TRAJECTORY-REFINED DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) has become a central post-training tool for large language models (LLMs), providing dense per-token teacher supervision along the student's own rollouts. In this work, we identify a common structural cause underlying OPD, which we call *prefix failure*. Under prefix failure, dense per-token supervision induces a bimodal teacher mixture and fragmented gradients that token-level loss truncation or reweighting fail to address. We thus propose Trajectory-Refined Distillation (TRD), a trajectory-level correction method that revises the student's rollout under the teacher guidance while within on-policy support. By correcting problematic prefixes before distillation, TRD mitigates prefix failure at its source. Moreover, TRD improves exploration by exposing the student to alternative valid derivations under teacher guidance, even when the original rollouts are already correct. TRD can also be applied to on-policy self-distillation (OPSD), a parameter-sharing variant that uses the student model conditioned on privileged information as the teacher. TRD improves average math accuracy while remaining competitive on code. Under OPSD, it solves $9/23$ AMOBench questions left unsolved by the Qwen3-8B base model, nearly doubling the strongest dense-KL baseline.

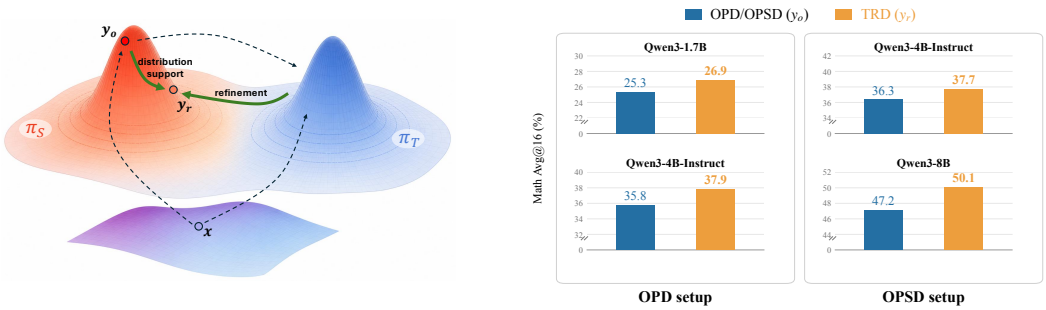


Figure 1: **Left**: TRD refines student-generated trajectories y_o into improved trajectories y_r, which are then used for distillation. **Right**: Mean Avg@16 across the five math benchmarks for OPD/OPSD and TRD under different base models.

1 INTRODUCTION

On-policy distillation (OPD), which computes per-token teacher supervision along the student's own rollouts, has quickly secured a place in modern large language model (LLM) post-training (Gu et al., 2024; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). Recent industry releases including Qwen3 (Yang et al., 2025), DeepSeek-v4 (DeepSeek-AI, 2026), MiMo-v2 (Xiao et al., 2026), and GLM-5 (Zeng et al., 2026) all incorporate an OPD stage alongside supervised fine-tuning (SFT) and reinforcement learning with verifiable rewards (RLVR). While OPD typically relies on a distinct teacher model to supervise training, On-policy self-distillation (OPSD) offers a lightweight alternative: it acts as the parameter-sharing variant of OPD in which the teacher and student are the same model under different contexts. The student is conditioned only on the problem statement, while the teacher is additionally conditioned on privileged information, such as the ground-truth answer (Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026).

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.

Despite these successes, recent studies show that vanilla OPD/OPSD recipes often fall short of their promise, exhibiting failure modes that turn the supervision signal noisy or totally uninformative (Fu et al., 2026; Xu et al., 2026b). Existing remedies address these issues at the token-loss level, reweighting or clipping per-token contributions while leaving the sampled trajectory itself unchanged. For example, Zhao et al. (2026) clip per-token losses above a fixed threshold to suppress destabilizing high-KL tokens. Unfortunately, by acting only at the per-token loss level, these interventions fail to mitigate *prefix failure*, an inherent limitation we formalize in this work. Prefix failure happens when the student's rollout takes a wrong reasoning path and almost no continuation of that prefix can reach the correct solution without backtracking or reflection. Under prefix failure, the per-token teacher distribution becomes a bimodal mixture between continuing the failed prefix and pivoting toward the correction (Sec. 4.1). Even with an ideal teacher, evaluating per-token KL along the student's *frozen rollout* fragments the gradient and yields supervision pairs that diverge from the correction path itself (Sec. 4.2). Recovery therefore requires a *trajectory-level* improvement; prior per-token approaches operate at the wrong scale, i.e., token-level, leaving the failed prefix structurally intact.

To mitigate the prefix failure, we propose Trajectory-Refined Distillation (TRD), a simple yet effective trajectory-level refinement strategy that retains on-policy support under teacher guidance. Concretely, for each problem, we first sample a raw on-policy rollout and then prompt the teacher model to produce a refined version of that rollout. The refined trajectory is then used as supervision for subsequent distillation (Sec. 5). TRD naturally extends to self-distillation, where the teacher additionally receives the reference solution. We empirically validate TRD in both OPD and OPSD on five competition-math benchmarks using Qwen3 models at multiple scales; we additionally evaluate code generation on three code generation tasks for OPD and OPSD variants. On math, TRD achieves the best average performance against baselines across settings, and on code it remains competitive while revealing benchmark-specific trade-offs. Under OPSD, Qwen3-8B with TRD also solves 9 of the 23 AMOBench questions that the base model leaves unsolved in 16 attempts, nearly doubling the strongest dense-KL baseline's Pass@16 on this subset (Appendix B.5).

2 Related Work

On-Policy Distillation. On-policy distillation (OPD) for post-training LLMs traces back to classical knowledge distillation (Hinton et al., 2015) and replaces the fixed-corpus targets there with per-token teacher supervision computed along the student's own rollouts. OPD couples with the on-policy sampling property and dense token-level learning signal by token-level KL loss (Gu et al., 2024; Agarwal et al., 2024; Song & Zheng, 2026; Yang et al., 2025; DeepSeek-AI, 2026; Xiao et al., 2026; Zeng et al., 2026). By contrast, RLVR offers only a sparse trajectory-level reward that scales poorly when most rollouts fail the verifier, whereas SFT relies on off-policy reference data and forfeits the on-policy structure that drives compute-efficient learning (Lu & Thinking Machines Lab, 2025).

On-policy Self-distillation. On-policy self-distillation (OPSD) is a special case of OPD that instantiates teacher and student from the same model under different privileged contexts, thereby enabling self-improvement without a separate teacher model (Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026). The privileged context typically can be involved in reference solutions, feedback, knowledge and experiences, or other auxiliary information that is unavailable to the student (Shi et al., 2026; Ye et al., 2026; Penaloza et al., 2026; Wang et al., 2026; Stein et al., 2026). Operating within the OPD paradigm, OPSD inherits most of the same failure modes, e.g., the privileged supervision can collapse into vacuous guidance as training progresses. A similar concern appears in Zhao et al. (2026), who clip unusually large per-token losses to suppress unreliable learning signals and stabilize training.

Common Failure Mode and Fix. Recent studies report that those vanilla distillation methods often underperform in practice and exhibit a range of failure modes, including mode collapse, trajectory inflation, and supervision signals that vanish or even actively mislead the student and more (Fu et al., 2026; Xu et al., 2026b; Luo et al., 2026; Yang et al., 2026a; Kim et al., 2026; Li et al., 2026a; Xu et al., 2026a; Song & Zheng, 2026). Most of these failure modes are addressed through token-level dense-KL loss interventions. For example, Fu et al. (2026) find that the teacher distribution under OPD can be dominated by a small number of high-loss tokens, and propose top-K truncation to

restrict supervision to high-confidence tokens. Xu et al. (2026b) upweights informative tokens, e.g., tokens with low student entropy but high teacher–student divergence, to achieve better results.

While the per-token interventions above may look contradictory, they share the goal of selecting informative learning signals while stabilize training, with the specific choice dictated by the divergence choice. These interventions are motivated by empirically observation, yet the empirical failures in OPD may partly be attributable to *prefix failure* (Sec. 4).

3 PRELIMINARIES

On-policy Distillation. Knowledge distillation trains a student model π_S to match the output distribution of a teacher π_T, beyond the hard reference label (Hinton et al., 2015). In on-policy distillation (OPD) of LLMs, supervision is computed on trajectories sampled from the current student rather than on fixed expert prefixes (Gu et al., 2024; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). Given a prompt $x \sim \mathcal{D}$ from the training dataset, the student samples an autoregressive rollout $y \sim \pi_\theta(\cdot \mid x)$. The teacher $\pi_T(\cdot \mid x, y_{<t})$ is then evaluated on the student-visited prefixes $y_{<t}$, producing a dense token-level learning signal. A representative OPD objective minimizes the per-token reverse KL divergence between student and teacher,

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} D\big(\pi_\theta(\cdot \mid x, y_{<t}) \,\big\|\, \pi_T(\cdot \mid x, y_{<t})\big) \right]. \tag{1}$$

On-policy Self-Distillation. On-policy self-distillation (OPSD) removes the need for a separate teacher by embodying teacher and student policies from the same model under different contexts (Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026). Given a problem-solution pair $(x, y^\star) \sim \mathcal{D}$, the teacher policy receives privileged information such as the reference answer or reasoning trace and is evaluated as $\pi_T = \pi_\theta(\cdot \mid x, y^\star, y_{<t})$, with teacher and student sharing the parameters θ of the same model. The standard OPSD loss is

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} D\big(\pi_\theta(\cdot \mid x, y_{<t}) \,\big\|\, \mathrm{sg}[\pi_\theta(\cdot \mid x, y^\star, y_{<t})]\,\big) \right], \tag{2}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator applied to the teacher branch. The choice of divergence is itself a design. Reverse KL is mode-seeking and is generally preferred for generative language-model distillation since it discourages the student from assigning probability to low-probability regions of the teacher, whereas forward KL is mode-covering (Gu et al., 2024; Lu & Thinking Machines Lab, 2025; Zhao et al., 2026; Jin et al., 2026; Jang et al., 2026). In practice, the full vocabulary $\mathcal{V}$ is widely adopted to reduce variance and stabilize gradient estimates relative to single-sample Monte Carlo estimation (Zhao et al., 2026; Yang et al., 2025). Having full access to both distributions makes the KL direction independent of the sampling distribution; see Jin et al. (2026) for detailed discussion.

4 PREFIX FAILURE IN TOKEN-LEVEL ON-POLICY DISTILLATION

4.1 MIXTURE OF DISTRIBUTION

Dense per-token KL relies on the teacher providing a reliable supervisory signal at every position along the student rollout. When the teacher faces a wrong reasoning path rolled out by the students, the supervision can become unreliable. Let $\mathsf{F}(y_{o,<t})$ denote the *prefix-failure* event: the prefix $y_{o,<t}$ contains reasoning errors that contradict $y^\star$ and cannot be extended to $y^\star$ without retraction or contradiction. Whenever $\mathsf{F}(y_{o,<t})$ holds, the teacher becomes a mixture distribution with two modes: one that continues the failed prefix $y_{o,<t}$ for sequence consistency and another that pivots back toward $y^\star$. This mixture structure turns the supposedly dense supervision into a noisy or even adversarial signal. A complementary failure mode arises on degenerate prefixes (e.g., repetition loops), where the teacher instead remains locally aligned with the student and the guidance signal vanishes entirely (Fu et al., 2026, Figure 3). Here, we focus on the supervision mismatch caused by prefix failure in dense

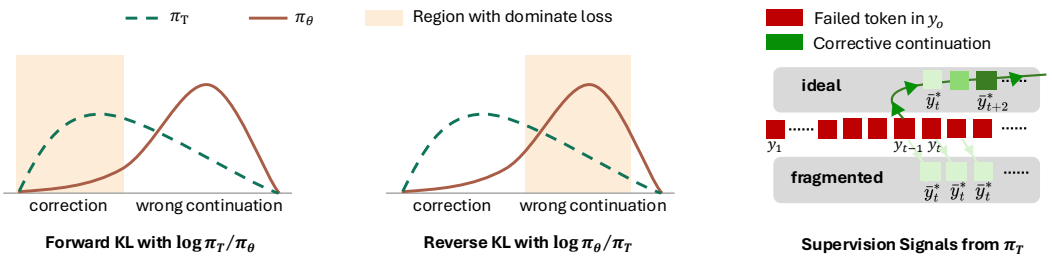


Figure 2: **Left and Middle:** Under prefix failure, the teacher distribution becomes a mixture with two modes. By their respective mode-covering and mode-seeking properties, forward KL is dominated by the correction-onset region, while reverse KL is dominated by the wrong-continuation region. **Right:** Supervision signal provided by the teacher under prefix failure.

on-policy distillation. SFT uses fixed reference trajectories. RLVR updates sampled trajectories using verifier-based rewards rather than dense teacher distributions at each prefix.

The choice of KL direction interacts with prefix failure asymmetrically. Recovering from $\mathsf{F}(y_{o,<t})$ requires a corrective continuation $\bar{y}^\star_{\geq t} = (\bar{y}^\star_t, \bar{y}^\star_{t+1}, \ldots, \bar{y}^\star_{t+k-1})^*$, generated autoregressively along the correction path, where $\bar{y}^\star_t$ is a correction-onset token ($\bar{y}^\star_t \in \{\texttt{Wait}, \texttt{Actually}, \ldots\}$) and $\bar{y}^\star_{t+1}, \ldots, \bar{y}^\star_{t+k-1}$ continue the recovery, with k denoting the length of this continuation. Under $\mathsf{F}(y_{o,<t})$, the ideal teacher partially shifts mass from the natural continuation toward $\bar{y}^\star_t$, while the student with high probability remains anchored on the wrong continuation. Forward KL $D(\pi_T \| \pi_\theta)$, weighted by π_T, is dominated by the correction-onset region; its mode-covering nature forces the student onto this out-of-distribution (OOD) mode. This can destabilize training and, in the worst case, lead to mode collapse (Zhao et al., 2026). In contrast, reverse KL $D(\pi_\theta \| \pi_T)$ is weighted by π_θ, so its mode-seeking behavior makes the loss dominated by the wrong-continuation region where the student already places high mass. Since $\pi_\theta(\bar{y}^\star_t)$ is small, the correction signal has limited effect, so updates concentrate on the failing trajectory rather than the recovery tokens.

Prior work has identified several failure modes of dense token-level KL training and proposed loss-level remedies. These failures are consistent with the prefix-failure mechanism above, even when not explicitly framed this way. Under forward KL, teacher-weighted correction or OOD modes can already dominate the loss, so Zhao et al. (2026) clip per-token losses to cap unstable high-KL terms. Under reverse KL, the correction mode is instead underweighted by the student-weighted objective; accordingly, Fu et al. (2026) use teacher top-K truncation, and Xu et al. (2026b) reweight losses by entropy and student-teacher disagreement to recover informative teacher-preferred tokens. Thus, these remedies control token-level dominance in opposite directions: clipping suppresses overly dominant forward-KL terms, while truncation or reweighting amplifies underweighted reverse-KL teacher modes. However, all of them leave the failed prefix unchanged.

4.2 Can a Perfect Teacher Recover the Correction Path?

Even granting an ideal teacher, dense per-token KL is structurally limited because it is a *post-hoc* per-token objective evaluated along the student's rollout. To make this precise, we trace the per-token OPD loss back to its sequence-level origin, which makes the underlying mechanism more transparent. Define the per-token log-ratio $\delta_t := \log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_T(y_t \mid x, y_{<t})$. Differentiating the sequence-level reverse KL without length normalization (full derivation in Appendix A) yields the policy-gradient form

$$\nabla_\theta \mathcal{J}_{\text{seq}}(\theta) \;=\; \mathbb{E}_{\substack{x\sim\mathcal{D} \\ y\sim\pi_\theta(\cdot|x)}} \left[\sum_{t=1}^{|y|} \left(\delta_t \;+\; \sum_{t'=t+1}^{|y|} \delta_{t'} \right) \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) \right],$$

where $-\delta_t$ acts as a token-level return. In practice, however, standard OPD implementations (Lu & Thinking Machines Lab, 2025; Yang et al., 2025) do not optimize this sequence-level objective

*$\bar{y}^\star_{\geq t}$ is *not* the literal suffix of $y^\star$, but any correct continuation conditioned on $y_{<t}$, i.e., $\text{Verify}(y_{<t} \cdot \bar{y}^\star_{\geq t}, y^\star) = 1$.

directly; instead, they retain only the immediate log-ratio at each position, yielding the per-token surrogate

$$g_{\text{tok}}(\theta) \;:=\; \mathbb{E}_{\substack{x\sim\mathcal{D}\\ y\sim\pi_\theta(\cdot|x)}}\left[\sum_{t=1}^{|y|}\delta_t\nabla_\theta\log\pi_\theta(y_t\mid x,y_{<t})\right]. \tag{3}$$

For comparison, an ideal teacher autoregressively unfolds $\bar{y}^\star_{\geq t}$ along the correction path, delivering the supervision pairs

$$\big\{(y_{o,<t},\,\bar{y}^\star_t),\;\;\big((y_{o,<t},\bar{y}^\star_t),\,\bar{y}^\star_{t+1}\big),\;\;\ldots,\;\;\big((y_{o,<t},\bar{y}^\star_{t:t+k-2}),\,\bar{y}^\star_{t+k-1}\big)\big\},$$

in which the context grows along the correction path itself. Define an illustrative correction-token update signal

$$u_{\text{ideal}} \;:=\; -\sum_{i=1}^{k}\delta_i^{\text{ideal}}\cdot\nabla_\theta\log\pi_\theta\big(\bar{y}^\star_{t+i-1}\,\big|\,x,\;y_{o,<t},\;\bar{y}^\star_{t:t+i-2}\big),$$

where $\bar{y}^\star_{t:t-1}$ is empty and $\delta_i^{\text{ideal}} := \log\pi_\theta(\bar{y}^\star_{t+i-1}\mid x,y_{o,<t},\bar{y}^\star_{t:t+i-2}) - \log\pi_T(\bar{y}^\star_{t+i-1}\mid x,y_{o,<t},\bar{y}^\star_{t:t+i-2})$. Yet under dense KL the contexts are dictated by the *frozen* student trajectory, not by the unfolding correction. At position t, the teacher conveys $\bar{y}^\star_t$ given $y_{o,<t}$, and the student updates its parameters to favor $\bar{y}^\star_t$. At position $t+1$, however, the teacher's supervision is conditioned on $y_{o,<t+1} = (y_{o,<t}, y_{o,t})$ (the original failed trajectory's own continuation), not on the correction path $(y_{o,<t},\bar{y}^\star_t)$. If the teacher repeatedly favors the same correction-onset token on these failed prefixes, the selected supervision pairs form the fragmented sequence

$$\big\{(y_{o,<t},\,\bar{y}^\star_t),\;\;(y_{o,<t+1},\,\bar{y}^\star_t),\;\;\ldots,\;\;(y_{o,<t+k-1},\,\bar{y}^\star_t)\big\},$$

in which the context grows along the wrong continuation while the target remains stuck at $\bar{y}^\star_t$. The analogous update signal

$$u_{\text{frag}} \;:=\; -\sum_{i=0}^{k-1}\delta_i^{\text{frag}}\cdot\nabla_\theta\log\pi_\theta\big(\bar{y}^\star_t\,\big|\,x,\;y_{o,<t+i}\big),$$

uses $\delta_i^{\text{frag}} := \log\pi_\theta(\bar{y}^\star_t\mid x,y_{o,<t+i}) - \log\pi_T(\bar{y}^\star_t\mid x,y_{o,<t+i})$. These illustrative signals compare selected (context, token) pairs; neither is a full KL gradient, and the negative sign denotes the update direction. When $\bar{y}^\star_t \neq y_{o,t}$, the two pair sets share only their first element $(y_{o,<t},\bar{y}^\star_t)$; beyond it, u_{ideal} follows $(y_{o,<t},\bar{y}^\star_t,\bar{y}^\star_{t+1},\ldots)$ while u_{frag} follows $(y_{o,<t},y_{o,t},y_{o,t+1},\ldots)$. The two trajectories diverge after a single step and never re-intersect:

$$\underbrace{\big\{\big((y_{o,<t},\bar{y}^\star_{t:t+i-2}),\,\bar{y}^\star_{t+i-1}\big)\big\}_{i=1}^{k}}_{\text{required by recovery}}\;\cap\;\underbrace{\big\{(y_{o,<t+i},\,\bar{y}^\star_t)\big\}_{i=0}^{k-1}}_{\text{delivered}}\;=\;\big\{(y_{o,<t},\,\bar{y}^\star_t)\big\}.$$

In this example, the privileged information $y^\star$ stays trapped in per-position marginals. Dense KL keeps recommending $\bar{y}^\star_t$ on ever-deepening wrong-continuation contexts but cannot supervise the multi-step unfolding of $\bar{y}^\star_{\geq t}$, and loss-level interventions only reweight terms within the visited pair set $\{(y_{o,<t+i},\bar{y}^\star_t)\}$ rather than move supervision onto the correction-path pair set. The u_{frag} signal is not useless, however; biasing the student toward $\bar{y}^\star_t$ can trigger self-correction, though unfolding the full $\bar{y}^\star_{\geq t}$ is bounded by the student's capacity. Our method TRD (Sec. 5) instead evaluates per-token KL along a teacher-refined trajectory, allowing supervision contexts to follow the refined path rather than the frozen failed prefix.

Complementary training diagnostics are reported in Appendix B.1. They characterize teacher-student discrepancies on raw and refined trajectories; the structural argument above concerns which supervision contexts are visited, rather than the absolute magnitude of the KL loss.

5 Trajectory-Refined Distillation

The previous section identifies *prefix failure* as one of the central bottlenecks in OPD. With prefix failure, dense supervision becomes noisy and can even fail to guide the student. Most existing mitigations operate through loss design and leave the offending prefix unchanged. Directly optimizing prefix failure is intractable: whether a prefix has failed is only revealed after the full trajectory is

verified, while locating the failure index would require searching $\mathcal{O}(|\mathcal{V}|^k)$ continuations of length k. We therefore relax the target to a trajectory-level surrogate that maximizes the expected verifier-pass rate over the dataset $\mathcal{D}$ under the support constraint of π_θ:

$$\max_q \mathbb{E}_{(x,y^\star)\sim\mathcal{D}}\left[\Pr_{y\sim q(\cdot|x,\cdot)}\left\{\text{Verify}\,(y,y^\star)=1\right\}\right] \quad \text{s.t.} \quad \pi_\theta(y\mid x)>0 \tag{4}$$

We note that this is a *trajectory-level* distribution support constraint and the optimization objective form mirrors a standard RLVR objective. We emphasize, however, that this objective is not optimized directly in the OPD update; rather, it defines an upstream trajectory-construction task before the standard OPD optimization over θ, namely to construct trajectories that attain higher verifier-pass rates while remaining within the current student's support. These trajectories then serve as the supervision for the subsequent OPD update through the standard distribution-matching loss in Eq. (1).

Two extreme choices of q illustrate the tension between objective and constraint. **(I)** Setting $q(\cdot\mid x,\cdot)=\pi_\theta(\cdot\mid x)$ fails back to standard OPD. The on-policy constraint is satisfied by construction, but this choice fails to mitigate *prefix failure* beyond current OPD algorithms, since the supervision trajectories are still drawn from π_θ. Reject sampling refines this choice by drawing rollouts and retaining only the verifier-passing ones (Brown et al., 2024; Stein et al., 2026), straining inference budgets linearly and yielding nothing on questions the student cannot solve. **(II)** Setting $q(\cdot\mid x,\cdot)=\pi^\star$, i.e., the expert policy that produces $y^\star$, attains $\text{Verify}=1$ exactly but generally violates the on-policy support constraint, so this choice falls outside the feasible set and breaks the on-policy character that OPD depends on.

To move beyond the two extremes above, we propose *Trajectory-Refined Distillation* (TRD), which operates at the trajectory level to optimize Eq. (4) by first drawing a raw on-policy rollout $y_o\sim\pi_\theta(\cdot\mid x)$, then asking the teacher to construct a refined trajectory y_r via:

$$y_r\sim q(\cdot\mid x,\cdot)\;:=\;\pi_T(\cdot\mid x,y_o). \tag{5}$$

In OPSD, the same backbone implements this teacher query by additionally conditioning on the reference solution y^*: $y_r\sim q(\cdot\mid x,\cdot):=\text{sg}[\pi_\theta(\cdot\mid x,y_o,y^*)]$. Crucially, conditioning on y_o anchors y_r to the reasoning patterns π_θ has already demonstrated, i.e., within the policy support, while π_T rewrites the erroneous portions to directly mitigate prefix failure. To our knowledge, this is the first trajectory-level optimization design that explicitly targets Eq. (4) while respecting the on-policy constraint without prohibitive additional compute overhead. A refined trajectory y_r is then used as supervision for the subsequent OPD update. Per-token KL along y_r reduces exposure to the bimodal teacher mixture (Sec. 4.1) and evaluates supervision on refined prefixes (Sec. 4.2), rather than the raw rollout y_o at higher risk of prefix failure.

Algorithm 1: Trajectory-Refined Distillation

Require: minibatch $\mathcal{B}\subset\mathcal{D}$, π_θ, π_T
1: **for** $x\in\mathcal{B}$ **do**
2: $\quad y_o\sim\pi_\theta(\cdot\mid x)$
3: $\quad y_r\sim\pi_T(\cdot\mid x,y_o)$ by Eq. (5)
4: $\quad$ Update θ by Eq. (6) on y_r
5: **end for**

TRD also boosts the student's exploration beyond standard OPD. On a correct y_o, standard OPD can still provide useful supervision, but the update is evaluated only on the prefixes of that sampled solution. Refinement additionally samples y_r from $\pi_T(\cdot\mid x,y_o)$ and can expose alternative valid derivations of the same answer, i.e., paths suggested by the teacher but rarely sampled from $\pi_\theta(\cdot\mid x)$ alone, thereby expanding the set of correct reasoning trajectories the student is supervised on. TRD therefore adds value in both regimes: mitigating failed prefixes when y_o exhibits prefix failure, and broadening the supervision distribution when y_o already succeeds. The training-data analysis in Appendix B.4 supports this interpretation (e.g., the correct subset of y_r exhibits a low-length mode absent in y_o) and is complemented by the test-time coverage analysis in Sec. 6.3.

Concretely, given y_r, we instantiate the distillation loss as the forward KL with full vocabulary matching over $\mathcal{V}$, which provides mode-covering supervision and stabilizes gradient estimates:

$$\mathcal{L}(\theta)=\mathbb{E}_{\substack{x\sim\mathcal{D}\\ y_o\sim\pi_\theta(\cdot|x)\\ y_r\sim\pi_T(\cdot|x,y_o)}}\left[\frac{1}{|y_r|}\sum_{t=1}^{|y_r|}D\big(\text{sg}[\pi_T(\cdot\mid x,y_o,y_{r,<t})]\;\big\|\;\pi_\theta(\cdot\mid x,y_{r,<t})\big)\right], \tag{6}$$

The exact prompt template is given in Appendix C.7; Fig. 1 (left) and Algorithm 1 illustrate the full procedure. Appendix B.1 reports complementary training diagnostics on y_o and y_r. These describe changes in the supervision distributions, while downstream effectiveness is evaluated in Sec. 6.

6 EXPERIMENTS

We evaluate TRD against SFT, GRPO, three on-policy distillation variants, and SKD under two teacher-information regimes across math and code benchmarks: a separate Qwen3-8B teacher and a privileged self-teacher. We first compare Avg@16 (Sec. 6.2) and then analyze solution discovery under increasing test-time sampling budgets (Sec. 6.3). Best-of-N comparisons under OPSD then test the trajectory-construction objective (Sec. 6.4). Additional objective-validation results, trajectory analyses, and a comparison of the two teacher regimes are provided in Appendices B.3 to B.6, with full experimental details in Appendix C.

6.1 EXPERIMENTS SETUP AND BASELINES

Models. We use the Qwen3 model family (Yang et al., 2025) and evaluate two teacher regimes. In the separate-teacher regime, a frozen Qwen3-8B teacher supervises Qwen3-1.7B and Qwen3-4B-Instruct-2507 students without access to the reference solution. In the privileged self-teacher regime, teacher and student share the same Qwen3-4B-Instruct-2507 or Qwen3-8B backbone, and the stop-gradient teacher branch receives the reference solution $y^\star$ as privileged context.

Training Datasets. For math, we train on the DeepScaleR math corpus (Luo et al., 2025) of roughly 40K problems with solutions; for code, we train on TACO (Li et al., 2023), an algorithmic code-generation corpus with roughly 25K training problems, with reference solutions and test cases. In the privileged self-teacher regime, privileged conditioning uses the dataset reference solution $y^\star$.

Baselines. We compare TRD with SFT trained on $y^\star$ and budget-matched GRPO (Shao et al., 2024), which optimizes verifier rewards on y_o. We also evaluate on-policy distillation using full-vocabulary forward KL (Lu & Thinking Machines Lab, 2025), top-K reverse KL (Fu et al., 2026), and clipped forward KL (Zhao et al., 2026). We denote these variants by +OPD with a separate teacher and +OPSD with a privileged self-teacher, adding "w/ Top-K" or "w/ Clip" where applicable. All three variants train on raw student trajectories y_o. Finally, we include Speculative Knowledge Distillation (SKD) (Xu et al., 2025), which uses interleaved student-teacher trajectories, denoted by $y_o + y_r$ in the tables. TRD trains only on refined trajectories y_r.

Evaluation. We evaluate on five math benchmarks, AIME24 (AI-MO, 2024), AIME25 (OpenCompass, 2025), HMMT25 (HMMT, 2025), BeyondAIME (Seed, 2025), and AMOBench (An et al., 2025). For code generation, we use HumanEval+ and MBPP+ (Liu et al., 2023), and LiveCodeBench v6 (Jain et al., 2024). Our primary metric is Avg@16. For each test question, we sample $K{=}16$ completions and grade each with an external verifier. Avg@16 averages their binary correctness scores within each question and then across the test set, estimating single-completion accuracy. We set maximum response lengths of $38,912$ and $16,384$ tokens for math and code tasks, respectively.

6.2 MAIN EXPERIMENTS

Separate-teacher results. Tab. 1 shows that TRD achieves the highest math and code macro-averages at both student scales. TRD achieves the best or tied-best results on $7/8$ benchmarks ($4/5$ in math) for Qwen3-1.7B and $5/8$ benchmarks ($3/5$ in math) for Qwen3-4B. On Qwen3-4B-Instruct-2507, TRD improves math macro-average Avg@16 over Base and vanilla OPD by 4.56 and 2.08 percentage points, respectively. The largest absolute gains occur on AIME24, where TRD reaches 65.4%, corresponding to relative improvements of 10.7% over Base and 9.0% over vanilla OPD ($+6.3$pp and $+5.4$pp, respectively). These gains over vanilla OPD suggest that a fixed teacher can provide more effective supervision when distillation follows refined trajectories. The gains over SKD may arise because TRD can revise earlier reasoning decisions using the full rollout as context, whereas SKD's token-level interventions leave previously accepted prefixes unchanged.

Privileged self-teacher results. Tab. 2 shows that TRD achieves the highest math macro-average at both student scales and the highest code macro-average on Qwen3-8B. TRD achieves the best or tied-best results on $5/8$ benchmarks ($5/5$ in math) for Qwen3-4B and $7/8$ benchmarks ($4/5$ in math) for Qwen3-8B. On Qwen3-8B, TRD improves math macro-average Avg@16 over Base and vanilla OPSD by 8.66 and 2.90 percentage points, respectively. The largest absolute gains occur

Table 1: Separate-teacher Avg@16 results (%). The +OPD family uses a frozen Qwen3-8B teacher without access to $y^\star$. Bold marks block best.

Method	Traj.	Math					Code		
		AIME24	AIME25	HMMT25	BeyondAIME	AMOBench	HumanEval+	MBPP+	LiveCodeBench
Qwen3-1.7B (w/ thinking)									
Base	–	44.8	33.8	21.2	20.1	1.9	60.3	48.0	30.3
+SFT	$y^\star$	44.5	33.3	22.7	19.8	2.1	60.1	49.3	29.3
+GRPO	y_o	47.4	35.2	23.6	**21.4**	2.9	62.2	50.1	31.5
+OPD	y_o	44.4	35.1	23.3	21.2	2.4	61.4	50.9	32.8
+OPD (w/ Top-K)	y_o	46.9	35.8	23.8	21.3	**3.0**	62.9	50.3	32.6
+OPD (w/ Clip)	y_o	48.1	34.4	22.9	20.8	2.1	62.6	50.3	32.4
+SKD	$y_o + y_r$	40.2	31.5	19.6	15.5	1.4	63.1	49.8	31.9
+TRD (ours)	y_r	**49.4**	**37.5**	**24.4**	20.1	**3.0**	**63.2**	**51.2**	**32.9**
Qwen3-4B-Instruct-2507									
Base	–	59.1	42.9	28.3	28.2	8.1	78.0	60.1	28.1
+SFT	$y^\star$	60.7	44.3	28.6	29.4	8.3	79.3	61.4	29.1
+GRPO	y_o	62.3	**48.8**	31.5	31.9	**11.1**	80.3	63.6	**32.3**
+OPD	y_o	60.0	46.7	30.4	32.0	9.9	81.7	64.2	31.5
+OPD (w/ Top-K)	y_o	61.7	45.8	30.6	31.2	9.8	81.8	64.3	32.0
+OPD (w/ Clip)	y_o	62.5	45.2	29.8	32.4	10.1	81.8	64.6	32.0
+SKD	$y_o + y_r$	62.7	48.5	28.8	**32.6**	10.1	81.8	64.8	32.0
+TRD (ours)	y_r	**65.4**	47.9	**33.2**	**32.6**	10.3	**82.1**	**65.2**	31.9

Table 2: Privileged self-teacher Avg@16 results (%). The +OPSD family uses a stop-gradient teacher that shares the student backbone and is conditioned on $y^\star$. Bold marks block best.

Method	Traj.	Math					Code		
		AIME24	AIME25	HMMT25	BeyondAIME	AMOBench	HumanEval+	MBPP+	LiveCodeBench
Qwen3-4B-Instruct-2507									
Base	–	59.1	42.9	28.3	28.2	8.1	78.0	60.1	28.1
+SFT	$y^\star$	60.7	44.3	28.6	29.4	8.3	79.3	61.4	29.1
+GRPO	y_o	62.3	48.8	31.5	31.9	11.1	80.3	63.6	**32.3**
+OPSD	y_o	60.3	48.8	30.8	32.3	9.5	81.1	**65.0**	31.9
+OPSD (w/ Top-K)	y_o	63.0	49.0	31.0	32.3	9.6	81.7	64.7	32.1
+OPSD (w/ Clip)	y_o	62.1	49.2	27.9	32.4	9.3	**82.1**	64.8	31.9
+SKD	$y_o + y_r$	61.3	46.0	30.0	31.3	9.3	81.3	64.8	32.0
+TRD (ours)	y_r	**63.1**	**49.4**	**32.1**	**32.7**	**11.2**	81.8	64.8	31.8
Qwen3-8B (w/ thinking)									
Base	–	72.3	59.7	30.5	35.6	8.9	70.3	57.6	47.5
+SFT	$y^\star$	73.2	61.9	32.9	38.5	11.3	73.1	59.8	49.2
+GRPO	y_o	75.6	68.5	41.9	**42.9**	15.9	78.5	62.5	52.1
+OPSD	y_o	74.8	65.4	40.3	39.6	15.7	79.5	62.2	52.3
+OPSD (w/ Top-K)	y_o	75.6	68.5	44.4	41.9	15.2	78.7	62.2	52.1
+OPSD (w/ Clip)	y_o	**76.5**	68.3	42.5	40.2	15.9	78.8	62.0	52.8
+SKD	$y_o + y_r$	72.1	66.7	42.3	39.7	15.6	79.6	62.3	51.7
+TRD (ours)	y_r	**76.5**	**69.2**	**44.5**	42.8	**17.3**	**79.7**	**62.6**	**53.3**

on HMMT25, where TRD reaches 44.5%, corresponding to relative improvements of 45.9% over Base and 10.4% over vanilla OPSD ($+14.0$pp and $+4.2$pp, respectively). These trends align with the separate-teacher results, extending the benefits of trajectory refinement to privileged self-distillation.

6.3 TRD Enhances Exploration

Beyond averaged completion accuracy, we evaluate whether TRD improves the student's ability to discover correct solutions as the test-time sampling budget increases. We measure this aspect of exploration with Pass@k, the probability of obtaining at least one correct completion within k samples, averaged across questions. Full Pass@16 comparisons across math and code under both teacher regimes are provided in Appendix B.2.

Fig. 3 compares vanilla OPD trained on y_o with TRD trained on y_r. The gain is modest at $k{=}1$, but widens as the sampling budget increases: at $k{=}128$, TRD reaches 53.8% compared with 46.7% for vanilla OPD, an improvement of 7.1 percentage points (15.2% relative).

These results show that TRD broadens solution coverage as the sampling budget grows. The trajectory analyses in Appendix B.4 provide complementary evidence: refinement improves verifier accuracy and produces shorter solutions even for initially correct training rollouts. At evaluation, vanilla OPD and TRD exhibit similar correct-rollout length distributions, so the coverage gain is accompanied by comparable lengths of successful solutions.

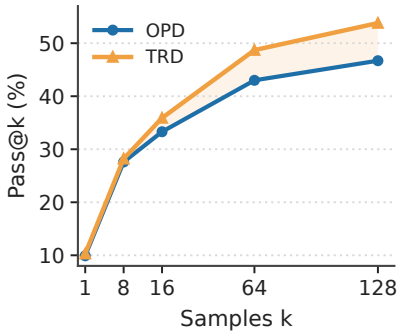


Figure 3: Pass@k on AMOBench for Qwen3-4B-Instruct-2507 under the separate-teacher regime.

6.4 Validating Trajectory Construction

We test whether optimizing Eq. (4) by selecting successful on-policy trajectories provides an effective surrogate for mitigating prefix failure and improving downstream performance. As discussed in Sec. 5, rejection sampling offers one way to pursue this objective while respecting the support constraint. We implement this strategy under OPSD through Best-of-N filtering, drawing N student rollouts per problem and retaining verifier-passing trajectories.

Figure 4 compares Best-of-N with TRD on the full Qwen3-4B-Instruct-2507 training corpus, allowing sampling cost to grow with N. Increasing N improves downstream performance, supporting the usefulness of Eq. (4) as a surrogate for mitigating prefix failure. However, finding successful trajectories through repeated sampling incurs substantial compute costs: even at $7.8\times$ the total wall-clock time of TRD, Best-of-8 does not match its performance. By using privileged guidance to refine sampled trajectories, TRD achieves stronger results with substantially less compute.

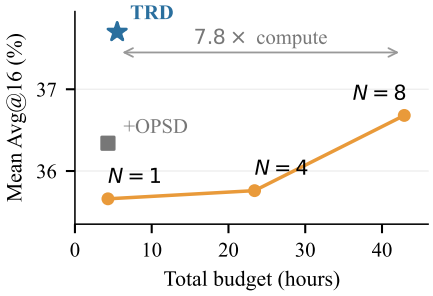


Figure 4: Full-corpus Best-of-N scaling under OPSD with Qwen3-4B-Instruct-2507 on math tasks.

Detailed experimental results are presented in Appendix B.3, including budget-matched Best-of-N comparisons, which also favor TRD overall (Tab. 6). For further analysis of Eq. (4), Tab. 5 shows that alternative supervision trajectories that may violate the support constraint underperform TRD overall.

7 Conclusion

We identify *prefix failure* as a structural limitation of on-policy (self)-distillation paradigms, where the per-token KL evaluated along the student's frozen rollout induces a bimodal teacher mixture and a fragmented gradient that loss-level fixes leave structurally intact. To address it, we propose TRD, a trajectory-level refinement that draws a refined trajectory y_r under teacher guidance and supervises the per-token KL along y_r, recovering the ideal supervision-pair structure while remaining on-policy. Under OPSD, Qwen3-8B with TRD solves $9/23$ AMOBench questions left unsolved by the base model in 16 attempts, nearly doubling the strongest dense-KL baseline's Pass@16 on this subset (Appendix B.5). TRD also outperforms Best-of-8, which requires $7.8\times$ total wall-clock time.

Limitations. First, TRD requires one extra sampling budget to construct y_r. This overhead is partially offset by faster KL training on shorter refined trajectories; on Qwen3-8B, the total wall-clock nearly matches the dense-KL baselines (Appendix C.4). Second, TRD relies on the teacher's ability to guide refinement in a way that mitigates prefix failure while keeping the refined trajectories close to the student's on-policy distribution. This limitation is less severe with a stronger teacher that can in principle optimize Eq. (4) with promise.

AI Use Statement Generative AI tools assisted with language polishing, including grammar, clarity, and readability. The authors reviewed the suggested edits and take responsibility for the final manuscript.

Reproducibility Statement To facilitate reproducibility, Appendix C documents the datasets, trajectory construction, training hyperparameters, hardware and compute accounting, diagnostics, evaluation protocols, and complete prompts. Appendix B provides supporting experimental analyses, and Appendix A presents the assumptions and full OPD policy-gradient derivation. The method and pseudocode are given in Sec. 5 and Algorithm 1.

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.

AI-MO. AIME 2024. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2024.

Shengnan An, Xunliang Cai, Xuezhi Cao, Xiaoyu Li, Yehao Lin, Junlin Liu, Xinxuan Lv, Dan Ma, Xuanlin Wang, Ziwen Wang, et al. Amo-bench: Large language models still struggle in high school math competitions. *arXiv preprint arXiv:2510.26768*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. Technical report, DeepSeek-AI, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro. Technical report and model card.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562*, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

HMMT. Harvard-MIT mathematics tournament, february 2025. https://www.hmmt.org/www/archive/results, 2025.

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

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. *arXiv preprint arXiv:2403.07974*, 2024.

Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation. *arXiv preprint arXiv:2601.07155*, 2026.

Woogyeol Jin, Taywon Min, Yongjin Yang, Swanand Ravindra Kadhe, Yi Zhou, Dennis Wei, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint arXiv:2603.07079*, 2026.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

Gengsheng Li, Tianyu Yang, Junfeng Fang, Mingyang Song, Mao Zheng, Haiyun Guo, Dan Zhang, Jinqiao Wang, and Tat-Seng Chua. Unifying group-relative and self-distillation policy optimization via sample routing, 2026a. URL https://arxiv.org/abs/2604.02288.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026b.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Kevin Lu and Thinking Machines Lab. On-policy distillation. Thinking Machines Lab: Connectionism, 2025. https://thinkingmachines.ai/blog/on-policy-distillation.

Feng Luo, Yu-Neng Chuang, Guanchu Wang, Zicheng Xu, Xiaotian Han, Tianyi Zhang, and Vladimir Braverman. Demystifying opd: Length inflation and stabilization strategies for large language models, 2026. URL https://arxiv.org/abs/2604.08527.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 3(5), 2025.

OpenCompass. AIME 2025. https://huggingface.co/datasets/opencompass/AIME2025, 2025.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

ByteDance Seed. Seed1.5-Thinking: Advancing superb reasoning models with reinforcement learning. https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME, 2025.

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

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential reinforcement learning. *arXiv preprint arXiv:2602.13949*, 2026.

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

Alex Stein, Furong Huang, and Tom Goldstein. Gates: Self-distillation under privileged context with consensus gating, 2026. URL https://arxiv.org/abs/2602.20574.

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

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EgJhwYR2tB.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, and Zhipeng Wang. Paced: Distillation and on-policy self-distillation at the frontier of student competence, 2026a. URL https://arxiv.org/abs/2603.11178.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. Tip: Token importance in on-policy distillation. *arXiv preprint arXiv:2604.14084*, 2026b.

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.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr, 2026a. URL https://arxiv.org/abs/2604.03128.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026b.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 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. *arXiv preprint arXiv:2601.18734*, 2026.

Appendix: Trajectory-Refined Distillation

A Derivation of the OPD Policy Gradient

We derive the exact sequence-level reverse-KL gradient, then distinguish the immediate-token surrogate in Eq. (3). The derivation parallels Yang et al. (2026b). Throughout, the teacher π_T is held fixed during differentiation, using stop-gradient in OPSD, and $\delta_t := \log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_T(y_t \mid x, y_{<t})$.

Step 1 (KL as a log-ratio expectation). The unnormalized sequence-level reverse KL $\mathcal{J}_{\text{seq}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}[D_{\text{KL}}(\pi_\theta(\cdot \mid x) \,\|\, \pi_T(\cdot \mid x))]$ expands to

$$\mathcal{J}_{\text{seq}}(\theta) \;=\; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[\log \pi_\theta(y \mid x) - \log \pi_T(y \mid x)\big],$$

where θ enters through both the sampling distribution and the integrand. This objective is distinct from the length-normalized token losses in Eqs. (1) and (2).

Step 2 (product rule and score-function trick). Differentiating under the expectation gives

$$\nabla_\theta \mathcal{J}_{\text{seq}}(\theta) \;=\; \mathbb{E}_x\Big[\sum_y \big(\nabla_\theta \pi_\theta(y \mid x)\big)\big(\log \pi_\theta(y \mid x) - \log \pi_T(y \mid x)\big) + \sum_y \pi_\theta(y \mid x)\, \nabla_\theta \log \pi_\theta(y \mid x)\Big].$$

The second sum vanishes because

$$\sum_y \pi_\theta(y \mid x)\, \nabla_\theta \log \pi_\theta(y \mid x) \;=\; \sum_y \nabla_\theta \pi_\theta(y \mid x) \;=\; \nabla_\theta 1 \;=\; 0.$$

Using $\nabla_\theta \pi_\theta = \pi_\theta\, \nabla_\theta \log \pi_\theta$, the remaining term becomes

$$\nabla_\theta \mathcal{J}_{\text{seq}}(\theta) \;=\; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[\big(\log \pi_\theta(y \mid x) - \log \pi_T(y \mid x)\big)\nabla_\theta \log \pi_\theta(y \mid x)\big]. \tag{7}$$

Step 3 (autoregressive decomposition). Factor $\log \pi_\theta(y \mid x) = \sum_t \log \pi_\theta(y_t \mid x, y_{<t})$ and similarly for π_T. Equation (7) expands to

$$\mathbb{E}_{x,y}\left[\sum_{t=1}^{|y|}\sum_{t'=1}^{|y|} \underbrace{\big(\log \pi_\theta(y_{t'} \mid x, y_{<t'}) - \log \pi_T(y_{t'} \mid x, y_{<t'})\big)}_{\delta_{t'}} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

Step 4 (causality, past log-ratios do not contribute). For any $t' < t$, $\delta_{t'}$ is measurable with respect to $(x, y_{<t})$, and conditioning on this prefix yields

$$\mathbb{E}_{y_t\sim\pi_\theta(\cdot|x,y_{<t})}\big[\nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\big] \;=\; \sum_{y_t} \nabla_\theta \pi_\theta(y_t \mid x, y_{<t}) \;=\; \nabla_\theta 1 \;=\; 0,$$

so all cross terms with $t' < t$ vanish in expectation.

Step 5 (exact gradient and per-token surrogate). Retaining $t' \geq t$ gives the exact sequence-level gradient,

$$\nabla_\theta \mathcal{J}_{\text{seq}}(\theta) \;=\; \mathbb{E}_{x,y}\left[\sum_{t=1}^{|y|}\left(\sum_{t'=t}^{|y|}\delta_{t'}\right)\nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

For gradient descent, $-\sum_{t'\geq t}\delta_{t'}$ acts as a return-to-go for token y_t. Discarding future log-ratios (Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025) gives the immediate-token surrogate used in Eq. (3),

$$g_{\text{tok}}(\theta) \;:=\; \mathbb{E}_{x,y}\left[\sum_{t=1}^{|y|}\big(\log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_T(y_t \mid x, y_{<t})\big)\nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right].$$

This surrogate omits future-return terms and is not the exact gradient of $\mathcal{J}_{\text{seq}}$. In Eqs. (1) and (2), implementations instead hold sampled trajectories fixed, evaluate full-vocabulary token KL, and apply the stated $1/|y|$ normalization. OPSD additionally conditions the stop-gradient teacher on $y^\star$.

B Additional Experiments

This section provides supporting analyses for the main experiments. Appendix B.1 examines training diagnostics related to prefix failure, and Appendix B.2 reports Pass@16 under both teacher regimes. Appendix B.3 examines alternative trajectory sources and Best-of-N filtering to validate the trajectory-construction objective and its compute trade-offs. Appendix B.4 examines training-trajectory quality and length in the privileged self-teacher regime, together with correct evaluation-rollout lengths in the separate-teacher regime. Appendix B.6 compares the separate teacher and privileged self-teacher for TRD on Qwen3-4B-Instruct-2507. Appendix B.5 reports AMOBench test-rollout coverage by base-model difficulty for Qwen3-8B under OPSD.

B.1 Experimental Validation of Prefix Failure

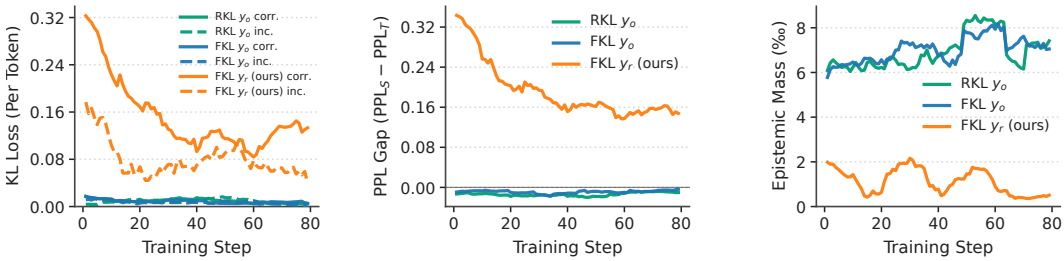


Figure 5: Training diagnostics in the privileged self-teacher setting: standard OPSD on y_o with forward or reverse KL, and TRD on y_r with forward KL. **Left**: Token-weighted mean KL, grouped by the original stage-1 rollout's verifier outcome. **Middle**: Teacher-student perplexity gap on the supervised response. **Right**: Mean teacher probability mass on a fixed epistemic-token set. Metric definitions and normalization are given in Appendix C.5.

We examine three training diagnostics related to the prefix-failure analysis in Sec. 4: per-token KL, the teacher-student perplexity gap, and epistemic-token mass (Fig. 5). Standard OPSD is evaluated on raw student rollouts y_o under forward and reverse KL; TRD is evaluated on refined trajectories y_r under forward KL. These measurements characterize supervision on the respective trajectories. They do not directly locate failed prefixes or establish the causal contribution of prefix correction to downstream gains.

Per-token Distributional Discrepancy (left). We group responses by the original stage-1 rollout's verifier outcome and compute token-weighted mean KL values D^{correct} and $D^{\text{incorrect}}$. Each mean divides the summed token-level KL by the number of unmasked response tokens, so it is already normalized for token count (Appendix C.5). The same outcome labels are used for the paired y_r responses. On y_o, both outcome groups exhibit small average KL, including the group whose original responses fail verification. Within the forward-KL runs, y_r exhibits a larger average teacher-student discrepancy than y_o. This comparison is descriptive: token-count normalization does not remove differences in trajectory content or prefix distributions, and forward and reverse KL are distinct quantities. A smaller mean KL does not imply an uninformative gradient, nor does a larger value establish more effective supervision. The observation on incorrect y_o shows that local distributional agreement can coexist with an incorrect final answer.

Teacher-student Perplexity Gap (middle). For each run, we compute both perplexities by exponentiating the token-weighted mean negative log-likelihood on the same supervised response tokens. The gap $\text{PPL}_S - \text{PPL}_T$ remains small on y_o, whereas y_r initially exhibits a larger positive gap that decreases during training. This describes the models' relative likelihoods of the observed responses.

Since the response distributions differ across runs, the gap alone does not measure the usefulness of privileged information or establish that a correction path has been learned.

Epistemic-token Mass (right). On y_o, the teacher assigns approximately 6 to 8 ‰ of its average next-token probability mass to a fixed token set constructed from 16 correction-related phrases (Appendix C.5); on y_r, this quantity falls below 2 ‰. The higher mass on raw trajectories is consistent with more frequent correction-related cues under failed-prefix conditioning, as discussed in Sec. 4.2. However, these tokens also occur in valid reasoning, and their aggregate mass neither identifies failed prefixes nor demonstrates repeated correction onsets at specific positions. We therefore treat this pattern as a supporting diagnostic of changed conditioning, rather than direct evidence of gradient fragmentation.

B.2 Pass@16 Results

This subsection provides the full Pass@16 results accompanying the exploration analysis in Sec. 6.3. Pass@16 measures the fraction of questions with at least one correct completion among 16 samples, complementing the Avg@16 results in Sec. 6.2. Tabs. 3 and 4 report this coverage metric for the same methods and model configurations as Tabs. 1 and 2, using the evaluation protocol in Sec. 6.1.

Table 3: Separate-teacher Pass@16 results (%). The +OPD family uses a frozen Qwen3-8B teacher without access to $y^\star$. Bold marks block best.

Method	Traj.	Math					Code		
		AIME24	AIME25	HMMT25	BeyondAIME	AMOBench	HumanEval+	MBPP+	LiveCodeBench
Qwen3-1.7B (w/ thinking)									
Base	–	63.3	**66.7**	46.7	42.0	12.8	68.2	55.1	40.3
+SFT	$y^\star$	60.0	63.3	43.3	39.0	10.3	65.8	50.8	41.2
+GRPO	y_o	76.6	63.3	**53.3**	44.0	12.8	76.2	62.7	**48.3**
+OPD	y_o	76.7	63.3	**53.3**	44.0	12.8	71.3	58.2	41.4
+OPD (w/ Top-K)	y_o	76.7	60.0	46.7	44.0	15.4	**78.0**	62.2	46.6
+OPD (w/ Clip)	y_o	**80.0**	56.7	50.0	**47.0**	10.3	77.4	61.9	47.2
+SKD	$y_o + y_r$	73.3	56.7	43.3	31.0	7.7	71.3	57.1	47.1
+TRD (ours)	y_r	**80.0**	**66.7**	**53.3**	45.0	**17.9**	**78.0**	**65.2**	46.8
Qwen3-4B-Instruct-2507									
Base	–	73.3	66.7	43.3	54.0	23.1	79.2	68.7	51.2
+SFT	$y^\star$	73.3	66.7	46.6	55.0	22.6	81.1	69.7	52.0
+GRPO	y_o	83.3	73.3	53.3	**62.0**	28.2	87.8	72.2	54.5
+OPD	y_o	80.0	73.3	50.0	61.0	33.3	87.2	72.6	53.3
+OPD (w/ Top-K)	y_o	80.0	70.0	53.3	57.0	28.2	87.2	72.2	54.5
+OPD (w/ Clip)	y_o	83.3	73.3	53.3	61.0	25.6	**88.4**	73.3	54.1
+SKD	$y_o + y_r$	**86.7**	73.3	**56.7**	**62.0**	25.6	87.2	73.5	**55.4**
+TRD (ours)	y_r	83.3	**76.7**	50.0	**62.0**	**35.9**	**88.4**	**73.8**	54.1

Separate-teacher results. Tab. 3 reports results with the frozen Qwen3-8B teacher. On Qwen3-1.7B, TRD ties the best AIME24, AIME25, and HMMT25 results and leads AMOBench at 17.9%, while +OPD (w/ Clip) leads BeyondAIME. On Qwen3-4B-Instruct-2507, TRD leads AIME25 and AMOBench and ties the best BeyondAIME result, while SKD leads AIME24 and HMMT25. Its AMOBench score rises from Base's 23.1% to 35.9%, a gain of 12.8. On code, TRD matches the best HumanEval+ value and leads MBPP+ at both student scales. GRPO and SKD give the strongest LiveCodeBench results on Qwen3-1.7B and Qwen3-4B, respectively. These results show that refinement improves coverage across most benchmarks, although other methods remain stronger on some math and code tasks.

Privileged self-teacher results. Tab. 4 reports results with the reference-conditioned self-teacher. On math, TRD is best or tied-best in nine of the ten model and benchmark settings. On Qwen3-4B-Instruct-2507, it leads AIME25, BeyondAIME, and AMOBench and ties the best AIME24 and HMMT25 values. On Qwen3-8B, it leads HMMT25 and AMOBench and ties the best AIME24 and AIME25 results; +OPSD (w/ Top-K) leads BeyondAIME. On AMOBench, TRD reaches 61.5%, compared with 51.3% for both GRPO and +OPSD, the strongest baselines at this scale. On code,

Table 4: Privileged self-teacher Pass@16 results (%). The +OPSD family uses a stop-gradient teacher that shares the student backbone and is conditioned on $y^\star$. Bold marks block best.

Method	Traj.	Math					Code		
		AIME24	AIME25	HMMT25	BeyondAIME	AMOBench	HumanEval+	MBPP+	LiveCodeBench
Qwen3-4B-Instruct-2507									
Base	–	73.3	66.7	43.3	54.0	23.1	79.2	68.7	51.2
+SFT	$y^\star$	73.3	66.7	46.6	55.0	22.6	81.1	69.7	52.0
+GRPO	y_o	83.3	73.3	53.3	62.0	28.2	87.8	72.2	54.5
+OPSD	y_o	**86.7**	76.7	53.3	58.0	33.3	87.2	**73.5**	55.3
+OPSD (w/ Top-K)	y_o	**86.7**	76.7	**56.7**	57.0	25.6	87.8	72.8	**55.6**
+OPSD (w/ Clip)	y_o	**86.7**	76.7	46.7	62.0	28.2	87.2	72.8	54.9
+SKD	$y_o + y_r$	80.0	66.7	53.3	57.0	28.2	85.4	69.0	46.1
+TRD (ours)	y_r	**86.7**	**80.0**	**56.7**	**64.0**	**33.8**	**88.4**	73.3	55.2
Qwen3-8B (w/ thinking)									
Base	–	83.3	76.6	66.7	62.0	41.0	84.5	69.3	64.1
+SFT	$y^\star$	86.7	80.0	70.0	64.0	46.2	82.3	68.7	63.4
+GRPO	y_o	**90.0**	83.3	70.0	66.0	51.3	**89.0**	**72.8**	68.2
+OPSD	y_o	86.7	83.3	73.3	68.0	51.3	88.6	71.7	68.4
+OPSD (w/ Top-K)	y_o	86.7	**86.7**	66.7	**69.0**	38.5	88.4	**72.8**	68.2
+OPSD (w/ Clip)	y_o	**90.0**	**86.7**	70.0	61.0	43.6	88.4	71.2	68.6
+SKD	$y_o + y_r$	86.6	83.3	73.3	65.0	46.4	87.8	72.5	67.4
+TRD (ours)	y_r	**90.0**	**86.7**	**76.3**	68.0	**61.5**	88.4	72.5	**69.1**

Table 5: Matched-budget trajectory-source comparison on five math benchmarks (%). y_t is generated by Qwen3-14B. Bold indicates the best or tied-best result in each model block.

		Avg@16					Pass@16				
Method	Traj.	AIME24	AIME25	HMMT25	BeyondAIME	AMOBench	AIME24	AIME25	HMMT25	BeyondAIME	AMOBench
Qwen3-4B-Instruct-2507											
Base	–	59.1	42.9	28.3	28.2	8.1	73.3	66.7	43.3	54.0	23.1
+OPSD	y_o	60.3	48.8	30.8	32.3	9.5	**86.7**	76.7	53.3	58.0	33.3
+OPSD	y_t	61.9	46.2	30.2	32.5	9.8	83.3	76.7	**56.7**	61.0	30.8
+OPSD	$y^\star$	62.5	49.0	31.0	**32.9**	9.6	83.3	76.7	46.7	58.0	30.8
TRD	y_r	**63.1**	**49.4**	**32.1**	32.7	**11.2**	**86.7**	**80.0**	**56.7**	**64.0**	**33.8**
Qwen3-8B											
Base	–	72.3	59.7	30.5	35.6	8.9	83.3	76.6	66.7	62.0	41.0
+OPSD	y_o	74.8	65.4	40.3	39.6	15.7	86.7	83.3	73.3	68.0	51.3
+OPSD	y_t	**77.9**	**69.2**	42.3	40.2	17.0	83.3	**86.7**	66.7	69.0	51.3
+OPSD	$y^\star$	74.8	69.1	44.0	42.3	**19.9**	**90.0**	**86.7**	66.7	**70.0**	56.4
TRD	y_r	76.5	**69.2**	**44.5**	**42.8**	17.3	**90.0**	**86.7**	**76.3**	68.0	**61.5**

TRD leads Qwen3-4B HumanEval+ and Qwen3-8B LiveCodeBench. +OPSD leads Qwen3-4B MBPP+, and +OPSD (w/ Top-K) leads Qwen3-4B LiveCodeBench. On Qwen3-8B, GRPO leads HumanEval+ and ties +OPSD (w/ Top-K) on MBPP+. Coverage gains are therefore strongest on math, with more varied rankings across code benchmarks.

B.3 Additional Validation of the Trajectory-Construction Objective

We expand the validation in Sec. 6.4 along two dimensions of Eq. (4): the role of the support constraint and the benefit of selecting successful on-policy trajectories. All comparisons use OPSD, with privileged self-teacher supervision. The trajectory-source and budget-matched Best-of-N comparisons report Avg@16 and Pass@16 on five math benchmarks for Qwen3-4B-Instruct-2507 and Qwen3-8B; the full-corpus scaling experiment in Fig. 4 uses Qwen3-4B-Instruct-2507. Together, these experiments assess the objective's practical value as a surrogate for mitigating prefix failure.

Trajectory Sources and the Support Constraint. To examine the role of the support constraint, we hold the sampling budget fixed and vary the trajectory used for the OPSD update: the raw student rollout y_o, a trajectory y_t generated by Qwen3-14B, the reference solution $y^\star$, or the refined trajectory y_r. Here, Qwen3-14B supplies only y_t; the subsequent distillation update still uses the privileged

Table 6: Matched-budget Best-of-N filtering in the privileged self-teacher regime (%). All Best-of-N rows, including Best-of-1, retain only verifier-passing y_o trajectories; TRD trains on y_r. Bold indicates the best or tied-best result in each model block.

		Avg@16					Pass@16				
Method	Traj.	AIME24	AIME25	HMMT25	BeyondAIME	AMOBench	AIME24	AIME25	HMMT25	BeyondAIME	AMOBench
Qwen3-4B-Instruct-2507											
Base	–	59.1	42.9	28.3	28.2	8.1	73.3	66.7	43.3	54.0	23.1
+OPSD	y_o	60.3	48.8	30.8	32.3	9.5	86.7	76.7	53.3	58.0	33.3
Best-of-1	y_o	60.7	47.2	29.3	32.1	9.0	86.7	73.3	50.0	58.0	20.5
Best-of-4	y_o	61.5	47.1	29.6	**32.7**	9.0	**90.0**	76.7	50.0	59.0	20.5
Best-of-8	y_o	60.6	46.5	30.2	32.1	10.6	86.7	76.7	**56.7**	61.0	33.3
TRD	y_r	**63.1**	**49.4**	**32.1**	**32.7**	**11.2**	86.7	**80.0**	**56.7**	**64.0**	**33.8**
Qwen3-8B											
Base	–	72.3	59.7	30.5	35.6	8.9	83.3	76.6	66.7	62.0	41.0
+OPSD	y_o	74.8	65.4	40.3	39.6	15.7	86.7	83.3	73.3	**68.0**	51.3
Best-of-1	y_o	75.0	67.7	40.1	40.1	15.8	86.7	83.3	73.3	64.6	46.2
Best-of-4	y_o	75.6	68.3	44.2	42.3	17.2	83.3	**86.7**	66.7	67.0	43.6
Best-of-8	y_o	74.0	68.8	44.0	**43.1**	17.0	86.7	**86.7**	66.7	**68.0**	51.3
TRD	y_r	**76.5**	**69.2**	**44.5**	42.8	**17.3**	**90.0**	**86.7**	**76.3**	**68.0**	**61.5**

self-teacher. Unlike student rollouts, y_t and $y^\star$ are not constrained to the current policy's support and may therefore violate the constraint in Eq. (4).

Tab. 5 shows that these alternative sources help on individual benchmarks but do not match refinement overall. The refined trajectory y_r is best or tied-best in 16 of 20 settings, including 9 of 10 Pass@16 settings. There are exceptions: on Qwen3-8B, y_t achieves higher AIME24 Avg@16 than y_r (77.9% vs. 76.5%), and $y^\star$ achieves higher AMOBench Avg@16 (19.9% vs. 17.3%). The overall pattern supports improving trajectory quality while retaining a connection to the student's own rollouts. This comparison is consistent with the support-constrained formulation, although it does not isolate support mismatch from other differences between trajectory sources.

Best-of-N on the Full Training Corpus. Best-of-N filtering offers a complementary way to pursue the objective while satisfying the support constraint: draw N trajectories from the current student and retain verifier-passing outputs. Increasing N raises the chance of obtaining a successful trajectory for each training problem. In Fig. 4, we sample from the full Qwen3-4B-Instruct-2507 training corpus and allow sampling cost to grow with N. Mean Avg@16 increases from 35.66% for Best-of-1 to 36.68% for Best-of-8, compared with 36.34% for vanilla OPSD. These gains support the usefulness of successful on-policy trajectories for downstream learning, consistent with Eq. (4) as a surrogate for mitigating prefix failure.

The gains come at a substantial compute cost, as anticipated in Sec. 5. Best-of-8 uses 42.9 hours of total wall-clock time compared with 5.5 hours for TRD, approximately 7.8× as much, yet remains below TRD on four benchmarks and ties only on BeyondAIME. Its mean Avg@16 also falls short of the 37.70% achieved by TRD. Thus, verifier-guided selection can improve the final policy, but refinement uses privileged guidance to obtain stronger supervision more efficiently than repeated unassisted sampling.

Best-of-N under a Matched Sampling Budget. We also fix the sampling budget to test whether Best-of-N remains competitive under resource constraints (Tab. 6). All Best-of-N variants, including Best-of-1, retain only verifier-passing student trajectories. Allocating more draws to each problem therefore trades coverage of distinct training problems for a higher chance of finding a successful trajectory. Best-of-8 matches or exceeds vanilla OPSD in 15 of 20 settings, compared with 10 for Best-of-1, but remains below TRD overall. Across all compared methods, TRD is best or tied-best in 18 of 20 settings; the exceptions are Qwen3-4B AIME24 Pass@16 and Qwen3-8B BeyondAIME Avg@16.

The two sampling regimes expose the same practical limitation: Best-of-N must spend additional draws to discover successful trajectories and provides no supervision for problems where all sampled candidates fail. TRD instead uses a failed rollout as context for privileged refinement, allowing those problems to contribute supervision through y_r. The results support optimizing the trajectory-

construction objective while favoring refinement as the more effective use of limited sampling resources.

B.4 Trajectory Analysis

We analyze training trajectories and correct evaluation rollouts to characterize how trajectory refinement changes supervision and solution length, complementing the coverage analysis in Sec. 6.3.

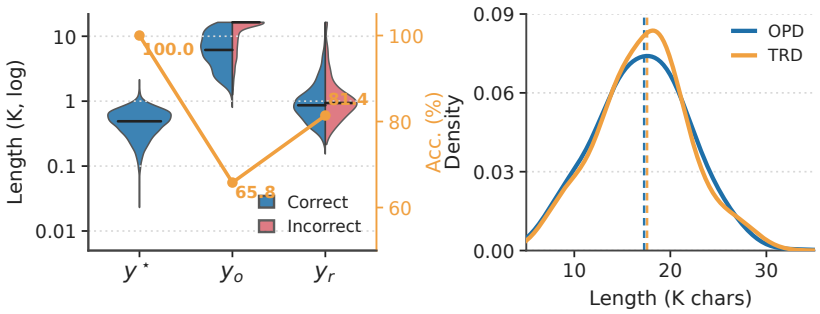


Figure 6: Trajectory evidence for the construction objective. **Left**: privileged self-teacher training-corpus trajectory length on Qwen3-8B, with the orange line showing verifier accuracy. **Right**: separate-teacher AMOBench correct-rollout length distribution for Qwen3-4B-Instruct-2507.

Training Trajectory Analysis (y_o vs. y_r). In the privileged self-teacher regime, the left panel of Fig. 6 contrasts $y^\star$, the raw rollout y_o, and the refined trajectory y_r over the Qwen3-8B training corpus. The constructed proposal increases the verifier-pass rate from 65.8% on y_o to 81.4% on y_r, while compressing the median length by roughly $9\times$, from 7.7K to 0.88K, toward the reference scale of roughly 0.49K. The same compression appears on the correct subset of y_o, where y_r exposes the student to alternative short derivations rather than only repairing failures. The shorter y_r also reduces Qwen3-8B training wall-clock by roughly 60%, partially offsetting the additional refinement pass (Appendix C.4).

Rollout Trajectory Analysis. We further analyze the AMOBench evaluation rollouts for Qwen3-4B-Instruct-2507 under the separate-teacher regime, using the same $K{=}128$ samples per question as in Sec. 6.3. The right panel of Fig. 6 shows that vanilla OPD and TRD produce highly similar correct-rollout length distributions, while TRD is slightly shorter on average (18.9K to 18.5K characters) with a modest density shift toward the 18K to 20K range. The broader coverage in Fig. 3 therefore accompanies only a small change in the lengths of successful solutions.

B.5 Test Rollout Analysis

Setup. We analyze AMOBench test rollouts from the Qwen3-8B checkpoints evaluated under OPSD in Tabs. 2 and 4. We compare the base model, vanilla forward-KL OPSD, and TRD, using $K{=}16$ completions per question and the generation and verification protocol in Sec. 6.1.

Base-difficulty buckets. Let $b_q \in \{0, \ldots, 16\}$ denote the number of correct base-model completions for question q. We partition the questions into three buckets: B0 ($n = 23$, $b_q = 0$), B1 to 8 ($n = 12$, $1 \le b_q \le 8$), and B9 to 16 ($n = 4$, $9 \le b_q \le 16$). The buckets are determined by the base model and held fixed when evaluating both trained methods.

Coverage on Questions Unsolved by the Base Model. Figure 7 shows that TRD solves 9 of the 23 questions in B0, achieving Pass@16 0.39, compared with 0.22 for vanilla forward-KL OPSD, the strongest dense-KL baseline on AMOBench Pass@16 in Tab. 4. This nearly doubles the baseline's coverage on the subset left unsolved by the base model under the same sampling budget.

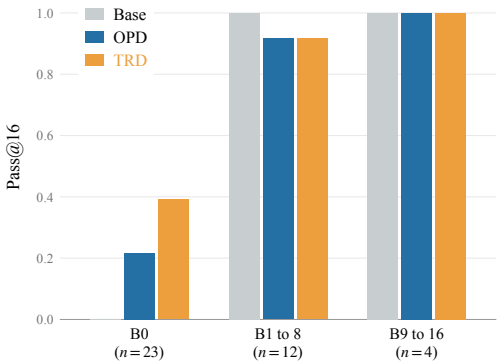


Figure 7: AMOBench Pass@16 by base-model difficulty for Qwen3-8B under OPSD, with $K=16$ completions per question. The buckets contain questions with 0, 1 to 8, or 9 to 16 correct base-model completions, respectively. B0 contains the 23 questions left unsolved by the base model in all 16 attempts. OPD denotes the vanilla forward-KL baseline.

Coverage on the Remaining Questions. In B1 to 8, both vanilla forward-KL OPSD and TRD achieve Pass@16 0.92, compared with 1.00 for the base model. All three methods achieve 1.00 in B9 to 16. The coverage advantage of TRD over vanilla forward-KL OPSD is therefore concentrated in B0.

B.6 Comparing Separate and Privileged Self-Teachers

Table 7: TRD comparison between the separate Qwen3-8B teacher and the privileged self-teacher on Qwen3-4B-Instruct-2507 math benchmarks. Bold marks the better value between the two teacher regimes.

	Avg@16					Pass@16				
Setting	AIME24	AIME25	HMMT25	BeyondAIME	AMOBench	AIME24	AIME25	HMMT25	BeyondAIME	AMOBench
Teacher	72.3	59.7	30.5	35.6	8.9	83.3	76.6	66.7	62.0	41.0
Separate teacher	**65.4**	47.9	**33.2**	32.6	10.3	83.3	76.7	50.0	62.0	**35.9**
Privileged self-teacher	63.1	**49.4**	32.1	**32.7**	**11.2**	**86.7**	**80.0**	**56.7**	**64.0**	33.8

Tab. 7 compares Avg@16 and Pass@16 between the two teacher regimes for TRD at the same Qwen3-4B-Instruct scale. Avg@16 is mixed: the separate teacher is stronger on AIME24 and HMMT25, while the privileged self-teacher is stronger on AIME25, BeyondAIME, and AMOBench. Pass@16 is generally stronger with the privileged self-teacher, which wins on four of five competition-math benchmarks.

The gain suggests that, for optimizing Eq. (4), privileged information can be more effective than teacher scaling: some questions may remain beyond the separate teacher's ability to refine, while the reference directly supplies the correct solution structure. Using a stronger separate teacher may reduce such refinement failures, but increases computational cost. Besides, because the privileged self-teacher refines with the student backbone conditioned on the reference, it potentially stays closer to the student's support and avoids mismatch between models (Fu et al., 2026; Li et al., 2026b).

C Experiment Details

This appendix follows the experimental workflow. We first describe training data and trajectory construction (Appendix C.1), followed by the OPD/OPSD and GRPO training configurations (Appendices C.2 and C.3). We then specify compute accounting, training diagnostics, and evaluation (Appendices C.4 to C.6). Complete initial-response and refinement prompts are collected in Appendix C.7.

C.1 Training Data and Trajectory Construction

Training uses DeepScaleR for math (Luo et al., 2025) and TACO for code (Li et al., 2023). For OPD/OPSD, Stage 1 samples a raw student rollout y_o conditioned on the problem; TRD adds Stage 2, in which the teacher refines y_o into y_r. We train on the selected trajectories for one epoch and merge the LoRA adapters before evaluation. Tab. 8 lists the generation settings, and Appendix C.7 provides the complete prompts.

Table 8: Training generation configuration. Stage 1 constructs y_o for the OPD/OPSD variants; Stage 2 constructs y_r only for TRD.

Setting	Stage 1: y_o	OPD Stage 2: y_r	OPSD Stage 2: y_r
Samples per problem	1	1	1
Temperature	0.6	0.6	0.6
Top-p	0.95	0.95	0.95
Top-k	20	20	20
Prompt budget	Math: 4,096; code: 2,048 tokens	18,432 tokens	22,528 tokens
Response budget	16,384 tokens	16,384 tokens	16,384 tokens
Maximum model length	Math: 20,480; code: 18,432 tokens	34,816 tokens	38,912 tokens

C.2 OPD/OPSD Hyperparameters

Models and teacher conditioning. OPD uses a frozen separate Qwen3-8B teacher and Qwen3-1.7B or Qwen3-4B-Instruct-2507 students (Yang et al., 2025). The teacher branch is never updated and its logits are detached before KL computation. OPSD uses Qwen3-4B-Instruct-2507 and Qwen3-8B as shared teacher/student backbones: the same base model produces privileged teacher logits under reference-solution conditioning, while LoRA adapters update only the student branch.

OPD and OPSD share the math/code corpora, prompt adapters, Stage-1 rollout sampler, KL implementation, optimizer, LoRA configuration, FSDP2 execution path, and evaluation sampling parameters. They differ in teacher identity, the teacher's access to $y^\star$, and the longer OPSD prompt budgets needed to include $y^\star$.

Shared optimization. All OPD/OPSD variants and TRD use AdamW, bfloat16, gradient checkpointing, and one training epoch, with the shared settings in Tab. 9. On one 8-GPU node, the per-GPU batch is 1 and gradient accumulation is 16, giving effective batch 128.

We apply LoRA of rank r=64, scaling α=128, dropout 0.05, on all attention and MLP linear layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`). After training we merge the adapters into the base weights before evaluation.

Table 9: Common optimization and generation hyperparameters used by the direct OPD/OPSD recipes.

Setting	Value
Optimizer	AdamW (β_1=0.9, β_2=0.999, ϵ=10^{-8})
Peak learning rate	5×10^{-6}
Precision	bfloat16
Gradient checkpointing	enabled
Per-GPU train batch	1
Gradient accumulation	16
Gradient clipping	1.0
LR schedule	linear warmup, cosine decay to 0.1× peak LR
Warmup ratio	0.1
Weight decay	0.005
Epochs per update	1
Full-vocab KL chunk size	512 tokens
LoRA save/merge	save adapter checkpoints and merge before evaluation
Sequence packing	remove-padding via verl FSDP2
Sequence parallel	ulysses, size 1
Rollout generation	temperature 0.6, top-p=0.95, top-k=20, max sequences 64

Distillation losses and method-specific settings. We use KL temperature T=1.0 and the Qwen3 vocabulary ($|\mathcal{V}| \approx$152K), with teacher top-32 truncation for the Top-K variant. Tab. 10 summarizes the trajectory, loss, teacher prompt, and context-length settings for each method.

Full-vocabulary variants compute the per-token KL at temperature T as

$$D^{(T)}\big(p \,\|\, q\big) \;=\; \sum_{v \in \mathcal{V}} p_T(v)\,\big(\log p_T(v) - \log q_T(v)\big), \tag{8}$$

with $p_T(v) \propto \exp(\ell_v/T)$. The clipping baseline applies a per-token cap $\min(D_{\mathrm{KL},t}, c)$ before averaging over the response mask, with c=0.1 in both OPD and OPSD. Top-K replaces $\mathcal{V}$ by the teacher's top-32 support $\mathcal{S}_t$ and renormalizes both p_T and q_T to sum to 1 on $\mathcal{S}_t$ before evaluating D.

Table 10: Method-specific OPD/OPSD settings. Clipped forward KL uses c=0.1 in both regimes. Max-length columns are training-time model lengths for teacher-forced KL.

Method	Traj.	KL dir.	Clip c	Top-K	Teacher prompt	OPD / OPSD max len.
Forward KL	y_o	forward	0	–	vanilla	18,432 / 22,528
Forward KL w/ Clip	y_o	forward	0.1	–	vanilla	18,432 / 22,528
Reverse KL	y_o	reverse	0	–	vanilla	18,432 / 22,528
Reverse KL w/ Top-K	y_o	reverse	0	32	vanilla	18,432 / 22,528
TRD (ours)	y_r	forward	0	–	refine	34,816 / 38,912

C.3 GRPO Algorithm and Hyperparameters

We implement GRPO (Shao et al., 2024) in verl using the task-specific configurations.[†] For each problem, we sample G=8 responses from the current policy and compute a scalar reward r_i for each response. The group-relative advantage is $A_i = (r_i - \bar{r})/(s_r + 10^{-6})$, where $\bar{r}$ and s_r are the within-group reward mean and sample standard deviation. This advantage is shared across the response tokens. The policy uses a clipped token-level objective with token-mean aggregation and a KL penalty to the frozen initial policy, without a learned critic. The KL penalty is applied in the actor loss; entropy regularization is disabled. Tab. 11 specifies the training configuration.

Rewards. For math, the reward is 0.67 for a correct extracted answer plus 0.33 for satisfying the prescribed `<think>` and `<answer>` format with a boxed answer. For code, the reward is 1 when all TACO tests pass; otherwise, it is the fraction passed among the first 10 tests (or all tests when fewer are available). Reference answers and tests are used for reward computation, while the policy is conditioned on the problem.

Table 11: GRPO training hyperparameters.

Setting	Value
Parameter updates	Full model, bfloat16, FSDP2
Optimizer	AdamW, β_1=0.9, β_2=0.999
Learning rate	10^{-6}, constant, no warmup
Weight decay / gradient clipping	0.01 / 1.0
Rollouts per problem	8
Problems per rollout batch	64
Responses per optimizer minibatch	256
Optimization epochs per rollout batch	1
Policy clipping	Lower/upper widths 0.2
Reference-policy KL	Coefficient 0.001
Training sampler	Temperature 1.0, top-p=1.0, top-k= -1
Maximum prompt length	Math: 2,048; code: 4,096 tokens
Maximum response length	16,384 tokens
Dynamic microbatch token budget per GPU	32,768; 18,432 for Qwen3-8B math

The rollout-and-training budget follows the GPU-time matching protocol in Appendix C.4.

C.4 Hardware and Compute

All runs use a single node of 8× H100 80GB GPUs with FSDP2 sharding via verl (Sheng et al., 2024).

[†]The configurations are available in `recipe/opd/run/grpo`.

GPU-time matching. We match GRPO and TRD by total GPU time covering all rollout generation and training, including both the raw-rollout and refinement stages for TRD. With the same 8 H100 GPUs, this is equivalent to matching the elapsed rollout-and-training time. The GRPO timer starts before the first rollout, and training stops after the current update completes once the time budget is reached, so matching is subject to the granularity of one update. Initialization, checkpoint saving, and final evaluation are outside this budget.

Measured wall-clock. Tab. 12 reports the training-pipeline wall-clock by model, setting, and method. The main trade-off is that TRD adds an extra sampling pass to construct y_r, but this overhead is partially offset by faster KL training because the refined trajectories are much shorter than y_o (see Appendix B.4). This offset becomes more pronounced as the backbone grows: on Qwen3-8B, TRD and Vanilla OPSD have nearly matched total wall-clock (9:20 vs. 9:40).

Table 12: Approximate OPD/OPSD training-pipeline wall-clock on a single 8× H100 80GB node. The y_o and y_r columns report rollout generation time; y_r is used only by TRD. Training covers the KL update only. Times are rounded to 10-minute bins.

Model	Setting	Method	y_o rollout	y_r rollout	Training	Total
Qwen3-1.7B	OPD	Vanilla OPD	3:30	–	1:10	4:40
		TRD	3:30	4:00	0:40	8:10
Qwen3-4B-Instruct-2507	OPD	Vanilla OPD	4:00	–	1:20	5:20
		TRD	4:00	4:00	1:00	9:00
	OPSD	Vanilla OPSD	2:10	–	2:10	4:20
		TRD	2:10	2:00	1:20	5:30
Qwen3-8B	OPSD	Vanilla OPSD	4:20	–	5:20	9:40
		TRD	4:20	2:50	2:10	9:20

C.5 Training Diagnostics

The diagnostic curves in Fig. 5, discussed in Appendix B.1, use the privileged self-teacher setting to control for differences in teacher and student backbone models. The same trainer can log these metrics for OPD, but the reported OPD direct-variant runs keep them off unless explicitly enabled. Let z_i denote the response scored in a run: $z_i = y_{o,i}$ for standard OPSD and $z_i = y_{r,i}$ for TRD. We use response-token masks $m_{i,t} \in \{0, 1\}$ and aggregate within each logging step. KL and epistemic-token mass are token-weighted means; perplexity is the exponential of a token-weighted mean negative log-likelihood. Prompt conditioning follows Sec. 5 and is suppressed in the probabilities below: the student receives the problem, while the teacher receives privileged reference information and, for TRD, the original rollout.

Per-token KL by rollout outcome. Each prompt x_i carries a fixed binary stage-1 outcome label b_i, set to 1 (*correct*) if the base model's original rollout $y_{o,i}$ passes the verifier and to 0 (*incorrect*) otherwise. The paired refined response $y_{r,i}$ inherits this label, so the bucket names do not describe the refined response's own correctness. The per-token KL between teacher and student is averaged separately within each bucket,

$$
\begin{aligned}
D^{\text{correct}} &= \frac{\sum_{i:\, b_i=1} \sum_t D_{\mathrm{KL},i,t}\, m_{i,t}}{\sum_{i:\, b_i=1} \sum_t m_{i,t}}, \\
D^{\text{incorrect}} &= \frac{\sum_{i:\, b_i=0} \sum_t D_{\mathrm{KL},i,t}\, m_{i,t}}{\sum_{i:\, b_i=0} \sum_t m_{i,t}},
\end{aligned}
\tag{9}
$$

where $D_{\mathrm{KL},i,t}$ uses the KL direction configured for that run and is evaluated at prefix $z_{i,<t}$. These are means per unmasked token, rather than trajectory sums. Normalizing by token count removes the direct scaling with the number of evaluated tokens, but does not equate the underlying prefix distributions or the forward- and reverse-KL objectives.

Teacher-student perplexity gap. Both perplexities use teacher-forced decoding on the same response tokens and mask within each run,

$$\begin{aligned} \mathrm{PPL}_S &= \exp\left(\frac{\sum_{i,t} -\log \pi_S(z_{i,t} \mid z_{i,<t})\, m_{i,t}}{\sum_{i,t} m_{i,t}}\right), \\ \mathrm{PPL}_T &= \exp\left(\frac{\sum_{i,t} -\log \pi_T(z_{i,t} \mid z_{i,<t})\, m_{i,t}}{\sum_{i,t} m_{i,t}}\right). \end{aligned} \tag{10}$$

We report $\mathrm{PPL}_S - \mathrm{PPL}_T$ using $T{=}1$ log-probabilities. This convention gives a consistent metric definition across runs, but runs scored on y_o and y_r evaluate different responses. Their gaps therefore describe likelihood differences on the respective response distributions, rather than a controlled comparison of supervision quality on a shared evaluation corpus.

Epistemic-token mass. Let $\mathcal{E} \subset \mathcal{V}$ be the set of *epistemic onset* tokens. We construct $\mathcal{E}$ from 16 phrases,

`Wait`, `Actually`, `However`, `Alternatively`, `Oops`, `Wrong`, `Error`, `Incorrect`, `Correction`, `Sorry`, `Hmm`, `Oh`, `Hold`, `Pause`, `Uh`, `Um`,

by tokenising each phrase under the student tokenizer in two variants (bare string and leading-space) and collecting the first sub-word id, then deduplicating. Thus, 16 is the number of source phrases, not necessarily the number of distinct token ids. At each response position (i, t) we measure how much teacher mass falls on $\mathcal{E}$ before any KL temperature scaling,

$$\mathrm{mass}_{i,t} = \sum_{v \in \mathcal{E}} \pi_T(v \mid z_{i,<t}) = \sum_{v \in \mathcal{E}} \mathrm{softmax}\big(\boldsymbol{\ell}^T_{i,t}\big)_v, \tag{11}$$

and report the token-weighted mean $\big(\sum_{i,t} \mathrm{mass}_{i,t}\, m_{i,t}\big) / \big(\sum_{i,t} m_{i,t}\big)$. The metric is computed only on the full-vocabulary path (i.e., when teacher logits are materialised) and is independent of the KL-loss temperature.

C.6 Evaluation Settings

Benchmarks. The math suite contains AIME24/25 (AI-MO, 2024; OpenCompass, 2025), HMMT25 (HMMT, 2025), BeyondAIME (Seed, 2025), and the 39 parser-graded AMOBench problems (An et al., 2025). The code suite contains HumanEval+, MBPP+ (Liu et al., 2023), and LiveCodeBench v6 (Jain et al., 2024).

Sampling and verification. Tab. 13 summarizes the evaluation configuration. We use $K{=}16$ completions per problem; temperature and response length follow the Qwen3 evaluation setup (Yang et al., 2025). OPD/OPSD code evaluation uses up to 128 concurrent sequences.

Table 13: Evaluation configuration. The $K{=}16$ sampling budget is our reporting protocol for Avg@16 and Pass@16.

Setting	Math evaluation	Code evaluation
Samples per problem	$K{=}16$	$K{=}16$
Temperature	0.6	0.6
Top-p	0.95	0.95
Top-k	20	20
Prompt budget	4,096 tokens	2,048 tokens
Response budget	38,912 tokens	16,384 tokens
Maximum model length	43,008 tokens	18,432 tokens
Verifier	rule-based boxed-answer verifier	benchmark unit-test executor

Math completions are scored with answer extraction from the final `\boxed{...}` block. HumanEval+ and MBPP+ are evaluated through EvalPlus; LiveCodeBench uses the `lcb_runner` code-generation scenario with release version 6.

Metrics. For each test question x_q we draw $K{=}16$ independent completions $y_q^{(i)} \sim \pi_\theta(\cdot \mid x_q)$ and score them with a verifier $\mathrm{Verify}(\cdot,\cdot)$: the boxed-answer verifier for math and unit tests for code. Over N test questions,

$$\mathrm{Avg}\,@K \;=\; \frac{1}{N}\sum_{q=1}^{N}\frac{1}{K}\sum_{i=1}^{K}\mathrm{Verify}\big(y_q^{(i)},\, y_q^{\star}\big), \quad \mathrm{Pass}\,@K \;=\; \frac{1}{N}\sum_{q=1}^{N}\max_{1\le i\le K}\mathrm{Verify}\big(y_q^{(i)},\, y_q^{\star}\big).$$

Avg@16 tracks average sample quality; Pass@16 tracks whether at least one of the 16 samples solves the problem.

Evaluation cost. Tab. 14 reports evaluation time separately from rollout generation and training.

Table 14: Approximate evaluation wall-clock on a single 8× H100 80GB node. Times are rounded to 30-minute bins and reported by model only.

Model	Math suite	Code suite
Qwen3-1.7B	2:00	6:30
Qwen3-4B-Instruct-2507	2:00	3:00
Qwen3-8B	4:00	–

C.7 Initial and Refinement Prompts

Stage-1 initial-response prompts are task-specific but shared by OPD and OPSD. They produce the raw rollout y_o used by the dense-KL baselines and by the Stage-2 refinement prompts. The refinement prompt is also task-specific and differs between OPD and OPSD only in whether the reference solution $y^\star$ is shown. OPD refinement uses the separate teacher and hides $y^\star$; OPSD refinement uses the shared model under privileged conditioning and includes $y^\star$.

Math Initial Response.

> {Problem}
> Please reason step by step, and put your final answer within `\boxed{}`.

Code Initial Response.

{PROBLEM}
Starter code: *optional*
```python
{STARTER CODE}
```
You will be given a programming problem. Write a correct Python program that solves it. Return only the code inside a single ```python code block.

OPD Math Refinement.

Your task is to rewrite your mathematical solution.
Problem:
{PROBLEM}
Your Initial Solution:
{INITIAL RESPONSE}
Instructions:

1. Preserve the overall structure and reasoning path of your original solution
2. Identify and fix errors in computation or logic
3. Keep correct intermediate steps and meaningful work
4. Output ONLY the rewritten solution

Please reason step by step, and put your final answer within `\boxed{}`.

OPSD Math Refinement.

Your task is to rewrite your mathematical solution using the reference solution as guidance.
Problem:
{PROBLEM}
Reference Solution:
{EXPERT SOLUTION}
Your Initial Solution:
{INITIAL RESPONSE}
Instructions:

1. Review the reference solution to understand the target reasoning and method
2. Rewrite your solution so it is consistent with the reference solution
3. Keep useful parts of your original structure and style when appropriate
4. Output ONLY the rewritten solution

Please reason step by step, and put your final answer within `\boxed{}`.

OPD Code Refinement.

Your task is to rewrite your Python solution.
Problem:
{PROBLEM}
Your Initial Solution:
{INITIAL RESPONSE}
Instructions:

1. Fix correctness issues and edge cases
2. Preserve useful parts of the original approach when appropriate
3. Output ONLY the rewritten Python solution

Return only the corrected Python code inside a single ```python code block.

OPSD Code Refinement.

Your task is to rewrite your Python solution using the reference solution as guidance.
Problem:
{PROBLEM}
Reference Solution:
```python
{EXPERT SOLUTION}
```
Your Initial Solution:
{INITIAL RESPONSE}
Instructions:

1. Fix correctness issues and edge cases
2. Preserve useful parts of the original approach when appropriate
3. Output ONLY the rewritten Python solution

Return only the corrected Python code inside a single ```python code block.