main contributions are:

- We introduce TrajTeach and its trajectory-query language, TrajQL, to let a teacher actively investigate and correct an RL student's behavior. Successive queries follow emerging questions and connect observed problems to decisions where useful corrections remain possible.
- We develop a dual reinforcement learning scheme that trains both inquiry and intervention through gains over the student's continuation. The student learns from the resulting transitions, coupling the teacher's ability to diagnose and correct behavior with the student's task learning.
- Through trajectory analyses across six games, we characterize how corrections depend on distant events, task prerequisites, and local action sequences. These cases clarify why useful guidance requires identifying both the relevant history and a decision from which the outcome can still be changed.

[1]Code: `https://anonymous.4open.science/r/TrajTeach-03FB/`.

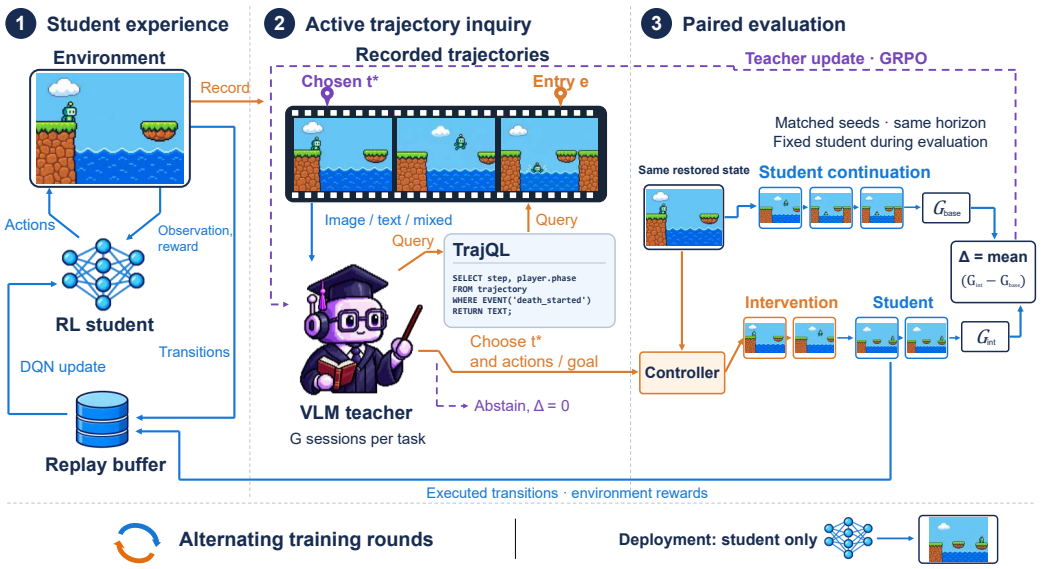


Figure 2: **Framework overview.** A student interacts with the environment; a foundation-model teacher queries its recorded trajectories and proposes interventions. Paired execution supplies return differences for teacher learning and additional transitions for student learning. Both policies are trained, with only the student retained at deployment. Game frames are schematic.

2 METHODOLOGY

In this section, we introduce three components of TrajTeach: the problem setup and organization of student experience (Section 2.1), active trajectory inquiry through TrajQL (Section 2.2), and dual reinforcement learning from teacher interventions (Section 2.3). Figure 2 provides an overview.

2.1 PROBLEM SETUP AND EXPERIENCE COLLECTION

We model the underlying game dynamics as a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma_S)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel P, reward function r, and discount factor γ_S. The RL student π_θ acts on observations o_t, such as frame stacks or text, which need not reveal the full state s_t. Its action a_t induces $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ and reward r_t. A foundation-model teacher π_ϕ guides the student's learning by examining its experience and proposing interventions during training. At test time, the student acts independently.

Each transition (o_t, a_t, r_t, o_{t+1}) is stored in a replay buffer $\mathcal{D}$ for student learning. Consecutive transitions from a rollout form a trajectory $\tau = (o_0, a_0, r_0, \ldots, o_T)$. We retain these trajectories, denoted collectively by $\mathcal{T}$, so that the teacher can examine how the student's decisions relate to preceding events and subsequent outcomes. The same experience thus supports both the student's policy updates and the teacher's trajectory inquiry.

2.2 ACTIVE TRAJECTORY INQUIRY WITH TRAJQL

Queryable trajectory representation. During student rollouts, we automatically record observations, actions, rewards, and structured game information, including player status, task progress, and object positions. These fields are extracted through game-engine interfaces and game-specific decoding rules, then stored with the corresponding observation steps. We design TrajQL (Trajectory Query Language) to make these recorded trajectories searchable: the teacher can retrieve both the original observations and selected state information. Access to internal game information is restricted to training; the student's observation interface remains unchanged.

Table 1: **Core capabilities of TrajQL.** SQL fragments show how the teacher searches and inspects recorded experience. A parameter such as `:t` can refer to a step returned by a preceding query.

Capability	TrajQL expression	Role in inquiry
Locate relevant moments	`WHERE EVENT('death_started') AND task.igloo_complete = true ORDER BY step DESC LIMIT 1`	Find an event satisfying a state condition and select the latest match.
Follow up on a result	`WHERE step BETWEEN :t - 8 AND :t ORDER BY step ASC`	Use a retrieved step to examine the preceding sequence.
Select information	`SELECT action, task.igloo_blocks`	Request the fields needed to assess a particular question.
Choose a return format	`RETURN TEXT` `RETURN IMAGE` `RETURN MIXED`	Inspect state records, game frames, or both.

Conditions also support `OR` and `CHANGED` for alternatives and state changes; temporal ranges can extend before or after a retrieved step.

Active inquiry with TrajQL. The teacher starts from a selected step e in a recorded trajectory τ. Its initial context x contains the frame at e (or a text observation), essential state information, and brief instructions about the game, the inquiry task, and available interfaces. This entry provides a place to begin examining the student's behavior; it need not be the step at which that behavior should change. Through TrajQL, the teacher can locate moments of interest by querying events, state changes, and temporal conditions, then inspect the corresponding frames, clips, or state information. These queries can reach both earlier and later parts of the recorded trajectory.

TrajQL lets the teacher specify *where to look*, *what to inspect*, and *how the information is presented* through SQL-style queries. Table 1 summarizes these capabilities. A returned event can anchor the next query, allowing the teacher to locate a relevant moment, inspect its surrounding actions, and request additional fields or images as its understanding develops. Appendix B gives a complete multi-round example.

For example, a teacher observing a Frostbite player in the water can search for the most recent completion of igloo construction before sinking:

```
SELECT step, player.x, player.y FROM trajectory
WHERE CHANGED(task.igloo_blocks, 'increase')
  AND task.igloo_complete = true AND animation.sinking_progress = 0
  AND step < :e
ORDER BY step DESC LIMIT 1 RETURN TEXT;
```

The returned step anchors a follow-up query for nearby frames, allowing the teacher to inspect platform motion and consider revising an earlier action. A lightweight Python query engine parses each expression into filtering, ordering, and retrieval operations over the recorded trajectory. Event predicates match recorded occurrences, while change predicates compare consecutive states. Selected fields and associated images are returned with their step indices to support further inquiry.

Each response informs the teacher's next query:

$$q_j \sim \pi_\phi(\cdot \mid x, q_1, y_1, \ldots, q_{j-1}, y_{j-1}), \qquad y_j = \mathrm{TrajQL}(\tau, q_j), \tag{1}$$

where q_j is the query and y_j its returned information. Thus, the teacher can follow a newly discovered event, revisit an earlier interpretation, or examine the recorded consequences of a decision before proposing a change. Queries retrieve the student's original experience; the consequences of a proposed intervention are obtained by execution in Section 2.3.

Intervention decisions. After inquiry, the teacher selects a previously inspected step $t^\star$ and an instruction u describing how to change the student's behavior from that point. The intervention point is itself determined through inquiry, allowing the teacher to trace an observed failure back to a decision where a useful correction remains possible. The resulting intervention is executed, after which the student resumes control. The teacher may also leave the student's continuation unchanged.

A complete teacher rollout is the sequence $z = (q_1, y_1, \ldots, q_m, y_m, d)$, where d is the final decision, including $(t^\star, u)$ when intervening. Appendix B provides implementation details for the query and action interfaces, including the teacher's prompts.

2.3 Dual Reinforcement Learning

TrajTeach learns from interventions in two ways: their gains over the student's continuation train the teacher, while their transitions train the student.

Evaluating guidance through intervention. For each context x, we sample G independent inquiry-and-decision rollouts from the pre-update teacher $\pi_{\phi_{\text{old}}}$. Each intervention $(t^\star, u)$ is compared with a continuation of the fixed student from the same restored state at $t^\star$. The intervention branch executes u before returning control to that student. Both branches run for horizon H; averaging discounted return differences over K paired trials yields

$$R_\Delta(x, z) = \frac{1}{K} \sum_{k=1}^{K} \sum_{h=0}^{H-1} \gamma_\Delta^h \left(r_{k,h}^{\text{int}} - r_{k,h}^{\text{base}} \right), \tag{2}$$

where $r_{k,h}^{\text{int}}$ and $r_{k,h}^{\text{base}}$ are environment rewards with and without intervention, and γ_Δ discounts from the intervention step. Positive gain rewards useful guidance; leaving the student unchanged receives zero.

Learning to inquire and intervene. GRPO (Shao et al., 2024) compares rollouts from the same context. For usable samples $i \in \mathcal{I}_x$, define $A_i = (R_i - \mu_x)/(\sigma_x + \epsilon)$, where R_i is the optimization reward, μ_x and σ_x its group mean and standard deviation, and $\epsilon > 0$ a stability constant. Evaluated rollouts use $R_i = R_\Delta(x, z_i)$. The teacher maximizes

$$\mathcal{J}(\phi) = \mathbb{E}_x \left[\frac{1}{|\mathcal{I}_x|} \sum_{i \in \mathcal{I}_x} \frac{1}{N_i} \sum_{j=1}^{N_i} \min(\rho_{i,j} A_i, \text{clip}(\rho_{i,j}, 1-\eta, 1+\eta) A_i) \right], \tag{3}$$

where $\rho_{i,j} = \pi_\phi(\xi_{i,j} \mid c_{i,j}) / \pi_{\phi_{\text{old}}}(\xi_{i,j} \mid c_{i,j})$ is the probability ratio for generated token $\xi_{i,j}$ given context $c_{i,j}$, N_i counts generated tokens, and η is the clipping threshold. Queries and the final decision share the rollout advantage, training both inquiry and guidance; returned observations condition generation but are excluded from the loss.

Learning from intervention experience. Both positive- and negative-gain intervention transitions enter replay $\mathcal{D}$ with their environment rewards. Alongside ordinary experience, they train the Rainbow student (Hessel et al., 2018) through

$$\mathcal{L}_{\text{student}}(\theta) = -\mathbb{E}_{\mathcal{D}} \left[w_t \sum_{j=1}^{M} \widehat{p}_{t,j} \log p_{\theta,j}(o_t, a_t) \right], \tag{4}$$

where $p_{\theta,j}$ is the predicted probability at the jth of M fixed return values, $\widehat{p}_{t,j}$ the projected multi-step target, and w_t the prioritized-replay importance weight. Each game uses a separate teacher–student pair. Training alternates between student interaction and learning, and teacher inquiry, evaluation, and learning, holding the student fixed throughout each teacher phase. Only the student is deployed. Appendix A details paired execution, sample masking and protocol penalties, target construction, replay ingestion, and the update schedule.

3 Experiments

3.1 Experimental Setup

Environments. We select a diverse set of games spanning platforming, shooting, survival, crafting, and object-based exploration to evaluate our method across complementary control and reasoning challenges. Frostbite, Seaquest, and River Raid from the Arcade Learning Environment (Bellemare et al., 2013) emphasize moving-platform traversal, combat under oxygen constraints, and

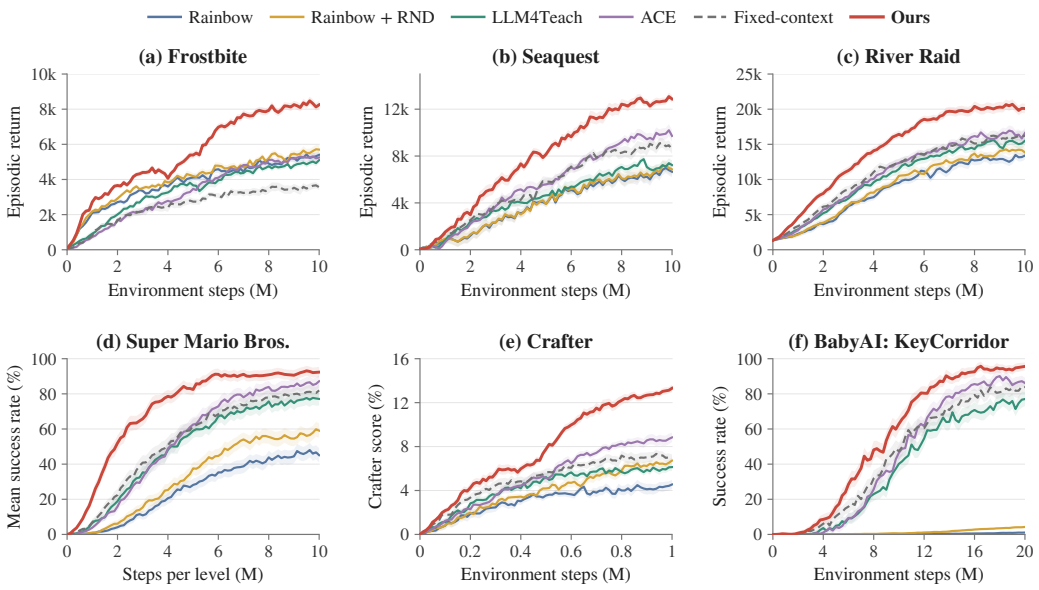


Figure 3: **Student learning dynamics.** Curves evaluate the student without teacher intervention. Mario averages three independently trained levels, with interaction steps counted per level.

shooting with fuel management, respectively. Super Mario Bros. (Hu et al., 2025), using levels 1-1, 1-2, and 1-3, adds obstacle traversal and coordinated movement. Beyond these action games, Crafter (Hafner, 2022) introduces dependencies among resources, tools, and crafting facilities, allowing us to examine whether the teacher can identify opportunities for progress within a student's past experience. We also include BabyAI's KeyCorridorS3R3 task (Chevalier-Boisvert et al., 2019) as an auxiliary environment for studying object retrieval and task prerequisites under partial observability. Together, these environments cover both local action timing and decisions whose interpretation requires connecting events across a trajectory.

Baselines. We compare value-based RL, intrinsic-motivation exploration, and teacher-guided learning. Rainbow (Hessel et al., 2018), our student backbone, provides the unassisted RL baseline; Rainbow + RND (Burda et al., 2019) adds a novelty-based intrinsic reward to encourage exploration. Among teacher-guided methods, LLM4Teach (Zhou et al., 2024) transfers the teacher's action guidance through policy distillation, while ACE (Wan et al., 2025) refines actions and rewards and fine-tunes the teacher on collected experience. We adapt both to Rainbow students with matched multimodal inputs and state access; Appendix C.7 details the adaptations from their original textual interfaces. Finally, a fixed-context variant of TrajTeach retains Dual RL and the same intervention interfaces, but receives frames and state descriptions selected from the trajectory by a predefined procedure. This comparison isolates the contribution of active inquiry.

Evaluation. Student evaluation uses held-out episodes with the teacher disabled. We report episodic return for the Atari games, achievement-based score for Crafter, and success rate for Mario and BabyAI. Evaluation aggregates independent training runs using interval estimates (Agarwal et al., 2021). Appendix C details the evaluation protocol, experience-ingestion controls, and resource accounting.

3.2 Student Performance and Learning Efficiency

Figure 3 shows higher final student performance across all six environments within the allocated interaction budgets. The advantage extends beyond unassisted exploration: our method also surpasses LLM4Teach, ACE, and the fixed-context variant. On Frostbite, the final return reaches 8,250, compared with 5,700 for the strongest baseline, Rainbow + RND, a 44.7% increase. Seaquest exhibits a similar advantage over a teacher-guided baseline, reaching approximately 12.9k return against

ACE's 9.7k. The relative ordering of the baselines, however, varies across tasks. LLM4Teach and ACE improve on Rainbow in Seaquest but finish slightly below it in Frostbite, illustrating that adding a teacher does not uniformly improve learning. Our method also finishes above the fixed-context variant in each plotted environment. This comparison is particularly relevant to active trajectory inquiry, since both variants retain Dual RL and the same intervention interfaces; the subsequent analyses examine how their guidance differs.

In Mario, our student reaches approximately 52.0% mean success after 2M steps per level, compared with 22.4% for the strongest baseline at that point, the fixed-context variant. This early separation narrows as other methods improve: the final success rates are 92.4% for our method and 87.2% for ACE. The per-level curves in Appendix C, Figure 8, show this early advantage across all three levels. Frostbite presents a different pattern: the gap widens during the latter half of training, with further improvement after the student has already acquired a partially effective policy. In BabyAI, teacher-guided methods progress beyond the prolonged low-success regime of Rainbow and Rainbow + RND, and our method rises earlier than the other guided approaches. Together, these patterns distinguish two benefits of guidance: accelerating the acquisition of useful behavior and improving performance later in training. Both are reflected in the student acting independently, with no teacher intervention during evaluation.

3.3 From Trajectory Inquiry to Better Decisions

Figure 4 connects observed problems to actionable corrections through different information needs. In Frostbite, locating igloo completion reveals that the remaining objective is a safe return; intervening before the fall reaches the igloo without sacrificing a life. River Raid requires three successive queries: locating a depot destroyed 237 steps before fuel exhaustion, establishing that the approach did not replenish fuel, and tracing firing to its onset. Delaying the shot permits refueling before destruction, raising return from 700 to 790 over the same 300-step horizon. Crafter and BabyAI instead require retrieving task prerequisites: a previously placed crafting table and a key matching a locked door. Mario's second query inspects controls around the retrieved jump onset and reveals an early button release, motivating a longer hold. In Seaquest, examining a diver's proximity to a shark supports choosing a safer rescue target. Across these cases, each finding helps determine what to inspect next and where to begin a useful correction, connecting distant events, missing prerequisites, and local action timing to changes in behavior.

3.4 Teacher–Student Learning Dynamics

Guidance as the student improves. To examine how guidance evolves during joint training, we compare the teacher's interventions with the current student's own continuations at successive training rounds. Figure 5(a) shows gains that generally rise and later narrow, with substantial differences across games in when they peak and how long they persist. This variation is consistent with different learning demands: guidance can help establish useful behavior early, while opportunities to improve more developed policies can emerge later. As student returns increase (Figure 3), diminishing paired gains suggest less remaining room for improvement over the student's own behavior. They need not indicate a weakening teacher: gains against fixed reference students continue to improve (Figure 5(b)). Together, these views connect the teacher's developing ability to help with the student's changing need for guidance. Appendix C.2 details the fixed-reference evaluation; Appendix C.1 further separates beneficial, harmful, and zero-gain interventions from decisions not to intervene.

3.5 Core Ablations

Figure 6 compares TrajTeach with four variants under matched interaction budgets on Frostbite, River Raid, and BabyAI: fixed context removes adaptive retrieval; frame-index inquiry allows repeated retrieval but removes content predicates; fixed intervention point requires corrections at the entry; and frozen teacher removes teacher updates. Repeated retrieval improves on fixed context, but content-based inquiry provides further gains: River Raid returns increase from 16,200 with fixed context to 17,640 with frame-index inquiry and 20,100 with TrajTeach. The importance of intervention timing varies across games. Fixing the intervention point reduces Frostbite return from 8,250 to 4,850, whereas BabyAI retains 94.1% success versus 95.6% for the full method, consistent with

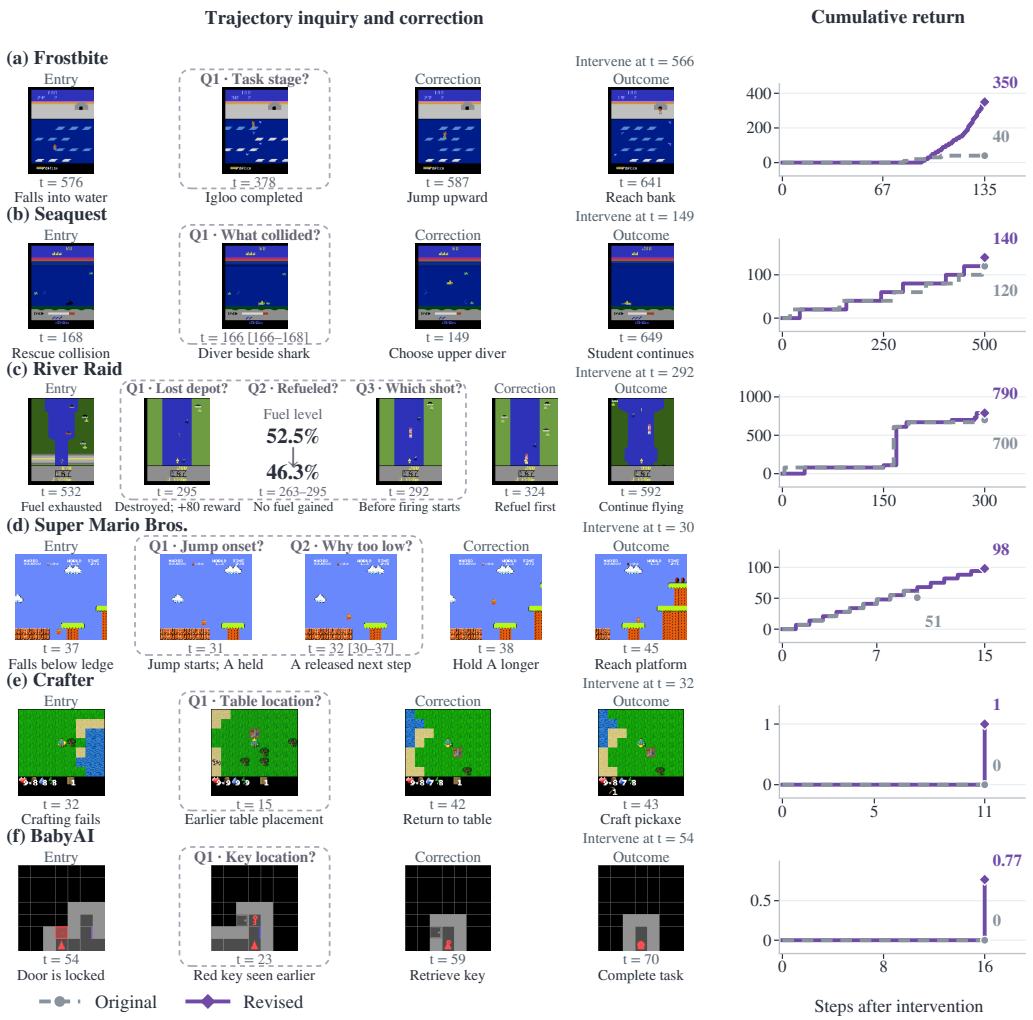
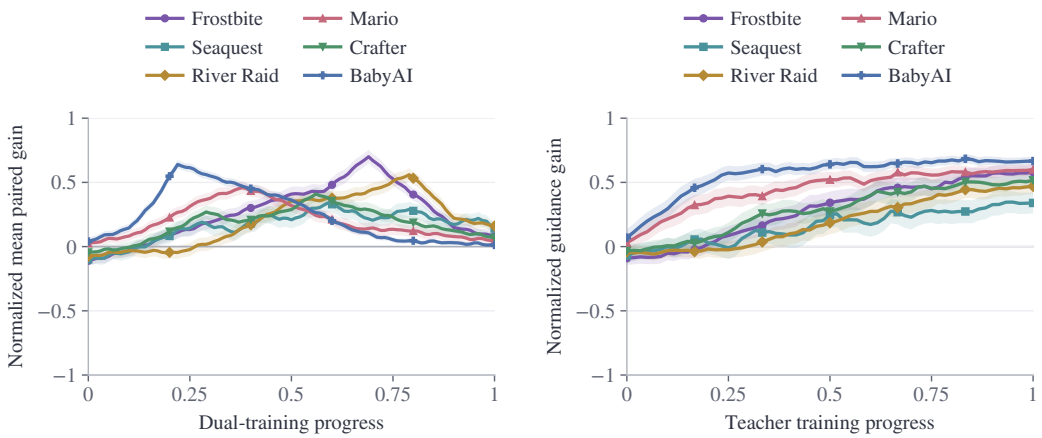


Figure 4: **From inquiry to correction.** Q1–Q3 mark successive queries; curves compare cumulative game rewards. Queries and replay details appear in Appendix C.8.

the distinction between revisiting an earlier mistake and addressing a prerequisite from the current state. Freezing the teacher reduces final performance and AUC in all three games; BabyAI's AUC falls from 55.0 to 48.1, indicating a contribution to early learning as well as final performance. Together, these comparisons distinguish locating useful information, selecting an actionable decision, and learning to provide guidance. Appendix C.4 specifies the controls; Appendices C.5 and C.6 examine action interfaces and retrieval budgets.

4 Related Work

Foundation models guide RL through exploration goals and rewards (Du et al., 2023; Klissarov et al., 2024; Sancaktar et al., 2025), action guidance and priors (Zhou et al., 2024; Yan et al., 2025; Bougie & Watanabe, 2025), and trajectory refinement (Wan et al., 2025). ACE additionally fine-tunes its teacher on collected experience. LEAP supplies its teacher with student trajectories and privileged task information to generate corrections (Choudhury & Sodhi, 2025); the information used for correction is assembled before the teacher's judgment. TrajTeach instead learns to acquire relevant information during the investigation, allowing successive findings to determine what to retrieve next. In visual game agents, VISTA provides lossless memory and tools for revisiting past frames and spatial regions to support the model's own play (Han et al., 2026). Our focus is



(a) Current students

(b) Fixed reference students

Figure 5: **Teacher guidance during learning.** Normalized mean paired gain against (a) the current students across dual-training rounds and (b) fixed reference students and problems across teacher checkpoints.

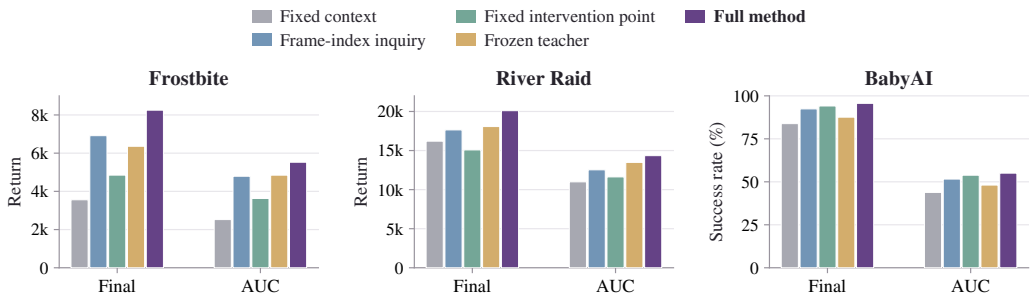


Figure 6: **Contributions of inquiry, intervention selection, and teacher learning.** Student-only final performance and horizon-normalized AUC. Exact values appear in Table 5.

guidance for a separate RL student: TrajQL locates experience through conditions on events, state changes, and entities, and the teacher selects where and how to intervene. The contribution lies in jointly learning this inquiry and intervention policy from gains over the student's continuation, while training the student on the resulting experience.

5 Conclusion

We presented TrajTeach, a dual reinforcement learning framework that enables a foundation-model teacher to learn how to guide an RL student through active trajectory inquiry. TrajQL allows the teacher to investigate recorded experience and decide whether, where, and how to intervene. Paired environment feedback trains the teacher's inquiry and guidance policy, while intervention transitions support student learning. The framework connects assistance during initial exploration with the refinement of established behavior, retaining only the student at deployment. Our central perspective is that effective teaching involves learning both how to understand the student's experience and how to change its behavior. Active inquiry makes these decisions part of the learning process.

AI Use Statement

We used generative AI tools to assist with language editing. The authors take responsibility for the final content of this work.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2204.14198.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the Atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 507–517. PMLR, 2020a. URL https://proceedings.mlr.press/v119/badia20a.html.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=Sye57xStvB.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. doi: 10.1613/jair.3912. URL https://arxiv.org/abs/1207.4708.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 449–458, 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

Nicolas Bougie and Narimasa Watanabe. Exploring Beyond Curiosity Rewards: Language-driven exploration in RL. In *Proceedings of the 16th Asian Conference on Machine Learning*, volume 260 of *Proceedings of Machine Learning Research*, pp. 127–142. PMLR, 2025. URL https://proceedings.mlr.press/v260/bougie25a.html.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2005.14165.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1810.08272.

Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: LLM agents that learn from privileged AI feedback. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1c60ed2b01120d383eebf12dc7a0e138-Paper-Conference.pdf.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8657–8677. PMLR, 2023. URL https://proceedings.mlr.press/v202/du23f.html.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021. doi: 10.1038/s41586-020-03157-9. URL https://www.nature.com/articles/s41586-020-03157-9.

Eran Eldar, Gaëlle Lièvre, Peter Dayan, and Raymond J. Dolan. The roles of online and offline replay in planning. *eLife*, 9:e56911, 2020. doi: 10.7554/eLife.56911. URL https://elifesciences.org/articles/56911.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Qiushi Han, Keya Hu, Linlu Qiu, Cathy Wu, and Kaiming He. VISTA: A visual harness for reasoning in an interactive world. Project blog, August 2026. URL https://vista-research.github.io/. Accessed September 19, 2026.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11796. URL https://ojs.aaai.org/index.php/AAAI/article/view/11796.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-Bench: How good are LLMs at playing games? *arXiv preprint arXiv:2505.15146*, 2025. URL https://arxiv.org/abs/2505.15146.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/a8de97fa77a09258b22880c230a83445-Abstract-Conference.html.

Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. Student-centered distillation narrows the agentic gap between small and large LLMs. *arXiv preprint arXiv:2509.14257*, 2026. URL https://arxiv.org/abs/2509.14257v3.

Marcelo G. Mattar and Nathaniel D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11):1609–1617, 2018. doi: 10.1038/s41593-018-0232-z. URL https://www.nature.com/articles/s41593-018-0232-z.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236.

Ida Momennejad, A. Ross Otto, Nathaniel D. Daw, and Kenneth A. Norman. Offline replay supports planning in human reinforcement learning. *eLife*, 7:e32548, 2018. doi: 10.7554/eLife.32548. URL https://elifesciences.org/articles/32548.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 2778–2787, 2017. URL https://proceedings.mlr.press/v70/pathak17a.html.

Cansu Sancaktar, Christian Gumbsch, Andrii Zadaianchuk, Pavel Kolev, and Georg Martius. SENSEI: Semantic exploration guided by foundation models to learn versatile world models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52745–52777. PMLR, 2025. URL https://proceedings.mlr.press/v267/sancaktar25a.html.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.05952.

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.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. URL https://mitpress.mit.edu/9780262039246/reinforcement-learning/.

Xu Wan, Wenyue Xu, Chao Yang, and Mingyang Sun. Think twice, act once: A co-evolution framework of LLM and RL for large-scale decision making. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 62102–62120. PMLR, 2025. URL https://proceedings.mlr.press/v267/wan25g.html.

Xue Yan, Yan Song, Xidong Feng, Mengyue Yang, Haifeng Zhang, Haitham Bou Ammar, and Jun Wang. Efficient reinforcement learning with large language model priors. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/797be96e4481c3fe5d675c1ba5352969-Abstract-Conference.html.

Zihao Zhou, Bin Hu, Chenyang Zhao, Pu Zhang, and Bin Liu. Large language model as a policy teacher for training reinforcement learning agents. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 5671–5679, 2024. doi: 10.24963/ijcai.2024/627. URL https://www.ijcai.org/proceedings/2024/627.

A Training Details

Paired evaluation and teacher batches. A teacher batch aggregates B initial inquiry contexts, with G independently sampled teacher rollouts per context. Each intervention is evaluated using K paired execution trials; G counts alternative inquiry and intervention proposals, whereas K counts repeated evaluations of a proposal. Both branches restore the same simulator state and student input history at the selected intervention step and use matched random seeds within a trial. The baseline is a fresh continuation of the fixed student checkpoint. The horizon and discount exponent both start at the intervention step, include all intervention actions, and continue through the handoff to the student. Terminated continuations contribute zero reward for the remaining horizon. The resulting return difference measures intervention value for that fixed student; student improvement after replay learning is a separate quantity.

Implementation and optimization. We implement multi-turn teacher training with verl 0.9.0 and vLLM 0.24.0. A custom agent loop executes trajectory queries and paired game continuations, while verl handles distributed policy optimization and synchronizes updated weights to vLLM. Table 2 gives the Frostbite training configuration. Each of the $G = 16$ rollouts is a complete inquiry-and-decision session; its intermediate model responses are not separate GRPO samples. The $K = 3$

Table 2: **Teacher training configuration.** Settings for the Frostbite recipe; query responses and image tokens count toward the session context limit.

Setting	Value
Teacher and optimization	
Initial teacher	Qwen3-VL-8B-Instruct
Framework / generation backend	verl 0.9.0 / vLLM 0.24.0
Questions per batch B / rollouts per question G	8 / 16 (128 sessions per batch)
Optimizer / learning rate	AdamW / 5×10^{-6}, constant
Adam coefficients / weight decay	$(0.9, 0.999)$ / 0.01
Optimization epochs per batch / gradient norm limit	1 / 1.0
Probability-ratio clipping η / dual-clip constant	0.2 / 3.0
KL penalty / entropy bonus	Disabled / 0
Group standard deviation / numerical offset	Population estimate / 10^{-6}
Language projection update rank / scaling	16 / 32
Parallelism / numerical precision	Four GPUs, FSDP2 / bfloat16
Generation and information access	
Sampling temperature / top-p	1.0 / 1.0
Session context / new tokens per response	16,384 / 768
Model responses / documentation requests per session	At most 10 / 6
Retrieved records / additional images per session	At most 24 / 3
Records / images per retrieval	At most 4 / 3
Retrieved text / documentation text	24,000 / 24,000 characters
Paired evaluation and alternating updates	
Paired trials K / continuation horizon H	3 / 256 agent steps
Teacher return discount γ	0.995
Primitive intervention length	At most 8 agent steps
Student collection per round / teacher batches per round	10,000 steps / 4 batches
Diagnostic recording length	At most 800 steps

paired trials evaluate a proposed intervention and do not generate additional teacher proposals. Each game has its own teacher, student, and optimizer state.

Only the rank-16 updates to language-model attention and MLP projections are optimized; the base weights and vision encoder remain fixed. We use gradient checkpointing, dynamic token batching with a 16,384-token budget per GPU, and parameter and optimizer offloading. The loss averages generated-token contributions within each session and then averages usable sessions within active question groups. Token-level rollout importance weights are capped at 2.0 to account for generation–training probability differences. Entropy is logged but receives no optimization bonus. The software stack uses PyTorch 2.11.0 and Transformers 5.6.2, with verl pinned to commit `483b8a0`.

Alternating schedule. The Frostbite continuation recipe initializes the student from its 1M-step Rainbow checkpoint. A round collects student experience, performs student updates, and records trajectories for teacher inquiry. The student snapshot remains fixed across four teacher batches, each containing eight questions and sixteen proposals per question. Evaluated intervention transitions are ingested before the following student phase. The stopping target is 100 effective teacher optimizer updates, with a cap of 200 attempted batches; a batch without an active reward group does not count as an optimizer update. These continuation settings are distinct from the from-scratch interaction budgets in Section 3.1.

Rainbow student objective. The Rainbow student in Equation 4 represents the return distribution with probabilities $p_{\theta,j}(o,a)$ on a fixed support $\{v_j\}_{j=1}^{M}$ (Bellemare et al., 2017). Its action values are $Q_\theta(o,a) = \sum_j v_j p_{\theta,j}(o,a)$. For an n-step replay item, define the accumulated reward $R_t^{(n)} = \sum_{h=0}^{n-1} \gamma_S^h \widetilde{r}_{t+h}$ and bootstrap discount $\Gamma_t = \gamma_S^n(1-b_t^{(n)})$. With $a^+ = \arg\max_a Q_\theta(o_{t+n}, a)$ selected by the online network, the target distribution is

$$\widehat{\mathbf{p}}_t = \Pi_{\mathbf{v}}\left[R_t^{(n)} + \Gamma_t Z_{\bar{\theta}}(o_{t+n}, a^+)\right]. \tag{5}$$

Here $\widetilde{r}_t$ denotes the student's preprocessed environment reward, $b_t^{(n)}$ indicates episode termination within the item, and w_t is the replay importance weight (Schaul et al., 2016). The target-network return distribution $Z_{\bar{\theta}}$ is projected onto the fixed categorical support by $\Pi_{\mathbf{v}}$. Targets are detached during optimization. The implementation combines this objective with Double-Q selection, dueling heads, noisy layers, prioritized replay, and multi-step returns (Hessel et al., 2018). For ordinary DQN, the one-step target instead uses the maximizing action of the target network. In all cases, the student receives environment rewards under the preprocessing used by its unassisted baseline.

Replay ingestion. We restore each evaluated intervention and verify the observation and reward sequence before ingestion. The first, predetermined paired trial supplies the intervention transitions; repeated evaluation trials do not multiply their replay weight. Positive and negative interventions enter ordinary replay. Multi-step accumulation follows the executed sequence across the controller-to-student handoff and stops at episode boundaries; incomplete suffixes use their actual length and bootstrap discount. An optional protected replay pool retains short prefixes of robustly positive interventions, mixed with ordinary replay at a configured fraction. Positive selection checks the predetermined trial, the mean paired gain, and the fraction of positive trials; task-specific terminal success can also qualify an experience. This retention mechanism changes sampling, not transition rewards.

Missing evaluations and protocol failures. Let $\mathcal{M}_x$ contain measured paired rewards, including the zero reward assigned to valid abstention. A session that exhausts its call or context budget after beginning generation receives optimization reward

$$R_{\text{budget}} = \min_{i \in \mathcal{M}_x} R_i - \max(\text{Std}_{i \in \mathcal{M}_x}(R_i), 1)\,. \tag{6}$$

For an empty measured set, the minimum and standard deviation are defined as zero, giving a penalty of -1. This is a protocol penalty: its environment-derived R_Δ remains unavailable and is not reported as a measured game return. Other unevaluable samples are masked individually. Group statistics include measured rewards and protocol penalties but exclude masked samples. A group contributes no update if fewer than two usable samples remain or its reward standard deviation is below the numerical threshold. Policy losses are normalized over active groups and usable sessions; tool observations are never treated as generated policy tokens.

State access, restoration, and scope. State adapters expose game-specific fields through a shared schema with field descriptions and provenance. Missing information remains unknown. A result count or a tool description alone does not authorize an intervention at an unseen frame; the selected step must have been delivered as observation evidence and pass the intervention eligibility checks. The recording available for queries can extend beyond the allowed intervention window. Both evaluation branches restore the selected environment state and the student's observation stack or equivalent input history. Controller actions consume the same evaluation horizon as student actions; returning control does not reset the discount exponent. Simulator calls used for restoration, controller search, and paired evaluation are additional training computation and should be accounted for separately from ordinary student interaction.

B TrajQL Interface and Inquiry Details

Automatic recording and state extraction. Each recorded observation is stored with a step index, the action and reward leading to it, and the game information captured during rollout. Game-specific state adapters implement mappings from engine interfaces or decoded emulator memory to named fields and entities. Once these mappings are implemented, extraction and recording run automatically without per-frame manual annotation. Events can be derived from changes between successive states. Field descriptions document their meanings and provenance; unsupported or uncertain values remain unknown. In Frostbite, for example, the adapter detects sinking from its animation counter but does not infer positive controllability merely from the absence of sinking. Figure 7 therefore uses construction progress and sinking state to retrieve a candidate for further inspection.

Trajectory storage. Each trajectory occupies a directory containing `metadata.json`, observation records in `records.jsonl`, RGB images in `frames/`, and periodic environment snapshots

in `snapshots/`. Metadata identifies the environment and its supported fields. Each record stores the observation step, incoming action and reward, structured state information, and image path. The executor loads the structured records for predicate evaluation and resolves the image paths when visual output is requested. State snapshots support restoration for intervention evaluation; restoration and prefix-replay checks are described in Appendix A. Trajectory records and snapshots serve inquiry and restoration, while the replay buffer supplies transitions for student updates.

Entry selection and temporal scope. Entry steps are sampled from salient events, sustained gaps in reward progress, and ordinary exploration, with temporal spacing to reduce redundancy. The selection reason accompanies the initial observation. Training-time retrospective inquiry can access the original recording both before and after the entry. An execution configuration specifies the permitted intervention range and eligible steps; the final choice must have been delivered to the teacher and support environment restoration. These execution constraints are separate from the entry used to initialize inquiry.

Query execution. The TrajQL parser accepts a read-only SQL-style grammar with `SELECT`, `FROM trajectory`, `WHERE`, `ORDER BY step`, `LIMIT`, and `RETURN`. Conditions support comparisons, `IN`, `BETWEEN`, nested `AND`/`OR`, event predicates, attribute changes, and entity predicates. Named parameters such as `:c` bind previously retrieved step indices; numerical offsets specify relative time ranges. Parsing produces a typed query, whose fields and operations are checked against the supported schema and the current game's capabilities. A Python executor evaluates the conditions over the permitted records, orders and limits the matches, and retrieves the requested fields and images. This execution requires no external database service. Unsupported expressions produce an error rather than being interpreted as executable code.

For predicate p_q, requested field set F_q, temporal order ω_q, result limit L_q, and presentation mode m_q, record retrieval is

$$S_q = \mathrm{Top}_{L_q}\big(\mathrm{Sort}_{\omega_q}\{v_t \in \tau : p_q(v_{t-1}, v_t) = \mathrm{true}\}\big)\,, \tag{7}$$

$$y_q = \mathrm{Render}_{m_q}\Big(\mathrm{Project}_{F_q}(S_q)\Big)\,. \tag{8}$$

Here $v_t \in \tau$ denotes the query record associated with step t of the recorded trajectory. Predicates on attribute changes compare consecutive records. Unknown values are distinguished from false matches. Image responses include essential lifecycle metadata; text and mixed responses can project requested state fields. Queries for local clips expand around matching steps, while counts, existence checks, and summaries operate over the authorized matching records. Query, record, image, and context budgets bound each session, and returned records identify their observation steps explicitly.

Initial context and progressive documentation. The initial prompt provides task instructions, the entry observation and essential state metadata, the selection reason, the permitted observation scope and intervention window, and a compact catalog of query and action interfaces. Detailed documentation is loaded on demand through `describe`, covering query operations, state fields, entity fields, and action tools. Descriptions specify supported parameters, field meanings, and execution constraints. Documentation responses consume their own budget and do not count as observations of the game. This keeps the initial interface description compact while allowing the teacher to obtain the details needed for a particular query or controller goal.

Action interfaces. The teacher can supply a primitive-action sequence or invoke a predefined action tool with parameters specifying a local goal. Tools expose their supported parameters and execution limits through the same documentation interface. Simple tools, such as holding a direction or moving toward a coordinate, use state feedback to select primitive actions. Game-specific tools include returning to a bank in Frostbite and surfacing in Seaquest. Some use bounded simulator search or candidate-policy evaluation to find an action sequence for the requested goal. In particular, Frostbite's `reach_bank` searches candidate continuations in a separate simulator and verifies the selected plan before execution. The teacher chooses the intervention step and instruction; the tool handles local execution. Execution stops when the goal is reached, a failure condition is detected, or its budget is exhausted, and control returns to the student. Tool success does not imply a positive intervention gain: all executed interventions are evaluated using the paired return difference in Section 2.3. Simulator search and verification are additional training computation.

ENTRY e Player observed in the water. *Could an earlier decision help?*

1 Locate the onset of the failure

```
SELECT step, player.phase, animation.sinking_progress
FROM trajectory
WHERE EVENT('death_started') AND step BETWEEN :e - 32 AND :e + 32
ORDER BY step ASC LIMIT 1 RETURN TEXT;
```

The returned death step `:d` anchors the search for an earlier opportunity.

2 Trace the completion of construction

```
SELECT step, task.igloo_blocks, animation.sinking_progress
FROM trajectory
WHERE step < :d AND step < :e
  AND CHANGED(task.igloo_blocks, 'increase')
  AND task.igloo_complete = true AND animation.sinking_progress = 0
ORDER BY step DESC LIMIT 1 RETURN TEXT;
```

If a candidate `:c` is found, examine the surrounding motion before deciding how to act.

3 Examine the approach and platform motion

```
SELECT frame, player.x, player.y, task.igloo_blocks
FROM trajectory
WHERE step BETWEEN :c - 3 AND :c + 3
ORDER BY step ASC LIMIT 7 RETURN MIXED;
```

DECISION Consider returning toward the bank from $t^\star = c < e$, based on the retrieved frames. The teacher can also continue querying or leave the student's behavior unchanged.

Figure 7: **From an entry observation to a targeted intervention.** An illustrative Frostbite inquiry, shown in SQL notation. Returned steps `:d` and `:c` guide subsequent queries; images reveal local motion alongside the selected state fields. Earlier construction progress identifies a candidate for visual inspection, while the entry e and proposed intervention $t^\star$ remain distinct.

B.1 FROSTBITE TEACHER PROMPT

The prompt combines the teacher's task, game rules, and TrajQL interface with a session-specific entry observation and tool catalog.

ROLE AND OBJECTIVE

Help the student improve its game return during training. The entry frame starts your investigation; it is not necessarily a mistake or the right place to intervene. Investigate relevant experience, then propose a useful local change or leave the student in control. Intermediate progress counts. Your proposal is evaluated against the student's continuation from the same state; their return difference provides feedback. The student resumes control after your intervention.

FROSTBITE RULES

Cross moving ice floes to build the igloo on the upper-right bank. Landing on white ice advances construction and turns it blue; blue ice remains usable. Entering the completed igloo yields additional reward. Water and hazards can kill the player, and temperature decreases over time. Directions control movement and jumping. FIRE reverses the occupied ice row and may remove an igloo block before completion; it is not the jump button. Temperature, score, and remaining lives are distinct HUD values.

ACTIVE INQUIRY WITH TRAJQL

Identify what is unclear, query information that could change your judgment, and let each result guide the next query. Search by events, state changes, or objects; inspect local sequences when motion or action timing matters. Select the fields you need and return TEXT, IMAGE, or MIXED. Request documentation for unfamiliar fields or tools from the supplied catalog.

For example, locate a construction milestone before the entry `:e`, then inspect motion around the returned step `:c`:

```
SELECT step, task.igloo_blocks FROM trajectory
WHERE step < :e AND CHANGED(task.igloo_blocks, 'increase')
ORDER BY step DESC LIMIT 1 RETURN TEXT;

SELECT frame, player.x, player.y FROM trajectory
WHERE step BETWEEN :c - 1 AND :c + 1
ORDER BY step ASC LIMIT 3 RETURN MIXED;
```

These queries show the syntax; choose conditions that address the current problem. Earlier and later records describe the original run, not the effects of your proposal. At step t, the recorded action and reward belong to $t-1 \to t$; your intervention starts with $t \to t+1$.

INTERVENTION AND FINAL RESPONSE

Choose an inspected, eligible start before, at, or after the entry. If the entry is already unrecoverable, investigate an earlier actionable state. Missing state values mean unknown; remaining lives alone do not establish control. Use a legal primitive sequence of at most eight steps, or an enabled action tool within its documented limits. Stop querying once you can decide; abstain when no useful change is supported.

Report: **decision** (intervene or abstain), **start step**, **actions or tool and parameters**, **supporting observation steps**, and a brief **diagnosis and expected effect**. For abstention, omit the start and instruction. Distinguish observed facts from predicted outcomes.

SESSION INPUT

Entry image and state record; sampling reason; searchable trajectory and eligible start steps; legal actions and enabled tools; field and tool summaries; remaining query, image, record, text, and model-call budgets. Follow these limits throughout the session.

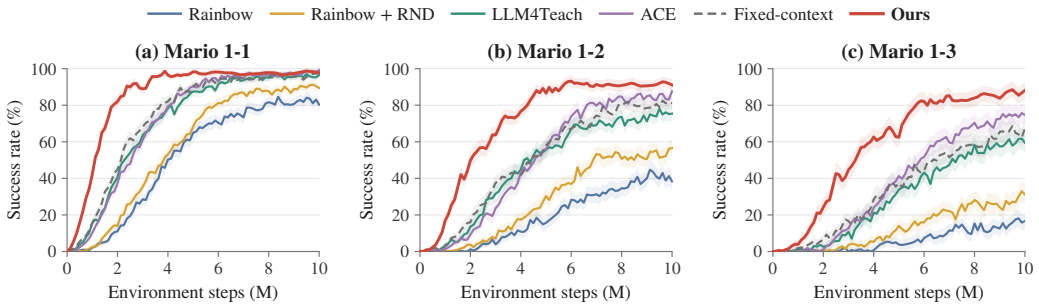


Figure 8: **Mario learning dynamics by level.** Each level is trained independently. The main Mario panel averages these level-wise success rates at matched per-level interaction counts.

C Evaluation Protocol and Additional Comparisons

Independent runs and held-out data. The formal comparison targets five independent training seeds per method and game. Each task and seed defines an independent teacher–student pair, with separate optimizer states and replay stores. Student evaluation episodes are excluded from both replay and teacher diagnosis. The diagnostic archive uses disjoint episodes for development and evaluation, with early, intermediate, and established student checkpoints represented as separate strata. Confidence intervals resample independent training runs; diagnostic intervals cluster by source episode and student checkpoint. Evaluation episodes and multiple proposals for one problem are not independent training seeds; proposals share their source problem's evaluation unit.

Mario level-wise evaluation. We train independent teacher–student pairs on Super Mario Bros. levels 1-1, 1-2, and 1-3. At each matched per-level interaction count, the main-panel success rate is the arithmetic mean of the three level-wise success rates. Thus, 10M steps on the main Mario axis corresponds to 10M steps per level, or 30M student interaction steps across the three levels.

Two initialization regimes. From-scratch results use a fixed total per-game interaction budget B_g. Continued-training results use a common unassisted checkpoint per seed, with its original training cost and additional budget B_g^{add} reported separately. Teacher-assisted continuations initialize their teachers from the same pretrained model. A continuation from an already assisted checkpoint is a separate condition. Table 3 compares only methods sharing the same Rainbow checkpoint and history; it does not compare independently initialized DQN variants to a pretrained Rainbow.

Table 3: **Continued student training from common unassisted checkpoints.** Return and evaluation-curve area use student-only episodes over the same additional interaction budget of 10M simulator calls. Initial checkpoint returns are 1,824 (Frostbite) and 1,488 (Seaquest). AUC is divided by the additional interaction horizon.

	Frostbite		Seaquest	
Method	Return	AUC	Return	AUC
Rainbow continuation	1,958	1,869	2,568	2,148
Matched branch sampling	2,134	1,948	2,718	2,294
Frozen TrajQL teacher	3,128	2,426	3,284	2,637
TrajTeach	4,276	3,074	4,128	3,216

Evidence and intervention controls. Teacher conditions share a pretrained model, game card, initial entry, candidate count, action vocabulary, goal-controller library, observation scope, and intervention window. Fixed full-recording context uses a prespecified mixture of uniform and event-based selection. Active frame-index retrieval returns the same projected fields and images available to TrajQL, with only the predicate language restricted. Primitive-only execution retains the same total action-duration cap as controller execution. All methods report both available and consumed

evidence budgets. Teacher adaptation comparisons use the same repeated-proposal and paired-evaluation opportunities for frozen and trainable teachers. Both the ordinary replay schedule and any protected replay mechanism are held fixed across the corresponding experience-ingestion controls.

Performance and behavioral metrics. Final student performance uses a prespecified evaluation checkpoint rather than the maximum over all checkpoints. Learning-curve area is the integral of student-only evaluation performance divided by the common interaction horizon. For teacher evaluation, positive and negative rates use all attempted sessions as their denominator; failed evaluations are separately counted and are not assigned fictitious game rewards. Mean paired gain is computed over measured sessions, including valid zero-gain abstentions, and is always accompanied by the measured fraction. Raw and discounted paired gains are reported separately. Abstention, requested intervention, and actual controller execution are distinct outcomes. Best-of-group gain, if included, is a search statistic rather than single-proposal performance.

Simulation and computational cost. We count all actual simulator calls used for student collection, diagnostic trajectory collection, restoration, controller search, and paired evaluation. Cached baseline continuations are charged when executed, with cache reuse recorded separately. Standard held-out evaluation is logged as evaluation cost and never enters replay. Teacher tokens and aggregate GPU time are reported separately from simulator calls. Wall-clock comparisons include teacher generation and optimization, and identify hardware, concurrency, and student update-to-data ratios. In particular, repeated controller-search or Monte Carlo trials do not enter replay multiple times merely because they were executed during evaluation.

Table 4: **Frostbite training-resource accounting for one seed.** Online steps plus extra simulator calls total 10M per method. Token counts include documentation and tool-context processing. GPU-hours sum active time across devices; wall time is in hours (M = millions).

Method	Online steps	Extra simulator calls	Teacher tokens	GPU-hours	Wall time (h)
Rainbow	10.0M	0	0	3.6	3.8
Matched branch sampling	8.0M	2.0M	0	5.7	4.9
Fixed-context teacher, frozen	8.0M	2.0M	318.4M	34.0	10.4
TrajQL teacher, frozen	8.0M	2.0M	246.8M	29.8	9.6
Fixed-context teacher, GRPO	8.0M	2.0M	326.1M	67.2	19.0
TrajTeach	8.0M	2.0M	258.6M	62.4	18.3

C.1 Guidance Relative to the Current Student

How broadly does guidance help? Figure 9 separates positive, negative, and zero-gain interventions from decisions not to intervene. Across the displayed stages, positive interventions first become more frequent and then recede to different extents. In BabyAI, their share falls from 48% at the middle checkpoint to 10% late in training, while decisions not to intervene reach 65%. Mario and Crafter show a similar shift toward retaining the student's behavior. Frostbite, Seaquest, and River Raid retain larger late-stage shares of beneficial interventions, accompanying the remaining gains in Figure 5(a). Negative interventions also become less frequent across these comparisons. A falling mean gain can therefore accompany fewer harmful changes and more decisions to leave the student in control. The outcome composition distinguishes this pattern from a mean near zero caused by frequent positive and negative gains canceling each other. Additional query-behavior analysis appears in Appendix C.3.

A changing student reference. At round k, paired execution compares the teacher's proposed intervention with the continuation of the student snapshot π_{θ_k}. This snapshot is held fixed during the paired branch evaluations, and replaced by the updated student snapshot for the next round. The difference is R_Δ in Equation 2; a fixed earlier student is not substituted for the current one. Query budgets, branch horizons, and discount settings stay fixed within each game. Problems are drawn from trajectories of the current student using the same selection procedure across rounds. Consequently, both the student reference and the available trajectory problems evolve, and this curve

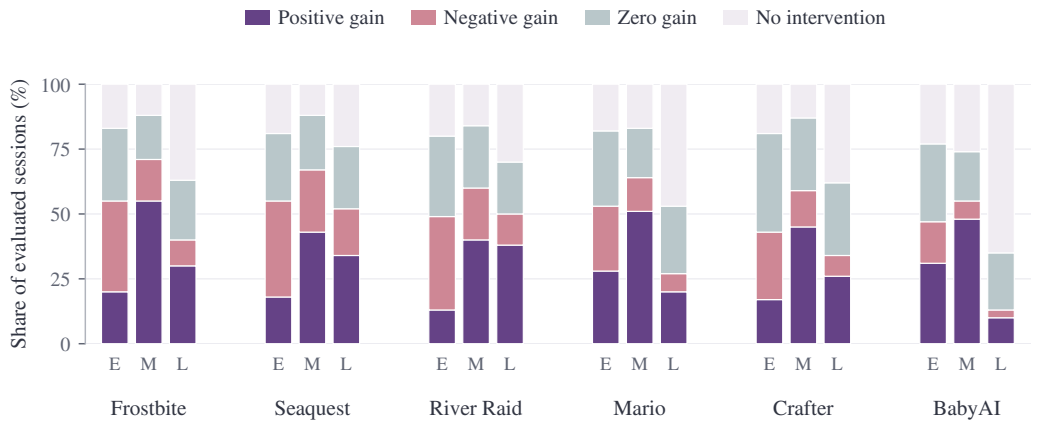


Figure 9: **Intervention outcomes over dual training.** E, M, and L denote the early, middle, and late checkpoints. Bars distinguish three intervention outcomes from decisions not to intervene; each sums to 100% of evaluated sessions.

tracks guidance in the changing training setting. The student interaction count associated with each round provides alignment with student-only evaluation.

Mean gain and training progress. The horizontal axis is the completed dual-training round divided by the scheduled round budget. Mean gain includes every session with an evaluated paired return, including negative and zero gains and valid decisions not to intervene, which receive zero. Unevaluated sessions are recorded separately with the evaluated fraction. Figure 5(a) uses the signed normalization in Equation 9, with the same fixed per-game scales as the reference-student comparison. Normalization follows averaging and does not change the reward supplied to teacher training. Gains are averaged per independent run before runs are combined with equal weights; Mario levels also receive equal weights.

Intervention outcome composition. The three snapshots in Figure 9 correspond to 10%, 50%, and 90% of the scheduled rounds, labeled early, middle, and late. They use the same student reference and problem population as the paired-gain curve at each selected round. Positive, negative, and zero-gain interventions are classified by the sign of their evaluated raw R_{Δ}. A valid decision not to intervene forms a separate category rather than being combined with an executed zero-gain intervention. Execution or evaluation failures are recorded separately; missing returns are never assigned zero. Within each run and stage, percentages use all evaluated sessions as their denominator, including valid non-intervention decisions, and sum to 100%. Percentages are then averaged across runs. The panel reports the composition among evaluated sessions, not an assumed evaluation-completion rate.

C.2 Teacher Guidance against Fixed Reference Students

Guidance quality. The curves in Figure 5(b) rise from near-zero or negative guidance gains to positive gains across all six games. Paired gain measures the improvement over the student's own continuation, so crossing zero marks the transition from neutral or harmful guidance to a net benefit. The progression differs across games. BabyAI and Mario show relatively early gains, with BabyAI approaching a plateau sooner. River Raid improves more slowly at first before gaining momentum later, while Seaquest follows a less regular trajectory with repeated setbacks. Together, these curves depict improving guidance as the teacher learns from intervention feedback. The reference students remain fixed across teacher checkpoints, so these curves isolate guidance against a common set of student policies.

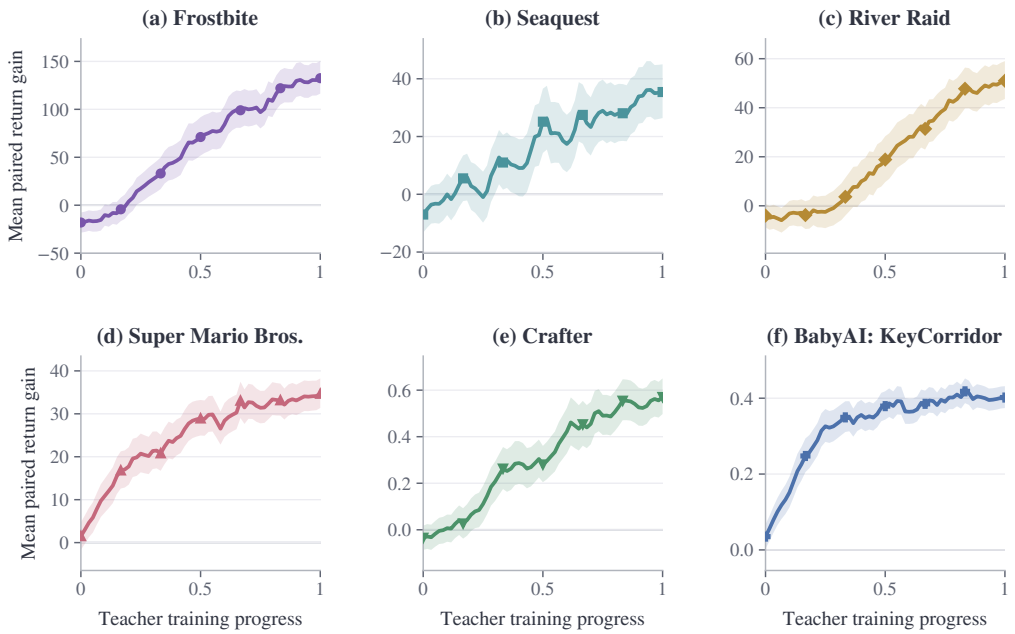


Figure 10: **Guidance gains in the original reward units.** Per-game views of Figure 5(b); zero denotes no improvement over the student's continuation. Vertical scales differ across games.

A common evaluation reference. Teacher checkpoints are compared on the same held-out problems, using their associated frozen student checkpoints for both branches. Early, intermediate, and established student policies remain represented with fixed weights, so a change in the teacher curve is not driven by an improving comparison student or a changing problem mixture. The protocol holds generation settings, the number of proposals per problem, query budgets, action interfaces, and paired-rollout settings fixed. Each proposal contributes its paired gain; selecting only the best proposal would instead measure search performance. Valid non-intervention decisions contribute zero. Missing evaluations remain missing and require a separately reported completion fraction, rather than an invented zero reward.

Display normalization. Let $\overline{R}_{\Delta,g}(k)$ be the mean evaluated paired gain for game g at checkpoint k. Both the current-student and fixed-reference panels use

$$\widetilde{G}_g(u) = \tanh\left(\frac{\overline{R}_{\Delta,g}(k)}{c_g}\right), \qquad u = \frac{k}{K_g}, \tag{9}$$

where K_g is the relevant training budget (dual rounds in Figure 5(a), teacher updates in Figure 5(b)) and $c_g > 0$ is a fixed per-game display scale. The scales are 200 for Frostbite, 100 for Seaquest, 100 for River Raid, 50 for Mario, 1 for Crafter, and 0.5 for BabyAI. Each scale remains constant along its curve. The transformation is applied after averaging paired gains; it maps them into $(-1, 1)$, preserves their sign and zero point, and does not force either endpoint to the axis limits. It is used only for presentation, without changing the teacher's optimization reward or its within-group GRPO advantages. Absolute vertical differences between games depend on these reference scales and are not a ranking of task difficulty. Figure 10 retains the original gain units.

Reward interpretation. These gains use the branch-return definition in Equation 2, including its evaluation horizon and discount. They are not final episodic scores or student success rates. In particular, Mario uses its environment reward for branch comparisons, while its student benchmark reports level completion; Crafter's branch reward is distinct from its aggregate achievement score. BabyAI's terminal reward reflects both success and completion time. A short horizon can omit a downstream benefit or cost, such as a later fuel failure in River Raid, even when a query correctly

identifies the preceding decision. Evaluation therefore fixes the horizon within each game and reports it with the paired-return settings.

Task-dependent guidance. The games place different demands on improving guidance. In BabyAI, identifying and completing a reusable key–door prerequisite can turn an unsuccessful continuation into a rewarded one; such progress in teacher guidance is distinct from the student's slower acquisition of a policy from scratch. Mario also offers local corrections, although precise action timing can still limit their execution. Frostbite and Crafter require connecting task progress to the next useful behavior, with rewards arriving at discrete construction or achievement milestones. In River Raid, guidance must account for the later consequences of an earlier fuel-management decision, making the evaluation horizon especially important. Seaquest combines rescue rewards with oxygen and collision constraints, so increasing one source of reward can come at the expense of another. These differences motivate examining both the direction and consistency of guidance gains, alongside the queries and intervention choices that produce them.

C.3 Inquiry Behavior

Changes in inquiry. Figure 11 contrasts two changes in TrajQL use: how the teacher locates relevant experience and how many rounds it spends investigating a problem. Conditional retrievals become more frequent across games, while inquiry depth changes in both directions. River Raid and Frostbite pair more condition-based searches with additional rounds; BabyAI and Crafter combine more targeted retrieval with fewer rounds. Mario retains a larger role for inspecting local temporal intervals. The comparison therefore distinguishes locating relevant moments from deciding how far to pursue an investigation. The counting rules and matched-checkpoint comparison are specified below.

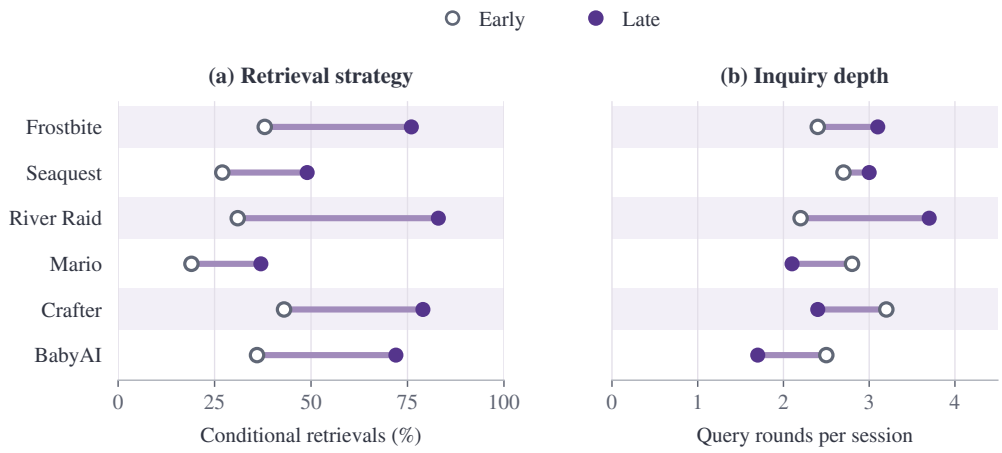


Figure 11: **Changes in teacher inquiry behavior.** Connected points compare early and late teacher checkpoints on a fixed problem set. Left: the share of retrievals using event, object, action, or state conditions. Right: mean query rounds per session.

Matched teacher checkpoints. The comparison uses the teacher before its first update and at its final update, denoted *early* and *late*. Both checkpoints receive the same held-out problems, initial observations, trajectory access, and query budget. Reference student checkpoints and generation settings remain fixed. Metrics are calculated independently for each training seed and then averaged across seeds, with equal weighting. Separate Mario level results are averaged with equal weights. This protocol keeps the question mixture constant while comparing how the teacher uses the query interface.

Conditional retrievals. A retrieval is conditional when its predicate contains an event, object, action, or non-temporal state condition. Examples include a death event, an observed key of a

specified color, a firing action, or a decrease in fuel. A query restricted only by step indices, a temporal interval, sorting, or a result limit does not qualify. Selecting state fields for return does not by itself make a query conditional. A query combining a state condition with a time bound qualifies once. The percentage divides the number of qualifying executed retrieval statements by the total number of executed retrieval statements at that checkpoint. Successfully executed queries returning no matches remain in this denominator. If no retrieval is executed, the ratio is undefined and the no-query frequency is reported separately.

Query rounds. A round is one teacher response containing at least one executed TrajQL retrieval. Several retrieval statements issued in the same response count as one round; several frames returned by one retrieval also count as one round. Schema documentation requests, final decisions, and action execution do not count as retrieval rounds. The mean includes every initiated session, including sessions that end without intervention or use zero queries. Failed sessions retain the rounds already executed and are separately recorded by status. Invalid statements are recorded separately from executed retrievals, so query counts and error counts remain distinguishable. These definitions allow the two panels of Figure 11 to distinguish the choice of retrieval conditions from the depth of interaction.

C.4 Core Ablation Protocol

Table 5: **Contributions of inquiry, intervention selection, and teacher learning.** Final performance and horizon-normalized AUC use student-only evaluation. Frostbite and River Raid report returns; BabyAI reports success rates (%). Higher is better.

	Frostbite		River Raid		BabyAI	
Variant	Final	AUC	Final	AUC	Final	AUC
Fixed context	3,559	2,522	16,200	10,992	83.8	43.8
Frame-index inquiry	6,915	4,791	17,640	12,531	92.4	51.6
Fixed intervention point	4,850	3,627	15,080	11,626	94.1	53.8
Frozen teacher	6,360	4,847	18,070	13,484	87.6	48.1
Full method	**8,250**	**5,525**	**20,100**	**14,346**	**95.6**	**55.0**

Shared training conditions. The variants in Table 5 share the student initialization for each seed, pretrained teacher, trajectory-selection procedure, game instructions, available state fields, and per-game interaction budget. Replay ingestion, student updates, teacher proposal counts, paired-evaluation horizons, and action interfaces remain fixed unless they are the component being removed. Each variant collects trajectories from its own evolving student; forcing all variants to reuse the full method's trajectories would change the training problem. The fixed-context and full-method entries use the corresponding runs from Section 3.2. Freezing the teacher does not reallocate its unused optimization cost to additional student interactions; teacher computation is reported separately.

Information and intervention controls. Fixed context uses the same predefined mixture of uniform and event-based record selection as the main baseline. Frame-index inquiry can retrieve any eligible step, temporal interval, or local clip, and can choose the returned fields and modality. It cannot select a record by its content or request an event-conditioned aggregate. The initial observation, searchable recording, and delivery limits are otherwise shared with TrajQL. Both variants can select an intervention point among their inspected, restorable steps. Fixed intervention point instead requires $t^\star = e$, while leaving the full recording available for inquiry. When the entry is already beyond recovery, the teacher may decline to intervene. These entries remain in the comparison, preserving the distinction between noticing a problem and locating a useful correction.

Student outcomes. Final performance uses the last scheduled student evaluation rather than the best checkpoint. For evaluation curve $J_g(b)$ over cumulative training interactions b and budget B_g, the learning-efficiency measure is

$$\mathrm{AUC}_g = \frac{1}{B_g} \int_0^{B_g} J_g(b)\, \mathrm{d}b, \tag{10}$$

computed by trapezoidal integration over scheduled evaluations. It retains the units of the game's reported score; BabyAI final performance and AUC are both expressed in percentage points. Interaction counts include branch evaluation, restoration, and controller search. Independent training seeds, rather than individual proposals or evaluation episodes, are the units for interval estimates. Learning curves and local paired gains are retained per seed before aggregation.

C.5 Action Interfaces across Games

Different demands on execution. The action interface serves different purposes across our environments (Table 6). In Frostbite and Seaquest, a correct local objective can still require coordinating several actions as the scene changes. BabyAI presents a different separation: choosing an object and satisfying its prerequisites are distinct from navigating to that object through observed space. Mario's button timing, River Raid's firing decisions, and Crafter's discrete crafting actions need not introduce a separate goal controller. We therefore distinguish ordinary action sequences from tools that select actions using state feedback. Repeating a specified primitive action is available in both conditions and does not by itself count as a goal controller.

Table 6: **Action demands differ across environments.** The teacher selects the correction and its starting point; an optional controller handles bounded execution.

Game	Decision made by the teacher	Role of the action interface
Frostbite	Choose a safe return or a target floe after examining progress and motion.	Feedback or bounded search coordinates crossings toward the chosen local goal.
Seaquest	Choose a rescue target or when to surface.	Local control pursues that goal while the scene and oxygen state change.
BabyAI	Identify the relevant object and next task prerequisite.	Navigation follows observed traversable cells to the selected target; it does not choose the task solution.
Mario	Choose jump onset, direction, and button duration.	Primitive button sequences express the timing correction directly.
River Raid	Choose when to fire or withhold fire, and where to steer.	Primitive controls can express the resource-preserving correction.
Crafter	Choose a resource, facility, or crafting operation.	Native movement and interaction actions express the chosen steps.

Crossing inquiry with execution. We compare frame-index inquiry and TrajQL under two interfaces: primitive sequences only, and primitive sequences plus the game's predefined controllers. The factorial comparison covers Frostbite, Seaquest, and BabyAI, where distinct goal controllers are available. It does not assign a controller condition to another game merely by renaming a primitive sequence. Table 7 reports guidance gain separately for each game; there is no assumption that control assistance should contribute equally to continuous motion and grid navigation.

For this local comparison, each teacher is adapted under its assigned query and action interface from the same pretrained initialization, with equal proposal and optimization budgets. Adaptation mixes retrieval allowances of 8, 16, and 32 records, while Table 7 evaluates all conditions with an allowance of 16. Adaptation and held-out evaluation use separate problem banks built from common reference student checkpoints spanning early and later learning. Reference students remain fixed, and all four conditions evaluate the same held-out problems with matched branch settings. This gives each teacher the opportunity to learn its available interface, rather than removing a familiar tool only at evaluation time. Unlike Table 5, this comparison measures intervention quality against common students, not downstream student training.

Where controllers contribute. The effect of control assistance varies substantially across the three games. With TrajQL, controllers raise Frostbite's normalized gain from 0.205 to 0.400, compared with 0.445 to 0.465 in BabyAI. The larger change in Frostbite is consistent with the difficulty of executing a return across moving platforms, even after choosing the right objective. Seaquest shows an intermediate increase. TrajQL also improves guidance within the primitive-only condition in every game, so its contribution persists when the teacher must express the correction directly as

Table 7: **Inquiry under matched action interfaces.** Normalized mean paired gain against common reference students. P: primitive sequences; P+C: primitives plus game-specific controllers.

	Frame-index inquiry		TrajQL	
Game	P	P+C	P	P+C
Frostbite	0.075	0.190	0.205	**0.400**
Seaquest	0.120	0.205	0.235	**0.320**
BabyAI	0.340	0.380	0.445	**0.465**

actions. Controllers and inquiry address different parts of the problem: executing a chosen correction and identifying which correction is useful.

Execution and accounting. Within each game, all conditions receive the same initial observations, trajectory access, and state-field sources. The maximum executed intervention duration is identical across interfaces; a controller call consumes its actual primitive steps and does not extend the paired horizon. The teacher must choose the target and starting step. Controllers may use only their documented state inputs and search procedures, with all simulator search and verification charged separately. We report controller invocation, executed duration, and additional simulator calls alongside paired gain. A completed tool goal is not counted as beneficial unless its continuation improves the paired return. Cross-game differences are interpreted in light of these execution demands rather than a pooled controller-success rate.

C.6 Guidance under Matched Retrieval Budgets

Finding useful history with limited access. This comparison asks whether TrajQL improves guidance by locating useful records, rather than simply exposing the teacher to more of the recording. For example, a teacher using frame indices may inspect several positions before reaching a fuel change, whereas TrajQL can request a record satisfying that condition directly. Both can then inspect the same surrounding actions. We compare the two interfaces with limits of 8, 16, and 32 returned records per session on Frostbite, River Raid, and BabyAI. Each interface has access to the same complete recording; the budget limits how much of it is delivered to the teacher, not how far backward or forward it can search.

What the budget counts. A record is one observation step returned with the game's prescribed image or text representation and a fixed set of state fields. For this comparison, both interfaces use the same representation, image resolution, and field set within a game. Every delivered step in a clip counts separately, and repeated delivery is charged again. The common entry observation is excluded from the retrieval allowance. TrajQL retains its content predicates, while frame-index inquiry specifies temporal positions; unrestricted summaries and aggregate queries are disabled in this comparison so that record retrieval remains the common unit. Both interfaces also share caps on query rounds, text length, image count, and total context, set to accommodate the largest record allowance. Documentation and empty results consume their usual call and token budgets even though they do not deliver observation records.

Guidance and consumed information. Table 8 pairs normalized mean gain with the mean number of records actually returned. These are allowances rather than quotas: the teacher can stop early or choose not to intervene. Query calls and consumed tokens, including image inputs and documentation, are recorded separately; equal record limits do not imply identical token costs. Paired evaluation uses the same problems, frozen reference students, action interface, proposal count, horizon, and reward normalization across all budget settings. Teachers are adapted separately for the two query interfaces with the same mixture of the three budget limits, then held fixed during evaluation across limits. Frostbite and BabyAI reuse the controller-enabled teachers and problem banks from Table 7, including their 16-record results; River Raid uses primitive actions. This comparison varies access at evaluation without retraining a new teacher at every budget.

More access and more targeted access. Larger allowances help frame-index inquiry in all three games, but the gap closes at different rates. In BabyAI, its gain increases from 0.285 to 0.425 as

Table 8: **Guidance at matched retrieval limits.** B is the maximum number of returned records. Gain is the normalized mean paired gain; used is the mean number of records delivered per session.

		Frostbite		River Raid		BabyAI	
B	Inquiry	Gain	Used	Gain	Used	Gain	Used
8	Frame-index	0.100	7.1	0.025	7.5	0.285	6.5
	TrajQL	0.250	6.2	0.240	6.8	0.410	5.2
16	Frame-index	0.190	13.6	0.115	14.3	0.380	10.9
	TrajQL	0.400	10.8	0.355	12.1	0.465	7.5
32	Frame-index	0.300	24.2	0.265	26.1	0.425	15.1
	TrajQL	0.440	13.9	0.390	18.4	0.475	8.8

the allowance grows from 8 to 32 records, approaching TrajQL's 0.475. River Raid retains a larger separation at the same upper limit: 0.265 versus 0.390. More informative retrieval can also offset a smaller allowance. With at most 16 records, TrajQL exceeds the frame-index method's gain at 32 records in each game, while consuming fewer records on average. Beyond 16 records, its additional gains are modest, particularly in BabyAI. Together, these patterns connect query effectiveness to how much history must be searched and how much of that history is needed to resolve the problem.

In both local comparisons, valid non-intervention decisions contribute zero paired gain. Failed or unevaluated executions remain separate, with the evaluated fraction reported for each condition. Mean gain uses the fixed game scales in Equation 9; it is not normalized by the best result at each budget. This preserves the meaning of differences across the three retrieval limits.

C.7 Baseline Input Adaptation

LLM4Teach and ACE originally provide their teachers with textual descriptions, rather than requiring the underlying environment to be a text game. Our adaptation protocol uses a common multimodal teacher and the same image and structured-state sources across teacher-guided conditions. It preserves LLM4Teach's option-based guidance and policy-distillation mechanism, and ACE's action refinement, environment validation, trajectory-level reward adjustment, and teacher fine-tuning. ACE retains its trajectory context. Context assembly follows each baseline's procedure; adaptive TrajQL retrieval is the component specific to our method. Where action controllers are available, their capabilities and execution costs are matched. This separates the effect of active inquiry from differences in visual access, privileged state, or action execution.

C.8 Trajectory Case Studies

Figure 4 uses six recorded games and their revised continuations. Each continuation starts from a restored state with an identical recorded prefix. The case editor preserves the original observations, incoming actions, native rewards, and structured fields. The displayed frames are selected from these recordings; Q1–Q3 indicate successive queries, while a displayed clip frame represents one result within the indicated retrieved interval. River Raid and Mario show queries whose time references are obtained from preceding results. River Raid first locates a destroyed depot, then checks its preceding 32 steps for fuel gains, and finally locates the firing onset after finding no replenishment in that interval. The middle card reports text fields rather than an additional screenshot. A query returning several frames is counted once. The curves use undiscounted native rewards and are distinct from the training metrics: a single Crafter return is not its benchmark achievement score, and a single BabyAI or Mario outcome is not an estimated success rate.

Replay settings. Frostbite and Seaquest use four ALE frames per action and a sticky-action probability of 0.25, with seeds 52000 and 52100, respectively. River Raid uses `ALE/Riverraid-v5`, four frames per action, and no sticky actions for the controlled replay. Mario uses level 1-3, the `COMPLEX_MOVEMENT` action set, and up to four NES frames per action. Crafter and BabyAI use one environment step per action; BabyAI uses `KeyCorridorS3R3`. These four cases use seed 52000. BabyAI images and object queries use the agent's local observations; the editor's full-map reference is excluded.

Frostbite: task progress changes the useful objective. The entry at step 576 shows a dying player. Querying the transition from an incomplete to a complete igloo locates step 378. Construction has already finished, so the subsequent correction focuses on returning to the bank. The branch begins at step 566; steps 587 and 641 show the upward return. The recording later enters the igloo at step 659 and reaches 350 additional points by step 701 while retaining three lives. The saved original branch stops at step 586 during the death animation. We preserve this prefix and continue it to step 701 with a frozen Rainbow student (seed 0, 1M training steps), without changing the revised recording. Across the shared 135-step window from step 566, the original continuation earns 40 points and has two lives at the endpoint, versus 350 points and three lives in the revised branch.

```
SELECT frame, step, task.igloo_blocks
FROM trajectory
WHERE step < 576
  AND CHANGED(task.igloo_complete, 'increase')
ORDER BY step DESC LIMIT 1 RETURN MIXED;
```

Seaquest: a reward opportunity can coincide with danger. At step 168, the original sequence picks up a diver but starts a collision animation. Inspecting the preceding frames reveals a diver beside a shark in the lowest occupied lane. From step 149, the revised branch instead approaches the diver in the lane above and records a pickup at step 168 without a crash. Both recorded action sequences are preserved through step 186, after which the same frozen Rainbow student (seed 0, 1M training steps) controls both continuations. Over a common 500-step window from step 149 to 649, native returns are 120 and 140 for the original and revised branches, respectively. The revised branch retains three lives, versus two in the original branch. The shorter window ending at step 186 yields returns of 20 and 0: the benefit of avoiding the collision only appears in the longer continuation.

```
SELECT frame, player.phase, task.divers, entities
FROM trajectory
WHERE step BETWEEN 166 AND 168
ORDER BY step ASC RETURN MIXED;

SELECT frame, step, entities
FROM trajectory
WHERE step = 149 AND HAS_ENTITY('diver', row = 3)
RETURN MIXED;
```

River Raid: a locally rewarding action removes a later resource. At the entry step 532, the plane has exhausted its fuel. A depot-destruction query identifies step 295, where destroying the depot yielded 80 points. A follow-up query retrieves the fuel record from steps 263–295: fuel falls from 52.55% to 46.27%, with no positive change between adjacent records in this interval. The final query locates the firing onset for this depot. The associated firing sequence starts with the action recorded at step 293; the corresponding pre-action state is step 292. The revision replaces `downfire` with `down` for steps 293–324. It first refuels, then destroys the depot at step 325 for the same 80 points. Both sequences use `downfire` thereafter. Across the 300-step window from step 292 to 592, returns are 700 and 790, giving a raw gain of 90. The original branch loses a life; the revised branch retains all four. Fuel is read from the engine, with the high byte divided by

255 for the displayed fraction. Refilling after respawn is distinguished from refueling at a depot.

```
-- Q1: which fuel depot was lost?
SELECT frame, step AS depot_step, reward
FROM trajectory
WHERE EVENT('fuel_depot_destroyed') AND step <= 532
ORDER BY step DESC LIMIT 1 RETURN MIXED;

-- Q2: did the approach include refueling?
SELECT step, fuel_fraction,
  fuel_fraction - LAG(fuel_fraction) OVER (ORDER BY step)
    AS fuel_change
FROM trajectory
WHERE step BETWEEN :depot_step - 32 AND :depot_step
ORDER BY step ASC RETURN TEXT;

-- Q3: where did firing at that depot begin?
WITH controls AS (
  SELECT step, firing,
    LAG(firing, 1, 0) OVER (ORDER BY step) AS previous_firing
  FROM trajectory
)
SELECT p.frame, p.step AS intervention_step, p.action
FROM controls AS c JOIN trajectory AS p ON p.step = c.step - 1
WHERE c.firing = 1 AND c.previous_firing = 0
  AND c.step <= :depot_step
ORDER BY c.step DESC LIMIT 1 RETURN MIXED;
```

Mario: the current position omits the preceding button sequence. At step 37, Mario is falling below the destination ledge. The jump begins at step 31, but the original sequence releases the jump button at step 32 and reaches a low apex at step 33. The branch starts at step 30 and holds `right+A` through step 38; subsequent motion reaches the platform. At the common step 40, the original sequence terminates with 51 reward accumulated after the fork, whereas the revised sequence has accumulated 68 and remains alive. Its later steps, shown through step 45, are not extrapolated for the terminated original branch. This case concerns one platform crossing rather than level completion.

```
-- Q1: locate the latest jump onset.
SELECT frame, step AS jump_step, action, y_pos
FROM trajectory
WHERE step <= 37 AND CHANGED(jump_held, 'increase')
ORDER BY step DESC LIMIT 1 RETURN MIXED;

-- Q2: inspect controls around the onset returned by Q1.
SELECT frame, step, action, y_pos, jump_held
FROM trajectory
WHERE step BETWEEN :jump_step - 1 AND 37
ORDER BY step ASC RETURN MIXED;
```

Crafter: progress depends on an earlier constructed facility. Repeated attempts to craft a wooden pickaxe at step 32 do not succeed. Retrieving the earlier `place_table` action locates step 15. Returning to the table at step 42 satisfies the proximity requirement, and the pickaxe is crafted at step 43. From the shared step 32, the 11-step continuations receive native returns of 0 and 1. The inventory and achievement records confirm the tool's acquisition.

```
SELECT frame, step, inventory, achievements
FROM trajectory
WHERE step < 32 AND action = 'place_table'
ORDER BY step DESC LIMIT 1 RETURN MIXED;
```

BabyAI: a prerequisite lies outside the current view. At step 54, the locked red door blocks progress. A query over previously observed objects retrieves the red key at step 23, outside the most recent 16 observations. The revised branch retrieves it at step 59, opens the door, drops the key to

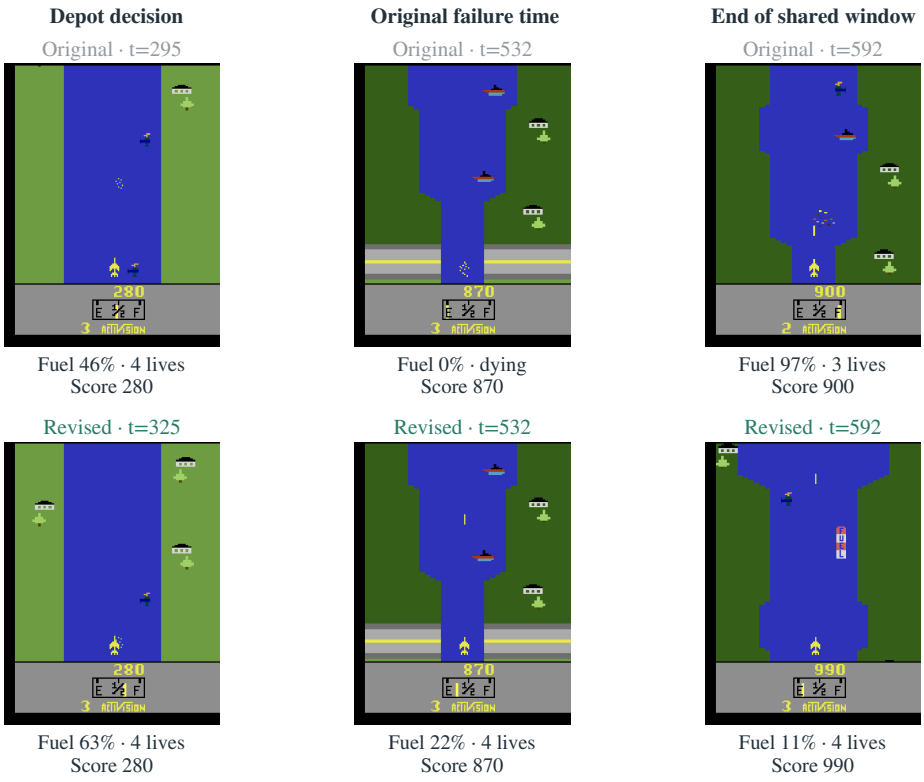


Figure 12: **River Raid branch comparison.** Left: the original sequence destroys the depot immediately, while the revised sequence first refuels and then obtains the same destruction reward. Middle: at the original failure step, only the revised plane retains fuel and remains in flight. Right: the shared evaluation endpoint, with one additional life and 90 additional points in the revised branch. Life counts include the current plane; the engine decrements the original count after the death animation.

free its carrying slot, and obtains the target ball at step 70. From the shared step 54, the 16-step control sequence receives zero reward, while the successful branch receives approximately 0.767.

```
SELECT frame, step, entities
FROM trajectory
WHERE step < 54
  AND HAS_ENTITY('key', color = 'red')
ORDER BY step DESC LIMIT 1 RETURN MIXED;
```