FLASH-OPD: FAST ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) provides dense teacher supervision on student-generated trajectories, but generating and evaluating long rollouts incurs substantial training cost. Existing acceleration methods reduce this cost through open-loop rollout schedules or closed-loop horizon adaptation. However, supervision compatibility can vary substantially across trajectories, making a single rollout horizon difficult to match their heterogeneous reliable lengths: an overly short horizon may truncate useful supervision, while an overly long one wastes computation beyond reliable regions. Our key insight is that the trajectory-specific reliability boundary need not be predicted before generation. By viewing reliability as the first-passage of accumulated low teacher–student compatibility events, the boundary is inherently unknown before sampling, yet whether it has been reached can be determined exactly from the observed prefix. Building on this insight, we propose `Flash-OPD`, which shifts from rollout-horizon control to adaptive trajectory-level boundary verification. `Flash-OPD` interleaves cached generation with teacher verification and independently stops each trajectory according to its observed compatibility events. To reduce verification overhead, the recent event rate is used only to schedule the next verification point, while the actual stopping decision always relies on the exact cumulative count. This separation prevents estimation errors from causing premature termination while enabling efficient verification during generation. Extensive experiments across diverse datasets and teacher–student settings show that `Flash-OPD` achieves $2.2\times$–$7.5\times$ speedups over full OPD while maintaining or improving accuracy.

1 INTRODUCTION

On-policy distillation (OPD) has become an important component of frontier language model post-training, enabling both capability transfer and the consolidation of specialized teachers (Yang et al., 2025; Xu et al., 2026; Zeng et al., 2026; Team et al., 2026). By providing dense teacher feedback on student-generated prefixes, OPD mitigates the training–inference mismatch of learning from fixed teacher-generated responses (Ross et al., 2011; Kim & Rush, 2016; Gu et al., 2024). Practical token-level formulations make this feedback locally available without accumulating future distillation signals into each update (Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). However, local supervision does not automatically yield efficient long-horizon training: generating and evaluating long trajectories remains costly, while teacher guidance may become less useful as student prefixes drift into regions of low teacher–student compatibility (Li et al., 2026; Wang et al., 2026).

Recent work addresses this bottleneck by controlling rollout length through predefined schedules (Zhang et al., 2026b;a) or feedback-driven adaptation (Yang et al., 2026c; Liang et al., 2026). Yet these methods select a shared horizon for each batch, despite substantial variation in supervision compatibility across trajectories. Our analysis in Figure 1 illustrates this granularity mismatch: a shared horizon can extend well beyond the compatibility-based boundaries of some samples, even as it adapts over training. Shortening that horizon uniformly may instead curtail useful supervision for other samples. Sample-level prefix filtering offers finer selection, but produces a binary keep-or-drop decision rather than adapting the continuation length of each retained trajectory (Zhao et al., 2026a). These limitations motivate allocating rollout computation at the trajectory level.

Our key insight is that a trajectory's stopping boundary cannot be known exactly before sampling, nor does it need to be: it can instead be self-verified during generation. We use teacher–student top-k overlap as a proxy for local compatibility and define the boundary as the first position where the

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

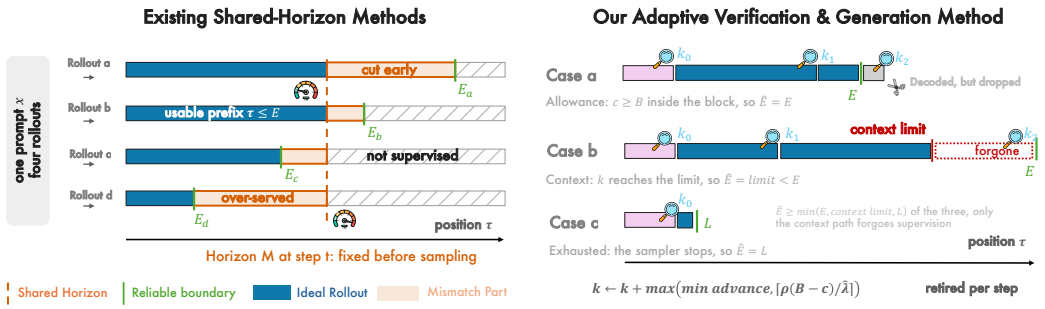

Figure 1: From a shared rollout horizon to adaptive verification. Existing OPD acceleration methods commit to a shared horizon before trajectory sampling, causing premature truncation or redundant generation when usable lengths vary across trajectories; `Flash-OPD` instead verifies reliability during generation and retires each trajectory at its own boundary, ensuring reliability and efficiency.

cumulative number of low-compatibility events reaches a prescribed budget. This accumulation tolerates isolated disagreements while adapting the boundary to each realized trajectory. Although its location is unknown in advance, whether the boundary has been crossed can be determined exactly from the observed prefix. Moreover, with model parameters fixed during rollout collection, extending a prefix leaves its existing token-level distillation signals unchanged. This enables interleaved generation and teacher evaluation, turning shared horizon into per-trajectory boundary verification.

Building on this insight, we propose `Flash-OPD`, which adaptively schedules verification and independently stops each trajectory within a reliable rollout horizon. At each verification point, `Flash-OPD` evaluates newly generated positions and checks the cumulative event count. If the budget remains unexhausted, it uses the recent event rate to schedule the next check; otherwise, it stops. Crucially, the estimated event rate determines *when to verify*, whereas the observed count determines *whether to stop*. For boundaries reached before EOS or the length cap, adaptive verification cannot trigger termination before the defined boundary. Resuming from cached prefixes ensures that each token is decoded only once, avoiding redundant generation. Through verified stopping at the trajectory level, `Flash-OPD` reduces rollout computation and substantially improves training efficiency over vanilla OPD while maintaining strong performance. Our contributions are threefold:

- We identify a granularity mismatch between batch-level rollout horizons and trajectory-specific supervision compatibility, motivating per-sample stopping for long-horizon OPD acceleration.
- We propose `Flash-OPD`, which interleaves cached generation with adaptive verification to determine each trajectory's rollout length from its observed compatibility events.
- Extensive experimental results across diverse datasets and teacher–student settings demonstrate that `Flash-OPD` achieves strong effectiveness, efficiency, robustness, and scalability.

2 Related Work

On-Policy Distillation. OPD alleviates distribution mismatch by evaluating a frozen teacher on trajectories sampled from the current student. MiniLLM (Gu et al., 2024) formalized this paradigm under reverse KL, while GKD (Agarwal et al., 2024) unified the on-policy / off-policy spectrum across divergences and demonstrated the benefits of on-policy supervision. Subsequent token-level formulations (Lu & Thinking Machines Lab, 2025) showed that OPD can achieve strong reasoning gains at substantially lower cost than outcome-reward RL, leading to its adoption in production post-training pipelines (Yang et al., 2025; Zeng et al., 2026; Team et al., 2026) and extensions to multi-teacher and self-distillation settings (Shenfeld et al., 2026; Hübotter et al., 2026; Ye et al., 2026; Zhao et al., 2026b). *However, because supervision is obtained along the student's current trajectories, long-horizon rollout remains a major computational bottleneck, even when later tokens provide limited useful signal. This makes rollout horizon a natural target for efficiency optimization.*

Rollout-Horizon Control. A growing body of work therefore treats rollout length as an explicit design variable. Open-loop methods determine the horizon from the training schedule: POPD pro-

Method	Length decision		Allocation		
	Driven by	Granularity	Spectrum	All seqs. trained	Stacks on a batch ctrl.
POPD (Zhang et al., 2026b)	training step	batch	single	✓	✗
TOPD (Zhang et al., 2026b)	training step	batch	single	✓	✗
Prefix OPD (Zhang et al., 2026a)	training step	batch	single	✓	✗
ADWIN (Liang et al., 2026)	gradient cosine	batch	single	✓	✗
PRUNE-OPD (Yang et al., 2026c)	top-k overlap	batch	single	✓	✗
PG-OPD (Zhao et al., 2026a)	probe overlap	candidate	two-valued	✗	✓
`Flash-OPD`(ours)	usable horizon	sequence	continuous	✓	✓

Table 1: Comparison of rollout-control methods for OPD. *Driven by*: what the length decision consumes. *Granularity*: the unit the decision is made per. *Spectrum*: the set of realized lengths within one step. *All seqs. trained*: whether every sampled sequence contributes to the loss. *Stacks on a batch ctrl.*: can run under an independent batch-level window controller instead of replacing it.

gressively expands it, TOPD permanently truncates it to a fixed fraction (Zhang et al., 2026b), and prefix distillation trains on short prefixes with a linear ramp (Zhang et al., 2026a). Closed-loop methods instead use runtime signals: PRUNE-OPD derives a drift signal from token-level top-k overlap to guide dynamic truncation (Yang et al., 2026c), while ADWIN selects the shortest window whose prefix gradient remains aligned with the full-rollout update (Liang et al., 2026). Despite their different criteria, these methods make horizon decisions at the batch level, forcing all sequences to share the same rollout length. We argue that this granularity is mismatched with the inherently sample-dependent reliability of teacher supervision. PG-OPD (Zhao et al., 2026a) moves to the sample level by probing fixed prefixes and retaining only high-scoring samples, but yields a binary keep-or-drop decision. *In contrast, we infer a continuous per-sample rollout budget within the existing batch-level horizon, enabling each trajectory to terminate according to its own supervision reliability.*

Table 1 compares the structural differences between our and other rollout control acceleration methods. Furthermore, we provide a more detailed introduction to additional related work in Appendix A.

3 PRELIMINARIES

3.1 ON-POLICY DISTILLATION

Let x be an input prompt and $y = (y_1, \ldots, y_L)$ a response sampled autoregressively from the student policy π_θ. At position τ write $y_{<\tau}$ for the prefix and define

$$p_\tau(v) \triangleq \pi_\theta(v \mid x, y_{<\tau}), \qquad q_\tau(v) \triangleq \pi_T(v \mid x, y_{<\tau}), \tag{1}$$

over the vocabulary $v \in \mathcal{V}$, with π_T a frozen teacher model. OPD minimizes a token-level divergence on student-sampled trajectories,

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{x,\, y\sim\pi_\theta(\cdot|x)}\Big[\textstyle\sum_{\tau=1}^{L} D_{\mathrm{KL}}\big(p_\tau \,\|\, q_\tau\big)\Big], \tag{2}$$

approximated in practice on the student's top-k candidates to give a per-position reward vector $r_{\tau,\cdot}$.

Eq. (2) exhibits two useful properties. *Locality:* $r_{\tau,\cdot}$ depends only on $(x, y_{\le\tau})$, not on later tokens. *Permanence:* once the prefix is fixed, $r_{\tau,\cdot}$ remains unchanged as generation continues. Thus, a trajectory can be scored incrementally, and stopping at τ forfeits only subsequent positions without affecting the supervision already obtained. We relegate the proofs of these properties to Appendix B.

3.2 RELIABILITY AND USABLE LENGTH

Locality and persistence allow supervision to accumulate gradually, but this does not mean that it maintains equal utility throughout the entire unfolding process. Prior empirical analysis (Li et al., 2026; Yang et al., 2026c) links successful OPD to progressive alignment on shared high-probability

tokens, showing that optimizing only these tokens recovers nearly all the gains of standard OPD. This finding motivates us to use top-k overlap as a proxy for local teacher–student compatibility. Formally, let K_τ^S and K_τ^T denote the top-k sets, and define a low-compatibility event as follow:

$$\phi_\tau = \mathbb{1}\big[\,|K_\tau^S \cap K_\tau^T|/k < \eta\,\big], \tag{3}$$

where $\eta \in (0, 1)$ is the compatibility threshold. To tolerate isolated mismatches, we accumulate these events and define the reliability boundary as a count budget $B \in \mathbb{N}_{>0}$ for such events:

$$c_\tau = \textstyle\sum_{i=1}^{\tau} \phi_i, \qquad E = \min\{\tau \in \{1, \ldots, L\} : c_\tau \geq B\}, \tag{4}$$

where E is the *usable length* of trajectory. Notably, we can naturally observe three crucial properties of E: ❶ it is *trajectory-specific*; ❷ it is the *exact first-passage time* of an integer-valued process rather than a quantity requiring estimation; ❸ it is *posterior*, hence unknowable prior to sampling y.

4 Methodology

4.1 From Horizon Control to Boundary Verification

Motivation. Existing OPD acceleration methods reduce training cost by shortening rollouts. However, as discussed above, Property ❶ of reliability length, *i.e.*, *trajectory specificity*, implies that different trajectories have different reliable lengths. For methods that share a rollout horizon across a batch, whether the horizon is scheduled by training progress or adapted from historical feedback, a single scalar is unlikely to fit these heterogeneous boundaries. We therefore argue that the key lies not in adapting a shared horizon, but in providing each trajectory with an appropriate rollout length.

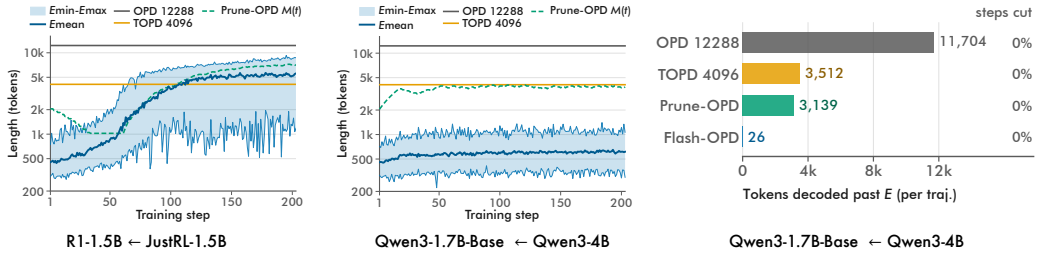

Figure 2: Heterogeneous and evolving reliability lengths expose the mismatch of fixed and adaptive batch-shared horizons (left and middle), while our method `Flash-OPD` substantially reduces excess rollout budget without prematurely truncating reliable prefixes in the illustrated setting (right).

Empirical Study. To quantify this phenomenon, we analyze training dynamics on two representative student–teacher pairs, comparing the rollout horizons of existing acceleration methods against the empirical reliability lengths of individual trajectories. The left and middle panels of Figure 2 reveal heterogeneous and dynamically evolving reliability boundaries that neither open-loop nor closed-loop shared-horizon methods can consistently track. In the left panel, TOPD's fixed horizon initially exceeds the entire reliability range but later falls below its mean, while Prune-OPD over-allocates rollout budget early in training. In the middle panel, both methods remain above $E_{\max}$, avoiding premature truncation at the cost of substantial computation beyond the reliable length. The right panel further confirms significant over-decoding beyond reliable lengths across existing horizon-control methods. These findings collectively reveal the inefficiency of existing methods.

Discussion. The above observations suggest that the granularity mismatch between shared horizon and per-trajectory boundaries cannot be resolved solely by a more sensitive horizon controller. A natural solution is to transform budget allocation into per-trajectory stopping decisions, allowing each trajectory to terminate independently based on its own compatibility events. However, the difficulty lies in the property ❸ of E: it can only be known after the trajectory is sampled, yet the budget must be set before sampling. Nevertheless, property ❷ offers a promising perspective: E is the first-passage time that depends on prefix counting. Therefore, instead of predicting the E of different trajectories in advance before generation, we can verify whether c_τ has reached B during trajectory unfolding, thereby naturally deriving a credible E. *Locality* and *Permanence* ensure that this does not alter the supervision already computed on the prefix. This motivates us to move the stopping decision inside the generation process, shifting from horizon control to boundary verification.

4.2 FLASH-OPD: ADAPTIVE BOUNDARY VERIFICATION

Building on theses insights, we propose `Flash-OPD` to robustly accelerate OPD via adaptive boundary verification. Rather than fixing unrolling steps pre-sampling, `Flash-OPD` dynamically stops each trajectory as it unrolls. Alternating between *generation* and *verification*, rollouts extend to verification points where cumulative counts are evaluated to either halt or continue. The exact usable length determines termination, while the estimated event rate sets the next verification point.

Algorithm 1 `Flash-OPD`: Fast OPD via Adaptive Verification

Require: prompts $\mathcal{D}$, student π_θ, teacher π_T, allowance B, initial verification position k_0, discount ρ, window W, minimum advance $k_{\min}$

for training step $t = 1, 2, \ldots$ **do**
  sample prompts $\{x^{(b)}\}_{b=1}^{B_{\text{sz}}} \sim \mathcal{D}$; set $k^{(b)} \leftarrow k_0$, $c^{(b)} \leftarrow 0$, mark all b active
  while any b is active **do**
    decode every active b from its current position to $k^{(b)}$, resuming from cached prefixes
    evaluate π_T on the newly produced positions; update $c^{(b)} \leftarrow c^{(b)}_{k^{(b)}}$ by Eqs. (3)–(4)
    for each active b **do**
      if $c^{(b)} \geq B$ **or** $k^{(b)} \geq L_{\max}$ **then**
        $\hat{E}^{(b)} \leftarrow k^{(b)}$; mark b inactive ▷ stops at its own boundary
      else
        $k^{(b)} \leftarrow k^{(b)} + \max\left(k_{\min}, \lceil \rho(B - c^{(b)})/\hat{\lambda}^{(b)} \rceil\right)$ ▷ Eq. (5)
      end if
    end for
  end while
  update θ on $\{y^{(b)}_{\leq \hat{E}^{(b)}}\}$ with rewards $r_{\tau,\cdot}$
end for

Implementation Details. We outline the overall procedure in Algorithm 1. Specifically, given an allowance B, an initial verification position k_0, a discount factor $\rho \in (0, 1]$, a trailing window W, and a minimum advancement step $k_{\min}$. Each trajectory maintains a verification position $k \leftarrow k_0$ and an accumulated count $c \leftarrow 0$, and repeats the following three steps:

(i) Generate to the verification position. Decode from the current position to k, resuming from the cached prefix rather than regenerating existing tokens.

(ii) Verify the accumulated count. Evaluate the teacher on the newly produced positions and update $c \leftarrow c_k$ using Eqs. (3)–(4). The resulting c_k is the exact count of observed low-compatibility events.

(iii) Stop or advance. If $c \geq B$ or $k \geq L_{\max}$, set $\hat{E} = k$ and stop the trajectory. Otherwise, advance the verification position according to

$$k \;\leftarrow\; k + \max\Big(k_{\min},\; \big\lceil \rho\,(B - c)/\hat{\lambda}_k \big\rceil\Big), \qquad \hat{\lambda}_k = \big(c_k - c_{k-W}\big)/W. \tag{5}$$

Here, $\hat{\lambda}_k$ estimates the local rate of low-compatibility events over the latest W positions. With $B - c$ events remaining before the boundary, $(B - c)/\hat{\lambda}_k$ estimates how many additional tokens would be needed if the current rate persisted. We advances only a fraction ρ of this estimated distance before measuring the rate again, yielding large steps far from the boundary and progressively smaller steps as the boundary approaches. The minimum step $k_{\min}$ ensures continued progress, while the actual stopping decision always relies on the exact count $c_k \geq B$. Figure 3 provides an intuitive example.

Theoretical Analysis. A potential concern is that an improper final rate estimate may compromise supervision. We show that extrapolation error only cause a slight decoding cost, not supervision:

Proposition 1. *$\hat{E}^{(b)} \geq E^{(b)}$ for every trajectory, for any $\rho \in (0, 1]$, any W, any realization of $\hat{\lambda}$.*

Proof. Termination occurs only under $c_k \geq B$, evaluated on the exact count. Since c_τ is non-decreasing and $E^{(b)} = \min\{\tau : c_\tau \geq B\}$, this cannot hold at any $k < E^{(b)}$. The estimate $\hat{\lambda}$ enters only the placement of subsequent verification boundaries, never the termination test. □

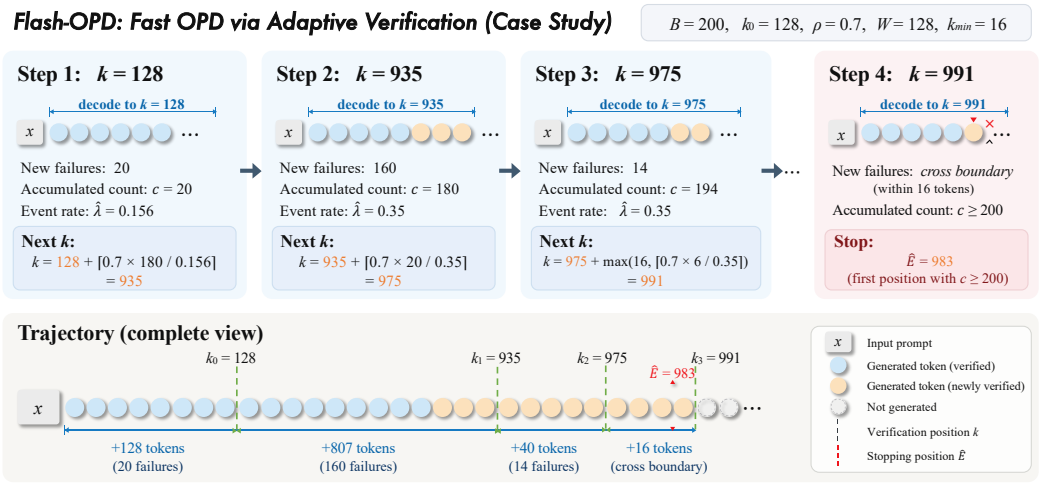

Figure 3: A schematic diagram of our rollout case, which adaptively adjusts verification positions according to the estimated local event rate and stops once the exact reliability boundary is reached.

Complexity Analysis. `Flash-OPD` replaces a long generation pass with R shorter ones, so the key questions are how large R can be and how much each round costs. Let R denote the number of verification rounds. Since each advance in Eq. (5) is at least $k_{\min}$ and $k \leq L_{\max}$, we have $R \leq 1 + \lceil (L_{\max} - k_0)/k_{\min} \rceil$. When the local event rate is approximately stable, each round consumes a fraction ρ of the remaining allowance, yielding $R = O(\log B)$. The per-step cost consists mainly of decoding and teacher evaluation, and our method do not incur additional overhead beyond these. Importantly, with prefix caching, each token is decoded only once, so the generation cost remains proportional to the actual rollout length. Compared with the full rollout length L_{max}, `Flash-OPD` substantially reduces this cost by terminating generation around the usable length E.

5 Experiments

In this section, we conduct extensive experiments to investigate the following research questions:

- **RQ1:** Does `Flash-OPD` consistently improve the accuracy–cost trade-off? ***(Effectiveness)***
- **RQ2:** Where exactly do the wall-clock savings of `Flash-OPD` come from? ***(Efficiency)***
- **RQ3:** How does `Flash-OPD` improve training dynamics through adaptive rollout? ***(Mechanism)***
- **RQ4:** Do the gains of `Flash-OPD` transfer reliably out of distribution? ***(Transferability)***
- **RQ5:** How do the efficiency gains of `Flash-OPD` scale with larger teacher size? ***(Scalability)***
- **RQ6:** How does `Flash-OPD` work, and does hyperparameter tuning affect it? ***(Robustness)***

5.1 Experimental Setup

Datasets and Baselines. We compare vanilla OPD (Lu & Thinking Machines Lab, 2025), and representative methods for open-loop and closed-loop control: TOPD (Zhang et al., 2026b), and Prune-OPD (Yang et al., 2026c), all trained on DAPO-Math-17K (Yu et al., 2026). Evaluation covers distribution-aligned math benchmarks (AMC23, AIME24/25, HMMT24/25) and transfer benchmarks (MATH-500, Minerva Math, Olympiad-Bench, GPQA), with GPQA testing beyond mathematics. Details are in Appx. D.1.

Models and Protocol. To assess generality across representative OPD settings, we pair DeepSeek-R1-Distill-Qwen-1.5B with JustRL-DeepSeek-1.5B (same-scale, post-training-enhanced teacher) or DeepSeek-R1-Distill-Qwen-7B (larger teacher, similar lineage), and Qwen3-1.7B/4B-Base with Qwen3-4B (Non-thinking), varying student scale and Base-to-post-trained distribution gaps. All methods train for 203 steps under identical optimization settings. We report mean@16 averaged over the final three steps (31,744-token decoding budget) and efficiency metrics, including training time. Baselines use recommended hyperparameters, while we use a fixed configuration unless otherwise specified: $\eta = 0.7$, $B = 200$, $\rho = 1.0$, and $k_0 = W = 128$. Full details in Appx. D.4.

Table 2: Overall performance-efficiency comparison under different benchmarks and settings.

Method	AMC23	AIME24	AIME25	HMMT24	HMMT25	Average	Time	Speed Up
Student / Teacher: DeepSeek-R1-Distill-Qwen-1.5B / JustRL-DeepSeek-1.5B								
OPD	78.16	46.46	32.92	24.79	18.12	40.09	7.46h	-
TOPD	79.67	45.83	33.33	24.79	19.38	**40.60**	3.51h	2.12×
Prune-OPD	78.92	44.79	31.25	22.92	20.62	39.70	3.36h	2.22×
Flash-OPD	79.37	46.46	32.92	22.29	20.83	40.37	**3.35h**	2.23×
Student / Teacher: DeepSeek-R1-Distill-Qwen-1.5B / DeepSeek-R1-Distill-Qwen-7B								
OPD	64.91	33.33	22.92	15.83	14.79	30.36	10.95h	-
TOPD	64.91	32.71	23.75	16.25	15.00	**30.52**	4.15h	2.64×
Prune-OPD	65.36	30.83	24.38	13.75	15.42	29.95	3.81h	2.88×
Flash-OPD	66.27	30.42	24.17	13.33	14.58	29.75	**2.06h**	5.32×
Student / Teacher: Qwen3-1.7B-Base / Qwen3-4B (Non-thinking)								
OPD	27.94	8.12	3.33	3.33	0.21	8.59	11.06h	-
TOPD	28.35	7.85	3.13	2.15	0.28	8.35	4.80h	2.30×
Prune-OPD	28.43	6.74	3.89	2.36	0.49	8.38	3.04h	3.64×
Flash-OPD	29.29	6.46	5.00	3.33	0.42	**8.90**	**1.55h**	7.14×
Student / Teacher: Qwen3-4B-Base / Qwen3-4B (Non-thinking)								
OPD	43.00	14.58	12.50	6.25	2.92	15.85	33.09h	-
TOPD	42.09	15.00	12.50	5.21	3.12	15.58	7.81h	4.24×
Prune-OPD	40.06	15.42	10.83	4.79	2.71	14.76	13.87h	2.39×
Flash-OPD	48.72	18.75	15.83	7.08	5.62	**19.20**	**4.40h**	7.52×

5.2 Effectiveness Study (RQ1)

Table 2 compares `Flash-OPD` with vanilla OPD and advanced OPD acceleration methods across four diverse student–teacher pairs and five in-domain benchmarks. We observe the following:

❶ `Flash-OPD` improves the accuracy–cost trade-off, outperforming existing acceleration baselines with 2.23×–7.52× speedups over vanilla OPD while largely preserving or improving average accuracy. ❷ Interestingly, on Qwen3-4B-B / Qwen3-4B, it also notably improves accuracy. We find that the student models in this group are prone to length runaway during distillation, thereby introducing low-quality or even harmful supervision. Our method truncates trajectories when supervision becomes unreliable, potentially providing an implicit curriculum learning and regularization effect.

5.3 Efficiency Study (RQ2)

To understand where the efficiency gains come from, Figure 4 decomposes the mean wall-clock time per training step into rollout generation, log-probability computation, teacher verification, policy updates, and residual overhead across the four model pairs. We make three observations:

❶ `Flash-OPD` substantially reduces per-step training time, primarily by shortening rollout generation. ❷ `Flash-OPD` evaluates both student and teacher during verification and reuses the results, eliminating the separate log-probability pass. ❸ By terminating rollouts early, `Flash-OPD` further reduces the costly policy-update time, extending generation savings to subsequent stages.

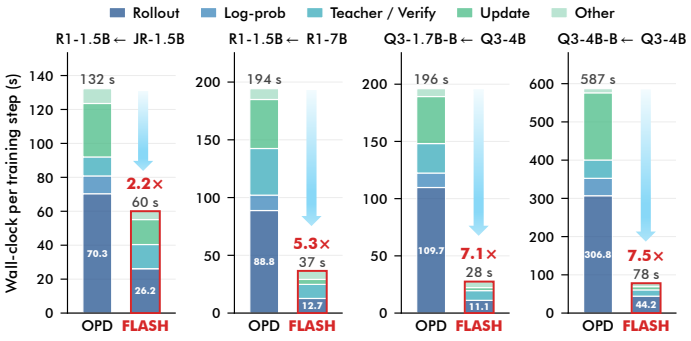

Figure 4: Mean wall-clock seconds per training step.

5.4 Mechanism Study (RQ3)

We further investigate the mechanisms of our method compared to vanilla OPD through training dynamics. Figure 5 compares the relationship between accuracy and wall-clock training time for `Flash-OPD` relative to OPD across four model pairs. We observe the following phenomena:

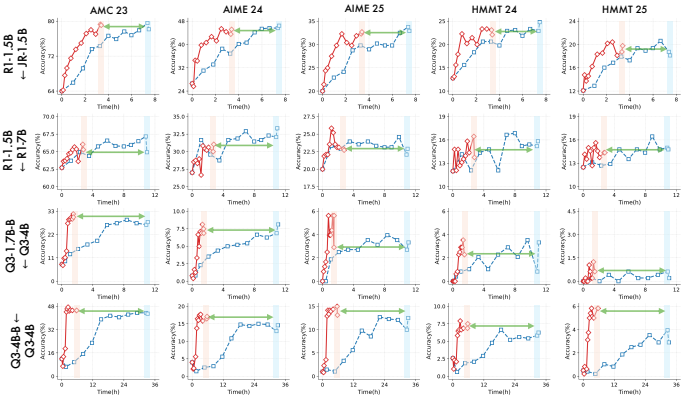

Figure 5: Wall-clock training dynamics of vanilla OPD and our `Flash-OPD` across four teacher-student model pairs and five benchmarks. We perform a benchmark evaluation every 20 steps.

❶ `Flash-OPD` consistently shifts learning curves upward and leftward, reaching vanilla OPD accuracy substantially earlier across pairs and benchmarks. ❷ For DeepSeek pairs, `Flash-OPD` largely preserves OPD's optimization trajectory while compressing training time in first setting; both methods show noisy trajectories and limited gains in the second. This aligns with prior findings (Li et al., 2026) that teachers from the same training pipeline may offer insufficiently novel, transferable supervision on student-visited states. ❸ For Qwen pairs, `Flash-OPD` improves sharply early in training, whereas Vanilla OPD progresses gradually over a much longer horizon. Notably, on Qwen3-4B-Base ← Qwen3-4B (Non-thinking), it achieves strong performance within hours and converges to higher accuracy on all five benchmarks, suggesting that adaptive truncation may improve optimization by filtering low-value supervision late in trajectories. ❹ Vanilla OPD often spends substantial time on plateaus with marginal gains or oscillations, while our `Flash-OPD` concentrates computation in earlier, productive stages. Shorter, higher-confidence supervision may be easier for the policy to absorb; prioritizing usable tokens over full teacher trajectories may reduce gradient variance and error accumulation.

5.5 Transferability Study (RQ4)

To examine whether the benefits of `Flash-OPD` extend beyond in-domain evaluation and same-lineag teacher–student pairs, we study transferability along two dimensions: evaluation distribution and model generation. Figure 6 summarizes the results. We make three observations:

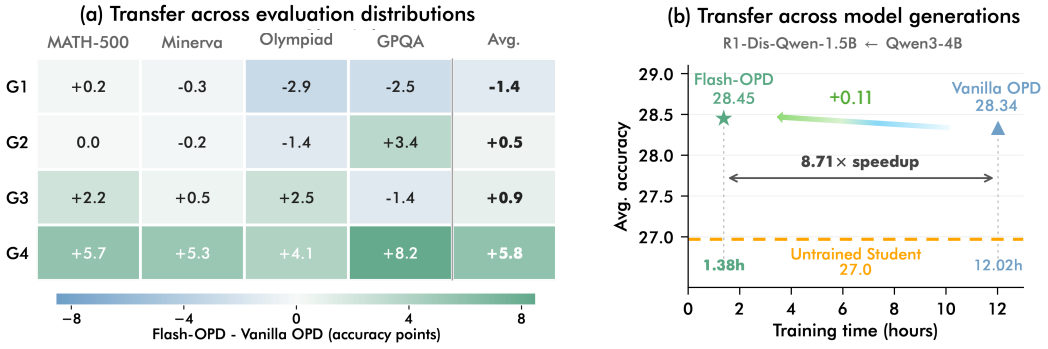

Figure 6: Transferability of `Flash-OPD` across evaluation distributions and model generations.

❶ Across unseen evaluation distributions, `Flash-OPD` largely preserves OPD performance: the macro differences on G1–G3 remain within 1.4 points, despite including GPQA, a non-mathematical benchmark. ❷ Transfer can also be favorable: on G4, `Flash-OPD` consistently improves all four held-out benchmarks, yielding a +5.84-point macro gain. ❸ The efficiency gain persists across model families without pair-specific tuning: on R1-1.5B←Qwen3-4B, `Flash-OPD` matches OPD in macro accuracy (28.45 vs. 28.34) while reducing training time from 12.02h to 1.38h (8.71×).

5.6 Scalability Study (RQ5)

To examine whether `Flash-OPD` remains effective with increasingly large teachers, Figure 7 scales the teacher from Qwen3-4B to 14B and 32B, fixing the Qwen3-1.7B-Base student. We observe that:

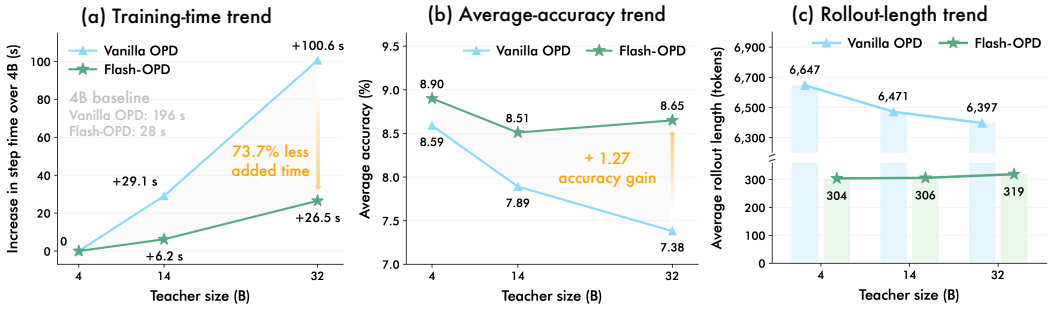

Figure 7: With a fixed student, `Flash-OPD` incurs smaller step-time increases than OPD while maintaining stable accuracy and consistently short rollouts as teacher size grows from 4B to 32B.

❶ `Flash-OPD` remains substantially faster: from 4B to 32B, per-step time rises by only 26.5 s *vs.* 100.6 s for vanilla OPD, a 73.7% reduction in added time. ❷ Accuracy remains stable at 8.5–8.9%, while OPD drops from 8.59% to 7.38%, giving `Flash-OPD` a 1.27 percentage-point advantage at 32B. ❸ Larger teachers do not consistently improve OPD, whose accuracy varies with rollout length. We maintains nearly constant rollout lengths, yielding more stable training across teacher scales.

5.7 Robustness Study (RQ6)

To understand how controller settings affect OPD behavior, Figure 8 examines the joint variation of the η and B, as well as the impact of ρ on validation. We can observe that:

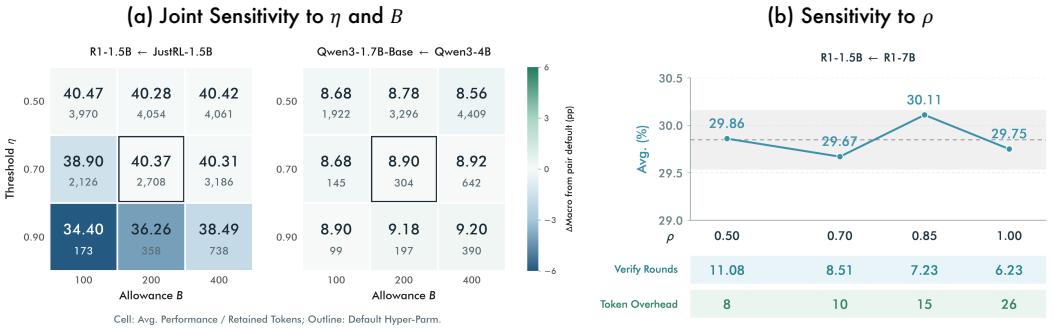

Figure 8: Sensitivity of `Flash-OPD` to the drift threshold η, allowance B, and step discount ρ.

❶ `Flash-OPD` remains robust to broad variations in controller parameters, with degradation mainly at extreme thresholds that aggressively shift stopping boundaries. ❷ The threshold and allowance interact: larger B partly mitigates degradation at high η but has little effect at low η. ❸ The step discount controls the verification trade-off; however, as proof shown earlier, the growth of redundant token generation is quite limited, and overall performance remains relatively stable.

6 Conclusion

In this paper, we investigate the mismatch between shared rollout horizons and trajectory-specific reliability boundaries in on-policy distillation. We propose `Flash-OPD`, an adaptive boundary verification framework that alternates generation and verification, using exact compatibility-event counts to guide stopping decisions and local event rates to schedule verification. Extensive experiments on multiple teacher-student model pairs demonstrate its effectiveness, efficiency, transferability, scalability, and robustness. In future work, we aim to extend this efficient acceleration method for adaptive boundary verification to multi-teacher online policy distillation and agent tasks.

AI USE STATEMENT

Following the conference guidelines regarding generative artificial intelligence models (GenAI), we declare that GenAI provided assistance only in literature search, code development, and manuscript drafting and polishing. Conceptual development, experimental methodology, and all aspects of the manuscript's core content were completed independently by the authors without GenAI assistance.

ETHICS STATEMENT

This paper presents a work aimed at promoting efficient training of post-training algorithms for large language models. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

REPRODUCIBILITY STATEMENT

We used publicly available datasets for training and evaluation, and plan to open-source the code. The numerical tables in the manuscript were manually double-checked and cross-verified against the records. The authors take responsibility for the text, claims, and research findings of this study.

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 International Conference on Learning Representations, volume 2024, pp. 21246–21263, 2024.

Yuchen Cai, Ding Cao, Liang Lin, Chunxi Luo, Xin Xu, Kai Yang, Weijie Liu, Saiyong Yang, Tianxiang Zhao, Guangzhong Sun, et al. Learning to foresee: Unveiling the unlocking efficiency of on-policy distillation. arXiv preprint arXiv:2605.11739, 2026.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo 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 International Conference on Learning Representations, volume 2024, pp. 32694–32717, 2024.

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

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.

Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggyu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation. In Findings of the Association for Computational Linguistics: ACL 2026, pp. 42217–42227, 2026.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 4163–4174, 2020.

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.

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.

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, 2016.

Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation. arXiv preprint arXiv:2603.11137, 2026.

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, 2026.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. In Findings of the Association for Computational Linguistics: ACL 2025, pp. 25366–25394, 2025.

Kun Liang, Chenming Tang, Clive Bai, Weijie Liu, Saiyong Yang, and Yunfang Wu. Adwin: Adaptive windows for horizon-aware on-policy distillation. arXiv preprint arXiv:2605.28396, 2026.

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

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. Crisp: Compressed reasoning via iterative self-policy distillation. arXiv preprint arXiv:2603.05433, 2026.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.

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

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. arXiv preprint arXiv:2607.24653, 2026.

Rui Wang, Hongru Wang, Yi Chen, Boyang Xue, Tianqing Fang, Wenhao Yu, and Kam-Fai Wong. Demystifying on-policy distillation: Roles, pathologies, and regulations. arXiv preprint arXiv:2607.13399, 2026.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Advances in neural information processing systems, 33:5776–5788, 2020.

Yecheng Wu, Song Han, and Hai Cai. Lightning opd: Efficient post-training for large reasoning models with offline on-policy distillation. arXiv preprint arXiv:2604.13010, 2026.

Anyi Xu, B Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, CC Yu, et al. Deepseek-v4. 1-flash: Pushing the limits of kv cache compression. arXiv preprint arXiv:2609.19969, 2026.

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

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, 2026a.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Yongxin Wang, Wenlei Shi, Yiwei Wang, Xiaodan Liang, and Jing Tang. Accordion-thinking: Self-regulated step summaries for efficient and readable llm reasoning. arXiv preprint arXiv:2602.03249, 2026b.

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. arXiv preprint arXiv:2605.07804, 2026c.

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online experiential learning for language models. arXiv preprint arXiv:2603.16856, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. Advances in Neural Information Processing Systems, 38:113222–113244, 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.

Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler II, Qian Qian, Gregory D Lyng, Sanjit Singh Batra, and Robert E Tillman. Fast and effective on-policy distillation from reasoning prefixes. In Findings of the Association for Computational Linguistics: ACL 2026, pp. 25553–25569, 2026a.

Yaocheng Zhang, Jiajun Chai, Yuqian Fu, Songjun Tu, Xiaohan Wang, Wei Lin, Guojun Yin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Are full rollouts necessary for on-policy distillation? arXiv preprint arXiv:2605.31490, 2026b.

Qingfei Zhao, Huan Song, Shuyu Tian, Jiawei Shao, and Xuelong Li. Prefix-guided on-policy distillation: Mining golden trajectories from rollouts. arXiv preprint arXiv:2606.21994, 2026a.

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, 2026b.

SUPPLEMENTARY MATERIAL

FLASH-OPD: FAST ON-POLICY DISTILLATION

TABLE OF CONTENTS

A More Related Work **14**

A.1 Distillation Objectives 14

A.2 Decoding and Training Efficiency 14

B More Analysis **14**

B.1 Locality and Permanence 14

B.2 Stopping Guarantee 14

B.3 Verification Complexity 15

C Method Details **15**

C.1 Batched Implementation 15

C.2 Decoding and Verification Cost 15

C.3 Rollout Engine Settings 16

D Experimental Details **17**

D.1 Datasets 17

D.2 Teacher–Student Pairs 17

D.3 Baselines 17

D.4 Protocol and Hyperparameters 19

E Additional Results **19**

E.1 Training-Time Breakdown 19

E.2 Trajectory Lengths and Stopping Statistics 20

E.3 Out-of-Domain Transfer 20

E.4 Cross-Generation Transfer 20

F Limitations and Future Work **21**

A More Related Work

A.1 Distillation Objectives

Classical knowledge distillation trains a student to match teacher predictions on a fixed corpus (Hinton et al., 2015; Sanh et al., 2019; Jiao et al., 2020; Wang et al., 2020). On-policy distillation instead queries the teacher on student-generated prefixes, reducing the mismatch between training states and those visited by the student (Ross et al., 2011; Gu et al., 2024; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). Recent work improves the supervision at each position through reward extrapolation (Yang et al., 2026a), revised divergence objectives (Jang et al., 2026; Jin et al., 2026; Ko et al., 2026; Fu et al., 2026), and analyses of teacher–student compatibility (Li et al., 2026; 2025; Kim et al., 2026). Self-distillation extends the source of supervision beyond a fixed external teacher (Shenfeld et al., 2026; Hübotter et al., 2026; Zhao et al., 2026b; Ye et al., 2026). These directions change the training signal, whereas `Flash-OPD` controls how far each trajectory is generated while retaining the token-level objective on its prefix.

A.2 Decoding and Training Efficiency

Speculative decoding reduces generation cost by verifying blocks of proposed tokens. Its verification determines which proposed tokens to accept. The verification in `Flash-OPD` instead determines whether a student trajectory should continue receiving distillation supervision; it does not replace generated tokens with teacher samples. Improvements to decoding or other training stages (Wu et al., 2026; Cai et al., 2026) address costs complementary to rollout truncation. Methods that train models to reason in fewer tokens at inference time (Yang et al., 2026b; Sang et al., 2026) target the deployed model's response length rather than the length of training rollouts.

B More Analysis

This section proves the prefix properties used in Section 3 and gives details of the stopping guarantee and verification complexity.

B.1 Locality and Permanence

Lemma 2 (Locality and Permanence)**.** *Fix the student and teacher parameters during rollout collection. At each position τ, the per-position reward $r_{\tau,\cdot}$ depends only on $(x, y_{<\tau})$. Any two trajectories with the same prefix $y_{<\tau}$ therefore receive the same reward at position τ, regardless of their later tokens.*

Proof. By causality, $p_\tau = \pi_\theta(\cdot \mid x, y_{<\tau})$ and $q_\tau = \pi_T(\cdot \mid x, y_{<\tau})$ are determined by the prefix. The student's top-k candidates and their weights are functions of p_τ. Both $D_{\mathrm{KL}}(p_\tau \| q_\tau)$ and its top-k approximation are therefore determined by the same prefix. Extending the trajectory changes neither distribution at position τ nor the resulting reward. □

Corollary 3 (Prefix supervision)**.** *For any retained length $S \leq L$, the truncated sum $\sum_{\tau=1}^{S} D_{\mathrm{KL}}(p_\tau \| q_\tau)$ equals the first S terms of the full-trajectory sum in Eq. (2).*

Proof. Truncation leaves every prefix $y_{<\tau}$ with $\tau \leq S$ unchanged, so Lemma 2 applies to each retained term. □

The implementation averages the loss over retained tokens. Truncation changes this normalization and removes the tail terms, but does not change the per-position rewards on the retained prefix.

B.2 Stopping Guarantee

For termination triggered by the allowance, Proposition 1 follows from the monotonicity of c_τ: the test $c_k \geq B$ cannot succeed before E. The rate estimate affects the next verification position, not this test. The implementation also handles EOS and the length limit as follows.

Proposition 4 (Stopping with EOS and a length limit)**.** *Let L be the natural termination length and set $E = \infty$ if the event count never reaches B. Algorithm 2, which locates the first crossing inside each verified block, retains*

$$\hat{E} = \min(E, L_{\max}, L).$$

Proof. If a verified block contains the first crossing, scanning its event counts identifies E and retains exactly $y_{\leq E}$. If no crossing occurs before $L_{\max}$, the algorithm retains all $L_{\max}$ generated positions. If the sampler terminates earlier without a crossing, it retains the complete response of length L. These cases give the stated minimum. □

Remark 1 (Verification position and retained length)**.** *Algorithm 1 uses the verification position as the retained length. The implementation in Algorithm 2 instead finds the first crossing within the verified block and drops tokens after it. For a fixed trajectory, changing ρ changes the verification positions but not this crossing. If the crossing is found at verification position k, the decoding overshoot is $k - E \geq 0$.*

B.3 Verification Complexity

Assume $k_0 \leq L_{\max}$. Each round after the first advances by at least $k_{\min}$, except for a final step clipped to $L_{\max}$. Hence the number of rounds satisfies

$$R \leq 1 + \left\lceil \frac{L_{\max} - k_0}{k_{\min}} \right\rceil.$$

EOS or reaching the allowance can end the loop earlier.

The logarithmic behavior in Section 4.2 describes a stable-rate approximation. Ignoring integer rounding, suppose a block of m tokens adds λm events and $\hat{\lambda} = \lambda > 0$. For residual allowance $a = B - c$, Eq. (5) advances $\rho a/\lambda$ tokens when the minimum step is inactive, leaving $(1 - \rho)a$. For fixed $0 < \rho < 1$, the residual falls below one event after $O(\log B)$ rounds. At $\rho = 1$, this approximation reaches the boundary in one additional round. Actual round counts depend on event-rate variation; the finite-length bound above holds independently of that approximation.

C Method Details

C.1 Batched Implementation

Algorithm 2 gives the batched implementation of `Flash-OPD`, including termination at EOS or the length limit.

Active trajectories. Each round processes only trajectories that have not stopped. Requests are padded to a multiple of the worker count and unpadded on return. Trajectory indices remain fixed during the loop so that each prefix is matched to its own event count; length-based reordering is applied after the loop.

Prefix resumption. Each request extends $x^{(i)} \| y^{(i)}$ by at most $\delta^{(i)}$ tokens. Prefix caching avoids regenerating tokens from earlier rounds. Verification computes student and teacher distributions, and the controller reads the event indicators for the newly generated positions.

Stopping within a block. The controller scans the new positions in order and retains the prefix through the first position where $c_\tau \geq B$. If no crossing occurs, it retains the complete response when the sampler terminates, or stops at $L_{\max}$. A zero estimated event rate sends the next verification position to $L_{\max}$.

C.2 Decoding and Verification Cost

Algorithm 2 records two token counts. D counts generated positions, each once. P sums the verified prefix lengths across rounds because verification forwards recompute the prefix without a persistent KV cache. Thus decoding cost depends on D, while verification cost depends on P.

Algorithm 2 `Flash-OPD`: detail implementation version, one training step.

Require: prompts $\{x^{(i)}\}_{i=1}^{N}$; student π_θ; frozen teacher π_T; allowance B; initial checkpoint k_0; discount ρ; window W; minimum advance $k_{\min}$; context limit $L_{\max}$; overlap threshold η; candidate-set size k

Ensure: retained prefixes $\{y^{(i)}_{\le \hat{E}^{(i)}}\}$; cost counters D (decoded positions), P (prefix positions)

$\mathcal{A} \leftarrow \{1, \dots, N\}$; $D \leftarrow 0$; $P \leftarrow 0$ ▷ $\mathcal{A}$ is the active set
for $i = 1, \dots, N$ **do**
  $y^{(i)} \leftarrow (\,)$; $v^{(i)} \leftarrow 0$; $c^{(i)} \leftarrow 0$; $k^{(i)} \leftarrow \min(k_0,\, L_{\max})$ ▷ v: verified positions, c: drift count
end for
while $\mathcal{A} \neq \emptyset$ **do**
  $\delta^{(i)} \leftarrow k^{(i)} - v^{(i)}$ for every $i \in \mathcal{A}$ ▷ tokens still owed to this trajectory's checkpoint
  $Y \leftarrow \textsc{Generate}\big(\{x^{(i)} \| y^{(i)}\}_{i\in\mathcal{A}}\big)$ with per-request cap $\delta^{(i)}$ ▷ resume from the prefix cache; retired rows not dispatched
  $m^{(i)} \leftarrow \min\big(|Y^{(i)}|,\, \delta^{(i)}\big)$ ▷ clamp: pad = eos over-counts by one at the cap
  $\Phi \leftarrow \textsc{Verify}\big(\{x^{(i)} \| y^{(i)} \| Y^{(i)}\}_{i\in\mathcal{A}}\big)$ ▷ π_θ pass, then π_T pass; outputs read on the new block only
  $\phi^{(i)}_{v^{(i)}+j} \leftarrow \mathbb{1}\big[\,|K^S \cap K^T|/k < \eta\,\big],\quad j = 1, \dots, m^{(i)}$ ▷ Eq. (3)
  for $i \in \mathcal{A}$ **do**
    $s \leftarrow \varnothing$
    for $j = 1, \dots, m^{(i)}$ **do** ▷ localise the crossing inside the verified block
      $c^{(i)} \leftarrow c^{(i)} + \phi^{(i)}_{v^{(i)}+j}$; **if** $c^{(i)} \ge B$ **and** $s = \varnothing$ **then** $s \leftarrow v^{(i)} + j$
    end for
    $y^{(i)} \leftarrow y^{(i)} \| Y^{(i)}_{1:m^{(i)}}$; $v^{(i)} \leftarrow v^{(i)} + m^{(i)}$; $D \leftarrow D + m^{(i)}$; $P \leftarrow P + v^{(i)}$
    if $s \neq \varnothing$ **then**
      $\hat{E}^{(i)} \leftarrow s$; $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i\}$ ▷ retire: allowance
    else if $m^{(i)} < \delta^{(i)}$ **then**
      $\hat{E}^{(i)} \leftarrow v^{(i)}$; $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i\}$ ▷ retire: exhausted (sampler emitted EOS)
    else if $v^{(i)} \ge L_{\max}$ **then**
      $\hat{E}^{(i)} \leftarrow v^{(i)}$; $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i\}$ ▷ retire: context
    else
      $W' \leftarrow \min\big(W,\, v^{(i)}\big)$; $\hat{\lambda} \leftarrow \frac{1}{W'} \sum_{\tau=v^{(i)}-W'+1}^{v^{(i)}} \phi^{(i)}_{\tau}$ ▷ trailing-window drift rate
      $a \leftarrow L_{\max}$ **if** $\hat{\lambda} = 0$, **else** $a \leftarrow \big\lceil \rho\,(B - c^{(i)})/\hat{\lambda} \big\rceil$
      $k^{(i)} \leftarrow \min\big(v^{(i)} + \max(k_{\min},\, a),\; L_{\max}\big)$ ▷ Eq. (5)
    end if
  end for
end while
return $\{y^{(i)}_{\le \hat{E}^{(i)}}\}_{i=1}^{N}$ and (D, P) ▷ row order unchanged; length rebalancing runs only now

Verification includes both a student forward pass, which supplies the student's top-k candidates, and a teacher forward pass. Table 7 reports the number of rounds, verification tokens per retained token, and decoding overshoot. Table 6 gives the corresponding time breakdown.

C.3 Rollout Engine Settings

Cache lifetime and memory. The vLLM engine remains active throughout the verification loop and is put to sleep once all trajectories stop. We set `gpu_memory_utilization` to 0.6 to leave memory for verification, with a verification micro-batch budget of 4096 tokens.

Reusing verification outputs. Student log-probabilities and teacher outputs can be cached for each verified span and reused in the subsequent loss computation. Reuse is enabled only when every retained trajectory has complete coverage.

D EXPERIMENTAL DETAILS

D.1 DATASETS

Table 3 summarizes the training and evaluation datasets. All methods train on DAPO-Math-17K (Yu et al., 2026), which contains 17,917 competition-mathematics prompts. Each run uses 203 steps of 64 prompts, for 12,992 sampled prompts without repetition.

Table 3: The training corpus and the five in-domain benchmarks are competition mathematics; the four out-of-domain benchmarks are not, and GPQA is not mathematics at all.

Dataset	**Domain**	**Split used**	**#Problems**	**Evaluation budget**
Training corpus				
DAPO-Math-17K (Yu et al., 2026)	Competition mathematics	train (all)	17,917	—
In-domain evaluation				
AMC23	Competition mathematics	test	83	31744 tok., mean@16
AIME24	Competition mathematics	test	30	31744 tok., mean@16
AIME25	Competition mathematics	test	30	31744 tok., mean@16
HMMT24	Competition mathematics	test	30	31744 tok., mean@16
HMMT25	Competition mathematics	test	30	31744 tok., mean@16
total			*203*	
Out-of-domain evaluation				
MATH-500	Mathematics, mixed level	test	250	31744 tok., mean@4
Minerva	Undergraduate STEM	test	250	31744 tok., mean@4
Olympiad-Bench	Olympiad mathematics	test	250	31744 tok., mean@4
GPQA	Graduate-level science	test	198	31744 tok., mean@4
total			*948*	

The in-domain evaluation contains 203 problems across AMC23, AIME24, AIME25, HMMT24 and HMMT25. For transfer evaluation, we sample 250 problems uniformly without replacement from each of MATH-500, Minerva and Olympiad-Bench, and use all GPQA. The resulting 948 prompts are fixed across methods. Each response has a decoding budget of 31,744 tokens, with 16 samples per prompt in-domain and 4 for transfer. GPQA evaluates generalization beyond mathematics.

D.2 TEACHER–STUDENT PAIRS

Table 4: The four teacher–student pairs and the controller each one runs.

Student	**Teacher**	**Relationship**	**Max resp.**	k_0	W	$k_{\min}$	$L_{\max}$
R1-Distill-Qwen-1.5B	JustRL-DeepSeek-1.5B	Same-size RL-improved copy of the student	12288	1024	1024	512	4400
R1-Distill-Qwen-1.5B	R1-Distill-Qwen-7B	Larger same-family reasoning model	12288	128	128	16	12288
Qwen3-1.7B-Base	Qwen3-4B (Non-thinking)	Stronger same-family teacher, base student	8192	128	128	16	8192
Qwen3-4B-Base	Qwen3-4B (Non-thinking)	Same-size post-trained teacher, base student	12288	128	128	16	12288

We refer to the four pairs in Table 4, in order, as G1–G4. All use $\eta = 0.7$, $B = 200$ and $\rho = 1.0$. G2–G4 use $k_0 = W = 128$ and $k_{\min} = 16$. G1 uses $k_0 = W = 1024$, $k_{\min} = 512$, and $L_{\max} = 4400$, since its trusted prefix is typically longer.

D.3 BASELINES

- **OPD** (Lu & Thinking Machines Lab, 2025) performs on-policy distillation by sampling trajectories from the current student policy and using a frozen teacher to provide token-level supervision on the student-visited states. We implement OPD following Eq. (2) without adaptive truncation, using the response-length limits specified in Table 4.

Table 5: Parameter setting of our method `Flash-OPD`.

Parameter	Value
Optimisation	
Train batch size (prompts per step)	64
PPO mini-batch size	64 (one update per step)
Samples per prompt n	4
Learning rate	1×10^{-6}, constant
Micro-batch per GPU	1, dynamic batching enabled
Total steps / epochs	203 / 1
Max prompt length	1024 tokens
Max response length (rollout)	8192 on G3, 12288 on G1, G2, G4
Parameter dtype	fp32
Loss aggregation	token-mean
KL penalty	disabled
Prompt shuffling	enabled
OPD objective, Eq. (2)	
Advantage estimator	`token_reward_direct`
Candidate set size k	16
Candidate set source	student top-k (`only_stu`)
Reward weight mode	`student_p`
Student temperature	1.0
Teacher temperature	1.0
Repetition penalty	1.0
Format reward	disabled
ICT controller, Eqs. (3)–(4) and (5)	
Drift threshold η	0.70
Allowance B	200
Initial checkpoint k_0	128; 1024 on G1
Trailing window W	128; 1024 on G1
Minimum advance $k_{\min}$	16; 512 on G1
Step discount ρ	1.00 on all four pairs
Context limit $L_{\max}$	max response length; 4400 on G1
Verification micro-batch	4096 tokens
Sampler resident across rounds	yes, at `gpu_memory_utilization`$= 0.6$
Evaluation	
Decoding budget	31744 tokens
Samples per prompt	16 in-domain, 4 out-of-domain
Sampling	`do_sample`, $T = 1.0$, top-$p = 0.95$
Cadence	every 20 steps; once at step 203 for ablations
Reported metric	mean@n accuracy, macro-averaged over benchmarks

- **TOPD** (Zhang et al., 2026b) accelerates OPD by truncating student rollouts to a fixed horizon, thereby reducing the number of generated and supervised tokens. We use a fixed rollout limit of 4096 tokens and keep its evaluation budget identical to that of OPD.
- **Prune-OPD** (Yang et al., 2026c) dynamically adjusts a batch-shared rollout horizon based on feedback from previous training steps, reducing unnecessary computation while allowing the horizon to evolve during training. We use threshold 0.7, $w_{\text{drop}} = 0.01$, $w_{\text{base}} = 0.5$, initial and minimum rollout lengths of 1024 tokens, and a hit ratio of 0.1.

Table 6: Where the speedup comes from: mean wall-clock seconds per training step.

Method	Step-time attribution (s/step)					Tokens vs. time		
	Rollout	Teacher	Log-prob	Update	**Step**	Length	Token ↓	Step ↓
R1-1.5B ← JustRL-1.5B								
OPD	70.3	11.2	10.5	31.5	132.3	6122	–	–
`Flash-OPD`	26.2	14.2	0.0	14.7	60.0	2708	2.3×	2.20×
R1-1.5B ← R1-7B								
OPD	88.8	40.4	13.3	42.4	194.3	7961	–	–
`Flash-OPD`	14.4	12.2	0.0	4.3	36.5	585	13.6×	5.32×
Qwen3-1.7B-Base ← Qwen3-4B								
OPD	109.7	25.8	12.6	41.1	196.1	6647	–	–
`Flash-OPD`	11.1	10.1	0.0	2.7	27.5	304	21.9×	7.13×
Qwen3-4B-Base ← Qwen3-4B								
OPD	306.8	47.8	45.6	175.7	586.9	11278	–	–
`Flash-OPD`	44.2	19.7	0.0	8.5	78.1	569	19.8×	7.52×

Table 7: Mechanism counters, averaged over all training steps.

Pair	Retained length $\hat{E}$			Stopping rule (of 256)			Verification cost		
	min	mean	max	allowance	$L_{\max}$	EOS	rounds	amplif.	overshoot
R1-1.5B ← JustRL-1.5B	830	2708	3509	130	96	30	2.64	1.54×	231 (8.5%)
R1-1.5B ← R1-7B	335	585	1052	256	0	0	6.23	2.41×	26 (4.5%)
Qwen3-1.7B-Base ← Qwen3-4B	126	304	672	243	0	13	4.98	2.23×	14 (4.5%)
Qwen3-4B-Base ← Qwen3-4B	293	569	2117	250	0	6	6.23	2.19×	62 (10.9%)

D.4 Protocol and Hyperparameters

Training and evaluation. Table 5 lists the optimization, controller and decoding settings. Each training batch contains 64 prompts with 4 responses each, giving 256 trajectories. Accuracy is the mean success rate over sampled responses, macro-averaged across benchmarks.

Hardware and timing. Each run uses one node with 8 NVIDIA H20 GPUs (approximately 96 GB each), two AMD EPYC 9K84 96-core CPUs and approximately 2.2 TiB of host memory. Reported training time sums training-step durations and excludes evaluation.

E Additional Results

This section provides the numerical results and trajectory statistics supporting the efficiency, mechanism, transferability and robustness studies in Section 5.

E.1 Training-Time Breakdown

Table 6 gives the values underlying Figure 4. Shorter rollouts reduce generation and policy-update time on all four pairs. Response-token reductions exceed step-time reductions because prompt processing, repeated verification and other per-step costs remain. The step-time ratios range from 2.20× to 7.52×, compared with response-token reductions of 2.3× to 21.9×.

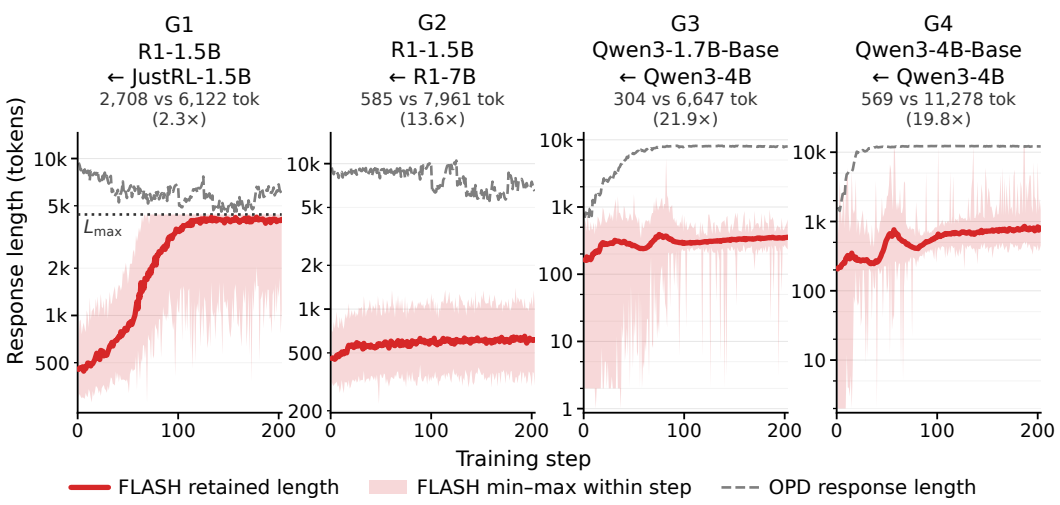

Figure 9: Retained lengths over training.

Table 8: Out-of-domain transfer.

Pair	Method	MATH-500	Minerva	Olympiad	GPQA	Avg.	textbfΔ
R1-1.5B ← JustRL-1.5B	OPD	90.40	38.40	61.60	36.93	56.83	
	`Flash-OPD`	90.60	38.10	58.70	34.39	55.45	tie
R1-1.5B ← R1-7B	OPD	85.20	34.50	51.40	11.80	45.73	
	`Flash-OPD`	85.20	34.30	50.00	15.23	46.18	tie
Qwen3-1.7B-Base ← Qwen3-4B	OPD	57.60	22.80	26.80	15.36	30.64	
	`Flash-OPD`	59.80	23.30	29.30	13.96	31.59	tie
Qwen3-4B-Base ← Qwen3-4B	OPD	73.50	32.90	43.60	16.24	41.56	
	`Flash-OPD`	79.20	38.20	47.70	24.49	47.40	+5.84

E.2 Trajectory Lengths and Stopping Statistics

Table 7 summarizes retained lengths, stopping reasons and verification costs over 203 training steps. The mean retained length ranges from 304 to 2708 tokens across pairs, with substantial variation within each batch. Figure 9 shows how these lengths evolve during training.

The allowance determines most stopping decisions on G2–G4. On G1, an average batch stops 130 trajectories at the allowance, 96 at the length limit and 30 at EOS. Across all four pairs, verification processes $1.54\times$–$2.41\times$ as many positions as are retained, and decoding overshoot is 4.5%–10.9% of the retained length.

E.3 Out-of-Domain Transfer

Table 8 gives the benchmark-level results for Figure 6. Macro differences on G1–G3 are within 1.4 points. G4 improves on all four benchmarks, with gains of 5.70, 5.30, 4.10 and 8.25 points on MATH-500, Minerva, Olympiad-Bench and GPQA, respectively. G1 has lower scores on Olympiad-Bench and GPQA, by 2.90 and 2.54 points.

E.4 Cross-Generation Transfer

Table 9 report cross-generation results. On R1-1.5B←Qwen3-4B, we reaches 28.45 average accuracy compared with 28.34 for OPD, while reducing training time from 12.02 to 1.38 hours.

Table 9: Cross-generation transfer.

Pair	Method	AMC23	AIME24	AIME25	HMMT24	HMMT25	Avg.	Time	Speed Up
R1-1.5B ← Qwen3-4B	Student	63.35	26.80	20.00	12.40	12.30	26.97	–	–
	OPD	60.02	31.25	22.92	12.92	14.58	28.34	12.02h	1.00×
	`Flash-OPD`	64.76	27.92	22.08	13.96	13.54	28.45	1.38h	8.71×

F Limitations and Future Work

While our results demonstrate that `Flash-OPD` substantially accelerates OPD while maintaining competitive performance, several limitations warrant further investigation.

❶ **Compatibility criterion.** `Flash-OPD` uses top-k overlap as an online proxy for local teacher–student compatibility. Accordingly, our stopping guarantee applies to the boundary induced by this criterion, rather than directly guaranteeing downstream accuracy or that all supervision beyond the boundary is uninformative. Future work may explore richer compatibility signals while retaining efficient online evaluation.

❷ **Truncation and optimization.** Although truncation preserves the token-level supervision already computed on the retained prefix, it removes subsequent terms and therefore does not preserve the full-rollout OPD objective exactly. Further analysis of gradient utility and compute-matched rollout allocation could better characterize when trajectory-specific stopping is preferable to uniformly shorter rollouts.

❸ **System and empirical scope.** Our reported speedups include verification overhead, but their magnitude may depend on hardware, batching, and verification implementation. Moreover, our experiments primarily use mathematics training and a limited set of model families and scales. Evaluating larger students, broader training domains, additional hardware settings, and independent training runs would further establish the generality of the observed accuracy–efficiency trade-off.

❹ **Beyond token-level supervision.** Our formulation assumes token-level teacher supervision that can be evaluated before a trajectory is completed. Extending adaptive stopping to sequence-level rewards, black-box teachers, or agentic settings is less straightforward because useful feedback may depend on later outcomes. Developing intermediate reliability signals for such settings is a promising direction for future work.