

# Fast and Effective On-policy Distillation from Reasoning Prefixes

Dongxu Zhang\* Zhichao Yang Sepehr Janghorbani Jun Han  
Andrew Ressler II Qian Qian Gregory D. Lyng  
Sanjit Singh Batra\* Robert E. Tillman\*  
Optum AI

## Abstract

On-policy distillation (OPD), which samples trajectories from the student model and supervises them with a teacher at the token-level, avoids relying solely on verifiable terminal rewards and can yield better generalization than off-policy distillation. However, OPD requires expensive on-the-fly sampling of the student policy during training, which substantially increases training cost, especially for responses with long reasoning traces. Our initial analysis shows that, during OPD, training signals are stronger in the prefix of each output reasoning trace, and that even a short teacher-generated prefix can significantly help the student produce the correct answer. Motivated by these observations, we propose a simple yet effective modification of OPD: we apply the distillation objective only to *prefixes* of student-generated outputs and terminate each sampling early during distillation. Experiments on a suite of AI-for-Math and out-of-domain reasoning benchmarks show that on-policy prefix distillation matches the performance of full OPD in long reasoning outputs while reducing training FLOP by 2×–40×.

## 1 Introduction

As large language models (LLMs) continue to scale in size, model compression (Buciluă et al., 2006) becomes increasingly important for deploying LLMs to real-world applications where resources are often limited. Knowledge distillation (Hinton et al., 2015) is a widely-used technique in machine learning for compressing a large teacher model into a smaller student model while retaining the teacher’s capabilities.

In the context of LLMs, distillation is commonly performed via *sequence-level knowledge distillation* (SeqKD), where the teacher first generates a

fixed set of off-policy token trajectories and the student is trained via next-token prediction on these teacher outputs (i.e., SFT on teacher-generated sequences) (Kim and Rush, 2016; Radford et al., 2019; Wei et al., 2022; Ouyang et al., 2022). Despite its broad adoption, this approach has a notable limitation, known as the *exposure bias problem*, where the training is exclusively using the teacher-generated data; therefore, the student rarely learns from its own induced states and errors can accumulate at test time (Lin et al., 2020). More broadly, recent work suggests that aggressive SFT can be associated with degraded generalization (Chu et al., 2025), catastrophic forgetting (Luo et al., 2025; Kalajdziewski, 2024), and increased hallucination rates (Gekhman et al., 2024; Kalai et al., 2025), highlighting the need for training signals that better match the student’s deployment-time state distribution. These limitations motivate moving beyond purely off-policy imitation.

On-policy learning (Williams, 1992), widely studied in reinforcement learning (RL), trains a model using trajectories sampled from its own policy, reducing train–test mismatch. In LLM training, this means sampling from the *current* student during optimization rather than relying on a fixed teacher-generated dataset. However, many on-policy RL methods require full rollouts to obtain terminal rewards (Lambert et al., 2024), which can be prohibitively expensive for long-form reasoning.

On-policy distillation (OPD) (Lin et al., 2020; Gu et al., 2024; Ko et al., 2024; Agarwal et al.; Lu and Lab, 2025) addresses key limitations of these issues from SFT and RL by minimizing the divergence between policies of student-generated outputs (SGO) and a teacher at token level. Compared to reward-based RL, OPD provides dense, token-level teacher feedback without requiring task-specific reward design. This dense supervision can make OPD more learning-efficient than sparse-reward RL; however, each update is still expen-

\*Correspondence to:  
{dongxu.zhang, sanjit.batra, rob.tillman}@optum.com

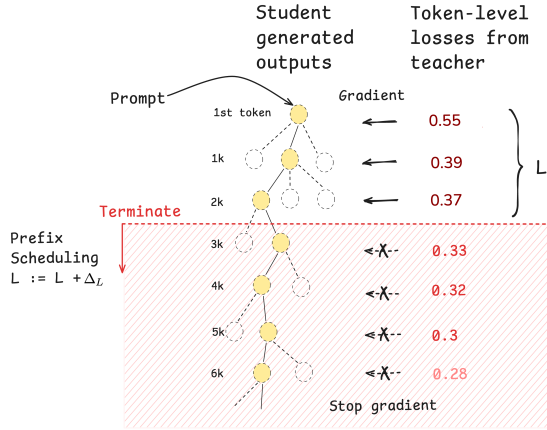


Figure 1: **On-policy prefix distillation.** During training, gradients are backpropagated only through the first  $L$  tokens of each student sampling, and rollout is terminated after  $L$ . Prefix scheduling increases  $L$  by a fixed amount at each training step. At test time, the student generates autoregressively until producing the final answer.

side: to preserve the on-policy property, OPD must generate student rollouts on the fly at each update rather than reusing a static off-policy dataset. In reasoning-heavy tasks, these solution trajectories can span tens of thousands of tokens, making both sampling and teacher scoring prohibitively expensive.

Because OPD provides token-level teacher feedback, full-length supervision may be unnecessary. This raises our central research question: **Can we preserve the benefits of on-policy distillation while *only* supervising the prefix of each student trajectory, thereby reducing the training cost?**

We test two hypotheses in long reasoning trace settings: (a) OPD’s learning signal is assigned unequally towards early tokens; (b) aligning this prefix is often sufficient for the student to complete the remaining reasoning correctly. Experiments in Section 4 on math reasoning benchmarks show that prefix OPD achieves performance close to full OPD while reducing training compute significantly.

### Contributions.

- We propose *on-policy prefix distillation*, which truncates rollouts during training and applies distillation only on the prefix.
- We introduce a simple *prefix scheduling* strategy that progressively increases the trained prefix length over training.
- On AI-for-Math and out-of-domain reasoning benchmarks, prefix OPD closely matches full

OPD while reducing training FLOP by 2×–40×.

## 2 Preliminary Study

In this section, we first describe the learning objective of on-policy distillation. Then, we provide experimental analysis to validate the idea of on-policy prefix distillation.

### 2.1 On-policy Distillation

On-policy distillation can be viewed as an on-policy variant of knowledge distillation: at each iteration, the student samples responses from its own policy  $\pi_s$  and is optimized to match a fixed teacher  $\pi_T$  under a KL-based objective.

$$\min_{\pi_s} \left\{ \text{KL}(\pi_s \parallel \pi_T) \right\}.$$

Since the expectation is taken over the student’s own state distribution, OPD avoids the distributional mismatch that plagues off-policy imitation and yields gradients aligned with the deployment policy.

We choose to use the reverse KL divergence because it encourages the student to assign high probability to teacher-preferred actions, reducing opportunities for reward hacking, which often happens when using forward KL. Its mode-seeking nature also reduces the risk of spreading the student policy over many suboptimal options (Gu et al., 2024; He and Lab, 2025)<sup>1</sup>. In addition, though previous work (Agarwal et al.) also allows using a mixture of forward and reverse KL, it shows that for arithmetic reasoning, such Jensen-Shannon divergence could lead to decreased performance.

During training, since it is intractable to sample all possible responses, we adopt the following unbiased estimation via sampling:

$$\begin{aligned} \text{KL}(\pi_s \parallel \pi_T) &= \mathbb{E}_{x_{1:L} \sim \pi_s} \left[ \sum_{t=1}^{L-1} \log \frac{\pi_s(x_{t+1} \mid x_{1:t})}{\pi_T(x_{t+1} \mid x_{1:t})} \right] \\ &\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^{L_i-1} \log \frac{\pi_s(x_{t+1}^{(i)} \mid x_{1:t}^{(i)})}{\pi_T(x_{t+1}^{(i)} \mid x_{1:t}^{(i)})}, \end{aligned} \quad (1)$$

where  $L$  is the maximum response length and  $N$  is the number of samples. Note that this also allows

<sup>1</sup>We use AI4Math benchmarks in this work for evaluation, and often a math problem can be solved by more than one solution path. For example, in Appendix G, we show that AIME-2024 has approximately 5.4 unique solutions per question.

one to avoid storing logits over the full vocabulary for each token in the trajectory<sup>2</sup>. Optionally, a student can also receive rewards from the environment and/or use KL regularization to its old policy for robustness and downstream needs. However, in this work we do not explore these losses for simplicity and focus on the on-policy distillation component. To mitigate off-policy drift introduced by floating-point non-associativity during rollout (He and Lab, 2025), we apply importance sampling during the gradient update<sup>3</sup>.

## 2.2 Does Prefix Receive More Training Signals?

To understand where the learning signal originates from, we measure the token-level average reverse-KL loss from Eq. 1 over tokens grouped into each positional bin along the trajectory during on-policy distillation training.

We use Qwen3-1.7B-Base and Qwen3-8B-Base (Yang et al., 2025) as the student models, and Qwen3-8B as the teacher model. Figure 2 shows the distribution of the average magnitude of reverse-KL loss at 10th, 20th, and 30th steps of full on-policy distillation training on the questions sampled from OpenThoughts3 (Guha et al., 2026) dataset. Each point in the curve corresponds to a positional bin covering 256 tokens, with a maximum output length of 16,384 tokens. During training, batch size 512, 4 samples per prompt and learning rate  $5e-5$  are used, following Lu and Lab (2025).

The distribution plot shows that early tokens consistently receive larger losses across training steps and model sizes. We conjecture that this pattern reflects the student model being weaker at high-level planning captured early in the trajectory than at executing later steps of a plan.

## 2.3 Can a Student Model Complete a Truncated Response from the Teacher without Training?

Another question is whether a student that receives supervision only at the beginning of its responses can subsequently finish the solution cor-

<sup>2</sup>A single sample of response with 16k tokens would take 4.8GB memory under BF16. This was not a challenge in the previous on-policy distillation work such as GKD, MiniLLM, DistillLM as the max response length was often 512.

<sup>3</sup>For more details, please refer to <https://tinker-docs.thinkingmachines.ai/losses#policy-gradient-importance-sampling>. We also follow this implementation and sum the token-level losses over the sequence length.

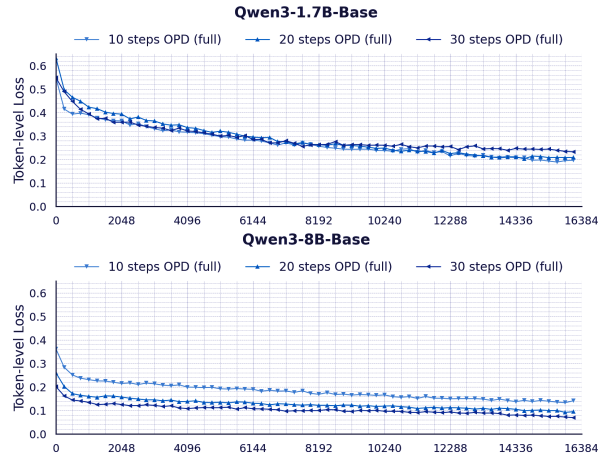


Figure 2: Mean token-level reverse-KL loss by output token position during OPD training for Qwen3-1.7B-Base (top) and Qwen3-8B-Base (bottom). We plot checkpoints after 10, 20, and 30 gradient steps.

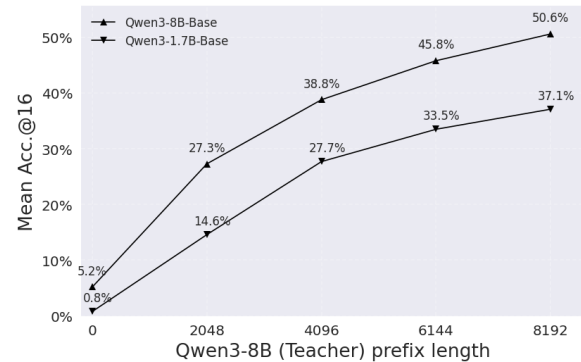


Figure 3: MEAN ACC.@16 on AIME-24 when an untrained student (Qwen3-8B-Base and Qwen3-1.7B-Base) continues from varying lengths of a teacher (Qwen3-8B)-generated prefix.

rectly. To probe this, we conduct a *teacher-prefix* study: we let the teacher LLM (Qwen3-8B) generate full solutions, feed the first  $L$  tokens ( $L \in \{2048, 4096, 6144, 8192\}$ ) to untrained students, and ask them to continue the reasoning unaided. This experiment probes the model’s ability to “fill in the tail”.

Figure 3 reports mean accuracy at 16 samples on AIME-24. During this experiment, we also ensure that teacher’s final answer is not likely in the prefix by enforcing  $L_p \leftarrow \min\{L_p, L_r - 512\}$  where  $L_p$  is the length of teacher’s prefix and  $L_r$  is the length of teacher’s full response for the current input. Note that teacher’s average output length on AIME-24 is 15k and maximum output length for this experiment is set to 32k.

The plot shows that teacher’s reasoning prefix does improve student performance, suggesting that

the early guidance from the teacher can effectively help students to infer the remaining steps even without further training. In addition, the largest performance gains come from the first few thousand tokens. This again verifies that the prefix is crucial during the reasoning process.

### 3 Methodology

Figure 1 shows the main idea of prefix distillation. In this work, we enforce prefix-only training by limiting the maximum number of tokens from each generated response during *training* to  $L^{\text{train}}$ . No additional modification to the loss or sampling procedure is required; we simply early-stop the on-policy generation once the budget is reached, backpropagate through the truncated trajectory for gradient updates, then proceed to the next batch. The on-policy property is preserved for the first  $L^{\text{train}}$  tokens.

At evaluation time we remove the cap<sup>4</sup>, ensuring that final task performance is not artificially constrained. This design choice keeps the implementation minimal, yet yields significant savings of computation during training.

**Special Tokens.** Though memory friendly for long reasoning scenarios, we observe that it is challenging for the *sample-based* KL estimation in Eq. 1 to learn from generating special tokens. When such supervision is provided from the teacher, the student repeatedly attempts to switch to a different token which leads to generating unreadable content. To avoid falling into such a scenario, in our training, `<think>` is enforced at the end of the input prompt, to align with the teacher’s behavior.

**Prefix Scheduling.** Although short prefixes reduce cost, later tokens in the student’s rollout can still carry valuable supervision when scored under the teacher. To progressively expose the student to longer-horizon feedback, we explore a simple linear schedule that increases the training prefix length after each gradient update:  $L^{\text{train}} \leftarrow L^{\text{train}} + \Delta_L$ . In our experiments,  $L^{\text{train}}$  starts with 1 and  $\Delta_L = 256$ .

## 4 Experiments

### 4.1 Setup

We randomly sample questions from OPENTHOUGHTS3 (Guha et al., 2026) for

<sup>4</sup>In the rest of the experiments, we allow up to  $L^{\text{test}}=16384$  tokens

training, use Qwen3-1.7B-Base / Qwen3-8B-Base as the student, and Qwen3-8B as the on-policy teacher. We leverage off-policy responses of QwQ-32B (Qwen, 2024) from OpenThoughts3 for SeqKD<sup>5</sup>.

**Hyperparameters.** Following Lu and Lab (2025), for SeqKD, we use batch size 128, optimize using AdamW with learning rate  $1 \times 10^{-4}$ ; for OPD and our prefix variants, we use batch size 512, 4 samples per prompt, learning rate  $5 \times 10^{-5}$ . More details on the learning rate sweep can be found in Appendix D. We implement training with VERL (Sheng et al., 2025) and leverage vllm (Kwon et al., 2023) to speed up the sampling process. All OPD experiments are run for 60 training steps, capped by a 7-day budget of 8xA100 GPUs for the full OPD training using Qwen3-8B-Base. We use in total 30.7k questions for OPD training, and 192k examples for SeqKD to reach a comparable FLOP.<sup>6</sup>

**Benchmarks.** Evaluation is performed on MATH500 (Hendrycks et al., 2021), AIME-24, AIME-25 as in-domain testing, GPQA (Rein et al., 2024), and a random sample of 500 questions from MMLU-Pro (Wang et al., 2024) as out-of-domain testing. These reasoning benchmarks have verifiable final answers for straightforward evaluation. During evaluation, temperature = 1.0, top\_p = 1.0 and maximum response length is 16384.

**Metrics.** For MATH500, GPQA and MMLU-Pro, we report mean accuracy<sup>7</sup> among 4 samples. For AIME-24 and AIME-25 we calculate mean accuracy over 16 samples to reduce the variance. We use AIME-24 as a development set and choose the best checkpoint for each baseline to report accuracy. More details can be found in Appendix F. To compare computational cost during training, we estimate the number of GPU FLOP (floating point operations) including forward sampling, log prob computation from teacher and student, backward passes, and gradient updates accordingly. For more details, please refer to Appendix B. We also report GPU hours<sup>8</sup> to take into account I/O, CPU and

<sup>5</sup>Note that QwQ-32B performs slightly better than Qwen3-8B on MATH-500, AIME-24 and AIME-25 based on Yang et al. (2025).

<sup>6</sup>OPD uses significantly higher FLOP per prompt than SeqKD because it requires multiple response sampling during training and needs to calculate  $\log \pi_T(x_{t+1} | x_1, \dots, x_t)$ .

<sup>7</sup>We adopt `math_metric` from `math-verify` with `ExprExtractionConfig` and `LatexExtractionConfig` parsers.

<sup>8</sup>We report hours on an 8xA100 VM. Evaluation time is excluded.

	In-Domain			Out-of-domain		Training cost	
	MATH500	AIME-24	AIME-25	GPQA	MMLU-Pro	FLOP	Hours
Qwen3-1.7B-Base	20.1	0.2	0.0	11.2	9.1	No post training	
<i>w. limited budgets</i>							
OPD - 10 steps	43.4	5.4	3.5	18.3	<b>30.7</b>	$8.2 \times 10^{18}$	12.2
SeqKD - 300 steps	<b>60.5</b>	4.6	6.9	<b>20.2</b>	<u>28.5</u>	$1.3 \times 10^{19}$	3.7
OPD (prefix 256)	50.0	4.8	6.3	1.0	1.5	$9.4 \times 10^{17}$	2.4
OPD (prefix 512)	47.6	4.2	6.0	1.1	1.0	$1.4 \times 10^{18}$	2.8
OPD (prefix 1024)	54.5	<b>8.1</b>	<b>9.0</b>	1.1	1.9	$2.5 \times 10^{18}$	3.4
OPD (prefix 2048)	<u>58.3</u>	<u>7.3</u>	<u>7.5</u>	4.5	12.0	$4.7 \times 10^{18}$	5.0
<i>w. sufficient budgets</i>							
OPD	67.3	10.0	11.5	24.6	41.3	$5.7 \times 10^{19}$	68.6
SeqKD	70.8	8.8	14.4	19.8	29.6	$6.6 \times 10^{19}$	18.3
OPD (prefix scheduling)	68.1	10.6	11.5	23.6	39.4	$2.4 \times 10^{19}$	30.8

Table 1: Mean accuracies and costs with Qwen3-1.7B-Base as the student.

	In-Domain			Out-of-domain		Training cost	
	MATH500	AIME-24	AIME-25	GPQA	MMLU-Pro	FLOP	Hours
Qwen3-8B-Base	39.0	5.2	2.1	17.6	26.5	No post training	
<i>w. limited budgets</i>							
OPD - 10 steps	77.6	23.3	24.2	<b>47.5</b>	<b>62.7</b>	$2.7 \times 10^{19}$	23.3
SeqKD - 300 steps	<b>88.3</b>	31.9	25.4	38.5	52.6	$4.4 \times 10^{19}$	11.3
OPD (prefix 256)	82.0	30.8	26.3	35.5	58.9	$3.4 \times 10^{18}$	3.2
OPD (prefix 512)	84.7	33.5	26.7	35.7	60.3	$5.2 \times 10^{18}$	4.8
OPD (prefix 1024)	84.1	<u>36.5</u>	<u>28.3</u>	37.1	60.5	$8.9 \times 10^{18}$	5.6
OPD (prefix 2048)	<u>87.1</u>	<b>39.2</b>	<b>31.3</b>	<u>40.0</u>	<u>61.3</u>	$1.7 \times 10^{19}$	9.8
<i>w. sufficient budgets</i>							
OPD	87.4	44.2	33.5	48.8	66.5	$1.8 \times 10^{20}$	148.3
SeqKD	91.3	41.7	34.6	42.2	55.4	$2.2 \times 10^{20}$	56.7
OPD (prefix scheduling)	86.9	44.0	33.5	48.3	66.1	$7.5 \times 10^{19}$	53.0

Table 2: Mean accuracies and costs with Qwen3-8B-Base as the student.

other computation cost during training.

**Baselines.** We compare the base model, off-policy SeqKD, OPD with maximum 16,384 tokens during training, and prefix OPD. For limited-budget comparisons we use 10-step OPD, 300-step SeqKD, and constant prefix OPD, as they have a similar order of magnitude in training FLOP. For sufficient-budget comparisons, we use 60-step OPD, 1500-step SeqKD, and prefix scheduling.

**Long-horizon reasoning.** All results in this paper are evaluated under a *chain-of-thought reasoning* regime, with thinking mode enabled. Although all benchmarks only require a short final answer, the model generates a long reasoning trace before emitting the final prediction. For example, on AIME-25 testset, the Qwen3-8B-Base student trained with full-trajectory OPD generates 13,629 tokens on average before the final answer, while the student trained with prefix OPD ( $L=2048$ ) generates 14,774 tokens on average. We also conducted a length constraint ablation study in Appendix E.

## 4.2 Main Results

Table 1 and 2 present our main results. All experiments are zero-shot. The best and second-best results within each column are highlighted with bold and underline accordingly, excluding results with sufficient budgets.

**In-domain.** From in-domain results, OPD with prefix 1024 / 2048 outperforms baselines with limited budgets in AIME-24 and AIME-25 with half the FLOP. On MATH500, OPD with prefix outperforms OPD (10 steps), while performing only slightly worse than SeqKD (300 steps). When using Qwen3-8B-Base as the student, prefix OPD approaches sufficient-budget performance while using less than 10% of the FLOP. These results indicate that prefix OPD can effectively reduce the training cost while maintaining a competitive in-domain reasoning performance.

**Out-of-domain** We observe different out-of-domain behavior depending on student size. For the 8B model, prefix OPD improves over the base model, and performance tends to move closer to full OPD as we increase  $L^{\text{train}}$  (Table 2). In con-

trast, for 1.7B model, training on very short prefixes can largely reduce out-of-domain accuracy below the pre-trained student (Table 1). One plausible explanation to this failure mode is that, for small students, optimizing only early tokens can improve how the solution starts but worsen the model’s ability to continue reliably over long responses (also see **Finding 3**). In these cases, prefix scheduling (gradually increasing  $L^{\text{train}}$  during training) is a more robust choice. As shown in Table 1, OPD with prefix scheduling leads to competitive out-of-domain performance.

**Scheduling.** Our linear prefix scheduling strategy provides benefits of both prefix distillation and full trajectory distillation. Across all settings, scheduled prefix OPD matches the upper-bound performance of full OPD while using less than 50% of the FLOP. These results suggest that allocating compute progressively to longer prefixes is effective. It is worth exploring more adaptive schedulers (e.g., curriculum based on validation performance or KL/entropy signals) in future work.

### 4.3 Ablation Studies

Prefix on-policy distillation can be viewed as restricting the training loss to a selected subset of token positions: only the first  $L_{\text{train}}$  tokens in each rollout contribute to the update, and tokens outside this region are masked out. This formulation allows us to test whether some parts of the rollout are more useful than others for improving downstream accuracy. To do so, we conduct an ablation study over supervision windows at different positions in the rollout. Specifically, we apply a fixed-length sliding window and compute gradients only for tokens inside that window. The evaluated windows are [1024, 2047], [2048, 3071], [3072, 4095], and a 1024-token window at the end of the rollout.

Table 3 shows that the prefix window leads to the best results among all ablation windows, suggesting that “earlier is more important” is an effective heuristic.

Method	Step 10	Step 20	Step 30
OPD (full)	23.3	40.8	40.0
OPD on prefix 1024	<b>26.3</b>	<b>32.5</b>	<b>34.8</b>
OPD on [1024, 2048)	14.0	16.0	16.0
OPD on [2048, 3072)	18.8	16.9	15.8
OPD on [3072, 4096)	9.4	19.2	22.5
OPD on tail 1024	17.9	23.8	23.8

Table 3: Ablation of masking strategies with Qwen3-8B-Base as student and Qwen3-8B as the teacher. We report Mean Acc.@16 on AIME-24.

### 4.4 Case Study

We use an AIME-24 example to illustrate how *early* guidance can prevent compounding reasoning errors and why prefix-only supervision can be effective. In this example, we use Qwen3-8B-Base as the initial student and Qwen3-8B as the teacher model.

**An AIME-2024 Problem.** Let  $x, y$  and  $z$  be positive real numbers that satisfy the following system of equations:  $\log_2\left(\frac{x}{yz}\right) = \frac{1}{2}, \log_2\left(\frac{y}{xz}\right) = \frac{1}{3}, \log_2\left(\frac{z}{xy}\right) = \frac{1}{4}$ . Then the value of  $|\log_2(x^4y^3z^2)|$  is  $\frac{m}{n}$  where  $m$  and  $n$  are relatively prime positive integers. Find  $m + n$ .

**A Valid Solution Sketch.** Let  $a = \log_2 x, b = \log_2 y, c = \log_2 z$ . Using  $\log_2\left(\frac{x}{yz}\right) = \log_2 x - \log_2 y - \log_2 z$ , the system becomes  $a - b - c = \frac{1}{2}, b - a - c = \frac{1}{3}, c - a - b = \frac{1}{4}$ . Solving yields  $a = -\frac{7}{24}, b = -\frac{3}{8}, c = -\frac{5}{12}$ , hence  $|\log_2(x^4y^3z^2)| = |4a + 3b + 2c| = \frac{25}{8}$ , so  $m + n = 33$ .

#### Behavior Comparison:

– **Base student (incorrect).** When prompted directly, the model makes an early conversion error, e.g., rewriting the system as  $x = 2^{1/4}yz, y = 2^{1/5}xz, z = 2^{1/6}xy$  (already inconsistent with  $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}$ ), then carries this mistake through and produces a confident but wrong final answer (e.g.,  $m + n = 5$ ), without detecting the inconsistency.

– **Base student + teacher prefix (correct).** If we prepend a *short* teacher-generated prefix (128 tokens from Qwen3-8B; containing only the initial reasoning, excluding the final answer), the same student follows the correct transformations (e.g.,  $\frac{x}{yz} = 2^{1/2}$ ), then leverages algebraic elimination and reaches the correct result  $m + n = 33$ .

– **Student after prefix OPD (correct).** After on-policy prefix distillation, the student no longer requires an explicit teacher prefix: it immediately introduces  $(a, b, c)$ , solves the linear equations, computes  $4a + 3b + 2c = -\frac{25}{8}$ , and returns  $m + n = 33$ .

This case study highlights that aligning the *start* of the student’s reasoning can prevent early, compounding errors and change the final outcome. Full prompts and generations are provided in Appendix A.

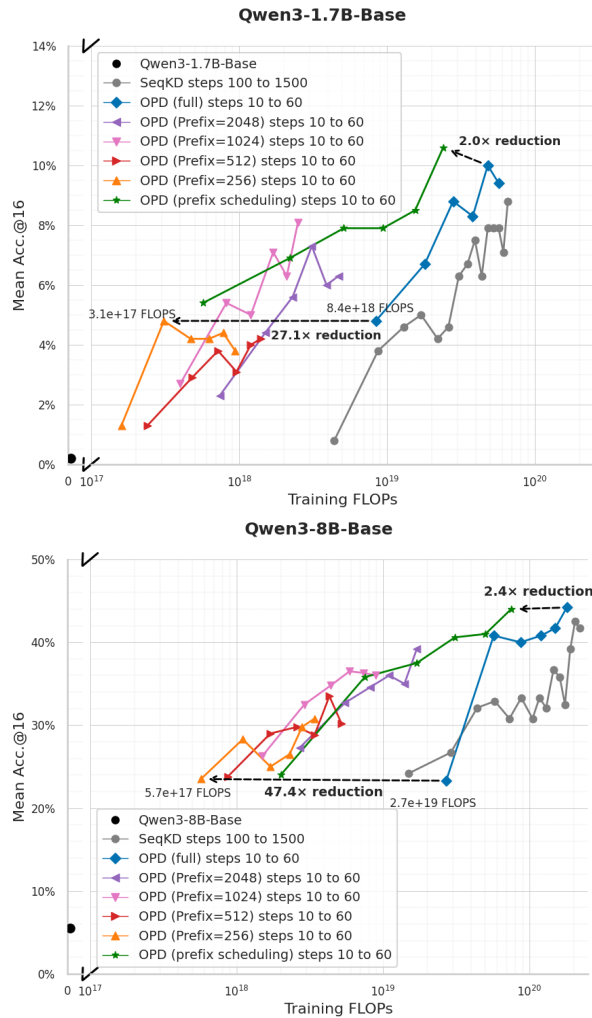


Figure 4: Comparison of training compute vs Mean Acc.@16 on AIME-24. **Top:** Student is Qwen3-1.7B-Base. **Bottom:** Student is Qwen3-8B-Base.

## 5 Discussion

### 5.1 Cost and Performance Trade-off

In this section, we discuss the tradeoff between the training cost and validation accuracy given different prefix lengths. Figure 4 shows the dynamics during training on AIME-24.

**Finding 1: Given a limited computing budget, distillation with a shorter prefix can enable significant performance gains. When the budget is sufficient, a longer prefix eventually performs better via prefix scheduling.** If we compare performances on the left sides (10th and 20th steps) of each curve, we can see that a shorter prefix can often yield significant accuracy gain in comparison to a longer prefix. For instance, the bottom figure shows that SeqKD requires  $1.5 \times 10^{19}$  to

reach a 22.5% accuracy<sup>9</sup>, and full OPD requires  $2.7 \times 10^{19}$  FLOP to reach a similar performance. In comparison, 10 steps of OPD on 256 tokens is able to increase the student model’s accuracy from 5% to 23.5% with only  $5.7 \times 10^{17}$  FLOP, a 47x reduction.

In contrast, if we compare accuracies of right sides (50th and 60th steps) of each curve, we often see that a longer prefix can lead to higher accuracies<sup>10</sup>. This suggests that when given sufficient budgets for training, a longer prefix is preferred for better performance, and that *prefix scheduling* is highly recommended rather than a full distillation from the beginning of training. This strategy can often save at least half of the training cost.

### 5.2 How Does Prefix Learning Affect the Tail?

To investigate the effect of prefix-only learning beyond the prefix length, we plot a token-level loss distribution with OPD with prefix=1024 on Qwen3-8B-Base on OpenThoughts3.

**Finding 2: Prefix on-policy distillation can reduce the loss on the tokens outside the prefix.** Fig 5 shows the distribution over different positions in the rollout at step 10, 20 and 30. It shows that with more prefix OPD training, token losses outside the prefix (position 1024) also decrease (shown as the down arrows). The decrease is most significant when close to the prefix and fades away on the tail. This is not surprising as the knowledge learned at the prefix may also be leveraged during the rest of the output through the model’s parameters.

**Finding 3: Learning the prefix benefits the tail for a student model with sufficient parameters, but can hurt the tail given a small model size.** Interestingly, from Fig 5, for a compact model like Qwen3-1.7B-Base, the loss is increased on the tail with more training steps (shown as the up arrow in the figure). This indicates that a small model size could lead to the prefix "competing" the loss reduction with the tail rather than "sharing" the benefit during on-policy distillation.

A potential concern is shortcut learning (Geirhos et al., 2020; Du et al., 2023). By supervising only the prefix, the student may rely more heavily on shallow stylistic or local reasoning cues rather than

<sup>9</sup>For SeqKD, we didn’t take into account the cost of generating the off-policy training data. In the real scenario, if we take this overhead into account, the cost could be significantly higher.

<sup>10</sup>Exceptions are often due to variances during training.

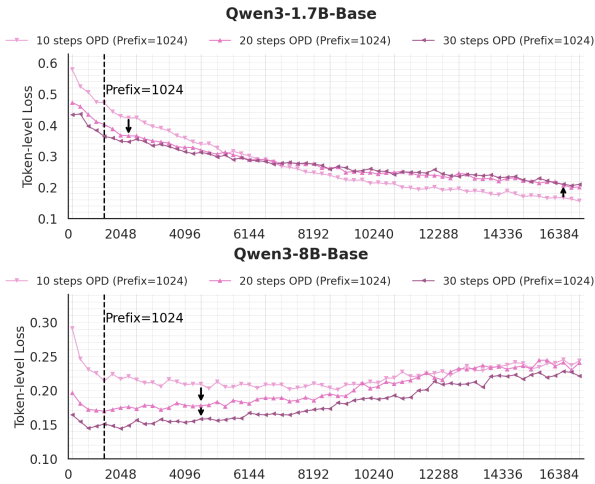


Figure 5: Mean token-level reverse-KL loss by output token position during OPD (prefix 1024) for Qwen3-1.7B-Base (top) and Qwen3-8B-Base (bottom). We plot checkpoints after 10, 20, and 30 gradient steps.

faithfully matching the full reasoning process. We investigate this in Appendix H through an additional reasoning-error analysis. From the analysis, we do not observe increased shortcut-learning errors.

### 5.3 Can Prefix-only Distillation Remain Effective with Off-policy Trajectories and/or Forward KL?

Several recent OPD variants extend the basic formulation by (i) leveraging off-policy trajectories and (ii) considering different divergence directions (forward KL vs reverse KL) (Agarwal et al.; Ko et al., 2024). In these methods, **off-policy data** is typically introduced to reduce the variance of purely on-policy updates and to improve training efficiency by reusing pre-collected trajectories, while **forward KL** is often used to encourage more diverse responses (i.e., less mode-seeking behavior than reverse KL). Since our previous discussion focuses on on-policy and reverse-KL distillation (Eq. 1), we now discuss the effect of prefix distillation on these settings.

**Setup.** We run *prefix-only* Generalized Knowledge Distillation (GKD) using the reference implementation.<sup>11</sup> We train on questions from OPENTHOUGHTS3 (Guha et al., 2026), using Qwen3-1.7B-Base as the student and Qwen3-8B as the teacher. We cap the training rollout length to  $L^{\text{train}}=256$  tokens, use batch size 512, and sample one rollout per prompt. We sweep learning rates in

<sup>11</sup><https://github.com/jongwooko/distillm>

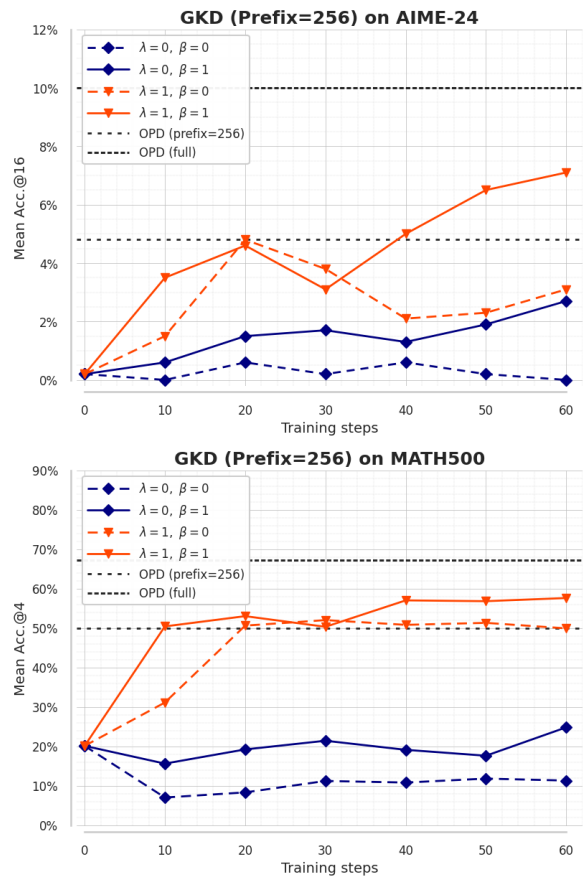


Figure 6: Performances of prefix generalized knowledge distillation. The student is Qwen3-1.7B-Base; Questions and off-policy signals are sampled from OpenThoughts3.  $\lambda=1$  uses on-policy data, and 0 uses off-policy data, while  $\beta=1$  uses reverse KL and 0 uses forward KL.

$\{5 \times 10^{-6}, 2 \times 10^{-5}, 5 \times 10^{-5}\}$  and select  $5 \times 10^{-6}$  as the best-performing setting for GKD under this configuration.

**Finding 4:** Figure 6 shows three consistent trends. (i) **On-policy + reverse KL** is the strongest and most stable setting under short-prefix truncation. (ii) **On-policy + forward KL is still effective:** it provides clear improvements over the base model, but consistently performs worse than on-policy reverse KL with the same prefix budget. (iii) In the **off-policy** setting, **reverse KL yields only mild gains**, whereas **forward KL hurts performance** (often falling below the base model). Similar findings are observed with Qwen3-8B-Base as the student. Please see Appendix C for more details with a larger student size.

**Why can off-policy be challenging under prefix truncation?** A plausible explanation is that truncation amplifies distribution mismatch: with

off-policy training, the student is optimized on prefix states induced by another policy that it may not reliably visit when sampling from its own policy. Since supervision is limited to only the first  $L^{\text{train}}$  tokens, the student must still generate the entire tail from these potentially mismatched states, which can negate the benefit of prefix-level distillation.

**Prefix is memory-efficient.** This is especially appealing for objectives such as ImitKD, GKD and DistiLLM where each supervised token requires a full teacher distribution over the vocabulary. Restricting distillation to a short prefix reduces the memory footprint of these token-level targets, making these objectives more accessible for long-form reasoning.

#### 5.4 Learning Rate and Generalization

Prior work reports that small learning rates can mitigate catastrophic forgetting in SFT and lead to better generalization on out-of-domain data (Pareja et al., 2025; Lin et al., 2026). To further evaluate the generalization ability on out-of-domain benchmarks, we *control* the in-domain performance by selecting checkpoints from different learning rates where accuracies on AIME-24 are similar. In our experiments, we select checkpoints from 10th step where learning rates are  $5e-5$  and  $2e-5$ , the checkpoint from 50th step where the learning rate is  $5e-6$ .<sup>12</sup>

**Finding 5: Reducing the learning rate does not significantly improve the generalization capability of prefix on-policy distillation.** Results are shown in Table 4. We did not observe out-of-domain performance decrease when using a large learning rate, e.g.  $5e-5$ , in comparison to a smaller learning rate of  $2e-5$  or  $5e-6$ .

	AIME-24 (control)	MMLU-Pro	GPQA
step 0	5.2%	26.5%	17.6%
lr= $5e-5$ , step 10	26.3%	<b>55.0%</b>	<b>27.9%</b>
lr= $2e-5$ , step 10	25.0%	49.3%	26.6%
lr= $5e-6$ , step 50	25.2%	47.8%	26.6%

Table 4: Effect of learning rate on generalization capability. We report validation performance from runs of up to 60 steps of OPD (Prefix=1024) on OpenThoughts3, using Qwen3-8B-Base as the student and Qwen3-8B as the teacher.

<sup>12</sup>We did not compare with learning rate  $2e-4$  as the learning rate is too large and dramatically hurt the model’s performance.

## 6 Related Work

**On-policy Distillation.** On-policy learning is a long-standing paradigm in reinforcement learning (Williams, 1992; Rummery and Niranjan, 1994), in which the data used for learning are collected by the same policy that is being optimized. This alignment between data collection and the learned policy reduces distributional mismatch between training and deployment, often leading to more robust behavior. Recent work on on-policy distillation (Lin et al., 2020; Gu et al., 2024; Ko et al., 2024; Agarwal et al.; Lu and Lab, 2025; Ko et al., 2025) adapts this idea to a distillation setting: the student samples its own responses, then a divergence loss aligns its token-level policy with that of a fixed teacher. Nevertheless, existing OPD methods typically require generating full, often long rollouts at each step and calculate divergence over the full vocabulary, which is prohibitively expensive for reasoning-heavy tasks. Our approach preserves the on-policy nature of OPD while drastically reducing training cost by supervising only short prefixes of each on-policy trajectory.

**Selective token supervision.** Prior work suggests that not all tokens in long reasoning traces contribute equally to learning, and several methods improve training efficiency by selecting or reweighting informative tokens (Lin et al., 2024; Huang et al., 2025; Xie et al., 2026; Wang et al., 2025). Most closely related to our motivation, Ji et al. (2025) argues that early segments of chain-of-thought often more self-consistent, and errors predominantly occur in later reasoning steps. Our work studies this idea in a different setting: on-policy distillation for long reasoning, where we show that restricting supervision to an early contiguous prefix preserves most of the gains of full OPD at substantially lower compute.

## 7 Conclusion

We introduced on-policy prefix distillation, a simple modification of on-policy distillation that truncates student rollouts and applies the token-level feedback only to the prefix of each sample. Experiments on several reasoning benchmarks demonstrate that our approach attains substantial gains over the base model and remains competitive with full on-policy distillation, while significantly reducing training cost.

## 8 Limitations

We evaluate our approach exclusively on reasoning tasks assuming the output contains long chain-of-thought reasoning. It remains unknown whether prefix distillation can be more generally adopted in other long-form generation scenarios such as summarization, story generation, etc. Moreover, prefix-only training presumes that the base model can already follow instructions and produce valid output formats; for weaker students, an initial warm-up phase of SFT or full OPD may still be necessary. Finally, like standard OPD, our method requires teacher and student to share the same vocabulary so that token-level losses are well-defined.

**Risks** Since gradients are applied only to the early portion of each student rollout, prefix-only learning may preferentially transfer the teacher’s initial reasoning style while under-training behaviors that typically emerge later in a response (e.g., safety refusals, calibration, or self-correction). As a result, the student may appear well-aligned in its opening tokens but deviate in longer completions. For applications where late-stage safety or calibration is important, we recommend prefix scheduling (progressively increasing the trained prefix length so later tokens are also supervised by the teacher by the end of training) and explicit end-to-end evaluation on safety/calibration benchmarks before deployment.

**AI Assistant within This Work** We used ChatGPT 5.2 to assist with grammar checking, language refinement, and drafting code for plotting figures/diagrams. ChatGPT was not used to generate experimental results, perform data analysis, or introduce new technical claims. All reported numbers and statements were verified by the authors against the underlying experiments and sources.

## 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*.
- Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.
- Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. *SFT memorizes, RL generalizes: A comparative study of foundation model post-training*. In *Forty-second International Conference on Machine Learning*.
- Mengnan Du, Fengxiang He, Na Zou, Dacheng Tao, and Xia Hu. 2023. Shortcut learning of large language models in natural language understanding. *Communications of the ACM*, 67(1):110–120.
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.
- Zorik Gekhman, Gal Yona, Roei Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. 2024. *Does fine-tuning LLMs on new knowledge encourage hallucinations?* In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA. Association for Computational Linguistics.
- Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. *MiniLLM: Knowledge distillation of large language models*. In *The Twelfth International Conference on Learning Representations*.
- Etash Kumar Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Rea Sprague, Ashima Suvarna, Benjamin Feuer, Leon Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 32 others. 2026. *Openthoughts: Data recipes for reasoning models*. In *The Fourteenth International Conference on Learning Representations*.
- Horace He and Thinking Machines Lab. 2025. *Defeating nondeterminism in llm inference*. *Thinking Machines Lab: Connectionism*. <https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/>.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. *Distilling the knowledge in a neural network*. *Preprint*, arXiv:1503.02531.
- Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. 2025. *Selectkd: Selective token-weighted knowledge distillation for llms*. *arXiv preprint arXiv:2510.24021*.
- Ke Ji, Jiahao Xu, Tian Liang, Qiuzhi Liu, Zhiwei He, Xiaoyuan Liu, Xingyu Chen, Junying Chen, Benyou Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025.

- The first few tokens are all you need: An efficient and effective unsupervised prefix fine-tuning method for reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Adam Tauman Kalai, Ofir Nachum, Santosh S Vempala, and Edwin Zhang. 2025. Why language models hallucinate. *arXiv preprint arXiv:2509.04664*.
- Damjan Kalajdzievski. 2024. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*.
- Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1317–1327.
- Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. 2025. **DistiLLM-2: A contrastive approach boosts the distillation of LLMs**. In *Forty-second International Conference on Machine Learning*.
- Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. **DistiLLM: Towards streamlined distillation for large language models**. In *Forty-first International Conference on Machine Learning*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024. **TÜlu 3: Pushing frontiers in open language model post-training**. *CoRR*, abs/2411.15124.
- Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. 2020. **Autoregressive knowledge distillation through imitation learning**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6121–6133, Online. Association for Computational Linguistics.
- Jiacheng Lin, Zhongruo Wang, Kun Qian, Tian Wang, Arvind Srinivasan, Hansi Zeng, Ruochen Jiao, Xie Zhou, Jiri Gesi, Dakuo Wang, Yufan Guo, Kai Zhong, Weiqi Zhang, sujay sanghavi, Changyou Chen, Hyokun Yun, and Lihong Li. 2026. **SFT doesn't always hurt general capabilities: Revisiting domain-specific fine-tuning in LLMs**. In *The Fourteenth International Conference on Learning Representations*.
- Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, yelong shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. 2024. **Not all tokens are what you need for pretraining**. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Kevin Lu and Thinking Machines Lab. 2025. **On-policy distillation**. *Thinking Machines Lab: Connectionism*. <https://thinkingmachines.ai/blog/on-policy-distillation>.
- Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*.
- Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant G Honavar. 2025. **Brains vs. bytes: Evaluating LLM proficiency in olympiad mathematics**. In *Second Conference on Language Modeling*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Aldo Pareja, Nikhil Shivakumar Nayak, Hao Wang, Krishnateja Killamsetty, Shivchander Sudalairaj, Wenlong Zhao, Seungwook Han, Abhishek Bhandwalidar, Guangxuan Xu, Kai Xu, Ligong Han, Luke Inglis, and Akash Srivastava. 2025. **Unveiling the secret recipe: A guide for supervised fine-tuning small LLMs**. In *The Thirteenth International Conference on Learning Representations*.
- Team Qwen. 2024. Qwq: Reflect deeply on the boundaries of the unknown, november 2024. *URL* <https://qwenlm.github.io/blog/qwq-32b-preview>.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jack-son Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Gavin A Rummery and Mahesan Niranjan. 1994. *Online Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. **Hybridflow: A flexible and efficient rlhf framework**. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297, New York, NY, USA. Association for Computing Machinery.

- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. [Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. [Finetuned language models are zero-shot learners](#). In *International Conference on Learning Representations*.
- Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.
- Xurong Xie, Zhucun Xue, Jiafu Wu, Jian Li, Yabiao Wang, Xiaobin Hu, Yong Liu, and Jiangning Zhang. 2026. Llm-oriented token-adaptive knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34070–34078.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

## A A Case Study with Full Details

### Prompt

```
<|im_start|>user
Let $x,y$ and $z$ be positive real numbers that satisfy the following
↪ system of equations:
\[\log_2\left(\frac{x}{yz}\right) = \frac{1}{2}\]
\[\log_2\left(\frac{y}{xz}\right) = \frac{1}{3}\]
\[\log_2\left(\frac{z}{xy}\right) = \frac{1}{4}\]
Then the value of $\frac{m}{n}$ is $\frac{m}{n}$
↪ where $m$ and $n$ are relatively prime positive integers. Find
↪ $\frac{m+n}{6}$.<|im_end|>
<|im_start|>assistant
```

### Qwen3-8B-Base Response (Incorrect)

```
[ImpactedPrompt]From the given log equations, we can express $x,y$ and
↪ $z$ in terms of each other:
$x = 2^{\frac{1}{4}}yz$,
$y = 2^{\frac{1}{5}}xz$,
$z = 2^{\frac{1}{6}}xy$.
Substituting the expressions for $x$ and $z$ into the second equation, we
↪ get:
$y = 2^{\frac{1}{5}} \cdot 2^{\frac{1}{4}}yz \cdot 2^{\frac{1}{6}}xy$,
which simplifies to:
$y = 2^{\frac{5}{20}} yz \cdot 2^{\frac{1}{6}} xy$.
Cancelling $y$ on both sides, we get:
$1 = 2^{\frac{5}{20}} z \cdot 2^{\frac{1}{6}} x$.
Substituting the expression for $x$ into this equation, we get:
$1 = 2^{\frac{5}{20}} \cdot 2^{\frac{1}{6}} yz \cdot 2^{\frac{1}{6}} \cdot 2^{\frac{1}{4}} yz$,
which simplifies to:
$1 = 2^{\frac{3}{5}} yz$.
Substituting the expression for $y$ into this equation, we get:
$1 = 2^{\frac{3}{5}} \cdot 2^{\frac{1}{5}} xz \cdot 2^{\frac{1}{6}} xy$,
which simplifies to:
$1 = 2^{\frac{2}{3}} xz$.
Again, substituting the expression for $x$ into this equation, we get:
$1 = 2^{\frac{2}{3}} \cdot 2^{\frac{1}{5}} yz \cdot 2^{\frac{1}{6}} xy$,
which simplifies to:
$1 = 2^{\frac{1}{5}} yz$.
Using the original equation $x = 2^{\frac{1}{4}}yz$, we can write:
$yz = 2^{\frac{3}{5}}$.
Similarly, using $y = 2^{\frac{1}{5}}xz$, we can write:
$xz = 2^{\frac{9}{10}}$,
and using $z = 2^{\frac{1}{6}}xy$, we can write:
$xy = 2^{\frac{6}{10}}$.
Multiplying these three equations together, we get:
$(xyz)^2 = 2^{\frac{18}{10}} = 2^{\frac{9}{5}}$.
Taking the square root of both sides, we get:
$xyz = 2^{\frac{9}{10}}$.
Finally, we want to find the value of $\frac{m}{n}$.
↪ Since $x,y$ and $z$ are positive, we know that $\log_2(x^4y^3z^2)$ is
↪ positive, so we can write:
$\log_2(x^4y^3z^2) = 4\log_2(x) + 3\log_2(y) + 2\log_2(z)$.
Substituting the expressions for $x,y$ and $z$, we get:
$\log_2(x^4y^3z^2) = 4\log_2(2^{\frac{1}{4}}yz) + 3\log_2(2^{\frac{1}{5}}xz) + 2\log_2(2^{\frac{1}{6}}xy)$.
Using the logarithm property $\log_b(a^c) = c\log_b(a)$, this expression
↪ simplifies to:
$4\left(\frac{1}{4}\log_2(2^{\frac{1}{4}}) + \log_2(y) + \log_2(z)\right) + 3\left(\frac{1}{5}\log_2(2^{\frac{1}{5}}) + \log_2(x) + \log_2(z)\right) + 2\left(\frac{1}{6}\log_2(2^{\frac{1}{6}}) + \log_2(x) + \log_2(y)\right)$,
which simplifies to:
$\frac{4}{4}\log_2(2) + \log_2(yz) + \frac{3}{5}\log_2(xz) + \frac{2}{6}\log_2(xy)$.
Substituting the expressions for $yz,xz$ and $xy$, this becomes:
$\frac{4}{4} + \frac{3}{5} + \frac{2}{6} = \frac{10}{10} + \frac{6}{10} + \frac{4}{10} = \frac{20}{10} = 2$.
Combining the fractions, we get:
$\frac{4}{4} + \frac{3}{5} + \frac{2}{6} = 2$.
Therefore, the value of $\frac{m}{n}$ is $\frac{2}{1}$.
↪ $\frac{m+n}{6} = \frac{2+1}{6} = \frac{3}{6} = \frac{1}{2}$.

Therefore, the final answer is $3 + 2 = \boxed{5}$. [ImpactedPrompt]
```

### Prompt with Teacher Prefix (128 tokens from Qwen3-8B)

```
<|im_start|>user
Let $x,y$ and $z$ be positive real numbers that satisfy the following
↪ system of equations:
\[\log_2\left(\frac{x}{yz}\right) = \frac{1}{2}\]
\[\log_2\left(\frac{y}{xz}\right) = \frac{1}{3}\]
\[\log_2\left(\frac{z}{xy}\right) = \frac{1}{4}\]
Then the value of $\frac{m}{n}$ is $\frac{m}{n}$
↪ where $m$ and $n$ are relatively prime positive integers. Find
↪ $\frac{m+n}{6}$.<|im_end|>
<|im_start|>assistant
<think>
Okay, so I need to solve this system of logarithmic equations and find
↪ the value of $\log_2(x^4y^3z^2)$. Let me start by writing down the
↪ given equations:

1. $\log_2(x/(yz)) = 1/2$
2. $\log_2(y/(xz)) = 1/3$
3. $\log_2(z/(xy)) = 1/4$
```

```
Hmm, logarithms can sometimes be tricky, but maybe I can convert them
↪ into exponential form to make it easier. Let me recall that $\log_b(a)
↪ = c$ means that $b^c = a$. So applying that
```

## Qwen3-8B-Base Response with Teacher Prefix (Correct)

```
to each equation:
1. $2^{1/2} = x/(yz)$
2. $2^{1/3} = y/(xz)$
3. $2^{1/4} = z/(xy)$
```

```
Okay, now I have three equations with variables $x, y$, and $z$. Maybe I can
↪ express everything in terms of one variable. Let's see:
From equation 1: $2^{1/2} = x/(yz)$ So, $x = yz \cdot 2^{1/2}$
From equation 2: $2^{1/3} = y/(xz)$ So, $y = xz \cdot 2^{1/3}$
From equation 3: $2^{1/4} = z/(xy)$ So, $z = xy \cdot 2^{1/4}$
```

```
Now, I have expressions for $x, y$, and $z$ in terms of the other variables.
↪ Maybe I can substitute one into another. Let's try substituting $x$
↪ from equation 1 into equation 2:
$y = (yz \cdot 2^{1/2}) \cdot z \cdot 2^{1/3}$ Wait, that seems a bit messy. Maybe
↪ there's a better way. Let's see:
I have:
$x = yz \cdot 2^{1/2}$
$y = xz \cdot 2^{1/3}$
$z = xy \cdot 2^{1/4}$
```

```
Perhaps I can multiply all three equations together. Let's try:
$x \cdot y \cdot z = (yz \cdot 2^{1/2}) \cdot (xz \cdot 2^{1/3}) \cdot (xy \cdot 2^{1/4})$
Simplify the right side:
$= yz \cdot 2^{1/2} \cdot xz \cdot 2^{1/3} \cdot xy \cdot 2^{1/4}$
$= x^2y^2z^2 \cdot 2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4}$
```

```
Now, $x \cdot y \cdot z = x^2y^2z^2 \cdot 2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4}$
Wait, that seems off. Let's double-check:
Actually, when I multiplied the right side, I should have:
$= (yz \cdot 2^{1/2}) \cdot (xz \cdot 2^{1/3}) \cdot (xy \cdot 2^{1/4}) = yz \cdot 2^{1/2} \cdot xz \cdot 2^{1/3} \cdot xy \cdot 2^{1/4}$
↪ $2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4} = 2^{13/12}$
But $x, y, z$ are on both sides, so perhaps I can cancel them out. Let's
↪ see:
Divide both sides by $x \cdot y \cdot z$:
$1 = 2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4}$
```

```
Now, let's simplify the right side:
$2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4} = 2^{(1/2) + (1/3) + (1/4)}$
Find a common denominator for the exponents: 2, 3, and 4 have a least
↪ common multiple of 12.
So, $(1/2) = 6/12, (1/3) = 4/12, (1/4) = 3/12$
Therefore, $(1/2) + (1/3) + (1/4) = 6/12 + 4/12 + 3/12 = 13/12$
So, $2^{(1/2) + (1/3) + (1/4)} = 2^{13/12}$
Thus, we have:
$1 = 2^{13/12}$
But $2^{13/12}$ is not equal to 1. This suggests that there might be an
↪ error in my calculations.
Let me revisit the multiplication step. Maybe I made a mistake there.
Original multiplied equation:
$x \cdot y \cdot z = (yz \cdot 2^{1/2}) \cdot (xz \cdot 2^{1/3}) \cdot (xy \cdot 2^{1/4})$
Let's simplify the right side correctly:
$(yz \cdot 2^{1/2}) \cdot (xz \cdot 2^{1/3}) \cdot (xy \cdot 2^{1/4}) = yz \cdot 2^{1/2} \cdot xz \cdot 2^{1/3} \cdot xy \cdot 2^{1/4}$
↪ $2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4} = 2^{13/12}$
Now, group like terms:
$= (y \cdot z \cdot x \cdot z \cdot x \cdot y) \cdot (2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4})$
$= (x^2y^2z^2) \cdot (2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4})$
Now, divide both sides by $x \cdot y \cdot z$:
$1 = (x^2y^2z^2) / (x \cdot y \cdot z) \cdot (2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4})$
$= x \cdot y \cdot z \cdot (2^{1/2} \cdot 2^{1/3} \cdot 2^{1/4})$
So, $1 = x \cdot y \cdot z \cdot 2^{13/12}$
Now, recall that $x \cdot y \cdot z = (x \cdot y \cdot z) \cdot 2^{13/12}$
This implies:
$(x \cdot y \cdot z) \cdot 2^{13/12} = 1$
So, $x \cdot y \cdot z = 1 / 2^{13/12}$
```

```
Now, let's go back to the original expressions:
$x = yz \cdot 2^{1/2}$
$y = xz \cdot 2^{1/3}$
$z = xy \cdot 2^{1/4}$
```

```
Let's express everything in terms of $z$.
From $x = yz \cdot 2^{1/2}$, we can express $y$ as:
$y = x / (z \cdot 2^{1/2})$
From $y = xz \cdot 2^{1/3}$, substitute $y$:
$x / (z \cdot 2^{1/2}) = xz \cdot 2^{1/3}$
Multiply both sides by $z \cdot 2^{1/2}$:
$x = xz^2 \cdot 2^{1/3} \cdot 2^{1/2}$
Divide both sides by $x$ (since $x \neq 0$):
$1 = z^2 \cdot 2^{1/3} \cdot 2^{1/2}$
So, $z^2 = 1 / (2^{1/3} \cdot 2^{1/2})$
```

```
Now, recall that $x \cdot y \cdot z = 1 / 2^{13/12}$
We have $z$ in terms of $x$ and $y$, but it's getting complicated. Maybe
↪ there's a better approach.
Let me consider the original equations again:
1. $\log_2(x/(yz)) = 1/2 \Rightarrow x/(yz) = 2^{1/2} \Rightarrow x = yz \cdot 2^{1/2}$
2. $\log_2(y/(xz)) = 1/3 \Rightarrow y/(xz) = 2^{1/3} \Rightarrow y = xz \cdot 2^{1/3}$
3. $\log_2(z/(xy)) = 1/4 \Rightarrow z/(xy) = 2^{1/4} \Rightarrow z = xy \cdot 2^{1/4}$
Let's multiply all three equations:
$(x = yz \cdot 2^{1/2}) \cdot (y = xz \cdot 2^{1/3}) \cdot (z = xy \cdot 2^{1/4})$
Simplify the right side:
$yz \cdot 2^{1/2} \cdot xz \cdot 2^{1/3} \cdot xy \cdot 2^{1/4} = x^2y^2z^2 \cdot 2^{(1/2) + (1/3) + (1/4)}$
↪ $2^{13/12}$
So, $x \cdot y \cdot z = x^2y^2z^2 \cdot 2^{13/12}$
Divide both sides by $x \cdot y \cdot z$:
$1 = x \cdot y \cdot z \cdot 2^{13/12}$
Thus, $x \cdot y \cdot z = 1 / 2^{13/12}$
```

```
Now, let's find $x \cdot y \cdot z$ in another way. From the first equation:
```

$x = yz * 2^{(1/2)}$   
 Multiply both sides by  $yz$ :  
 $x * yz = yz * 2^{(1/2)} * yz \Rightarrow x * yz = y^2 z^2 * 2^{(1/2)} \Rightarrow x = y^2 z^2 * 2^{(1/2)}$   
 $\hookrightarrow 2^{(1/2)} / yz \Rightarrow x = yz * 2^{(1/2)}$   
 Wait, that brings us back to the original equation. Maybe I should use the expressions differently.  
 Let me try to express  $x, y, z$  in terms of one variable. Let's assume  $z = k$ , where  $k$  is a positive real number.  
 Then, from the third equation:  
 $z = xy * 2^{(1/4)} \Rightarrow k = xy * 2^{(1/4)} \Rightarrow xy = k / 2^{(1/4)}$   
 From the second equation:  
 $y = xz * 2^{(1/3)} \Rightarrow y = xk * 2^{(1/3)}$   
 Now, substitute  $y$  into the expression for  $xy$ :  
 $x * (xk * 2^{(1/3)}) = k / 2^{(1/4)} \Rightarrow x^2 k * 2^{(1/3)} = k / 2^{(1/4)}$   
 Divide both sides by  $k$  (since  $k \neq 0$ ):  
 $x^2 * 2^{(1/3)} = 1 / 2^{(1/4)}$   
 Multiply both sides by  $2^{(1/4)}$ :  
 $x^2 * 2^{(1/3)} * 2^{(1/4)} = 1 \Rightarrow x^2 * 2^{((1/3) + (1/4))} = 1$   
 $(1/3) + (1/4) = 4/12 + 3/12 = 7/12$   
 So,  $x^2 * 2^{(7/12)} = 1 \Rightarrow x^2 = 1 / 2^{(7/12)} \Rightarrow x = 1 / 2^{(7/24)}$   
 Now, find  $y$  using  $y = xk * 2^{(1/3)}$ :  
 $y = (1 / 2^{(7/24)}) * k * 2^{(1/3)} = k * 2^{((1/3) - (7/24))} = k * 2^{((8/24) - (7/24))} = k * 2^{(1/24)}$   
 $\hookrightarrow - (7/24) = k * 2^{(1/24)}$   
 Now, find  $x * y * z$ :  
 $x * y * z = (1 / 2^{(7/24)}) * (k * 2^{(1/24)}) * k = k^2 * 2^{((1/24) - (7/24))} = k^2 * 2^{(-6/24)} = k^2 * 2^{(-1/4)}$   
 $\hookrightarrow (7/24) = k^2 * 2^{(-1/4)} = 1 / 2^{(13/12)}$   
 But earlier, we found that  $x * y * z = 1 / 2^{(13/12)}$   
 So,  $k^2 * 2^{(-1/4)} = 1 / 2^{(13/12)}$   
 Divide both sides by  $2^{(-1/4)}$ :  
 $k^2 = 1 / (2^{(13/12)} * 2^{(1/4)}) = 1 / 2^{((13/12) + (1/4))} = 1 / 2^{((13/12) + (3/12))} = 1 / 2^{(16/12)} = 1 / 2^{(4/3)}$   
 Thus,  $k^2 = 1 / 2^{(4/3)} \Rightarrow k = 1 / 2^{(2/3)}$   
 Now, recall that  $z = k = 1 / 2^{(2/3)}$   
 We can find  $x$  and  $y$  as well:  
 $x = 1 / 2^{(7/24)}$   
 $y = k * 2^{(1/24)} = (1 / 2^{(2/3)}) * 2^{(1/24)} = 2^{((1/24) - (16/24))} = 2^{(-15/24)} = 2^{(-5/8)}$   
 $\hookrightarrow 2^{((1/24) - (10/24))} = 2^{(-9/24)} = 2^{(-3/8)}$   
 Now, we need to find  $|\log_2(x^4 y^3 z^2)|$   
 First, compute  $x^4 y^3 z^2$ :  
 $x^4 = (1 / 2^{(7/24)})^4 = 1 / 2^{(28/24)} = 1 / 2^{(7/6)}$   
 $y^3 = (2^{(-5/8)})^3 = 2^{(-15/8)}$   
 $z^2 = (1 / 2^{(2/3)})^2 = 1 / 2^{(4/3)} = 1 / 2^{(5/6)}$   
 So,  $x^4 y^3 z^2 = (1 / 2^{(7/6)}) * 2^{(-15/8)} * (1 / 2^{(5/6)}) = 1 / (2^{(7/6)} * 2^{(15/8)} * 2^{(5/6)})$   
 $\hookrightarrow 2^{(5/6)} * 2^{(9/8)} = 1 / (2^{((7/6) + (5/6) + (15/8))})$   
 Find a common denominator for the exponents: 6 and 8 have a least common multiple of 24.  
 So,  $(7/6) = 28/24, (5/6) = 20/24, (9/8) = 27/24$   
 Thus,  $(7/6) + (5/6) + (9/8) = (28/24) + (20/24) + (27/24) = 75/24$   
 So,  $x^4 y^3 z^2 = 1 / 2^{(75/24)}$   
 Now, compute  $|\log_2(x^4 y^3 z^2)|$ :  
 $|\log_2(1 / 2^{(75/24)})| = | - (75/24) | = 75/24$   
 Simplify 75/24:  
 Divide numerator and denominator by 3: 25/8  
 So,  $|\log_2(x^4 y^3 z^2)| = 25/8$   
 Thus,  $m = 25, n = 8, \text{ and } m + n = 33$   
 $\boxed{33}$

## Qwen3-8B-Base after Prefix On-policy Distillation (Correct)

Okay, so I need to solve this system of logarithmic equations and find the value of  $|\log_2(x^4 y^3 z^2)|$ . The problem gives me three equations involving logs with base 2 of ratios of  $x, y, \text{ and } z$ . Let me write them down again to make sure I have them right:
 

- $\log_2(x/(yz)) = 1/2$
- $\log_2(y/(xz)) = 1/3$
- $\log_2(z/(xy)) = 1/4$

 And then I need to find  $|\log_2(x^4 y^3 z^2)|$ , which simplifies to  $|4 \log_2 x + 3 \log_2 y + 2 \log_2 z|$ . But maybe there's a better way to approach this. Let me think.
   
 First, I remember that  $\log_b(a/c)$  is equal to  $\log_b(a) - \log_b(c)$ . Maybe I can rewrite each equation using that property. Let me try that.
   
 Starting with the first equation:  
 $\log_2(x/(yz)) = \log_2 x - \log_2(yz) = \log_2 x - \log_2 y - \log_2 z = 1/2$ .
   
 Similarly for the second equation:  
 $\log_2(y/(xz)) = \log_2 y - \log_2(xz) = \log_2 y - \log_2 x - \log_2 z = 1/3$ .
   
 Third equation:  
 $\log_2(z/(xy)) = \log_2 z - \log_2(xy) = \log_2 z - \log_2 x - \log_2 y = 1/4$ .
   
 So now I have three equations:
 

- $\log_2 x - \log_2 y - \log_2 z = 1/2$
- $-\log_2 x + \log_2 y - \log_2 z = 1/3$
- $-\log_2 x - \log_2 y + \log_2 z = 1/4$

 Hmm, this looks like a system of linear equations in terms of  $\log_2 x, \log_2 y, \text{ and } \log_2 z$ . Let me denote them as variables for simplicity.
   
 $\hookrightarrow$  Let me let:
   
 $a = \log_2 x,$   
 $b = \log_2 y,$   
 $c = \log_2 z.$ 
  
 Then the equations become:
 

- $a - b - c = 1/2$
- $-a + b - c = 1/3$
- $-a - b + c = 1/4$

So now I have three equations with three variables. Let me write them again:
   
 Equation 1:  $a - b - c = 1/2$   
 Equation 2:  $-a + b - c = 1/3$   
 Equation 3:  $-a - b + c = 1/4$

I need to solve for  $a, b, c$ . Let me try to solve this system step by step.
   
 First, maybe I can add equations together to eliminate variables. Let me see.
   
 Let me try adding equations 1 and 2:  
 Equation 1 + Equation 2:  
 $(a - b - c) + (-a + b - c) = 1/2 + 1/3$   
 Simplify left side:  $a - a - b + b - c - c = -2c$   
 Right side:  $1/2 + 1/3 = 3/6 + 2/6 = 5/6$   
 So  $-2c = 5/6 \Rightarrow c = -5/12$ .
   
 Okay, so  $c = -5/12$ .
   
 Now let me use equation 1 and equation 3 to find another variable.
   
 Wait, equation 1:  $a - b - c = 1/2$   
 Equation 3:  $-a - b + c = 1/4$

Let me add equations 1 and 3:  
 $(a - b - c) + (-a - b + c) = 1/2 + 1/4$   
 Left side simplifies:  $a - a - b - b - c + c = -2b$   
 Right side:  $1/2 + 1/4 = 3/4$   
 So  $-2b = 3/4 \Rightarrow b = -3/8$ .
   
 Now, since we have  $c$  and  $b$ , we can plug them back into one of the equations to solve for  $a$ .
   
 $\hookrightarrow$  equations to solve for  $a$ .
   
 Let me use equation 1:  
 $a - b - c = 1/2$   
 So  $a - (-3/8) - (-5/12) = 1/2$   
 Calculate:  
 $a + 3/8 + 5/12 = 1/2$

Convert to common denominators. Let me find the least common denominator for 8, 12, and 1. Which is 24.
   
 So:  
 $a + (9/24) + (10/24) = 12/24$   
 So  $a + 19/24 = 12/24$   
 Thus,  $a = 12/24 - 19/24 = -7/24$ .
   
 So  $a = -7/24$ .
   
 So now we have:  
 $a = \log_2 x = -7/24$   
 $b = \log_2 y = -3/8$   
 $c = \log_2 z = -5/12$

Now, we need to compute  $|\log_2(x^4 y^3 z^2)|$ . Let me express that in terms of  $a, b, c$ .  
 $\log_2(x^4 y^3 z^2) = 4 \log_2 x + 3 \log_2 y + 2 \log_2 z = 4a + 3b + 2c$ .
   
 So let me compute that:  
 $4a + 3b + 2c = 4*(-7/24) + 3*(-3/8) + 2*(-5/12)$

Let me compute each term step by step.
   
 First term:  $4*(-7/24) = (-28)/24 = -7/6$   
 Second term:  $3*(-3/8) = -9/8$   
 Third term:  $2*(-5/12) = -10/12 = -5/6$

Now adding them together:  $-7/6 - 9/8 - 5/6$ .
   
 Let me convert all terms to sixteenths to add them. Wait, maybe better to find common denominator.
   
 $\hookrightarrow$  find common denominator.
   
 The denominators are 6, 8, 6. The least common denominator is 24.
   
 Convert each term:  
 $-7/6 = -28/24$   
 $-9/8 = -27/24$   
 $-5/6 = -20/24$

So total is:  $-28/24 - 27/24 - 20/24 = (-28 - 27 - 20)/24 = (-75)/24$ .  
 Simplify  $-75/24$ . Divide numerator and denominator by 3:  $-25/8$ .
   
 Therefore,  $\log_2(x^4 y^3 z^2) = -25/8$ .
   
 But the problem asks for  $|\log_2(x^4 y^3 z^2)|$ , which is  $|-25/8| = 25/8$ .
   
 $\hookrightarrow$  Therefore,  $m/n$  is  $25/8$ , and  $m + n$  is  $25 + 8 = 33$ .
   
 Wait, but let me check my calculations again because sometimes I make arithmetic errors.
   
 First, let me check the values of  $a, b, c$  again.
   
 From earlier:  
 $c = -5/12, b = -3/8, a = -7/24$ . Let me check if they satisfy equation 2  
 $\hookrightarrow$  and equation 3.
   
 Let me check equation 2:  $-a + b - c = 1/3$ .  
 Plug in  $a = -7/24, b = -3/8, c = -5/12$ .  
 Compute  $-a + b - c = 7/24 - 3/8 - (-5/12) = 7/24 - 3/8 + 5/12$ .  
 So sum:  $7/24 - 3/8 + 5/12$ .  
 Convert all to 24 denominator:  
 $7/24 - 9/24 + 10/24 = (7 - 9 + 10)/24 = 8/24 = 1/3$ . Correct!
   
 Equation 3:  $-a - b + c = 1/4$ .  
 Plug in:  
 $-a - b + c = 7/24 - (-3/8) - 5/12 = 7/24 + 3/8 - 5/12$ .  
 Convert to 24 denominator:

$7/24 + 9/24 - 10/24 = (7 + 9 - 10)/24 = 6/24 = 1/4$ . Correct!

Okay, so the values of  $a$ ,  $b$ ,  $c$  are correct.

Now, compute  $4a + 3b + 2c$ :  
 $4*(-7/24) + 3*(-3/8) + 2*(-5/12) = -28/24 - 9/8 - 5/6$ .  
Convert all to 24 denominator:  
 $-28/24 - 27/24 - 20/24 = (-28 - 27 - 20)/24 = -75/24$ . Simplify: divide  
↪ numerator and denominator by 3:  $-25/8$ .  
So  $\log_2(x^4y^3z^2) = -25/8$ . Then absolute value is  $25/8$ . So  $m/n$  is  $25/8$ ,  
↪ which is already in simplest terms since 25 and 8 are coprime.  
↪ Therefore,  $m + n = 25 + 8 = 33$ .

But let me check again the calculation steps.

First,  $4a$ :  $4*(-7/24) = -7/6$ . Correct.  
Then  $3b$ :  $3*(-3/8) = -9/8$ . Correct.  
Then  $2c$ :  $2*(-5/12) = -10/12 = -5/6$ . Correct.

Adding them:  
 $-7/6 - 9/8 - 5/6$ .  
Convert to 24ths:  
 $-7/6 = -28/24$ ,  $-9/8 = -27/24$ ,  $-5/6 = -20/24$ . Adding gives  $-75/24$ .  
Yes, which is  $-25/8$ . So absolute value is  $25/8$ .

So the answer is  $25/8$ ,  $m + n$  is 33. Wait, but let me check if I did the  
↪ problem correctly. Wait, the problem says to compute the absolute  
↪ value. So the value is negative, but the absolute value is positive.

But maybe I made a mistake in the sign? Wait, let's check the original  
↪ problem.  
The question is  $|\log_2(x^4y^3z^2)|$ . So if  $\log_2(x^4y^3z^2)$  is negative,  
↪ then the absolute value is positive. So  $25/8$ .

Alternatively, maybe my calculation of the coefficients is wrong. Let me  
↪ check again:  
The expression to compute is  $\log_2(x^4y^3z^2)$ . Which is  $\log_2(x^4) +$   
↪  $\log_2(y^3) + \log_2(z^2) = 4 \log_2 x + 3 \log_2 y + 2 \log_2 z$ . Yes,  
↪ that is correct.  
So  $4a + 3b + 2c$  is  $-25/8$ , so absolute value is  $25/8$ .

Wait, but let me check if there's a miscalculation in the addition of  
↪ fractions.

Let me do it again step by step:  
First term:  $4a = 4*(-7/24) = -28/24$ . Simplify to  $-7/6$ .  
Second term:  $3b = 3*(-3/8) = -9/8$ .  
Third term:  $2c = 2*(-5/12) = -10/12 = -5/6$ .  
So sum is  $(-7/6) + (-9/8) + (-5/6)$ .  
Combine the first and third terms:  
 $(-7/6 - 5/6) = -12/6 = -2$ . Then add  $-9/8$ .  
So  $-2 - 9/8 = -16/8 - 9/8 = -25/8$ . Correct.  
Therefore, that is correct. So the absolute value is  $25/8$ .

Therefore, the answer is  $25/8$ , so  $m = 25$ ,  $n = 8$ ,  $m + n = 33$ .

But let me check if the problem says positive real numbers. Since  $x$ ,  $y$ ,  $z$   
↪ are positive, all the logs are defined. So that's okay.

Alternatively, maybe there's a mistake in my equations setup. Let me  
↪ check the original equations again.  
Original equations:  
1.  $\log_2(x/yz) = 1/2$   
Which is  $\log_2 x - \log_2 y - \log_2 z = 1/2$ . Correct.  
Similarly for others. So equations are correct.  
Therefore, the answer should be  $25/8$ .

Wait, but the problem says "the value of  $|\log_2(x^4y^3z^2)|$  is  $m/n$ ". So my  
↪ conclusion is that  $m/n$  is  $25/8$ , so  $m + n$  is 33. But let me check once  
↪ more with another approach. Maybe I can compute  $x$ ,  $y$ ,  $z$  directly?  
Wait, but that might take longer. Let me try.

Let me express  $x$ ,  $y$ ,  $z$  in terms of exponents. Let me recall that if  $\log_2$   
↪  $x = a$ , then  $x = 2^a$ , similarly for  $y$  and  $z$ . So:  
 $x = 2^{(-7/24)}$ ,  $y = 2^{(-3/8)}$ ,  $z = 2^{(-5/12)}$   
Then  $x^4y^3z^2 = 2^{(4a + 3b + 2c)}$   
But we already found that  $4a + 3b + 2c = -25/8$ . Therefore,  
↪  $\log_2(x^4y^3z^2) = -25/8$ , so absolute value is  $25/8$ .  
Therefore, the answer is  $25/8$ .

Alternatively, maybe I can compute  $x^4y^3z^2$ ? Let me check:  
 $x^4 = (2^{(-7/24)})^4 = 2^{(-28/24)} = 2^{(-7/6)}$   
 $y^3 = (2^{(-3/8)})^3 = 2^{(-9/8)}$   
 $z^2 = (2^{(-5/12)})^2 = 2^{(-10/12)} = 2^{(-5/6)}$   
Multiply them together:  
 $2^{(-7/6)} * 2^{(-9/8)} * 2^{(-5/6)} = 2^{(-7/6 - 9/8 - 5/6)}$   
Compute the exponents:  
Convert to common denominator 24:  
 $-7/6 = -28/24$   
 $-9/8 = -27/24$   
 $-5/6 = -20/24$   
Total exponent:  $-28 - 27 - 20 = -75/24 = -25/8$ . So  $\log_2(x^4y^3z^2) =$   
↪  $-25/8$ . Therefore, absolute value is  $25/8$ .  
Therefore, I think my answer is correct. So  $25 + 8 = 33$ . Therefore, the  
↪ answer is 33.

Wait, but let me check once more for possible errors in the equations.

↪ Let me verify equations with the values of  $a$ ,  $b$ ,  $c$ .  
Equation 1:  $a - b - c = -7/24 - (-3/8) - (-5/12) = -7/24 + 3/8 + 5/12$ .  
Convert to 24 denominator:  
 $-7/24 + 9/24 + 10/24 = (-7 + 9 + 10)/24 = 12/24 = 1/2$ . Correct.  
Equation 2:  $-a + b - c = 7/24 - 3/8 - (-5/12) = 7/24 - 3/8 + 5/12$ .  
Convert:  
 $7/24 - 9/24 + 10/24 = (7 - 9 + 10)/24 = 8/24 = 1/3$ . Correct.

Equation 3:  $-a - b + c = 7/24 + 3/8 - 5/12 = 7/24 + 9/24 - 10/24 = (7 + 9$

↪  $- 10)/24 = 6/24 = 1/4$ . Correct.

Therefore, equations are satisfied. Therefore, the calculations are

↪ correct.

Therefore, the answer is  $25/8$ , so  $m + n$  is 33.

\*\*Final Answer\*\*  $\boxed{33}$

## B FLOP Estimation Details

We report training compute using an estimated GPU FLOP count to facilitate *relative* cost comparisons across methods. This estimate is not meant to be an exact measurement of hardware-executed FLOP, since realized cost depends on implementation details (e.g., kernels, padding, KV cache, and parallelism). We therefore use a single, consistent accounting scheme for all experiments.

**Training FLOP from VERL.** We use the FLOP counter in VERL (Sheng et al., 2025) for Qwen3 models to estimate the compute of one student training update (including forward + backward + optimizer step) on a sequence of length  $L$ . We denote it by  $F_{\text{train}}(M, L)$  for model  $M$ .<sup>13</sup>

**Forward-only approximation.** OPD additionally requires forward-only computation (sampling and log-prob evaluation). We approximate the forward-only FLOP as  $F_{\text{fwd}}(M, L) \approx \frac{1}{3} F_{\text{train}}(M, L)$ , following the common rule-of-thumb that training cost is roughly  $\approx 3\times$  the forward cost. This approximation is applied uniformly across methods.

**Per-rollout estimate for OPD.** For each student rollout of supervised length  $L$ , we estimate total compute as  $F_{\text{OPD}}(L) \approx F_{\text{train}}(S, L) + 2F_{\text{fwd}}(S, L) + F_{\text{fwd}}(T, L)$ , where  $S$  and  $T$  denote the student and teacher, respectively. Intuitively, this sums one student update plus forward passes for (i) sampling, (ii) student log-probs, and (iii) teacher log-probs. For prefix OPD,  $L$  is capped by the training prefix budget  $L^{\text{train}}$ .

**Total FLOP.** If each optimization step uses batch size  $B$  prompts and  $K$  sampled rollouts per prompt, then  $F_{\text{step}} \approx B \cdot K \cdot F_{\text{OPD}}(L)$ , and total training FLOP are obtained by summing  $F_{\text{step}}$  over steps (evaluation excluded).

**Notes.** This estimate does not capture effects such as KV-cache reuse, communication overhead,

<sup>13</sup>We slightly modified the reference code to return cumulative FLOP (rather than a rate) by removing the division by elapsed time: [https://github.com/volengine/verl/blob/c936ec7d5cebcc1d40f50296d28699696fbad6f5/verl/Utils/FLOP\\_counter.py#L130](https://github.com/volengine/verl/blob/c936ec7d5cebcc1d40f50296d28699696fbad6f5/verl/Utils/FLOP_counter.py#L130).

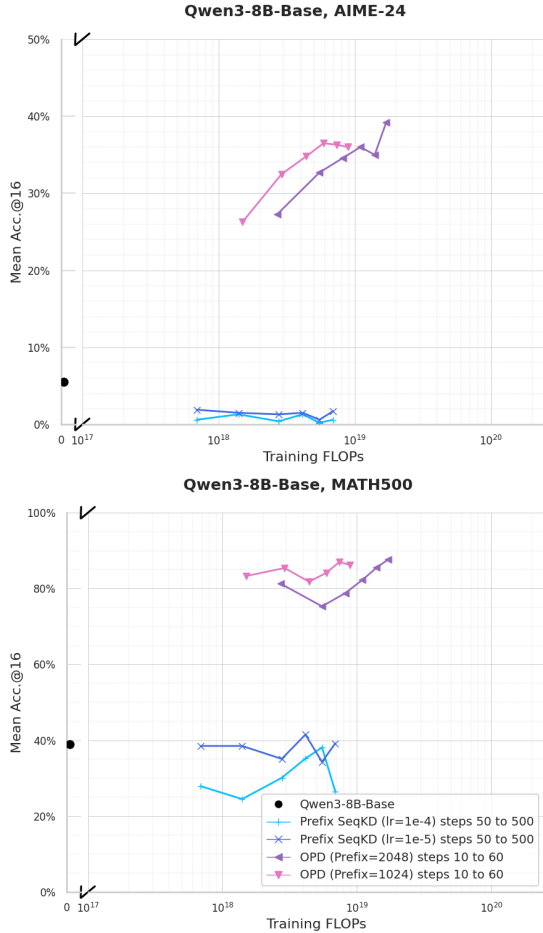


Figure 7: Performances of off-policy prefix distillation. The student is Qwen3-8B-Base; Off-policy prefix signals are sampled from OpenThoughts3. **Top**: AIME-24 **Bottom**: MATH500

CPU or I/O costs. We therefore treat FLOP as a consistent proxy for comparing training cost within our experimental setup.

### C Prefix SeqKD with Qwen3-8B-Base

In this section, we would like to explore whether prefix distillation can be applied to SeqKD (off-policy + forward KL) with a larger student size. Figure 7 shows some preliminary results. We implement off-policy prefix distillation by setting the maximum context length of SeqKD to 2048, and evaluate the results on AIME-24 and MATH500. We also plot results from OPD with prefix 2048 into the same diagram for easier comparison.

From the result, prefix SeqKD with large learning rate quickly harms the student model’s performance, while a smaller learning rate only slows down the performance decrease. When further looking into sampled outputs of student model with prefix SeqKD, we found that the beginning of out-

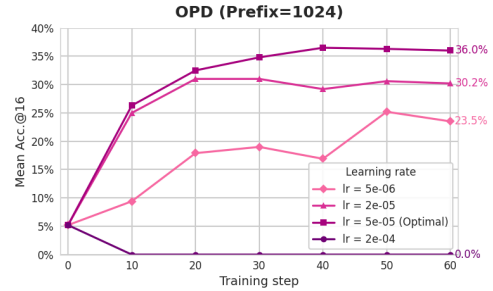


Figure 8: The effect of learning rate on AIME-24 during the training of on-policy prefix distillation on OpenThoughts3. The student is Qwen3-8B-Base and the teacher is Qwen3-8B.

Max response length	Prefix=256	2048
16,384 tokens (default)	26.3	31.3
2,048 tokens (ablation)	9.2	4.6

Table 5: AIME-25 length-constrained decoding ablation with Qwen3-8B-Base as student and Qwen3-8B as the teacher. Numbers are Mean Acc.@16 (%). Short decoding budgets at test time substantially reduce performance.

puts are readable and logically coherent, while the tail often contains self-repeating and even unreadable tokens.

### D Learning Rate Sweep

We conducted a learning rate sweep for prefix OPD. As shown in Figure 8, performance differs significantly with different learning rates. Consistent with Lu and Lab (2025), 5e-5 turns out to be the optimal learning rate with batchsize 512 in our experiment setting. We leave extensive analysis on a grid search of learning rate, learning rate scheduling, step size, batch size and prefix length for future work.

### E Ablation with Output Length Constraints

To justify that the effectiveness of prefix distillation is not due to short output lengths, we conduct an ablation study where the inference is artificially constrained to short outputs. As shown in Table 5, limiting the maximum response length to 2,048 tokens causes a large accuracy drop for both prefix settings, relative to the standard decoding budget of 16,384 tokens. This confirms that the reported performance gains rely on long reasoning generation at inference time, rather than only on producing short final answers.

## F Early Stop with Dev

Table 6 and Table 7 contain the detailed information of which checkpoints we used for testing. We use AIME-24 as the development set and then test on the rest of the benchmarks.

Prefix length	Checkpoint for testing
256	Step 20
512	Step 60
1024	Step 60
2048	Step 40
Prefix scheduling	Step 60
Full	Step 50

Table 6: Checkpoints for testing with Qwen3-1.7B-Base as the student.

Prefix length	Checkpoint for testing
256	Step 60
512	Step 50
1024	Step 40
2048	Step 60
Prefix scheduling	Step 60
Full	Step 60

Table 7: Checkpoints for testing with Qwen3-8B-Base as the student.

## G Number of Solutions for AIME-2024

See Table 8.

Problem ID	Number of Solutions
1	2
2	6
3	5
4	2
5	7
6	7
7	12
8	5
9	2
10	8
11	2
12	3
13	4
14	10
15	6
Average	5.4

Table 8: Number of solutions for each question in AIME-2024 from AoPS forum [https://artofproblemsolving.com/wiki/index.php?title=2024\\_AIME\\_I](https://artofproblemsolving.com/wiki/index.php?title=2024_AIME_I)

## H Shortcut Learning Analysis

One concern with prefix-only supervision is *shortcut learning*: the student may match shallow stylistic or local reasoning patterns in the teacher-generated prefix without faithfully reproducing the

Error type	OPD	OPD (prefix 2048)
Total responses	161	150
Correct	107	107
Proof by example	2	0
Proposal without verification	20	10
Inventing wrong facts	0	0
Begging the question	1	0
Solution by trial-and-error	5	6
Calculation mistakes	25	23
Other errors	30	22

Table 9: Reasoning-error analysis on AIME-25 for Qwen3-8B-Base. We compare correct-answer responses from models trained with full-trajectory OPD and prefix OPD ( $L=2048$ ), and annotate them using the error taxonomy of Mahdavi et al. (2025).

full reasoning process. While this concern is especially relevant for open-ended generation tasks such as summarization and story generation, it can also arise in reasoning tasks, where a model may arrive at the correct final answer despite flawed intermediate reasoning. Note that each response can have zero, one or multiple reasoning errors.

To examine this issue, we compare reasoning traces from Qwen3-8B-Base trained with full-trajectory OPD and prefix OPD ( $L=2048$ ) on AIME-25. We collect responses with correct final answers and use GPT-5.2 (which shows 100% accuracy on AIME-25<sup>14</sup>) as a judge to annotate them with the reasoning-error taxonomy from Mahdavi et al. (2025). Table 9 reports the resulting counts.

Overall, both methods can produce correct final answers with imperfect intermediate reasoning, but we do not observe evidence here that prefix OPD is more prone to such errors than full OPD. In particular, the frequency of several error categories is comparable or lower under prefix OPD. We view this as preliminary evidence that the gains of prefix OPD in our setting are not caused by increased shortcut learning, though broader analysis across tasks and judges remains an important direction for future work.

<sup>14</sup><https://openai.com/index/introducing-gpt-5-2/>