VIABLEOPD: GUIDING ON-POLICY DISTILLATION WITH PREFIX VIABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) transfers teacher capabilities by providing dense, token-level teacher supervision on student-generated trajectories. This paradigm implicitly assumes that the entire trajectory can benefit from teacher supervision. However, once a student-generated prefix deviates from the task objective, teacher supervision is conditioned on an erroneous context, making distillation at such states ineffective or even harmful. To characterize when teacher supervision remains effective, we introduce *prefix viability*, defined as the probability that the original task objective can still be achieved from a student-generated prefix. We estimate prefix viability by normalizing the teacher's success probability conditioned on the prefix by its success probability from the original input. Based on this criterion, we propose Viability-Guided On-Policy Distillation (ViableOPD), which identifies viable rollouts for teacher supervision, adjusts token-level supervision strength within selected rollouts, and adapts the rollout horizon according to the fraction of viable rollouts. Across multiple benchmarks and teacher–student pairs, ViableOPD consistently outperforms strong OPD baselines while reducing training time by 33.6% on average relative to standard OPD. These results suggest that effective OPD requires prefixes to be not only visited by the student, but also viable for continued supervision.

1 INTRODUCTION

On-policy distillation (OPD) transfers the capabilities of a teacher model to a student model by providing dense, token-level teacher supervision over trajectories generated by the student (Gu et al., 2024; Zhao et al., 2026b). Unlike offline distillation, which relies on fixed training data such as pre-collected teacher trajectories or offline datasets, OPD obtains teacher feedback at the states the student visits during generation, reducing the distribution mismatch between training and inference (Kim & Rush, 2016; Agarwal et al., 2024). However, this paradigm implicitly relies on a critical assumption: every state along a student-generated trajectory can benefit from teacher supervision.

This assumption does not always hold. During generation, a student model may gradually deviate from the original task objective due to local errors, inappropriate decisions, or unsupported intermediate assumptions. Although subsequent generations may remain coherent and locally plausible, the student has already entered a state from which the original task objective is unlikely to be achieved. In such cases, the teacher can still provide supervision based on the current state, but this supervision no longer effectively guides the student toward a valid output; instead, it encourages imitation of the teacher's continuation from an erroneous context. Therefore, distilling from such states not only fails to transfer teacher capabilities effectively but may also reinforce task-misaligned generation patterns, ultimately degrading distillation performance.

To characterize whether a student-generated state remains suitable for teacher supervision, we introduce *prefix viability*, defined as the probability that the original task objective remains achievable when generation continues from a prefix. To disentangle degradation caused by the student-generated prefix from the intrinsic task difficulty, we estimate prefix viability by comparing the teacher model's success probability conditioned on the student-generated prefix with that obtained when solving the same task directly from the original input.

Based on this task-level criterion, we propose Viability-Guided On-Policy Distillation (ViableOPD), a framework that uses prefix viability to guide supervision during on-policy distillation. Specifically,

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.

ViableOPD evaluates the viability of student rollouts and performs distillation only on rollouts that remain viable. Within viable regions, it further adjusts the token-level supervision strength according to the estimated distillation utility of each token. Moreover, ViableOPD adaptively updates the rollout horizon according to the fraction of viable rollouts, allowing the supervision horizon to evolve with the student's capability. By distinguishing student-visited states from viable states, ViableOPD enables OPD to focus teacher guidance on states where it can provide effective supervision.

We evaluate ViableOPD across multiple teacher–student pairs and benchmarks. ViableOPD consistently outperforms standard OPD and strong alternatives, while reducing training time by 33.6% on average. These results suggest that effective OPD requires more than supervising the states visited by the student: it requires determining whether those states remain viable for distillation.

Our contributions are summarized as follows:

- We identify a fundamental limitation of OPD: not all student-generated states can benefit from teacher supervision. When a state has deviated from the original task objective, teacher guidance conditioned on it may become ineffective or misleading.
- We introduce prefix viability, a task-level criterion that characterizes whether a student-generated state remains suitable for teacher supervision, providing a principled measure of OPD effectiveness.
- We propose ViableOPD, a viability-guided OPD framework that leverages prefix viability as a task-level criterion to determine supervision allocation and adaptively control the rollout horizon during on-policy distillation.
- Extensive experiments across multiple teacher–student pairs and benchmarks demonstrate that ViableOPD consistently improves over standard OPD and strong baselines, achieving better performance with substantially reduced training cost.

2 Standard OPD and the Implicit Assumption

Let π_θ denote the student policy and π_T a teacher policy. Given a problem $\mathbf{x} \sim \mathcal{D}$, the student generates a rollout $\mathbf{y} = (y_1, \ldots, y_L) \sim \pi_\theta(\cdot \mid \mathbf{x})$. Each generation step induces a student-visited state $s_l = (\mathbf{x}, y_{<l})$, at which both models define a distribution over the next token.

Standard OPD applies teacher supervision at these student-visited states (Agarwal et al., 2024; Zhao et al., 2026b). Using reverse KL for concreteness, the population loss can be written as

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\left[\frac{1}{L}\sum_{l=1}^{L} D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid s_l) \,\|\, \pi_T(\cdot \mid s_l)\right)\right]. \tag{1}$$

This formulation implicitly assumes that every student-generated state can benefit from teacher supervision. However, this assumption does not always hold. During generation, a student may gradually deviate from the original task objective, entering a task-misaligned state from which achieving the original objective becomes unlikely (Lightman et al., 2023; Wang et al., 2023). Although the teacher can still provide supervision conditioned on this state, this supervision no longer reliably directs the student toward the original task objective. Instead, it teaches the student to imitate the teacher's continuation under an unreliable context. Consequently, applying distillation on such states may fail to transfer useful teacher capabilities and even reinforce undesirable generation behaviors, ultimately impairing distillation effectiveness.

3 Prefix Viability in On-Policy Distillation

3.1 Definition of Prefix Viability

To characterize whether a student-generated state remains valid for teacher supervision, we introduce *prefix viability* V, defined as the probability that the original task objective can still be achieved when generation continues from a student-generated prefix.

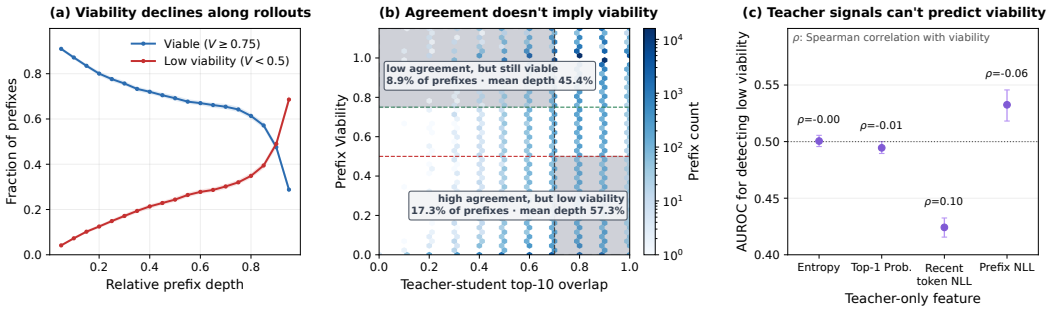

Figure 1: Diagnosing prefix viability in OPD. (a) Viability declines as student rollouts progress. (b) Teacher–student agreement is not a reliable indicator of viability. (c) Teacher–only local statistics show near–chance performance in detecting low-viability prefixes.

Specifically, c denotes a teacher-generated continuation, and $\mathrm{Verify}(\cdot)$ is the task-specific verifier that returns 1 if the generated solution satisfies the original task objective and 0 otherwise. For a problem $\mathbf{x}$, let $P_0(\mathbf{x})$ denote the probability that the teacher solves the problem from scratch:

$$P_0(\mathbf{x}) = \Pr_{c \sim \pi_T(\cdot|\mathbf{x})} \left[\mathrm{Verify}(\mathbf{x}, c) = 1\right]. \tag{2}$$

Given a student prefix $y_{\leq l}$, we similarly define the prefix-conditioned completion probability:

$$P_l(\mathbf{x}, y_{\leq l}) = \Pr_{c \sim \pi_T(\cdot|\mathbf{x}, y_{\leq l})} \left[\mathrm{Verify}(\mathbf{x}, y_{\leq l} \oplus c) = 1\right]. \tag{3}$$

We estimate prefix viability of state $s_l = (\mathbf{x}, y_{\leq l})$ by the normalized ratio:

$$\hat{V}(s_l) = \frac{P_l(\mathbf{x}, y_{\leq l})}{P_0(\mathbf{x})}. \tag{4}$$

Normalizing by P_0 accounts for the intrinsic difficulty of the task and measures the effect of the student prefix. When $P_0(\mathbf{x}) = 0$, we use the prefix-conditioned task completion probability $P_l(\mathbf{x}, y_{\leq l})$ directly for viability estimation. Values greater than one can occur when conditioning on the prefix improves the teacher's success probability.

3.2 Visited Does Not Mean Viable

To examine whether student-visited states remain viable throughout generation, we conduct an analysis on DAPO (Yu et al., 2025), a large-scale mathematical reasoning dataset, using DeepSeek-R1-Distill-Qwen-1.5B as the student and JustRL-DeepSeek-1.5B as the teacher (DeepSeek-AI et al., 2025). For each student rollout, we estimate the viability of intermediate states and categorize states with viability score below 0.5 as low-viability and those above 0.75 as high-viability.

Figure 1(a) shows that prefix viability consistently decreases as generation proceeds. Specifically, 38.4% of rollouts enter the low-viability region before termination, and 45.6% of all generated tokens are produced after the trajectory has entered this region. These results indicate that many student-visited states fall into low-viability regions where the original task objective is unlikely to be achieved, yet standard OPD continues to apply teacher supervision to these states.

3.3 Local Signals Do Not Reveal Viability

Existing approaches for evaluating the effectiveness of OPD supervision often rely on local signals, such as teacher–student agreement (Wang et al., 2026; Li et al., 2026) and teacher-side prediction statistics (Jin et al., 2026). To examine whether existing local signals can reliably identify viable states, we evaluate their capability to predict prefix viability on student-generated prefixes.

We first examine teacher–student top-10 overlap, which is commonly used to assess the teacher–student agreement. As shown in Figure 1(b), prediction agreement does not reliably reflect prefix viability. High agreement may occur in low-viability states, where the teacher and student produce

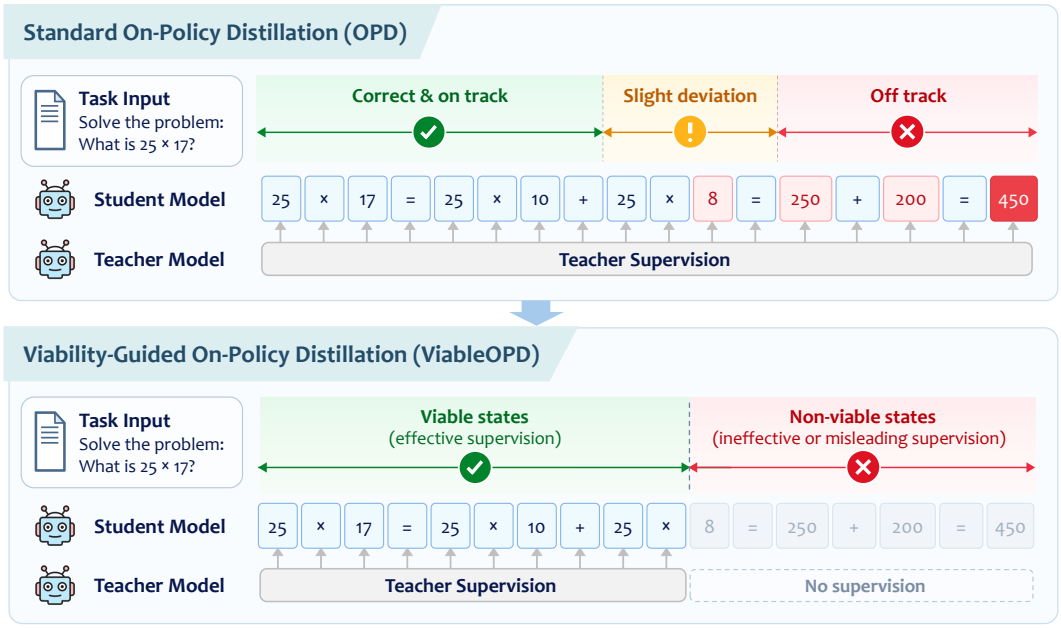

Figure 2: Overview of ViableOPD. Standard OPD applies teacher supervision to all student-generated states, whereas ViableOPD restricts distillation to regions where the original task remains achievable.

similar predictions conditioned on a task-misaligned context. Conversely, low agreement may appear in viable states where the two models follow different but valid generation paths. We further analyze teacher-only local statistics, including prediction uncertainty, confidence, and prefix likelihood measures. As shown in Figure 1(c), these signals exhibit near-zero correlation with prefix viability and achieve near-chance performance in identifying low-viability prefixes.

These results reveal that local signals reflect local teacher–student behaviors but cannot determine whether the original task remains completable from that state. Prefix viability captures this task-level property, providing a more principled criterion for identifying effective supervision regions.

4 Method: ViableOPD

In this section, we introduce ViableOPD, a framework that uses prefix viability to guide the distillation process. An overview of ViableOPD is shown in Figure 2.

4.1 Viability–Guided Supervision Selection

For a student-generated state $s_l = (\mathbf{x}, y_{\le l})$, we determine whether the state remains viable for supervision based on whether the original task can still be successfully completed from the current state. Specifically, we define a binary supervision indicator $I_{\text{via}}(s_l)$ as

$$I_{\text{via}}(s_l) = \begin{cases} \text{Verify}(\mathbf{x}, y_{\le l}), & \text{if the generation has terminated,} \\ \mathbb{I}[\hat{V}(s_l) \ge \tau_v], & \text{otherwise.} \end{cases} \tag{5}$$

where τ_v is the threshold applied to the prefix viability score $V(s_l)$.

Although $I_{\text{via}}(s_l)$ is defined for arbitrary intermediate states, precisely identifying the transition from viable to non-viable regions requires viability estimation over multiple prefixes within each rollout. Accordingly, we apply the viability criterion at the rollout level. For a batch of B sampled rollouts $\{y^{(i)}\}_{i=1}^{B}$ with a generation horizon H, ViableOPD evaluates the viability of each rollout at its boundary state $s_H^{(i)} = (\mathbf{x}^{(i)}, y_{\le H}^{(i)})$ and restricts distillation to viable rollouts:

$$\mathcal{L}_{\text{ViableOPD}} = \frac{\sum_{i=1}^{B} I_{\text{via}}(s_H^{(i)}) \ell_{\text{distill}}(y_{\le H}^{(i)})}{\sum_{i=1}^{B} I_{\text{via}}(s_H^{(i)})}, \tag{6}$$

where $\ell_{\text{distill}}(y_{\le H}^{(i)})$ denotes the OPD loss computed over the i-th rollout. When a batch of rollouts contains no viable rollouts, we skip the update to avoid optimizing with ineffective supervision.

4.2 Learning an Online Viability Judge

Although prefix viability provides the desired criterion for supervision selection, directly estimating it during online distillation is computationally prohibitive. We therefore estimate prefix viability offline to obtain binary supervision labels and train a lightweight viability judge for online distillation.

Specifically, for each student-generated state $s_l = (\mathbf{x}, y_{\le l})$, we first estimate its empirical viability $\hat{V}(s_l)$ using the continuation-based procedure described in Section 3.1, and define the binary supervision label as $\mathbb{I}[\hat{V}(s_l) \ge \tau_v]$.

Based on this dataset, we train a parameterized viability judge J_ϕ with a binary cross-entropy loss to predict the probability that a student prefix remains viable for supervision:

$$J_\phi(s_l) = J_\phi(\mathbf{x}, y_{\le l}) \in [0, 1]. \tag{7}$$

During online distillation, we approximate the ideal indicator I_{via} using the viability judge J_ϕ:

$$\hat{I}_{\text{via}}(s_l) = \begin{cases} \text{Verify}(\mathbf{x}, y_{\le l}), & \text{if the generation has terminated,} \\ \mathbb{I}[J_\phi(s_l) \ge \tau_J], & \text{otherwise.} \end{cases} \tag{8}$$

where τ_J is the threshold for converting the judge confidence into a binary viability decision.

4.3 Distillation within Viable Regions

Prefix viability determines where teacher supervision should be applied. Within viable regions, ViableOPD further adjusts supervision strength according to token-level distillation utility.

We consider three token-level signals: teacher–student top-k prediction overlap O, student top-k entropy E, and teacher–student projected JS divergence D. Tokens are assigned different supervision weights based on these signals: High-agreement tokens receive full supervision, informative low-agreement tokens retain partial supervision, and other tokens are down-weighted. Specifically, we assign the supervision weight $w_l^{(i)}$ to the l-th token of the i-th rollout as follows:

$$w_l^{(i)} = \begin{cases} 1, & O_l^{(i)} \ge \tau_O, \\ \lambda_{\text{info}}, & O_l^{(i)} < \tau_O \wedge \left(E_l^{(i)} \ge \tau_E \vee D_l^{(i)} \ge \tau_D \right), \\ \lambda_{\text{low}}, & O_l^{(i)} < \tau_O \wedge E_l^{(i)} < \tau_E \wedge D_l^{(i)} < \tau_D. \end{cases} \tag{9}$$

where τ_O, τ_E, and τ_D are the thresholds for the three token-level signals, and λ_{info} and λ_{low} denote the weights assigned to informative and low-value tokens, respectively.

The weighted distillation loss for the i-th rollout is defined as:

$$\ell_{\text{distill}}^{w}(y_{\le H}^{(i)}) = \frac{\sum_{l=1}^{H} w_l^{(i)} \ell_{\text{distill}}(y_l^{(i)})}{\sum_{l=1}^{H} w_l^{(i)}}. \tag{10}$$

where $\ell_{\text{distill}}(y_l^{(i)})$ denotes the distillation loss at the l-th position of the i-th rollout.

The final ViableOPD objective is:

$$\mathcal{L}_{\text{ViableOPD}} = \frac{\sum_{i=1}^{B} \hat{I}_{\text{via}}(s_H^{(i)}) \ell^w_{\text{distill}}(y^{(i)}_{\leq H})}{\sum_{i=1}^{B} \hat{I}_{\text{via}}(s_H^{(i)})}. \tag{11}$$

4.4 Adaptive Viability Horizon

Prefix viability defines a viability horizon that characterizes how far along a student trajectory teacher supervision remains valid. This horizon evolves with the student's capability: stronger students can maintain viable states over longer trajectories. A fixed generation horizon may therefore either extend supervision beyond this range or restrict learning to overly conservative regions. ViableOPD adaptively adjusts the generation horizon to track this evolving viability horizon.

Let H_k denote the generation horizon at optimization step k. We evaluate the viability of each rollout sampled at this step with $\hat{I}_{\text{via}}$ and compute the viability ratio:

$$r_k = \frac{1}{B} \sum_{i=1}^{B} \hat{I}_{\text{via}}(s_{H_k}^{(i)}). \tag{12}$$

Given a target range for the viability ratio, $[\alpha_{\text{low}}, \alpha_{\text{high}}]$, we update the generation horizon as

$$H_{k+1} = \begin{cases} \min(H_k + \Delta, H_{\max}), & r_k > \alpha_{\text{high}}, \\ \max(H_k - \Delta, H_{\min}), & r_k < \alpha_{\text{low}}, \\ H_k, & \text{otherwise.} \end{cases} \tag{13}$$

where Δ denotes the horizon update step size, and $H_{\min}$ and $H_{\max}$ denote the minimum and maximum allowable generation horizons, respectively.

5 Experiments

5.1 Experimental Setup

Tasks, Models and Metrics. We construct experiments on two types of tasks. For mathematical reasoning, we consider two student–teacher model pairs: DeepSeek-R1-Distill-Qwen-1.5B/JustRL-DeepSeek-1.5B (DeepSeek/JustRL) and OpenMath-Nemotron-1.5B/JustRL-Nemotron-1.5B (OpenMath/JustRL). Following previous work (Yang et al., 2026; Zhao et al., 2026a), we use DAPO-Math-17K (Yu et al., 2025) for training and evaluate performance on AMC23, AIME25, and HMMT25. For code generation tasks, we use TACO (10K sampled) (Li et al., 2023) for training and consider the Qwen2.5-Coder-1.5B-Base/Qwen2.5-Coder-7B-Base (Qwen2.5-1.5B/Qwen2.5-7B) student–teacher pair (Hui et al., 2024). The models are evaluated on HumanEval+, MBPP+, and LiveCodeBench. We report pass@1 accuracy as the performance metric and wall-clock training time to evaluate efficiency.

Baselines. We compare ViableOPD with standard OPD and several representative variants: (1) OPD: standard OPD with supervision on all student-generated tokens. (2) Fixed-Horizon: limits supervision to a predefined generation horizon. (3) Random-Horizon: assigns supervision horizons randomly across trajectories. (4) Prune-OPD (Yang et al., 2026): identifies unreliable states based on teacher–student compatibility and reduces supervision on subsequent tokens. (5) PG-OPD (Zhao et al., 2026a): selects trajectories for extended supervision based on teacher–student compatibility. All methods are trained under the same settings and optimization configurations for fair comparison.

Training Details. For all experiments, we use the AdamW optimizer with a learning rate of 1e-6. The models are trained for one epoch with a global batch size of 64 on four NVIDIA A100 GPUs. We fine-tune Qwen3-1.7B (Yang et al., 2025) and Granite-3.3-2B-Instruct (Granite Team, 2024) as viability judge models for mathematical reasoning and code generation, respectively. Each judge is trained on mixed prefixes from multiple teacher–student pairs in its domain and is shared across all experiments within that domain. Additional implementation details are provided in Appendix C.

Table 1: Performance comparison of different OPD variants. Results are reported in terms of pass@1 accuracy (%) and wall-clock training time. The best and second-best results are highlighted in **bold** and underlined, respectively.

Method	Mathematical Reasoning								Code Generation			
	DeepSeek / JustRL				OpenMath / JustRL				Qwen2.5-1.5B / Qwen2.5-7B			
	AMC23	AIME25	HMMT25	Time (h)	AMC23	AIME25	HMMT25	Time (h)	HumanEval+	MBPP+	LiveCodeBench	Time (h)
OPD	86.88	34.17	16.88	26.6	95.31	58.33	27.91	41.7	15.63	59.20	17.32	2.6
Fixed-Horizon	86.41	32.92	18.75	**11.7**	94.53	59.38	28.33	**12.4**	13.80	58.14	17.47	1.6
Random-Horizon	87.19	34.17	17.71	17.9	95.31	59.79	28.12	22.4	15.02	58.41	17.38	1.7
Prune-OPD	85.94	34.79	18.75	13.7	95.42	56.67	27.71	35.8	15.44	59.47	17.49	**1.0**
PG-OPD	86.56	32.50	20.00	15.2	94.69	57.92	27.50	23.1	13.80	58.14	17.10	2.0
ViableOPD	**89.22**	**35.42**	**21.88**	14.3	**96.14**	**61.12**	**30.71**	30.2	**16.24**	**60.51**	**18.35**	1.9

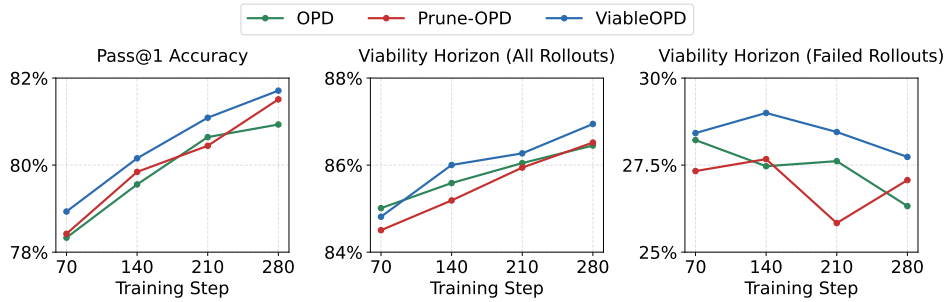

Figure 3: Evolution of rollout viability horizon during training on the OpenMath/JustRL mathematical reasoning setting. ViableOPD consistently achieves a longer viability horizon, indicating improved preservation of viable generation states.

5.2 Overall Performance of ViableOPD

Table 1 summarizes the performance of ViableOPD compared with baseline methods. Across different teacher–student pairs and benchmarks, ViableOPD consistently achieves better task performance than existing OPD variants, while maintaining lower training cost than standard OPD.

ViableOPD consistently outperforms standard OPD, confirming that student trajectories contain states with different supervision viability and that excluding non-viable states improves supervision effectiveness. Compared with Fixed-Horizon and Random-Horizon variants, the superior performance of ViableOPD demonstrates that effective supervision horizons should be determined by state viability rather than predefined rules. Moreover, its advantage over Prune-OPD and PG-OPD shows that task-level viability provides a more effective criterion for supervision selection than local token-level signals. Finally, ViableOPD achieves the best performance–efficiency trade-off among evaluated methods, delivering stronger performance without the full computational cost of standard OPD.

Although ViableOPD reduces online cost, it requires a one-time offline investment to obtain viability supervision and train the viability judge. For mathematical reasoning, this process requires approximately 500 A100 GPU-hours, which is amortized after 10–11 distillation runs given the 46.0–49.2 A100 GPU-hours saved per run. For code generation, the corresponding cost is approximately 58 A100 GPU-hours, leading to a break-even point of 20–21 runs with a saving of 2.8 A100 GPU-hours per run. Therefore, the judge construction cost represents an amortizable one-time investment, making ViableOPD increasingly cost-effective when the judge is reused across multiple distillation runs.

5.3 Evolution of Viability Horizon During Training

To further understand the effect of viability-aware supervision, we track the evolution of the viability horizon of student-generated trajectories during training on the OpenMath/JustRL mathematical reasoning setting. Specifically, at each checkpoint, we sample student rollouts and estimate their

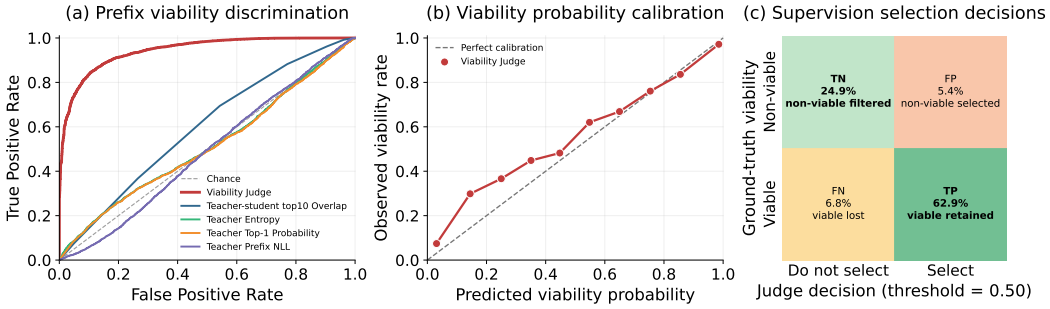

Figure 4: Effectiveness of the mathematical reasoning viability judge. (a) ROC curves of the viability judge and existing local signals for prefix viability discrimination. (b) Calibration curve of the predicted viability probability. (c) Supervision selection analysis under the deployment threshold.

Table 2: Ablation study of key components in ViableOPD on the DeepSeek/JustRL mathematical reasoning setting. Results are reported as pass@1 accuracy (%). The best and second-best results are highlighted in **bold** and underlined, respectively.

Method	Prefix Selection	Token Weighting	AMC23	AIME25	HMMT25
Effect of Prefix Selection					
Random Selection	Random	Utility-Aware	87.03	32.29	17.08
Overlap-based Selection	Top-k Overlap	Utility-Aware	86.88	34.17	18.33
Effect of Token-Level Weighting					
Uniform Weighting	Viability Judge	Uniform	<u>88.34</u>	<u>34.73</u>	20.47
Shuffled Weighting	Viability Judge	Shuffled	87.95	34.58	<u>21.04</u>
Our Method					
ViableOPD	Viability Judge	Utility-Aware	**89.22**	**35.42**	**21.88**

viability horizons as the relative positions where prefix viability first falls below the predefined threshold, and then average these values across rollouts.

As shown in Figure 3, ViableOPD consistently maintains a longer viability horizon than OPD and Prune-OPD throughout training. This indicates that viability-aware supervision enables the student to remain in viable generation states for longer and reduces the likelihood of entering task-misaligned regions. Furthermore, this advantage is also observed on failed rollouts, suggesting that the improvement is not solely attributed to higher final accuracy. Instead, ViableOPD delays the transition into erroneous states and maintains higher viability throughout generation.

5.4 Effectiveness of Viability Estimation

We evaluate the effectiveness of the mathematical reasoning viability judge in Figure 4. Specifically, in Figure 4(a), we compare the viability judge with existing local signals for identifying viable prefixes. The viability judge achieves an AUROC of 0.943, outperforming existing local signals and showing that viability estimation captures task-level properties beyond local prediction statistics. We further evaluate the reliability of the predicted viability scores. Figure 4(b) shows that the predicted probabilities closely match empirical viability rates, demonstrating that the judge provides well-calibrated estimates for threshold-based supervision selection. Figure 4(c) analyzes supervision decisions under the deployment threshold. The judge correctly retains or filters most prefixes, with only a small fraction of incorrect selections, validating its effectiveness in guiding teacher supervision.

5.5 Analysis of Key Factors in ViableOPD

We conduct ablation studies on the DeepSeek/JustRL mathematical reasoning setting to analyze two key components: viability-based supervision selection and token-level supervision weighting.

Effect of Supervision Selection. To examine whether ViableOPD improves by identifying viable rollouts rather than simply reducing supervision, we compare different supervision selection strategies under a matched supervision budget. Random Selection randomly selects the rollouts for supervision, while Overlap-based Selection uses teacher–student prediction overlap as a local signal for identifying suitable prefixes. As shown in Table 2, ViableOPD achieves the best performance, showing that its gain comes from restricting teacher supervision to viable prefixes, not merely reducing the number of supervised prefixes. Moreover, its advantage over Overlap-based Selection indicates that local prediction agreement is insufficient to determine whether a state remains suitable for supervision.

Effect of Token-Level Supervision Weighting. After restricting supervision to viable rollouts, we further investigate whether token-level supervision weighting improves distillation effectiveness. We compare ViableOPD with two variants: Uniform Weighting applies identical supervision strength to all tokens within viable regions, while Shuffled Weighting randomly permutes the token weights within each rollout, preserving the weight distribution but removing the correspondence between weights and token positions. As shown in Table 2, ViableOPD consistently outperforms Uniform Weighting and Shuffled Weighting, demonstrating that, beyond identifying viable regions, effective supervision requires assigning different strengths according to token-level distillation utility.

6 Related Work

Existing OPD methods estimate supervision utility using sequence position, teacher uncertainty, token teachability, or short-range future behavior (Xie et al., 2026; Liu et al., 2026b; Jiang & Ferraro, 2026). Related approaches reduce computation by stopping, pruning, or selectively extending student rollouts (Zhou et al., 2026; Yang et al., 2026; Zhao et al., 2026a; Xin et al., 2026). However, their criteria primarily rely on position or local teacher–student behavior.

Concurrent and independently developed work studies trajectory-level interventions in OPD. TRD refines problematic trajectories (Jiang et al., 2026b), while PW-OPSD weights supervision according to teacher-token reliability (Liu et al., 2026a). Counterfactual recoverability examines whether erroneous states admit recovery through alternative trajectory analysis (Jiang et al., 2026a). ViableOPD systematically characterizes the mismatch between local token-level signals and task-level prefix viability. By evaluating prefix viability over student-visited states, ViableOPD identifies both viable states overlooked by local criteria and non-viable states mistakenly retained by them, and uses this criterion to guide supervision allocation and adaptively adjust rollout horizon.

7 Conclusion

In this work, we identify a fundamental limitation of OPD: not all student-generated states can benefit from teacher supervision. When a state has deviated from the original task objective, teacher guidance conditioned on it may become ineffective or misleading. To characterize whether a student-generated state remains suitable for teacher supervision, we introduce prefix viability, a task-level criterion that measures whether the original objective remains achievable from a student-generated prefix. Based on this criterion, we propose ViableOPD, which restricts supervision to viable regions, adjusts token-level supervision within these regions, and adaptively controls the rollout horizon according to the evolving viability horizon. Experiments across multiple teacher–student pairs and benchmarks demonstrate the effectiveness of ViableOPD, showing that effective supervision for model-generated trajectories depends not only on the states a model visits, but also on whether those states remain viable for achieving the original objective.

8 Limitations and Future Work

Our evaluation is limited to verifiable reasoning tasks; extending ViableOPD to open-ended tasks requires suitable verifiers or semantic evaluation criteria. The viability horizon assumes an approximately monotonic transition from viable to non-viable regions, whereas actual reasoning trajectories may exhibit non-monotonic viability dynamics, where self-correction can occasionally recover a viable path. Finally, ViableOPD requires a substantial one-time investment to construct the domain-level judge. Future work will investigate viability-guided supervision across broader task settings, develop more flexible models of viability dynamics, and improve the efficiency of viability estimation.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing, grammar correction, and improving the clarity and readability of the manuscript, as well as drafting and refining code snippets and command-line instructions. These tools were not used to formulate the core research ideas, design the methodology, interpret the experimental results, or determine the scientific conclusions. All AI-assisted text, code, and commands were manually reviewed by the authors, and the code and commands were verified and tested before use. The authors take full responsibility for the accuracy, originality, and final content of this work.

REPRODUCIBILITY STATEMENT

To facilitate the reproduction of our work, we provide detailed descriptions of the proposed algorithm, experimental settings, training procedures, and hyperparameter configurations in the main paper and Appendix C. The source code, implementation details, and evaluation scripts are publicly available at `https://anonymous.4open.science/r/ViableOPD`.

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. URL `https://openreview.net/forum?id=3nN6nQ0x6q`.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Granite Team. Granite 3.0 language models, 2024.

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, Keming Lu, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

De Jiang, Zhengyang Zhang, Kehong Yuan, and Shaohua Ma. Not every divergence should be suppressed: Counterfactual recoverability in on-policy distillation, 2026a. URL `https://arxiv.org/abs/2608.04408`.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation, 2026b. URL `https://arxiv.org/abs/2606.08432`.

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

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, 2026. URL `https://arxiv.org/abs/2603.07079`.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327. Association for Computational Linguistics, 2016. URL `https://aclanthology.org/D16-1139`.

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, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe, 2026. URL https://arxiv.org/abs/2604.13016.

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

Xiaogeng Liu, Xinyan Wang, Yingzi Ma, Yechao Zhang, and Chaowei Xiao. When are teacher tokens reliable? position-weighted on-policy self-distillation for reasoning, 2026a. URL https://arxiv.org/abs/2605.21606.

Yanjiang Liu, Jie Lou, Xinyan Guan, Yuqiu Ji, Hongyu Lin, Ben He, Xianpei Han, Le Sun, Xing Yu, and Yaojie Lu. Your teacher can't help you here: Combating supervision fidelity decay in on-policy distillation, 2026b. URL https://arxiv.org/abs/2605.30833.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yuanyi Wang, Su Lu, Yanggan Gu, Pengkai Wang, Yifan Yang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, and Hongxia Yang. Not all disagreement is learnable: Token teachability in on-policy distillation, 2026. URL https://arxiv.org/abs/2605.26844.

Yan Xie, Sijie Zhu, Tiansheng Wen, Bo Chen, and Yifei Wang. On the position bias of on-policy distillation, 2026. URL https://arxiv.org/abs/2606.22600.

Haoran Xin, Anhao Zhao, Ying Sun, Jin Li, Xiaoyu Shen, and Hui Xiong. Escaping the KL agreement trap in on-policy distillation, 2026. URL https://arxiv.org/abs/2606.09471.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, et al. Qwen3 technical report, 2025.

Zhicheng Yang, Zhijiang Guo, Yifan Song, Minrui Xu, Yongxin Wang, Yiwei Wang, Xiaodan Liang, and Jing Tang. Prune-OPD: Efficient and reliable on-policy distillation for long-horizon reasoning, 2026. URL https://arxiv.org/abs/2605.07804.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yue Yu, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Qingfei Zhao, Huan Song, Shuyu Tian, Jiawei Shao, and Xuelong Li. Prefix-guided on-policy distillation: Mining golden trajectories from rollouts, 2026a. URL https://arxiv.org/abs/2606.21994.

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, 2026b. URL https://arxiv.org/abs/2601.18734.

Ziheng Zhou, Jiaqi Li, Huacong Tang, Ying Nian Wu, and Demetri Terzopoulos. Less is more: Early stopping rollout for on-policy distillation, 2026. URL https://arxiv.org/abs/2605.27028.

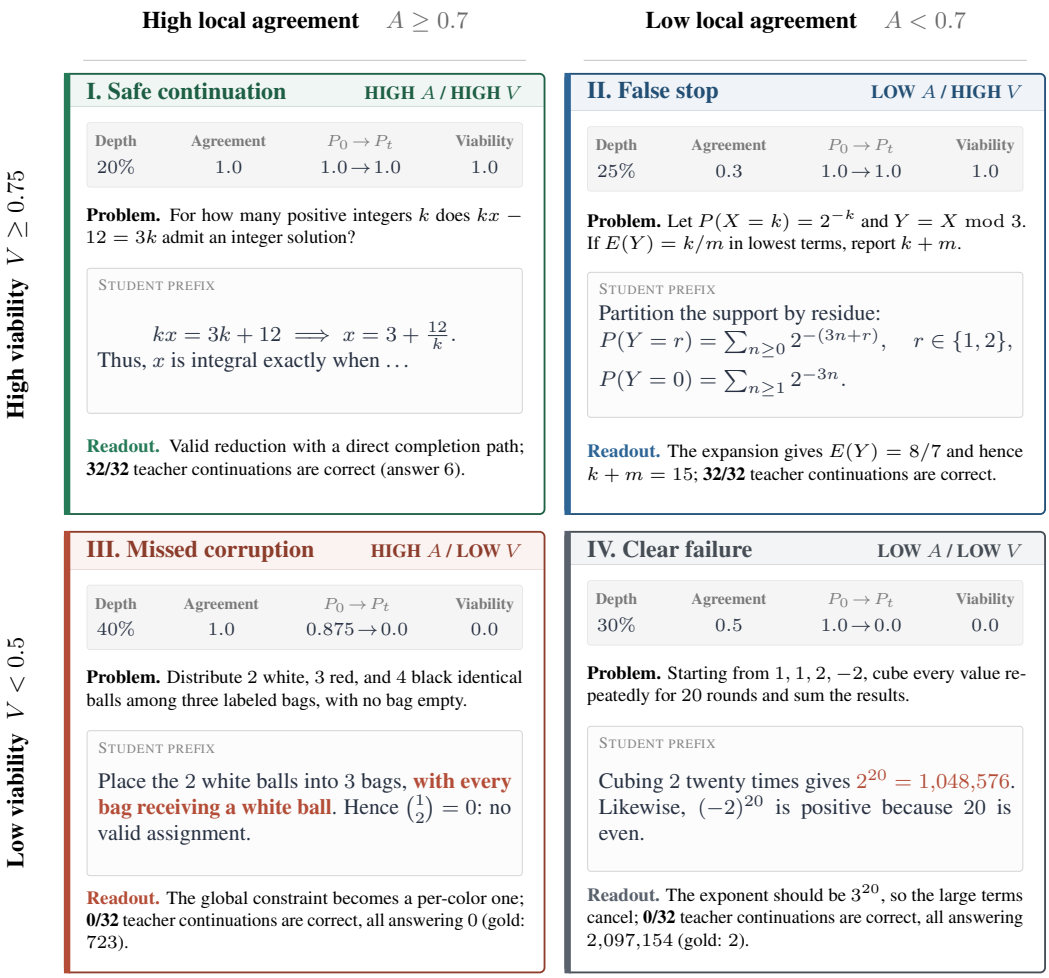

Figure 5: Four qualitative regimes of prefix viability. Prefixes are shortened only for presentation. The off-diagonal cases carry the main distinction: local disagreement can trigger a false stop on a recoverable path (II), while local agreement can miss a corrupted task state (III).

A Further Analysis of Prefix Viability

This section further analyzes the disagreement between local teacher–student agreement and prefix viability observed in Section 3. While the aggregate results demonstrate that local compatibility cannot fully characterize whether teacher supervision remains effective, we investigate where such mismatches occur during generation and what semantic patterns underlie them. We use the same thresholds as in the main text: local agreement is considered high when the teacher–student top-10 support overlap satisfies $A \geq 0.7$, and a prefix is considered high-viability when its normalized viability score satisfies $V \geq 0.75$ and low-viability when $V < 0.5$.

The two disagreement regimes exhibit distinct temporal patterns during rollout. High-agreement, low-viability prefixes appear later in the trajectory, with an average relative depth of 57.3%, whereas low-agreement, high-viability prefixes occur earlier, at 45.4%. This difference reflects two fundamentally different failure modes. Later high-agreement, low-viability prefixes typically emerge after the student has introduced an incorrect equation, invalid branch, or erroneous intermediate conclusion. Although the teacher continues generating coherently from this corrupted state, the original task objective has already become unrecoverable. Conversely, earlier low-agreement, high-viability prefixes usually correspond to alternative but valid reasoning processes, such as different notation, decomposition, or solution strategies. The teacher and student may disagree locally, while the student trajectory still preserves a valid path to the final answer.

Figure 5 illustrates all four agreement–viability regimes with representative trajectories. The off-diagonal cases provide direct evidence that local compatibility alone cannot determine whether teacher supervision remains effective. In Case II, the student exhibits low token-level agreement because it expands the expectation through residue classes in a different local formulation. However, this alternative derivation preserves the underlying reasoning structure and remains fully recoverable. In contrast, Case III shows high token-level agreement despite low viability: after the student incorrectly changes a global non-emptiness constraint into a per-color constraint, the teacher continues reasoning consistently from the corrupted premise. Although the local predictions remain aligned, the original task has already been compromised.

Table 3: Main training and evaluation configurations.

Setting	**Mathematical Reasoning**	**Code Generation**
Training dataset	DAPO-Math-17K	TACO (10K sampled)
Optimizer	AdamW	AdamW
Learning rate	10^{-6}	10^{-6}
Weight decay	0.01	0.01
Gradient clipping (L2 norm)	1.0	1.0
Training epochs	1	1
Batch size	64	64
Rollouts per prompt	4	4
Maximum prompt length	1,024 tokens	1,024 tokens
Maximum response length	12,288 tokens	4,096 tokens
Temperature / Top-p	1.0 / 1.0	1.0 / 1.0
Top-K strategy	Student Top-K	Student Top-K
LogProb top-K	16	16
Seed	42	42
Signal thresholds τ_O / τ_E / τ_D	0.7 / 0.75 / 0.75	0.7 / 0.75 / 0.75
Informative token weight λ_{info}	0.8	0.8
Low-value token weight λ_{low}	0.5	0.5
Initial horizon H_0	2,048 tokens	512 tokens
Minimum horizon $H_{\min}$	1,024 tokens	128 tokens
Maximum horizon $H_{\max}$	9,216 tokens	4,096 tokens
Horizon increment Δ	256 tokens	32 tokens
Viability ratio range α_{low} / α_{high}	0.4 / 0.6	0.4 / 0.6

Table 4: Default configurations for viability supervision data collection.

Setting	**Mathematical Reasoning**	**Code Generation**
Source dataset	DAPO-Math-17K	TACO
Sampled problems	3,000	3,000
Student rollouts per problem	3	3
Relative prefix positions	5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 60%, 70%, 80%, 90%	5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 60%, 70%, 80%, 90%
Minimum prefix length	32 tokens	32 tokens
Scratch completions	32	32
Prefix continuations	32	32
Temperature	0.7	0.7
Top-p	0.95	0.95
Maximum new tokens	12,288	2,048
Seed	42	42
Train / validation / test split	85% / 7.5% / 7.5%, by problem ID	85% / 7.5% / 7.5%, by problem ID

Table 5: Training settings of the viability judge.

Setting	Mathematical Reasoning	Code Generation
Judge backbone	Qwen3-1.7B	Granite-3.3-2B-Instruct
Offline viability threshold τ_v	0.75	0.75
Judge fine-tuning	2 epochs; AdamW; lr 2×10^{-4}	2 epochs; AdamW; lr 2×10^{-4}
LoRA rank / alpha / dropout	16 / 32 / 0.05	16 / 32 / 0.05
Online judge threshold τ_J	0.5	0.5

B Prompt Specifications

This section provides the prompt templates used for task generation and viability estimation.

B.1 Mathematical Reasoning

For mathematical reasoning, we use the prompt template distributed with DAPO (Yu et al., 2025) for training, rollout collection, and evaluation. The template encourages step-by-step reasoning and requires the final answer to be provided in a standardized format:

```
Solve the following math problem step by step. The last line of
your response should be of the form Answer: $Answer (without quotes)
where $Answer is the answer to the problem.

{problem}

Remember to put your answer on its own line after "Answer:".
```

B.2 Code Generation

For code generation, we adopt a completion-style Python prompt for TACO training and rollout collection. During evaluation, the official function signatures are preserved for HumanEval+ and MBPP+, while LiveCodeBench follows its full-program evaluation protocol.

````
You are an expert Python programmer. Produce a correct program that
follows the specification and passes all tests.

### Problem
{problem}

Read input from stdin and write the answer to stdout.
Return only the solution inside one ```python code block.

### Answer
```python
````

B.3 Viability Judges

The viability judge evaluates whether the original task objective can be achieved from a partial student trajectory. Therefore, the input contains both the original problem and the student-generated prefix.

For mathematical reasoning, the judge receives the following prompt:

```
Problem:
{problem}

Student partial solution:
{student_prefix}
```
```

Table 6: Student and teacher performance on mathematical reasoning benchmarks (pass@1, %).

Model	AMC23	AIME25	HMMT25
DeepSeek-R1-Distill-Qwen-1.5B	72.81	22.71	13.96
JustRL-DeepSeek-1.5B	92.34	53.33	24.79
OpenMath-Nemotron-1.5B	87.81	45.42	23.96
JustRL-Nemotron-1.5B	95.62	58.96	25.00

Table 7: Student and teacher performance on code generation benchmarks (pass@1, %).

Model	HumanEval+	MBPP+	LiveCodeBench
Qwen2.5-Coder-1.5B-Base	13.02	57.47	15.55
Qwen2.5-Coder-7B-Base	21.95	66.14	26.94

```
Judge whether the original task objective can still be achieved
from this prefix.
```

For code generation, the judge receives the following prompt:

````
Programming problem:
{problem}

Immutable student code prefix:
```
{student_prefix}
```

Judge whether the original task objective can still be achieved
from this prefix.
````

C Experimental Details

We provide additional implementation details for reproducing ViableOPD. Unless otherwise specified, all experiments follow the protocols described in Section 5. Table 3 summarizes the main training and evaluation configurations; Table 4 lists the configurations for collecting judge training data; and Table 5 presents the viability judge training settings.

D Student and Teacher Benchmark Performance

Tables 6 and 7 report the performance of student and teacher models, providing reference results for evaluating the capability gap between the student and teacher models used in our experiments.