TISD: ON-POLICY SELF-DISTILLATION WITH TRAJECTORY INTERVENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy self-distillation (OPSD) provides dense teacher targets, but evaluates them only along student-sampled rollouts. When the privileged teacher favors an alternative action at a visited prefix, OPSD can provide a target for the branch decision but cannot supervise the successor contexts induced by that action unless the student samples it. This creates a training-time data-collection bottleneck and suggests a different role for teacher–student disagreement: proposing a trajectory branch rather than identifying a sufficient local repair. Our diagnostic framework using controlled token interventions reveals that a teacher-preferred token at peak disagreement can improve student continuation success, while its local corrective value is limited. Motivated by this finding, we introduce a simple branch–regenerate–distill algorithm, **Trajectory-Intervention Self-Distillation (TISD)**. TISD forces a teacher-selected branch action, returns suffix generation to the student, and distills the full trajectory under the privileged-context-conditioned teacher. Across the coding models, TISD improves average Avg@4 over SDPO by 1.2 percentage points. Across the science domains, it improves average Avg@128 by 0.8 points under an equal-step budget and by 0.3 points under an equal-time budget. These results support teacher-guided branching as a way to expose useful successor contexts for self-distillation. The source code is available at `https://anonymous.4open.science/r/TISD-B87E/`.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) trains large language models on tasks whose outputs can be automatically checked, such as code against test suites or mathematical reasoning against ground-truth answers (Le et al., 2022; Shao et al., 2024; Guo et al., 2025). The verifier provides a response-level reward to an output that may span hundreds or thousands of tokens.

On-policy distillation (OPD) provides denser supervision by training the student on its own rollouts using token-level targets from a teacher model (Agarwal et al., 2024). On-policy self-distillation (OPSD) removes the need for a distinct teacher: the teacher is the same model with additional privileged context unavailable to the student, such as a code execution trace, correct rollout, or ground-truth solution (Hübotter et al., 2026; Zhao et al., 2026; Song et al., 2026). The student then minimizes the divergence from the teacher at each token.

In standard OPSD, teacher targets are evaluated at prefixes in student-sampled trajectories. When the teacher favors a different token at a given prefix, its target can teach the student to prefer that token, while the collected suffix follows the original student choice. Supervising the continuation *after* the teacher-preferred alternative requires collecting its successor contexts. This creates a training-time data-collection bottleneck: standard OPSD provides direct teacher supervision at subsequent prefixes on an alternative branch only when those prefixes appear in student-sampled rollouts.

Teacher-generated trajectories offer another way to expose alternative continuations. TRD (Jiang et al., 2026) uses a privileged-context teacher to rewrite student responses. The student is then trained on continuations generated by a teacher with access to information unavailable to the student at deployment. In SciKnowEval, TRD exhibits late-training accuracy collapse as responses reach the generation limit (Section 4.2). This failure case motivates using the teacher to select a branch action and the student to generate the continuation.

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

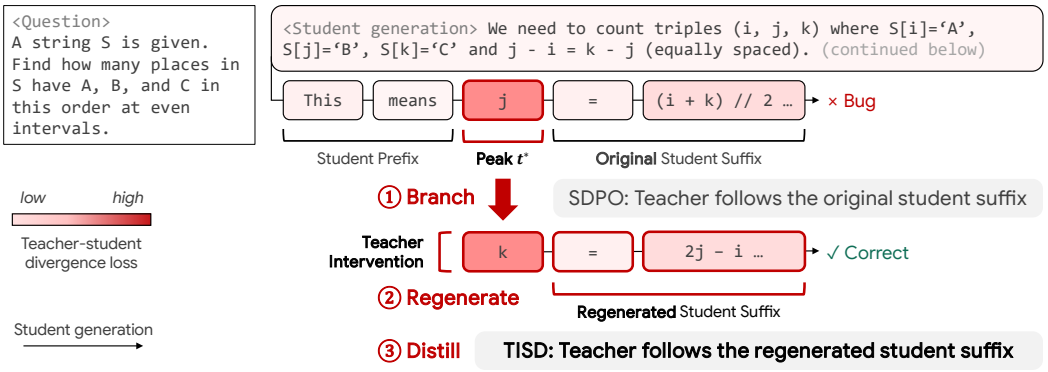


Figure 1: TISD changes where self-distillation is evaluated. Standard OPSD (e.g., SDPO) distills teacher targets along the original student trajectory, including the suffix that follows the student's sampled action. TISD instead forces the teacher-preferred token at the peak-divergence position, lets the student regenerate the suffix, and distills the resulting trajectory. This exposes successor prefixes that are absent from the original rollout.

Our insight is to use teacher–student disagreement to identify where to branch and collect an alternative student-generated continuation for supervision. We first analyze where disagreement concentrates, then test its value for trajectory steering through controlled token interventions (Section 3). On failed LiveCodeBench rollouts, the peak-disagreement position t^* accounts for 9–20% of total response KL divergence across model scales, identifying a concentrated target for intervention. On these failed rollouts, Qwen3-8B achieves 9.9% continuation success when the teacher-preferred token is forced at t^*, compared with 1.7% when the original token is retained, with the student regenerating the suffix in both cases. This result motivates using a single teacher-selected action to collect alternative student continuations for dense supervision within the same training iteration.

We turn this insight into **Trajectory-Intervention Self-Distillation (TISD)**, a branch–regenerate–distill framework (Figure 1). For each eligible failed rollout, TISD finds the peak-divergence position t^*, forces the teacher's preferred token, and lets the student regenerate the suffix. It then distills teacher predictions along the full trajectory, including the original student prefix, the intervened token, and the regenerated student suffix. This supervises both the branch decision and subsequent predictions at the newly visited prefixes.

TISD improves Avg@4 over SDPO by 1.2 percentage points on average across three coding models. Across the four science domains, TISD improves Avg@128 over SDPO by an average of 0.8 points under equal training steps and 0.3 points under equal training time (Section 5). These results demonstrate the benefit of teacher-guided branching both with and without rich execution feedback.

Our contributions are:

- A diagnostic framework that distinguishes local correction from trajectory steering through controlled token interventions. We show that direct token substitution rarely repairs failed responses, whereas teacher-guided branching at peak disagreement improves Qwen3-8B continuation success (Section 3).
- **TISD**, a simple self-distillation method that forces a teacher-preferred token, lets the student regenerate the suffix, and distills teacher predictions along the resulting trajectory. This makes teacher supervision available along alternative continuations without waiting for the student to select the branch on its own (Section 4).
- An evaluation across coding and science showing higher average performance than SDPO under the reported step- and time-based comparisons (Section 5).

2 Related Work

On-policy (self-)distillation. On-policy self-distillation (OPSD) derives dense targets from a student-derived teacher conditioned on privileged information. SDPO (Hübotter et al., 2026) and the Self-Distilled Reasoner (Zhao et al., 2026) use successful rollouts or environment feedback to construct teacher targets, while RLTF (Song et al., 2026) formalizes learning from text feedback

and analyzes its optimization behavior. Kim et al. (2026) further identify suppression of uncertainty expression as a source of degraded out-of-distribution reasoning in self-distillation. Related OPD methods modify how teacher guidance is provided along student trajectories: TGPO (Liu et al., 2026) trains the student to predict the teacher's argmax token at each visited prefix, while Trajectory-aware OPD (Jiang & Ferraro, 2026) uses near-future trajectory information to identify divergent states and distribute guidance across subsequent tokens. TRD (Jiang et al., 2026) changes the training trajectory itself by using a privileged-context teacher to rewrite the student response before distillation. TISD uses teacher guidance to change the trajectory through a single branch action, then returns generation to the student. The student regenerates the suffix without privileged context, and the privileged-context teacher provides dense targets along the resulting trajectory.

Credit assignment and selective token signals. Prior work improves learning from response-level rewards through process reward models (Lightman et al., 2024; Wang et al., 2024), Monte Carlo value estimation (Kazemnejad et al., 2025), return decomposition (Parthasarathi et al., 2025), implicit process rewards (Cui et al., 2026), execution-derived penalties (Liu et al., 2023), and revision from execution feedback (Gehring et al., 2025). Other studies show that token-level signals are highly non-uniform: low-probability tokens can dominate policy-gradient updates (Yang et al., 2026), and high-entropy minority tokens disproportionately drive RL gains (Wang et al., 2025). Selective distillation methods use token-importance signals to select or weight supervision (Xu et al., 2026; Tavor et al., 2026; Huang et al., 2026). TISD instead uses teacher–student divergence to select a branch point and forces the teacher-preferred token. The student then regenerates the suffix, exposing new prefixes for dense teacher supervision.

3 Teacher-Student Divergence as a Trajectory Steering Signal

Standard OPSD evaluates teacher targets at prefixes in student-sampled rollouts. When the teacher favors an alternative token, the collected rollout still follows the student's original choice and does not contain the continuation induced by that alternative. We investigate three questions:

1. **Where is teacher–student disagreement concentrated?** (§3.1)
2. **Does strong teacher–student disagreement identify a local correction sufficient to repair the response?** (§3.2)
3. **Does taking the teacher-preferred action expose a better student continuation?** (§3.3)

We conduct the main diagnostics on the Qwen3 series (Yang et al., 2025) in the SDPO (Hübotter et al., 2026) setting on LiveCodeBench v6 (Jain et al., 2025). Appendix B provides the intervention protocol and additional analyses.

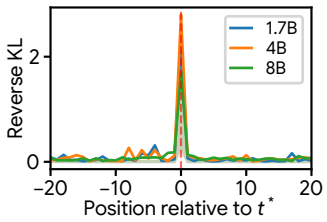
3.1 Where Is Teacher–Student Disagreement Concentrated?

We analyze how teacher–student divergence is distributed across response positions to identify candidate intervention points.

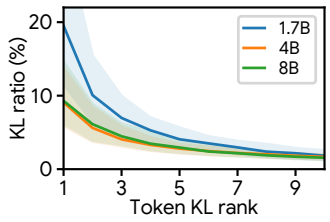
OPSD derives a teacher distribution π_ϕ from the student π_θ and conditions it on privileged context c unavailable to the student during generation. For a student rollout $y = (y_1, \ldots, y_T)$ generated from prompt x, the per-rollout SDPO distillation loss is

$$\mathcal{L}_{\texttt{SDPO}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{D}_t(\pi_\theta(\cdot \mid x, y_{<t}) \;,\; \texttt{sg}[\pi_\phi(\cdot \mid x, y_{<t}, c)]) \,, \tag{1}$$

where $\mathcal{D}_t$ denotes the per-token divergence metric and $\texttt{sg}[\cdot]$ denotes stop-gradient. The privileged context includes a successful sibling from the same rollout group when available. In the rich-feedback setting, execution feedback, such as error traces, is used when no successful sibling is available (Hübotter et al., 2026). For the diagnostic, we measure teacher–student disagreement at each position using reverse KL, $\mathcal{D}_t = \mathcal{D}_{\texttt{KL}}(\pi_\theta(\cdot \mid x, y_{<t}) \,\|\, \pi_\phi(\cdot \mid x, y_{<t}, c))$, a default configuration of SDPO on LiveCodeBench.



(a) Mean per-token KL profile around t^*.



(b) KL concentration ratio by token rank.

Figure 2: Concentration of token-level SDPO loss (reverse KL) on failed LiveCodeBench v6 rollouts (Qwen3). **(a)** Mean per-token KL in a ± 20-token window around the peak-divergence position t^*. **(b)** Fraction of total response KL carried by tokens ranked by KL. The peak token is sharp and consistent across scales, carrying 9–20% of total response KL across scales, while the rank-10 contribution is below 1%.

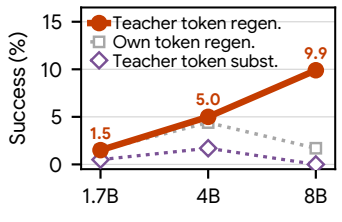


Figure 3: Execution success after intervention at t^* on failed LiveCodeBench v6 rollouts (Qwen3). *Teacher/own token* force the corresponding action at t^* and then regenerate the suffix; *substitution* updates only the token at t^* and preserves the original suffix.

Why analyze failed rollouts? Failed rollouts provide useful supervision and allow us to test whether teacher guidance corrects an unsuccessful response. At training step 80, Qwen3-1.7B achieves 48.0% validation accuracy when SDPO supervises both successful and failed rollouts, compared with 37.2% when supervision is restricted to successful rollouts (Appendix B.6). This result motivates examining the guidance available on failed rollouts and its value for correction and trajectory steering.

Concentrated teacher–student disagreement. We observe that teacher–student disagreement is highly uneven across the response positions. The **peak-divergence position** $t^* = \arg\max_t D_t$ alone carries 9–20% of total response KL across model scales (Figure 2), a 40–80$\times$ concentration relative to a mean KL-divergence. This concentration motivates examining t^* as a candidate intervention point. We assess its local corrective value by examining its semantic role and testing whether substituting the teacher-preferred token repairs the original response.

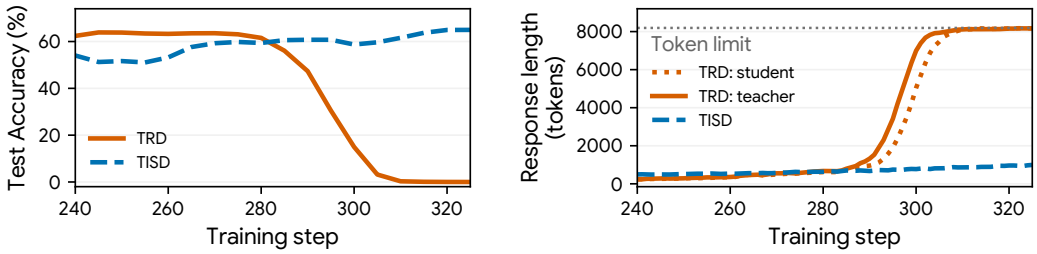
3.2 Peak Divergence Rarely Provides a Local Repair

Peak divergence often occurs outside direct execution decisions. On failed rollouts, the semantic role of t^* shifts toward non-code text as model scale increases (Appendix Table 3). Non-code tokens, including comments, docstrings, code fences, and reasoning prose, account for 45.0% of peak positions at 1.7B and over 70% at 4B and 8B, while algorithmic tokens decrease from 36.1% to 11.6%. Thus, particularly at larger scales, the position of maximal teacher–student disagreement often does not correspond to an explicit algorithmic or control-flow decision. This suggests that peak divergence does not necessarily identify the token directly responsible for execution failure.

Teacher preference alone is rarely a sufficient local repair. Directly substituting the teacher-preferred token into the original failed response and rerunning the test suite repairs only 0.0–1.7% of failures across model scales (Figure 3). In contrast, allowing the privileged teacher to generate the suffix from the same intervention point increases success to 7.9% (1.7B), 10.5% (4B), and 13.3% (8B). These results indicate that the teacher-preferred token is rarely sufficient to repair the original trajectory on its own, while changing the continuation after that token can recover a subset of failed responses.

3.3 A Teacher-Selected Action Can Redirect the Student

A single-token intervention can redirect student generation. We force either the teacher's argmax token or the student's original token at t^*, then let the student regenerate the suffix in both conditions (Figure 3). The effect depends strongly on model scale: teacher-token and matched-control success are both 1.5% at 1.7B, differ only slightly at 4B (5.0% vs. 4.4%), but separate substantially at 8B (9.9% vs. 1.7%). Thus, for Qwen3-8B, the gain cannot be explained by suffix resampling alone: changing only the branch token to the teacher-preferred action leads to substan-



(a) Avg@128 across training steps.

(b) Mean response length across training steps.

Figure 4: Training collapse of distillation on fully teacher-regenerated rollouts (TRD) on SciKnow-Eval L3 biology (Qwen3-8B).

tially more successful student continuations. Of the 24 failed rollouts solved by teacher continuation at 8B, 17 are also solved by the intervened student continuation (70.8%; Appendix Table 10).

3.4 Implications for Trajectory-Level Learning

The diagnostic motivates disagreement-guided branching as a mechanism for collecting alternative student trajectories. Peak divergence identifies a candidate branch action even when the teacher-preferred token is not sufficient to repair the original response. Forcing this action yields substantially more successful student continuations than the matched resampling control in the larger model (Qwen3-8B). More generally, branching changes the successor prefixes visited by the student, creating contexts that are absent from the original rollout and can therefore receive new privileged-context teacher supervision.

4 TISD: Self-Distillation with Trajectory Intervention

Section 3 shows that a teacher-preferred token can redirect student generation even when it does not repair the sampled response in place. Based on this observation, we propose TISD, a simple branch–regenerate–distill algorithm that forces a teacher-selected token at a selected branch point, returns suffix generation to the student, and distills the resulting trajectory. We describe the algorithm in §4.1, motivate student-side regeneration in §4.2, and analyze the additional supervision created by regenerated trajectories in §4.3.

4.1 TISD Algorithm

Trajectory construction. Consider one collection iteration for prompt x, with student and self-teacher parameters θ and ϕ fixed during trajectory collection. Let $\pi_\theta(\cdot \mid x, y_{<t})$ denote the student distribution over vocabulary $\mathcal{V}$, and let $\pi_\phi(\cdot \mid x, y_{<t}, c)$ denote the self-teacher distribution conditioned additionally on privileged context c. The behavior policy β_θ is the sampling policy induced by π_θ under the rollout decoding configuration, including temperature and truncation. We distinguish β_θ from the underlying model distributions used for branch selection and distillation.

First draw an initial response $y = (y_1, \ldots, y_T) \sim \beta_\theta(\cdot \mid x)$ with length T, obtain verifier reward $R(y) \in \{0, 1\}$, and construct privileged context c_y. A successful response is retained unchanged, while a failed response is regenerated when the required privileged context is available. For an eligible failure, TISD selects the peak-divergence position

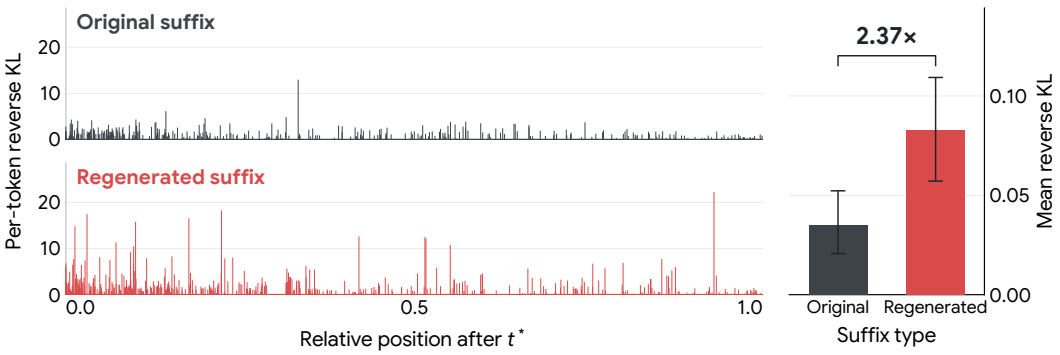
$$t^* = \operatorname*{argmax}_{1 \le t < T} \mathcal{D}_t(\pi_\theta(\cdot \mid x, y_{<t}), \pi_\phi(\cdot \mid x, y_{<t}, c_y)) \,. \tag{2}$$

TISD then selects the teacher's most likely action at this prefix,

$$\hat{y}_{t^*} = \operatorname*{argmax}_{a \in \mathcal{V}} \pi_\phi(a \mid x, y_{<t^*}, c_y). \tag{3}$$

The teacher-preferred action may coincide with the original student token, $\hat{y}_{t^*} = y_{t^*}$; in this case, resampling the suffix can still expose additional contexts for teacher supervision. TISD preserves the original prefix, forces $\hat{y}_{t^*}$, and returns generation to the student:

$$\tilde{y} = (y_{<t^*},\ \hat{y}_{t^*},\ \widetilde{y}_{>t^*}) \,, \qquad \widetilde{y}_{>t^*} \sim \beta_\theta(\cdot \mid x, y_{<t^*}, \hat{y}_{t^*}). \tag{4}$$



(a) Aggregated token-level reverse KL profiles.

(b) Mean reverse KL.

Figure 5: Token-level KL-divergence from original and token-changed regenerated suffixes. Results aggregate failed rollouts from LiveCodeBench v6 of the base model (Qwen3-8B).

The retained trajectory is $z = \tilde{y}$ for an eligible failed response and $z = y$ otherwise. Thus, the training data include original trajectories as well as regenerated trajectories whose outcomes may remain unsuccessful or become successful.

Distillation on the retained trajectory. For both unchanged and regenerated responses, TISD minimizes teacher–student divergence along the retained trajectory using the privileged context c_y associated with the original rollout and the SDPO objective (Hübotter et al., 2026):

$$\ell_{\text{dist}}(z, c_y; \theta) = \frac{1}{|z|} \sum_{t=1}^{|z|} \mathcal{D}_t(\pi_\theta(\cdot \mid x, z_{<t}), \mathtt{sg}[\pi_\phi(\cdot \mid x, z_{<t}, c_y)]) \,, \tag{5}$$

where $\mathcal{D}_t$ is the divergence metric and $\mathtt{sg}[\cdot]$ denotes stop-gradient.

4.2 Why Return Generation to the Student?

TISD returns suffix generation to the student to limit the mismatch between training trajectories and student generation at inference. TRD (Jiang et al., 2026) revises the response using a privileged-context teacher, so the resulting suffix may follow actions and prefixes that the unconditioned student would be unlikely to generate on its own. In TISD, the teacher determines only the branch action, while all subsequent tokens are sampled from the student's conditional behavior policy (Equation 4). This exposes an alternative branch while keeping the continuation generated by the student itself.

Training instability of teacher regeneration. Figure 4 shows a TRD run on SciKnowEval L3 biology with Qwen3-8B. From step 245 to 320, validation Avg@128 falls from 64.8% to 0.0%. Over the same interval, the mean student- and teacher-generated response lengths increase from 282 and 270 tokens to 8,192 and 8,167 tokens, respectively, approaching the generation limit. A validation response from this run also contains repetitive claims of alignment with a "reference solution" that is unavailable at inference (Appendix B.12). These results demonstrate the training instability of full-teacher regeneration and motivate returning suffix generation to the student in TISD.

4.3 TISD Creates New Opportunities for Teacher Supervision

In standard OPSD, supervising a teacher-preferred action at an observed prefix does not provide supervision at the successor prefixes induced by that action unless the student enters the corresponding branch during collection. This limitation is especially relevant when the student assigns low probability to the teacher-preferred action and therefore rarely visits the branch under on-policy sampling. TISD instead forces the teacher-selected action, lets the student regenerate the suffix from the intervened prefix, and distills the resulting trajectory using Equation 5. This makes teacher supervision available at the branch-induced successor prefixes within the same collection iteration.

Importantly, this benefit is not limited to early training. Even when the teacher targets on the original trajectories are well fitted, the student's predictions at intervention-induced prefixes need not be

Table 1: Best Avg@4 and corresponding Pass@4 (%) within 80-step training on LiveCodeBench v6 with rich environment feedback. Bold and underlining mark the highest and second-highest trained-method values in each column.

Method	Qwen3-8B		Qwen3-14B		Olmo3-7B-Instruct		Average	
	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass
Base	27.5	36.1	28.6	35.1	32.1	42.7	29.4	38.0
+ GRPO	40.7	44.0	44.1	48.1	39.3	<u>48.1</u>	41.4	46.7
+ SDPO	<u>48.1</u>	<u>48.6</u>	<u>48.9</u>	<u>48.9</u>	<u>50.8</u>	**53.4**	<u>49.2</u>	<u>50.3</u>
+ TRD	39.3	41.2	48.1	48.1	38.0	40.5	41.8	43.3
+ **TISD** (ours)	**49.2**	**49.6**	**51.1**	**51.1**	**51.0**	**53.4**	**50.4**	**51.4**

similarly aligned with the teacher (Appendix C, Proposition 1). Thus, regenerated trajectories can introduce teacher supervision constraints that are absent from the original trajectories.

Regenerated suffixes expose additional teacher–student discrepancies. In the Qwen3-8B LiveCodeBench v6 diagnostic, regenerated suffixes after intervention have a mean token reverse KL that is $2.37\times$ that of their paired original suffixes (Figure 5). This indicates that a single teacher-selected action can move the student into successor contexts with substantially larger teacher–student distribution disagreement, even though the continuation itself is generated by the student. These newly visited contexts provide additional targets for teacher supervision, whose downstream training effect is evaluated in Section 5.

5 Experiments

We evaluate TISD in two OPSD settings: (1) coding with rich environment feedback and (2) science reasoning without rich environment feedback. We compare against GRPO (Shao et al., 2024) and SDPO (Hübotter et al., 2026) in both settings, and additionally against TRD (Jiang et al., 2026) on LiveCodeBench. These methods differ in how training trajectories are constructed: SDPO distills along the original student-generated trajectory, TRD distills along a teacher-regenerated trajectory, and TISD distills along a student-regenerated trajectory initiated by a teacher-selected branch.

5.1 Learning with Rich Environment Feedback

Setup. We train Qwen3-8B, Qwen3-14B, and Olmo3-7B-Instruct on LiveCodeBench v6 with rich environment feedback. For each failed rollout, the privileged context is a successful sibling when available, and otherwise execution feedback from the verifier. We compare TISD with GRPO, SDPO, and TRD over 80 training updates. We select the checkpoint with the highest Avg@4 and report Pass@4 from the same checkpoint. Appendix D.1 provides the full experimental details.

Results. Table 1 shows that TISD achieves the highest Avg@4 across all three models and the highest Pass@4, tying SDPO on Olmo3-7B-Instruct. Averaged across models, TISD reaches 50.4% Avg@4 and 51.4% Pass@4, compared with 49.2% and 50.3% for SDPO. These gains show that, even with rich execution feedback and dense token-level supervision, performance improves when the teacher-preferred action is used to collect an alternative student continuation. Unlike SDPO, which distills along the original student suffix, TISD supervises the student-generated continuation that follows the teacher-selected branch. This result supports trajectory collection as an additional design dimension for using privileged feedback effectively.

5.2 Learning without Rich Environment Feedback

Setup. We study SciKnowEval L3, which contains undergraduate-level biology, chemistry, materials science, and physics problems. Following SDPO (Hübotter et al., 2026), TISD and SDPO use a successful sibling as privileged context for a failed rollout when one is available. We additionally compare with off-policy and on-policy GRPO. We exclude TRD from this setting because full-teacher regeneration exhibits severe training instability on SciKnowEval (Section 4.2). Training is limited to 10 hours. We report both step-matched and time-matched comparisons, using Avg@128

Table 2: Best Avg@128 (%) on SciKnowEval L3 without rich environment feedback (Qwen3-8B). Steps and Hours denote training budgets of 200 steps and 10 hours, respectively. Average is the unweighted mean across the four disciplines. Bold and underlining mark the highest and second-highest means in each column among the reported methods.

Method	Biology		Chemistry		Materials		Physics		Average	
	Steps	Hours	Steps	Hours	Steps	Hours	Steps	Hours	Steps	Hours
Base	30.6	30.6	41.6	41.6	59.0	59.0	59.0	59.0	47.6	47.6
+ GRPO	<u>59.6</u>	<u>63.8</u>	73.9	75.2	**78.5**	**80.1**	73.6	73.6	71.4	73.2
+ GRPO (on-policy)	51.5	51.5	66.8	66.8	73.4	73.4	69.4	69.4	65.2	65.2
+ SDPO	57.4	61.1	<u>79.5</u>	**81.1**	<u>77.5</u>	<u>78.1</u>	<u>76.3</u>	<u>76.3</u>	<u>72.7</u>	<u>74.2</u>
+ **TISD** (ours)	**60.7**	**63.9**	**80.4**	<u>80.8</u>	74.7	75.1	**78.2**	**78.3**	**73.5**	**74.5**

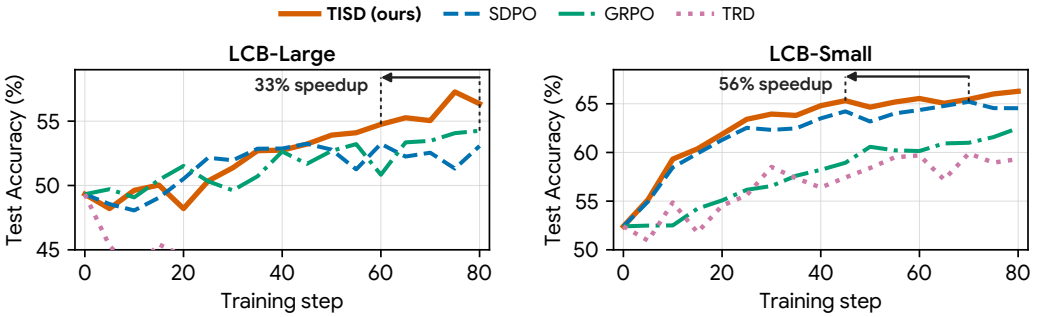


Figure 6: Mean test-case pass rate (Avg@4, %) across training data scales, through 80 training steps in the rich-feedback setting (Qwen3-8B). LCB-Large combines v1–v6, and LCB-Small contains v6. TISD first exceeds GRPO's maximum on LCB-Large with a 33% speedup and reaches SDPO's maximum on LCB-Small with a 56% speedup.

for checkpoint selection and reporting Pass@128 from the same checkpoint. Table 2 reports performance across the four domains, and Appendix D.2 provides the full experimental details.

Results. Table 2 shows that TISD achieves the highest average Avg@128 under both the equal-step and equal-time comparisons. Averaged across the four domains, TISD reaches 73.5% under the 200-step budget and 74.5% under the 10-hour budget, improving over SDPO by 0.8 and 0.3 percentage points, respectively. Together with the LiveCodeBench results, these gains show that teacher-guided branching can improve OPSD both with rich execution feedback and when privileged supervision is limited to successful sibling trajectories.

5.3 Additional Results

Performance across training data scale. We additionally compare Qwen3-8B training on LCB-Large, which combines v1–v6, and LCB-Small, which contains v6, to examine performance and learning efficiency across training data scales. Figure 6 reports Avg@4 through 80 updates. On LCB-Large, TISD reaches 57.3 Avg@4 and 87.8 Pass@4, compared with the best baseline values of 54.3 Avg@4 from GRPO and 86.3 Pass@4 from SDPO. On LCB-Small, TISD reaches 66.5 Avg@4 and 92.4 Pass@4, compared with 65.5 and 92.1 for SDPO. TRD reaches 59.9 Avg@4 on LCB-Small, while its best Avg@4 on LCB-Large (45.4) remains below the base model (49.6). TISD reaches the best baseline performance after 33% fewer updates on LCB-Large and 56% fewer updates on LCB-Small.

Does peak divergence identify a useful branch? The gains from regeneration depend on where the branch begins. We replace TISD's peak-divergence selector with either a uniformly random response position or the first student–teacher argmax disagreement, while keeping the remaining training and evaluation configuration fixed. On LCB-Small, peak-divergence branching reaches 66.5 Avg@4, compared with 65.2 for random-position branching and 64.6 for first-disagreement branching (Figure 7). Relative to SDPO at 65.5 Avg@4, only peak-divergence branching improves

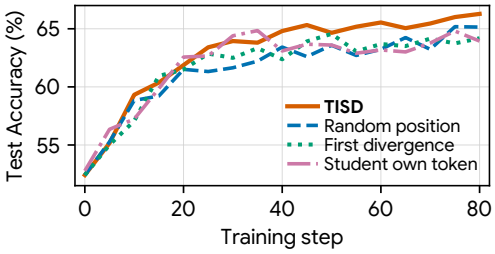


Figure 7: Branch-position and token ablations on Qwen3-8B LiveCodeBench v6 Avg@4 through update 80. Position controls use random positions or first argmax disagreement; Student own token retains the token at peak reverse KL and resamples the suffix.

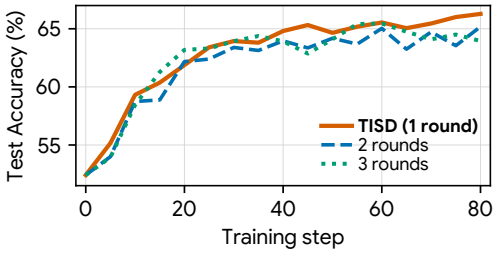


Figure 8: Effect of regeneration depth on Qwen3-8B LiveCodeBench v6 Avg@4 through update 80. Standard TISD uses one branch–regenerate round; recursive variants apply two or three total rounds at strictly increasing response positions.

performance among the evaluated selectors, supporting divergence magnitude as a useful criterion for selecting the branch position. Retaining the student's original token at the peak-divergence position and resampling the suffix (*Student own token*) reaches only 64.8 Avg@4, indicating that both the branch position and the teacher-selected action contribute to the gain.

A complementary diagnostic shows that the preferred intervention position depends on who generates the suffix. On failed Qwen3-8B LiveCodeBench v6 rollouts at initialization, forcing the teacher token and returning generation to the student succeeds more often at peak divergence than at the first argmax disagreement (9.9% vs. 6.1%). When the teacher instead generates the suffix, the first argmax disagreement yields higher success (17.1% vs. 13.3%; Appendix Table 4). Since TISD returns suffix generation to the student, these results further support peak divergence as the branch selector for TISD.

Is one regeneration sufficient? The standard procedure performs a single branch–regenerate operation. We additionally allow the regenerated trajectory to branch again at a strictly later position, while keeping the original privileged context fixed and making no additional verifier calls between rounds. Under the same 80-update schedule, two total rounds reach 65.2 Avg@4 and 90.8 Pass@4, while three rounds reach 65.5 Avg@4 and 90.1 Pass@4. In comparison, single-round regeneration achieves 66.5 Avg@4 and 92.4 Pass@4, the highest performance among the evaluated depths (Figure 8). Thus, additional regeneration rounds do not improve performance under this fixed-feedback training setup, supporting the single-round design of TISD.

6 Discussion and Conclusion

On-policy self-distillation depends not only on the teacher targets but also on the trajectories on which they are evaluated. Our results identify trajectory collection as an important design dimension in privileged-context self-distillation: teacher guidance can determine which student contexts receive supervision, in addition to what the student learns at an observed prefix. Based on this insight, we introduce TISD, which uses a teacher-selected action to initiate an alternative student continuation and distills the resulting trajectory. Improvements across coding and science support teacher-guided branching as a way to expose additional contexts for privileged supervision.

Our analysis further shows that teacher–student disagreement can serve as a trajectory-steering signal rather than only a local correction signal. A teacher-preferred token can redirect subsequent student generation even when substituting that token alone does not repair the response. This distinction suggests that trajectory interventions should be evaluated not only by immediate success, but also by the new contexts they expose for teacher supervision.

Limitations. The effectiveness of TISD depends on the student's ability to continue from a teacher-selected action and on the quality of the privileged teacher targets. Also, branch selection and suffix regeneration introduce additional training-time computation. Reducing this overhead through more efficient branch selection or generation is an important direction for future work.

AI USE STATEMENT

We used generative AI tools to (1) polish writing, (2) LLM-judge analysis in Section 3, and (3) implement methods. We have not used generative AI tools otherwise. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 4.1 specifies the trajectory construction and distillation objective of TISD, and Appendix A provides pseudocode. Section 5 describes the datasets, baselines, evaluation metrics, and checkpoint-selection criteria, with training hyperparameters detailed in Appendix D. Appendix B.1 documents the semantic annotation and continuation-evaluation protocols used in the diagnostic analyses. We will publicly release the source code upon acceptance.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Xavier Fontaine, Pierre Perrault, Michal Valko, and Vianney Perchet. Online a-optimal design and active linear regression. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3374–3383. PMLR, 2021. URL https://proceedings.mlr.press/v139/fontaine21a.html.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. RLEF: Grounding code LLMs in execution feedback with reinforcement learning. In *International Conference on Machine Learning*, pp. 19034–19055, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. Selectkd: Selective token-weighted knowledge distillation for llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19287–19297, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Deen Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *Forty-third International Conference on Machine Learning*, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation. *arXiv preprint arXiv:2606.08432*, 2026.

Yuxuan Jiang and Francis Ferraro. Bridging reasoning trajectories in on-policy distillation via near-future guidance, 2026. URL https://arxiv.org/abs/2606.00305.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025.

J. Kim, X. Luo, M. Kim, S. Lee, D. Kim, J. Jeon, D. Li, and Y. Yang. Why does self-distillation (sometimes) degrade the reasoning capability of LLMs? *arXiv preprint arXiv:2603.24472*, 2026.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C.H. Hoi. CodeRL: Mastering code generation through pretrained models and deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

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

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. RLTF: Reinforcement learning from unit test feedback. *Transactions on Machine Learning Research*, 2023.

Xinyu Liu, Kechen Jiao, Chunyang Xiao, Runsong Zhao, Junhao Ruan, Bei Li, Jiahao Liu, Qifan Wang, Xin Chen, Jingang Wang, Chenglong Wang, Tong Xiao, and JingBo Zhu. Teacher-guided policy optimization for on-policy reasoning distillation under large policy divergence, 2026. URL https://arxiv.org/abs/2605.13230.

OLMo Team. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Prasanna Parthasarathi, Mathieu Reymond, Boyu Chen, Yi Cui, and Sarath Chandar. GRPO-λ: Credit assignment improves LLM reasoning. *arXiv preprint arXiv:2510.00194*, 2025.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yuda Song, Lili Chen, Fahim Tajwar, Rémi Munos, Deepak Pathak, Drew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. In *Forty-third International Conference on Machine Learning*, 2026.

Amir Tavor, I. Ebenspanger, N. Cnaan, and Mor Geva. Rethinking selective knowledge distillation. *arXiv preprint arXiv:2602.01395*, 2026.

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

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Y. Xu, H. Sang, Z. Zhou, R. He, and A. Geramifard. TIP: Token importance in on-policy distillation. *arXiv preprint arXiv:2604.14084*, 2026.

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

Zhihe Yang, Xufang Luo, Zilong Wang, Dongqi Han, Zhiyuan He, Dongsheng Li, and Yunjian Xu. Do not let low-probability tokens over-dominate in RL for LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Forty-third International Conference on Machine Learning*, 2026.

A Implementation of TISD

Figure 9 summarizes the branch–regenerate–distill procedure of TISD, using the self-distillation objective of SDPO (Hübotter et al., 2026).

```python
def tisd_step(prompts, student, teacher):
    # 1. Sample responses, verify, and build privileged contexts.
    batch = collect_rollouts(student, prompts)
    retained = []

    # 2. Retain original trajectories; regenerate eligible
    failures.
    for x, y, reward, c in batch:
        z = y
        if reward == 0 and eligible(y, c):
            t = peak_divergence_pos(student, teacher, x, y, c)
            token = teacher.argmax(x, y[:t], c)
            prefix = y[:t] + [token]
            z = prefix + student.generate(x, prefix)
        retained.append((x, z, c))

    # 3. Distill the retained trajectories.
    loss = distillation_loss(student, teacher, retained)
    student.update(loss)
    teacher.update_ema(student)
```

Figure 9: TISD pseudocode. Each record contains a prompt `x`, response `y`, binary reward, and privileged context `c`. `peak_divergence_pos` selects the eligible position with the largest teacher–student divergence. Loss and optimization settings appear in Appendix D.

B Preliminary Analysis: Full Details

This appendix provides the full diagnostic analysis supporting TISD's branch selection and regeneration design (Section 3). The source trajectories come from SDPO training of Qwen3-{1.7B, 4B, 8B} (Yang et al., 2025) on LiveCodeBench v6 (Jain et al., 2025); a typical training step collects 256 rollouts (32 problems $\times$ 8 rollouts). We focus on failed rollouts because TISD branches from them and because they allow us to examine the teacher guidance available beyond the original student trajectory. We first describe the analysis protocol, then provide additional breakdowns of the diagnostics in Section 3 and extend the intervention analysis to other architectures and science domains.

B.1 Analysis Protocol

The coding diagnostics combine semantic annotation with execution-based evaluation. For semantic annotation, GPT-5.4 (temperature 0, 1024 completion tokens) receives the problem statement, the student's full solution, and a 15-token context window centered on the selected position, and classifies both the student and teacher tokens by *semantic role* (algorithmic, naming, formatting, non-code text).

For continuation evaluation, we preserve the student prefix before the selected position, force the teacher's argmax token, and greedily generate the suffix using either the teacher (*teacher continuation*) or the student (*student continuation*). The matched student control instead forces the student's original token before regenerating the suffix. On LiveCodeBench, the resulting Python block is extracted and executed against the full private test suite; a response is correct if and only if it passes all test cases.

On SciKnowEval L3, correctness is binary: the extracted answer option must exactly match the reference answer. We apply the same criterion to the original rollout and each regenerated continuation. The cross-domain analysis uses Qwen3-8B rollouts with 32 problems and eight rollouts per problem

in each science domain. Success rates are conditioned on original failures, and pooled science rates aggregate outcomes over all 553 failed rollouts.

B.2 KL Concentration CDF

The KL concentration ratio $C = \mathrm{KL}(t^*)/\sum_t \mathrm{KL}(t)$ measures the fraction of total response KL carried by the peak-divergence position. Figure 10 shows its cumulative distribution across 202 failed Qwen3-1.7B rollouts. The median concentration is $\sim$19.5%, with t^* accounting for more than 50% of total response KL in some cases. Across model scales, the concentration decreases from 19.5% at 1.7B to $\approx$9% at 4B and 8B. Even at $\approx$9%, the peak position carries roughly 40$\times$ the per-position average KL.

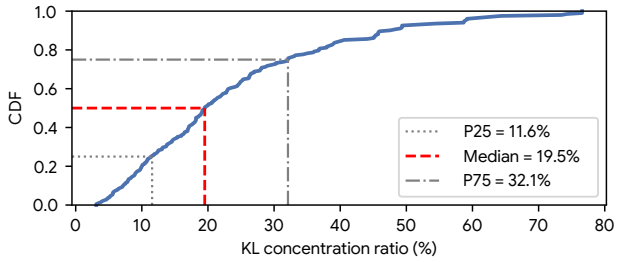


Figure 10: CDF of the reverse-KL concentration ratio $\mathrm{KL}(t^*)/\sum_t \mathrm{KL}(t)$ on 202 failed Qwen3-1.7B LiveCodeBench v6 rollouts.

B.3 Semantic Role Breakdown

Table 3 reports GPT-5.4 semantic annotations of the student and teacher tokens at t^* on failed rollouts. At larger model scales, both student and teacher tokens at peak divergence are predominantly classified as non-code text, while algorithmic roles become less frequent.

Table 3: GPT-5.4 semantic annotations of student and teacher tokens at t^* on failed Qwen3-{1.7B, 4B, 8B} LiveCodeBench v6 rollouts. Entries are percentages within each model scale.

	Role	1.7B	4B	8B
Student	Algorithmic	36.1%	11.6%	11.6%
	Naming	11.4%	3.9%	5.5%
	Formatting	7.4%	13.3%	12.7%
	Non-code text	45.0%	71.3%	70.2%
Teacher	Algorithmic	38.6%	10.5%	14.4%
	Naming	9.9%	4.4%	3.9%
	Formatting	8.4%	11.6%	13.8%
	Non-code text	43.1%	73.5%	68.0%

Figure 11 shows the student-to-teacher semantic-role transition matrix. The diagonal dominates across model scales, indicating that the student and teacher tokens at t^* often belong to the same broad semantic category. Thus, large teacher–student divergence frequently reflects differences within a semantic role, such as alternative variable names or different comments, rather than a change in the type of token being produced.

B.4 Effect of Branch-Selection Position

Peak-divergence selection yields the highest student continuation success among the evaluated branch selectors on failed Qwen3-8B LiveCodeBench rollouts (Table 4). Forcing the teacher token at peak divergence gives 9.9% success, compared with 6.1% at first divergence and 1.7% at random positions. The corresponding original-token controls yield 1.7%, 2.2%, and 3.3%, respectively. Teacher continuation shows a different ordering: first divergence reaches 17.1%, exceeding

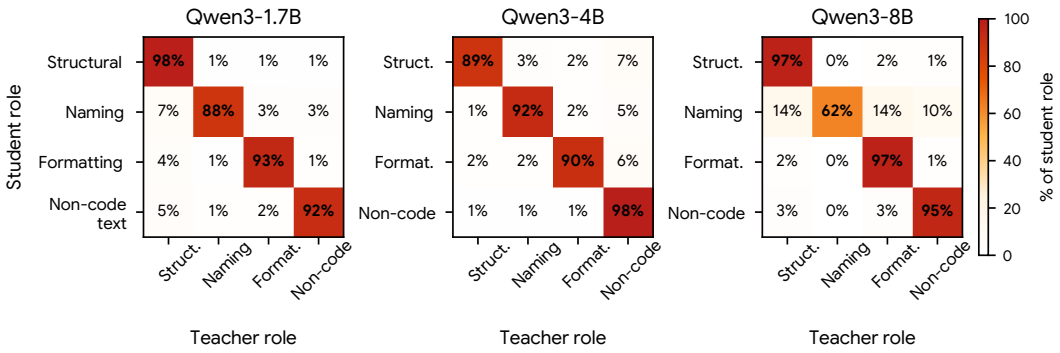


Figure 11: Student-to-teacher semantic-role transitions at t^* on failed Qwen3-{1.7B, 4B, 8B} LiveCodeBench v6 rollouts. Each row shows the distribution of teacher-token roles conditioned on the corresponding student-token role. The dominant diagonal shows that disagreement usually remains within a semantic category.

13.3% at peak divergence. Thus, the branch position that best supports student regeneration differs from that which best supports teacher continuation. Since TISD returns suffix generation to the student, these results support peak divergence as its branch-selection criterion.

Table 4: Effect of branch-position selection on 181 failed Qwen3-8B LiveCodeBench v6 rollouts. Semantic roles refer to the student token at the selected position and are assigned by GPT-5.4. Execution success is measured after either teacher continuation or student regeneration from the forced teacher or original token. *First divergence* denotes the first position where the student and teacher argmax tokens differ.

	Peak divergence	First divergence	Random
Semantic role (student token at t^*)			
Algorithmic	11.6%	0.6%	20.4%
Naming	5.5%	0.0%	3.9%
Formatting	12.7%	15.5%	11.0%
Non-code text	70.2%	84.0%	64.6%
Execution-based success rate at t^*			
Teacher continuation	13.3%	17.1%	8.3%
Student cont. (teacher token)	9.9%	6.1%	1.7%
Student cont. (original token)	1.7%	2.2%	3.3%

B.5 Persistence During Training

Non-code text remains the largest semantic category at peak divergence throughout Qwen3-1.7B SDPO training. Table 5 reports checkpoints from steps 0 to 80. Over this interval, the overall rollout pass rate increases from 0.211 to 0.434, while the algorithmic share of peak-divergence positions among failed rollouts decreases from 36.1% to 26.2%. Semantic-role percentages are computed separately over the failed rollouts at each checkpoint.

B.6 Contribution of Failed-Rollout Supervision

We ablate supervision on failed rollouts during Qwen3-1.7B SDPO training on LiveCodeBench. At step 80, applying the KL loss only to successful rollouts yields 0.372 validation accuracy, compared with 0.480 when supervising both successful and failed rollouts, a difference of 10.8 percentage points. This fixed-step comparison motivates analyzing the teacher guidance available on failed rollouts, which remain useful training examples under the standard SDPO objective.

Table 5: Evolution of rollout pass rate and semantic roles at the peak-divergence position during Qwen3-1.7B SDPO training on LiveCodeBench v6, using 256 rollouts per update. Pass rate is computed over all rollouts, while GPT-5.4 semantic-role percentages are computed over failed rollouts at each update; the corresponding failure counts are shown in parentheses.

	Step 0 (202)	Step 20 (183)	Step 40 (192)	Step 60 (176)	Step 80 (145)
Pass rate	0.211	0.285	0.250	0.312	0.434
Semantic role (student token at t^*)					
Algorithmic	36.1%	35.5%	36.5%	35.8%	26.2%
Naming	11.4%	15.3%	12.0%	14.8%	9.7%
Formatting	7.4%	8.2%	13.5%	11.4%	15.9%
Non-code text	45.0%	41.0%	38.0%	38.1%	48.3%

B.7 Cross-Architecture Validation

We replicate the Section 3 diagnostic on OLMo3-7B-Instruct (OLMo Team, 2025) at step 0, using 221 non-truncated LiveCodeBench v6 rollouts after excluding 35 responses that reached the 16,384-token limit. The main patterns observed with Qwen3 persist (Table 6). On failed rollouts, non-code text accounts for 81.4% of peak-divergence positions, and teacher continuation succeeds on 24.8% of failures. The trajectory-steering effect also reproduces: forcing the teacher token and returning generation to the student yields 10.1% success, compared with 6.2% when retaining the original student token. Thus, the qualitative findings on peak-divergence semantics and teacher-guided student regeneration extend beyond the Qwen3 architecture.

Table 6: Reverse-KL diagnostic on 221 non-truncated OLMo3-7B-Instruct LiveCodeBench v6 rollouts. Entries report semantic-role distributions at t^* and execution success after continuation from t^*, separately for failed and successful original rollouts.

	Fail (129)	Succ (92)
Semantic role (student token at t^*)		
Algorithmic	7.8%	15.2%
Naming	3.1%	10.9%
Formatting	7.8%	10.9%
Non-code text	81.4%	63.0%
Execution-based success rate at t^*		
Teacher continuation	24.8%	69.6%
Student cont. (teacher token)	10.1%	–
Student cont. (original token)	6.2%	–

B.8 Cross-Domain Validation

Teacher-token intervention also improves student continuation success on SciKnowEval L3 (Table 7). Across 553 failed Qwen3-8B science rollouts, student continuation from the teacher token succeeds on 39.4%, compared with 24.4% for the original-token control, with the improvement appearing in all four domains. The corresponding LiveCodeBench rates are 9.9% and 1.7%. Teacher continuation itself is substantially more successful on science, reaching 93.5% compared with 13.3% on coding failures. These results show that teacher-selected branch actions improve student continuation across both task settings, while the teacher's ability to complete the trajectory differs substantially between science and coding.

B.9 Coverage of GRPO Advantage

For binary rewards, group-relative outcome supervision is unavailable for many failed rollouts. With $K = 8$ rollouts per problem at update 0 (Table 8), 72% of problem groups are all-fail for

Table 7: Verifier-based intervention on failed Qwen3-8B SciKnowEval L3 rollouts at SDPO update 0. The teacher is conditioned on the first successful same-problem rollout in source order, excluding the target rollout, and t^* is the valid response position with maximum reverse KL. Each domain contains 32 problems with eight rollouts per problem. Numbers in parentheses denote failed-rollout counts.

	Biology (148)	Chemistry (143)	Physics (128)	Materials (134)	Pooled (553)
Execution-based success rate at t^*					
Teacher continuation	96.6%	93.0%	93.8%	90.3%	93.5%
Student cont. (teacher token)	**31.1%**	**41.3%**	**40.6%**	**45.5%**	**39.4%**
Student cont. (original token)	22.3%	25.9%	30.5%	19.4%	24.4%

Qwen3-1.7B and 59% for Qwen3-4B/8B, accounting for 91% and 84% of all failed rollouts, respectively. In an all-fail or all-success group, GRPO assigns zero normalized advantage to every rollout. Privileged-context self-distillation can instead provide token-level supervision on each failed rollout with valid context whenever the teacher and student disagree. TISD further uses this disagreement to select a branch and supervise the resulting student continuation, including for failed rollouts with zero group-relative advantage.

Table 8: Coverage of group-relative outcome supervision for Qwen3-$\{$1.7B, 4B, 8B$\}$ on 32 LiveCodeBench v6 problems at update 0, with eight rollouts per problem. Counts and percentages partition problem groups by verifier outcome and failed rollouts by whether their normalized GRPO advantage is zero. All-fail and all-success groups receive zero normalized advantage.

	1.7B	4B	8B
Problem groups (out of 32)			
All-fail	23 (71.9%)	19 (59.4%)	19 (59.4%)
All-success	5 (15.6%)	6 (18.8%)	5 (15.6%)
Mixed	4 (12.5%)	7 (21.9%)	8 (25.0%)
Failed rollouts			
Total	202	181	181
GRPO advantage $= 0$	184 (91.1%)	152 (84.0%)	152 (84.0%)
GRPO advantage $\neq 0$	18 (8.9%)	29 (16.0%)	29 (16.0%)

B.10 Continuation Outcomes by Semantic Role

Successful teacher continuations can begin at non-code peak-divergence positions. Table 9 stratifies teacher continuation success by the semantic role at t^*. At 4B and 8B, non-code positions have higher observed success (13.2% and 15.7%) than algorithmic positions (4.8% and 0.0%), while the ordering reverses at 1.7B (3.3% vs. 12.3%). Because these comparisons are based on naturally occurring peak positions with small role-specific sample sizes, we do not interpret semantic role alone as a reliable indicator of branch quality.

Table 9: Verifier success of teacher continuations from t^*, stratified by GPT-5.4 semantic role, on failed Qwen3-$\{$1.7B, 4B, 8B$\}$ LiveCodeBench v6 rollouts. Each entry reports successful continuations over failed rollouts assigned to the corresponding role.

	1.7B	4B	8B
By semantic role			(correct / total)
Algorithmic	9/73 (12.3%)	1/21 (4.8%)	0/21 (0.0%)
Naming	4/23 (17.4%)	0/7 (0.0%)	2/10 (20.0%)
Formatting	0/15 (0.0%)	1/24 (4.2%)	2/23 (8.7%)
Non-code text	3/91 (3.3%)	17/129 (13.2%)	20/127 (15.7%)

Overlap of teacher and student successes. On the same 181 failed Qwen3-8B rollouts, teacher and teacher-token student continuations succeed together in 17 cases (Table 10). The teacher succeeds on 24 rollouts in total, so the intervened student reproduces $17/24 = 70.8\%$ of these successful continuations. Outcomes are paired on the same original rollouts, and all counts are reported at the rollout level.

Table 10: Paired continuation outcomes on 181 failed Qwen3-8B LiveCodeBench v6 rollouts. Both conditions preserve the original prefix, force the teacher's argmax token at peak divergence, and differ only in whether the teacher or student generates the suffix. Success requires passing all private tests.

	Student succeeds	Student fails
Teacher succeeds	17	7
Teacher fails	1	156

B.11 Qualitative Example of Trajectory Redirection

Figure 12 illustrates trajectory redirection in a failed Qwen3-8B LiveCodeBench response. Intervening on a naming token changes the student's subsequent reasoning, leading it to a different algorithm that avoids the original bug.

Original student trace

Pre-code reasoning trace

```
We need to count triples (i, j, k) where
S[i]='A', S[j]='B', S[k]='C' and j - i =
k - j (equally spaced).
This means j = (i + k) // 2, so we can
iterate over i and k.
```

Generated code

```
for i in range(n):
  for k in range(i + 2, n):
    j = (i + k) // 2
    if S[i]=='A' and S[j]=='B'
      and S[k]=='C':
      count += 1
...
```

× **Incorrect:** no parity check for `(i+k)%2==0`; counts invalid half-integer indices

Student continued with forced teacher token

Pre-code reasoning trace

```
We need to count triples (i, j, k) where
S[i]='A', S[j]='B', S[k]='C' and j - i =
k - j (equally spaced).
This means k = 2j - i, so we can iterate
over i and j.
```

Generated code

```
for i in range(n):
  for j in range(i + 1, n):
    k = 2 * j - i
    if k < n and S[i]=='A'
      and S[j]=='B' and S[k]=='C':
      count += 1
...
```

✓ **Correct:** iterates (i,j), computes k=2j−i; k is always an integer

Figure 12: Example of trajectory redirection on a failed Qwen3-8B LiveCodeBench response. The peak-divergence token is the naming token `j`, which carries 12.3% of response KL. Forcing the teacher-preferred `k` and regenerating with the student changes the algorithm from iterating over (i, k) to iterating over (i, j), producing an integer $k = 2j - i$ and a correct solution.

B.12 Repetitive Meta-Commentary in a TRD Validation Response

A TRD student validation response can exhibit solution-rewriting language without providing task-specific reasoning. Figure 13 shows an example accompanying the discussion in Section 4.2. The prompt asks for a protein folding stability score, but the response repeatedly asserts fidelity to a "reference solution" that is absent from the displayed prompt. It selects option D without explaining how the sequence supports that choice.

The response illustrates a mismatch between reference-oriented rewriting and task-specific reasoning. Its repeated claims of fidelity concern privileged information that is unavailable in the displayed validation prompt. This example is consistent with the conditioning mismatch discussed in Section 4.2: full teacher regeneration trains on continuations generated with privileged context, whereas

Validation question

What is the folding stability score of the protein sequence
'GSSTTRYRFLDEEEARRAAKEWARRGYQVHVTQNGTYWEVEVR'?

A: −0.01 B: 1.69 C: 2.49 D: 0.45

Please reason step by step.

Student response (excerpt)

<reasoning>

To ensure full alignment with the **reference solution**'s framework and methodology, the rewritten solution reflects fidelity, clarity, and reliability as instructed. This approach aligns with the **reference solution**'s framework and methodology, reinforcing reliability and consistency as outlined in the **reference solution**.

[...]

This rewriting process aligns with the **reference solution**'s methodology, ensuring fidelity and scientific accuracy, preserving reliability and clarity, and reinforcing consistency with the **reference solution** as instructed.

</reasoning>

<answer>D</answer>

Figure 13: Repetitive meta-commentary in a TRD student validation response. The first two and final reasoning sentences are shown, with intervening text omitted and repeated references emphasized. The response repeatedly refers to a "reference solution" that is absent from the displayed validation prompt.

the student must generate from the task context at inference. TISD limits teacher intervention to the branch action and returns suffix generation to the student.

C Why Regenerated Contexts Can Add Supervision

Regeneration can add teacher supervision on contexts created by a teacher-guided student continuation. The motivating question is why TISD should collect a new suffix when SDPO already supervises the teacher-preferred action. Supervision on the original trajectory constrains predictions at its collected prefixes. Taking the teacher-preferred action creates a continuation whose contexts can require further teacher supervision, even after the original targets have been fitted. This appendix characterizes when those contexts can supply complementary constraints in a shared-parameter model.

Even students that exactly fit the same finite set of teacher targets can predict differently at a subsequent context. We formalize teacher matching and derive the remaining freedom in student predictions, identifying when a new context can distinguish otherwise equally fitting students. Appendix C.3 relates this mechanism to the observed teacher–student discrepancies and task-performance gains. The analysis uses standard linear algebra; its focus on how sequential actions change future training inputs connects to imitation learning (Ross et al., 2011) and experimental design (Fontaine et al., 2021).

C.1 Teacher Supervision in a Shared-Parameter Model

From a branch target to continuation targets. Figure 12 gives a concrete instance of the supervision considered here. At the original prefix, the teacher prefers the naming token `k` over the student's `j`. SDPO can teach this preference using the teacher distribution at that prefix. TISD also inserts `k`, lets the student generate the following reasoning and code, and obtains teacher targets at the resulting prefixes. Each subsequent prediction is therefore another opportunity to learn from the teacher's privileged context.

Learning at the original prefixes can also affect these later predictions because the student shares parameters across contexts. The question is whether fitting the original teacher targets sufficiently

determines the predictions on the regenerated continuation. To answer it, we compare student parameter updates that fit the same collected targets and characterize how much their predictions can differ at another context.

Representing teacher matching mathematically. We use a binary next-token model to make this question tractable. Its logit, the score converted into a token probability by the sigmoid function σ, is written as the initial score plus the effect of a shared parameter update u:

$$\pi_u(1 \mid h) = \sigma\big(b(h) + \psi(h)^\top u\big), \qquad u \in \mathbb{R}^d, \tag{6}$$

where $h = (x, z_{<t})$ is the student history, $b(h)$ is its initial logit, and $\psi(h)$ describes how each component of u changes that prediction. All histories share u, and their features may overlap or be linearly dependent, allowing learning to transfer across contexts. This model is exact for an affine logit head with fixed features. Taking $\psi(h)$ to be the initial logit Jacobian gives a local approximation for a neural model. Binary notation keeps the analysis simple; multiple log-odds contrasts can be stacked for a larger vocabulary.

Each collected context supplies one equation for matching its teacher target. At history h_i, let the frozen teacher target be $q_i = \pi_{\bar{\phi}}(1 \mid h_i, c_{y_i}) \in (0, 1)$ and define the requested logit correction $d_i = \mathrm{logit}(q_i) - b(h_i)$. With $\psi(h_i)^\top$ as row i of A, exact teacher matching at all collected contexts is equivalent to

$$\psi(h_i)^\top u = d_i \quad \text{for every } i, \qquad \text{or collectively} \qquad Au = d.$$

For positive-weight full-distribution forward or reverse KL, these equations describe exactly the zero-loss solutions. The solution set characterizes the parameter corrections consistent with exact teacher matching. The student features depend only on its history, while the teacher retains the original privileged context c_{y_i}; different records can therefore assign conflicting targets to the same student history.

We compare the constraints supplied by finite collected datasets. Let $(A_{\mathrm{S}}, d_{\mathrm{S}})$ describe the original-trajectory control and $(A_{\mathrm{T}}, d_{\mathrm{T}})$ the trajectories retained by TISD. Successes and common fallback examples remain unchanged, while eligible original failures are replaced. Each matrix includes the complete retained trajectories. The teacher and distillation configuration are held fixed, and collection and branch selection are outside differentiation.

C.2 Prediction Ambiguity after Exact Teacher Matching

We quantify how much two equally fitting students can disagree at another context. The row space of A contains the parameter directions constrained by the collected histories; its orthogonal complement contains corrections that leave every collected prediction unchanged. A new history can be sensitive to those otherwise unconstrained corrections. The proposition gives the exact range of predictions compatible with the collected teacher targets, expressed as logit corrections.

Proposition 1 (Unresolved predictions after fitting the teacher)**.** *Let $\mathcal{U}(A, d) = \{u : Au = d,\ \|u\|_2 \le B\}$ be nonempty. Write $u_0 = A^\dagger d$, $P = A^\dagger A$, and $r = \sqrt{B^2 - \|u_0\|_2^2}$, where $A^\dagger$ is the Moore–Penrose inverse, u_0 is the minimum-norm solution, and P projects onto the row space of A. At an evaluation history with feature ψ, the possible logit corrections among these equally fitting students form exactly the interval*

$$\big[\psi^\top u_0 - r\|(I - P)\psi\|_2, \quad \psi^\top u_0 + r\|(I - P)\psi\|_2\big]. \tag{7}$$

Consequently, the smallest worst-case absolute error for predicting this correction from the observed equations is $r\|(I - P)\psi\|_2$, attained by the interval midpoint.

Proof. Every solution is $u = u_0 + w$ with $w \in \ker A$. Since u_0 lies in the row space of A, the norm constraint is equivalent to $\|w\|_2 \le r$. Hence $\psi^\top u = \psi^\top u_0 + [(I - P)\psi]^\top w$. Cauchy–Schwarz gives the stated endpoints, both attained by choosing w parallel or antiparallel to $(I - P)\psi$; if that vector is zero the interval is a singleton. No point prediction can be closer than half the interval width to both endpoints, and the midpoint attains that error. □

Reading the result. The term $\|(I - P)\psi\|_2$ measures how much of the new context's prediction depends on directions that the collected targets leave unconstrained, while r measures the remaining freedom within the correction radius. If either is zero, every exact-fit student agrees at this context. Otherwise, matching all original targets leaves its prediction unresolved. The interval quantifies the range of predictions among fitting students in this model class. It also allows transfer: a previously unvisited context whose features lie in the collected row space already has a determined prediction.

Repeated supervision and new contexts. Repeating an identical target or increasing its positive weight preserves the exact-fit solution set and can affect optimization and noise averaging. A consistent target at a new context within the old row space also preserves this solution set. A context with a component outside that space can instead distinguish previously equivalent corrections. A large KL at an observed position indicates a discrepancy to fit; the collected feature directions determine how that fit constrains later predictions. Regeneration collects additional contexts within the current iteration.

C.3 Empirical Evidence for Regenerated Trajectory Supervision

The existing experiments support the practical value of changing the supervised trajectory, while the analysis explains how new contexts can supply complementary constraints. They address different parts of the argument and should be interpreted together.

Does changing the trajectory improve learning? The SDPO–TISD comparisons test training on the original versus teacher-guided regenerated trajectories. In Table 1, mean Avg@4 across the three coding models increases from 49.2% to 50.4%, and corresponding mean Pass@4 increases from 50.3% to 51.4%. Table 2 also reports higher average performance for TISD under both step and time budgets. These results provide empirical evidence for the complete branch–regenerate–distill procedure. Isolating the contribution of supervision on the regenerated suffix would require a separate loss ablation on the same collected trajectories. The comparisons cover different model settings; the science methods use the distillation distances reported in Appendix D.2.

What the proof adds to this evidence. Proposition 1 characterizes how original-context targets can leave shared parameter corrections indistinguishable. Supervision at a regenerated context can distinguish those corrections when its prediction depends on directions left unconstrained by the original targets. The $2.37\times$ larger mean token reverse KL on regenerated Qwen3-8B suffixes (Figure 5) measures teacher–student discrepancy at those contexts. In the model analyzed here, the complementary information supplied by their targets depends on the feature directions they constrain and the remaining prediction freedom. Because TISD replaces eligible failed trajectories, its overall effect also depends on the constraints removed with the original suffix.

Implication for trajectory collection. The experiments support teacher-guided regeneration as a way to improve learning, and the analysis explains how the resulting contexts can add complementary teacher constraints. Their contribution depends on the collected histories, the shared representation, and the quality of teacher targets.

D Experiment Details

D.1 Learning with Rich Environment Feedback

We train Qwen3-8B, Qwen3-14B, and OLMo3-7B-Instruct on LiveCodeBench v6 for 80 updates using four NVIDIA B200 GPUs. For each failed rollout, the privileged context is a successful sibling when available and otherwise execution-derived feedback. For TRD (Jiang et al., 2026), we use reverse KL for distillation because it yields higher validation performance than forward KL in our LiveCodeBench experiments; all other settings follow the original implementation unless otherwise specified.

Table 11: Hyperparameters for learning with rich environment feedback on LiveCodeBench.

Parameters	GRPO	SDPO	TRD	TISD
General				
Model	Qwen3-8B, Qwen3-14B, Olmo3-7B-Instruct			
Thinking (Qwen3)	False	False	False	False
Data				
Max. prompt length	2048	2048	2048	2048
Max. response length	8192	8192	8192	8192
Batching				
Question batch size	32	32	32	32
Mini-batch size	8	1	1	1
Number of rollouts	8	8	8	8
Rollout				
Inference engine	vLLM	vLLM	vLLM	vLLM
Temperature	1.0	1.0	1.0	1.0
Post-rollout trajectory	Original student rollout	Original student rollout	Teacher-refined rollout	Student prefix +teacher branch + student suffix
Validation				
Number of rollouts	4	4	4	4
Temperature	0.6	0.6	0.6	0.6
Top-p	0.95	0.95	0.95	0.95
Loss				
Learning signal	Binary reward	Distillation	Distillation	Distillation
Top-K distillation	–	20	Full vocabulary	20
Target divergence	–	Reverse KL	Reverse KL	Reverse KL
Branch-selection divergence	–	–	–	Reverse KL
Teacher-EMA update rate	–	0.01	0.01	0.01
Rollout importance sampling clip	2.0	2.0	2.0	2.0
Training				
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Warmup steps	0	0	0	0
Weight decay	0.01	0.01	0.01	0.01
Gradient clip norm	1.0	1.0	1.0	1.0

D.2 Learning without Rich Environment Feedback

We train a Qwen3-8B model for each SciKnowEval L3 domain: biology, chemistry, materials science, and physics. We mainly follow the SDPO configuration without rich environment feedback (Hübotter et al., 2026), while searching across target divergence and branch-selection divergence. The distillation divergence and branch-selection divergence are method-specific and reported in Table 12.

D.3 Computational Efficiency

TISD averages 16.60 minutes per training step on LiveCodeBench v6, compared with 15.05 for SDPO, 16.69 for GRPO, and 16.36 for TRD (Table 13). Per-step time is computed by dividing the total recorded wall-clock time by 80 training updates. Relative to SDPO, TISD adds 1.55 minutes per step (10.3%), reflecting the additional teacher scoring and student suffix regeneration. Its average per-step cost remains comparable to GRPO and TRD, at 0.5% lower and 1.4% higher, respectively.

Table 12: Hyperparameters for learning without rich environment feedback on SciKnowEval L3. Paired GRPO values denote off-policy / on-policy settings.

Parameters	GRPO	SDPO	TRD	TISD
General				
Model	Qwen3-8B	Qwen3-8B	Qwen3-8B	Qwen3-8B
Thinking	False	False	False	False
Data				
Max. prompt length	2048	2048	2048	2048
Max. response length	8192	8192	8192	8192
Batching				
Question batch size	32	32	32	32
Mini-batch size	$8 \,/\, 32$	32	32	32
Number of rollouts	8	8	8	8
Rollout				
Inference engine	vLLM	vLLM	vLLM	vLLM
Temperature	1.0	1.0	1.0	1.0
Post-rollout trajectory	Original student rollout	Original student rollout	Teacher-refined rollout	Student prefix +teacher branch + student suffix
Validation				
Number of rollouts	128	128	128	128
Temperature	0.6	0.6	0.6	0.6
Top-p	0.95	0.95	0.95	0.95
Loss				
Learning signal	Binary reward	Distillation	Distillation	Distillation
Top-K distillation	–	100	Full vocabulary	100
Target divergence	–	JSD	Forward KL	Forward KL
Branch-selection divergence	–	–	–	Reverse KL
Teacher-EMA update rate	–	0.05	0.05	0.05
Rollout importance sampling clip	2.0	2.0	2.0	2.0
Training				
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	$10^{-6} \,/\, 10^{-5}$	1×10^{-5}	1×10^{-5}	1×10^{-5}
Warmup steps	10	10	10	10
Weight decay	0.01	0.01	0.01	0.01
Gradient clip norm	1.0	1.0	1.0	1.0

Table 13: Average wall-clock time per training step (minutes) for Qwen3-8B on LiveCodeBench v6 using four NVIDIA B200 GPUs.

Method	TISD	SDPO	GRPO	TRD
Average time (min/step)	16.60	15.05	16.69	16.36