Efficient Paths and Dense Rewards: Probabilistic Flow Reasoning for Large Language Models

Yan Liu[1,2*†] **Feng Zhang**[1,3*] **Zhanyu Ma**[1] **Jun Xu**[1‡] **Jiuchong Gao**[1‡]
Jinghua Hao[1] **Renqing He**[1] **Han Liu**[4] **Yangdong Deng**[2,5‡]
[1]Meituan, [2]Tsinghua University, [3]Peking University, [4]Dalian University of Technology
[5]FuturististAI Lab, Shanghai Tsinghua International Innovation Center

Abstract

High-quality chain-of-thought has demonstrated strong potential for unlocking the reasoning capabilities of large language models. However, current paradigms typically treat the reasoning process as an indivisible sequence, lacking an intrinsic mechanism to quantify stepwise information gain. This granularity gap manifests in two limitations: inference inefficiency from redundant exploration without explicit guidance, and optimization difficulty due to sparse outcome supervision or costly external verifiers. In this work, we propose **CoT-Flow**, a framework that reconceptualizes discrete reasoning steps as a continuous probabilistic flow, quantifying the contribution of each step toward the ground-truth answer. Built on this formulation, CoT-Flow enables two complementary methodologies: flow-guided decoding, which employs a greedy flow-based decoding strategy to extract information-efficient reasoning paths, and flow-based reinforcement learning, which constructs a verifier-free dense reward function. Experiments on challenging benchmarks demonstrate that CoT-Flow achieves a superior balance between inference efficiency and reasoning performance.

1 Introduction

Large Language Models (LLMs) have demonstrated emergent reasoning capabilities on complex tasks, driven largely by Chain-of-Thought (CoT) (Wei et al., 2022). By explicitly generating intermediate reasoning steps, CoT enables models to systematically decompose complex logical problems. To further enhance these capabilities, researchers have increasingly integrated reinforcement learning, which utilizes outcome correctness or human annotations to align reasoning behaviors (Lightman et al., 2024; Wang et al., 2024; Shao et al., 2024; Yu et al., 2025a). Despite these advancements, a fundamental problem persists in how reasoning processes are modeled: intermediate reasoning steps are typically treated as opaque sequences without quantitative utility attribution. Unlike final answers, which can be explicitly verified, the intermediate steps lack an intrinsic measure of their contribution to the problem-solving goal. The absence of stepwise awareness leads to two critical limitations: inference inefficiency and optimization difficulty.

Existing approaches to mitigate these challenges remain fragmented and constrained by inherent trade-offs. Methodologies targeting inference efficiency (Zhang et al., 2025a; Lou et al., 2025; Kang et al., 2025) often depend on handcrafted or synthesized hybrid datasets containing paired long and short reasoning chains. This requirement limits their generalization across diverse tasks. In parallel, strategies for reasoning optimization struggle to balance scalability with precision: Process Reward Models (PRMs) (Lightman et al., 2024) are bound by prohibitive annotation costs, while verifier-free proxies (Tang et al., 2025; Wang et al., 2025a) frequently yield sparse signals that lack awareness of the intermediate reasoning process. Crucially, these isolated studies lack a unified framework to explicitly and continuously quantify the contribution of intermediate reasoning steps toward the final answer.

To bridge this gap in fine-grained quantification, we revisit the reasoning process from a continuous perspective, as illustrated in Figure 1. Inspired by the rectified flow theory (Liu et al., 2023), we model the reasoning process as a probabilistic flow that transports the model's information state from the initial query to the ground truth answer. In this view, each reasoning step is physically interpreted as a velocity vector that drives the reasoning process towards the target. To strictly quantify this velocity, we introduce Probabilistic Flow Progress (PFP), which measures the instantaneous gain in

*Equal contribution.
†Work done during internship at Meituan.
‡Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 26397–26414
July 2-7, 2026

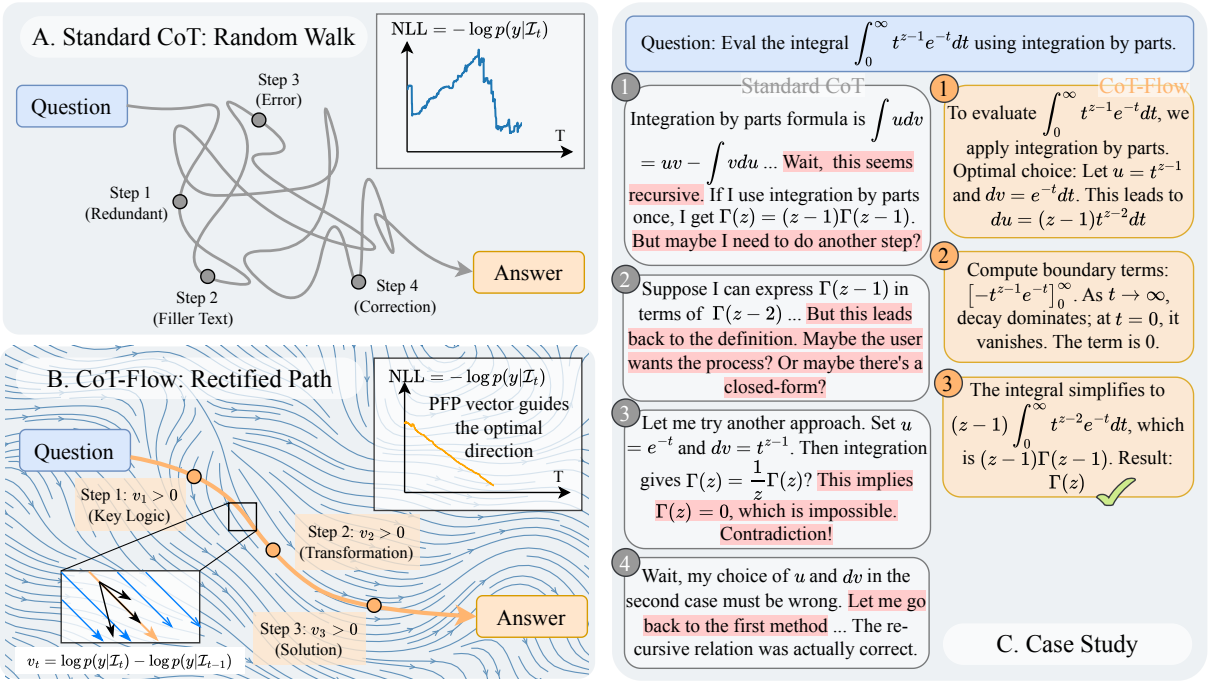


Figure 1: Illustration of our proposed CoT-Flow. Panel A: Standard CoT reasoning often exhibits a random walk behavior in the information space, characterized by unstructured exploration, redundancy, and reliance on sparse outcome-based rewards. Panel B (Ours): CoT-Flow models reasoning as a probabilistic flow. By optimizing probabilistic flow progress, it rectifies the reasoning trajectory into the shortest path from the question to the answer, providing dense supervision signals without external verifiers.

the log-likelihood of the ground truth answer. This definition provides a unified standard to evaluate reasoning. Valid logical steps exhibit positive velocity by reducing uncertainty, whereas redundant or erroneous steps manifest as zero or negative velocity, stalling the flow.

To this end, we propose CoT-Flow, a unified framework designed to streamline both reasoning inference and model training. To enhance **inference efficiency**, we employ Flow-Guided Decoding, which greedily selects tokens with high PFP scores to construct concise reasoning paths. To improve **optimization process**, we utilize Flow-based Reinforcement Learning, where the cumulative flow serves as a dense, verifier-free reward signal for robust policy alignment. Extensive experiments on mathematical and general reasoning tasks demonstrate substantial gains. Notably, on the AIME 2024 benchmark, CoT-Flow improves the performance of a Qwen3-4B model by 15.9% while reducing the average inference length by over 15%, establishing a superior Pareto frontier between efficiency and performance. Our main contributions are summarized as follows[1]:

- We propose CoT-Flow, a theoretical framework that maps LLM reasoning to continuous flow models, which can explicitly quantify the information gain of intermediate steps.
- Two flow-based algorithms are introduced, incorporating flow-guided decoding for efficient inference and a verifier-free dense reward mechanism for robust reinforcement learning.
- Experimental results on different benchmarks validate the effectiveness of CoT-Flow.

2 Background

2.1 Reinforcement Learning for Reasoning

Chain-of-thought prompting empowers large language models to decompose complex problems into intermediate reasoning steps (Wei et al., 2022). To further align these reasoning behaviors with human intent, Reinforcement learning has become the standard paradigm. Formally, given a prompt $\boldsymbol{x}$, the model generates a reasoning chain $\boldsymbol{s} = (s_1, \ldots, s_T)$ and a final answer $\boldsymbol{y}$. The optimization objective is typically to maximize the expected reward: $\mathcal{J}(\theta) = \mathbb{E}_{\boldsymbol{s} \sim \pi_\theta(\cdot|\boldsymbol{x})}[R(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y})]$. Existing approaches diverge primarily in their reward for-

[1]Code available at https://github.com/LVYUERLVR/CoT-Flow

mulation. Outcome-based methods, such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025a), utilize a sparse reward signal determined solely by the correctness of the final answer $\boldsymbol{y}$. This introduces the *credit assignment problem*, as the scalar signal at step T fails to distinguish the contribution of each token s_t. Conversely, process-based methods employ process reward models to assign step-wise scores (Lightman et al., 2024; Wang et al., 2024), mitigating sparsity but relying on costly dense annotations.

2.2 Rectified Flow and Optimal Transport

Rectified Flow (Liu et al., 2023) is a unified framework for learning ordinary differential equation (ODE) models to transport samples between two empirically observed distributions, denoted as π_0 and π_1. Unlike diffusion models that rely on specific noise schedules, Rectified Flow learns a deterministic transport map by minimizing the transport cost. Formally, let Z_t be the state at time $t \in [0, 1]$, evolving according to an ODE $dZ_t = v(Z_t, t)dt$. To transport π_0 to π_1 efficiently, Rectified Flow enforces the trajectory to follow a straight line connecting coupled samples $(X_0, X_1) \sim \pi_0 \times \pi_1$. The velocity field v is optimized via a nonlinear least squares objective:

$$\min_v \int_0^1 \mathbb{E}\left[\|(X_1 - X_0) - v(X_t, t)\|^2\right] dt, \quad (1)$$

where $X_t = tX_1 + (1-t)X_0$ represents the linear interpolation path.

In this work, we bridge continuous optimal transport and discrete reasoning. Rather than transporting pixels, we adapt the flow framework to transport the probability mass from an initial state of high uncertainty of the correct answer to a target state of deterministic certainty, rectifying the reasoning chain into the most efficient trajectory through the semantic manifold.

3 Methodology

Given a sample $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$, we aim to obtain the ground-truth answer through the reasoning process $\mathcal{I}_{\text{target}} = (\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y})$, where $\boldsymbol{x}$ is the input query, $\boldsymbol{y}$ is the corresponding answer and $\boldsymbol{s}$ is the chain-of-thought. The i-th decoding state is defined as $\mathcal{I}_i = (\boldsymbol{x}, s_1, s_2, ..., s_i)$, where s_i means the i-th token, and $\mathcal{I}_0 = \boldsymbol{x}$. We employ the negative log-likelihood as the difficulty of the current state, $D(\mathcal{I}_i) = -\log p(\boldsymbol{y}|\mathcal{I}_i)$. Then, the velocity $v(s_i)$ is as follows:

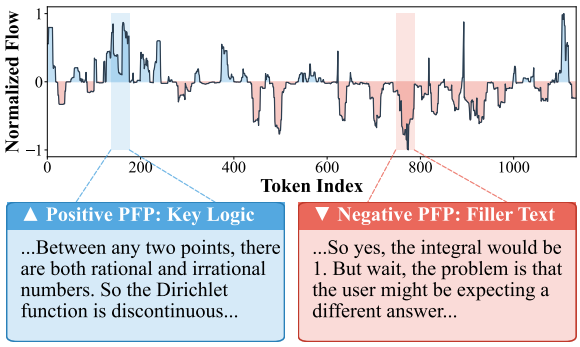


Figure 2: Visualization of velocity $v(s_i)$ over a chain-of-thought segment. Blue denotes higher velocity scores, and red denotes lower ones.

$$v(s_i) = D(\mathcal{I}_{i-1}) - D(\mathcal{I}_i) = \log \frac{p(\boldsymbol{y}|\mathcal{I}_i)}{p(\boldsymbol{y}|\mathcal{I}_{i-1})}, \quad (2)$$

where $p(\boldsymbol{y}|\mathcal{I}_i) = p(\boldsymbol{y}|\mathcal{I}_{i-1}, s_i)$ is the probability of predicting $\boldsymbol{y}$ given $\mathcal{I}_i$. A higher velocity indicates that the token yields a greater reduction in difficulty. Based on the Bayes' theorem $p(\boldsymbol{y}|\mathcal{I}_{i-1}, s_i) = \frac{p(s_i|\mathcal{I}_{i-1}, \boldsymbol{y}) \cdot p(\boldsymbol{y}|\mathcal{I}_{i-1})}{p(s_i|\mathcal{I}_{i-1})}$, we can obtain that

$$v(s_i) = \log \frac{p(s_i|\mathcal{I}_{i-1}, \boldsymbol{y})}{p(s_i|\mathcal{I}_{i-1})}, \quad (3)$$

where $p(s_i|\mathcal{I}_{i-1})$ is the probability of s_i only given the previous context $\mathcal{I}_{i-1}$, i.e., prior probability, and $p(s_i|\mathcal{I}_{i-1}, \boldsymbol{y})$ is the probability of s_i given both the previous context $\mathcal{I}_{i-1}$ and the reference answer $\boldsymbol{y}$, i.e., posterior probability. To empirically understand the role of velocity, we visualize how the velocity $v(s_i)$ reflects semantic importance within chain-of-thought reasoning in Figure 2. As shown in Figure 2, regions with high velocity concentrate on key numerical transformations and logical connectives that substantially increase the certainty of producing the correct answer, whereas regions with low velocity correspond to filler templates and repetitive statements that are largely independent of final answer. The observations show that velocity is a reliable signal for capturing the underlying logical skeleton while filtering out redundant content.

3.1 Train-Free Greedy Flow Decoding

To generate effective chain-of-thought reasoning samples, we propose a test-time greedy flow decoding strategy π_{flow} to maximize the velocity of each decoding step, thus ensuring that each step

Prompt Template for Posterior Probability

<|im_start|>user
{{question}}
Note: You have been provided with the ground truth answer: {{ground_truth}}. Your task is to generate a step-by-step reasoning process (Chain-of-Thought) that logically arrives at the conclusion. <|im_end|>

Posterior Approximation:
{{ground_truth}} ∈ {Gold Label, Random Label, ∅}

Figure 3: The posterior prompt template.

provides the greatest information gain towards the answer:

$$s_i^* = \pi_{\text{flow}}(\mathcal{I}_{i-1}) = \arg\max_{s_i \in \mathcal{V}_\tau} \{v(s_i)\}, \quad (4)$$

where $\mathcal{V}_\tau = \{s_i \in \mathcal{V} | \log p(s_i|\mathcal{I}_{i-1}) > \tau\}$ is the refined vocabulary obtained by imposing a prior-probability constraint on the original vocabulary $\mathcal{V}$. This constraint restricts candidate tokens to those whose prior probability exceeds the threshold τ, thereby promoting semantic coherence in the generated chain-of-thought.

According to Eq. (3), the velocity $v(s_i)$ is determined by the log-probability ratio between a posterior and a prior model, while the target answer $\boldsymbol{y}$ is unavailable during inference. Empirically, we observe that the velocity field is remarkably robust to the specific content of $\boldsymbol{y}$. Leveraging this property, we approximate $\log p(s_i|\mathcal{I}_{i-1}, \boldsymbol{y})$ using a fixed posterior prompt template, i.e., $\log p(s_i|\mathcal{I}_{i-1}, \text{Prompt}_{\text{post}})$, as listed in Figure 3. The ground truth answer may be selected from the gold label, a random label, or the latent label (i.e., empty content). Here, we adopt the latent label, which enables train-free and label-free refinement at test time. At each decoding step, we select s_i^* accordingly, ensuring the reasoning path adheres to the information-theoretic geodesic. Further implementation details and robustness analysis are provided in Appendix A.

Theoretical Analysis. We provide a theoretical analysis for the greedy flow decoding strategy by comparing its expected velocity against standard sampling. First, we analyze the expected velocity under the standard reference policy V_{ref} as follows:

$$\begin{aligned} V_{\text{ref}} &= \mathbb{E}_{s_i \sim \pi_\theta(\cdot|\mathcal{I}_{i-1})}[v(s_i)] \\ &= \sum_{s_i} p(s_i|\mathcal{I}_{i-1}) \cdot \log \frac{p(s_i|\mathcal{I}_{i-1}, \boldsymbol{y})}{p(s_i|\mathcal{I}_{i-1})} \\ &= -\mathbb{D}_{\text{KL}}\left[\pi_\theta(\cdot|\mathcal{I}_{i-1}) || \pi_\theta(\cdot|\mathcal{I}_{i-1}, \boldsymbol{y})\right]. \end{aligned} \quad (5)$$

Since the Kullback-Leibler divergence is non-negative, we obtain $V_{\text{ref}} \leq 0$, implying on average, standard sampling tends to adhere to the prior rather than actively steering towards the conditional target $\boldsymbol{y}$. In contrast, the greedy flow decoding strategy seeks to maximize the instantaneous velocity at each step. The expectation of velocity V_{flow}, is defined by the maximum possible velocity at step i, obtained by selecting the token s_i^* that maximizes the instantaneous velocity function $v(s_i)$, where $V_{\text{flow}} = v(s_i^*)$. By definition of the maximum, V_{flow} must be greater than or equal to the expected velocity under the policy π_θ, thus $V_{\text{flow}} \geq 0$:

$$V_{\text{flow}} \geq \mathbb{E}_{s_i \sim \pi_\theta(\cdot|\mathcal{I}_{i-1})}[v(s_i)]. \quad (6)$$

This analysis demonstrates that the reference policy yields a non-positive expected velocity, reflecting potential information decay or lack of direction relative to $\boldsymbol{y}$, whereas the flow-based strategy ensures a positive velocity, effectively directing the generation process.

3.2 Flow-Based Dense Rewards in RL

When extending the RL training paradigm to general reasoning tasks, answer verification becomes challenging due to sparse outcome rewards and the prohibitive cost of annotating process rewards. Leveraging the additivity of flow, our framework inherently yields dense rewards as a natural corollary, eliminating the need for artificial design.

3.2.1 Global Reward and Stop Gradient

According to the definition of flow, the total information gain of a complete trajectory $\boldsymbol{s}$ equals the accumulation of PFP along the path: $R_{\text{global}} = \sum_{i=1}^{T} v(s_i)$. To prevent reward hacking in model reinforcement learning by manipulating the reference baseline, we propose a stop gradient (sg) operation. By treating the previous state as a fixed environmental baseline, we ensure that the model focuses solely on improving the current policy:

$$v(s_i) = \log p(\boldsymbol{y}|\mathcal{I}_i) - \text{sg}[\log p(\boldsymbol{y}|\mathcal{I}_{i-1})]. \quad (7)$$

3.2.2 Orthogonal Decomposition of Gradients

Our optimization objective is to maximize the expected reward $\mathcal{J}(\theta) = \mathbb{E}_{\boldsymbol{s} \sim \pi_\theta(\cdot|\boldsymbol{x})}\left[R_{\text{global}}\right]$. The gradient of this objective can be strictly decom-

posed into two parts (derivation in Appendix C):

$$\nabla_\theta \mathcal{J}(\theta) = \underbrace{\mathbb{E}_{\boldsymbol{s}}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_t|\mathcal{I}_{t-1})\right) \cdot \hat{A}\right]}_{\text{Term A: RL Gradient}} + \underbrace{\mathbb{E}_{\boldsymbol{s}}\left[\sum_{i=1}^{T} \nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i)\right]}_{\text{Term B: Flow Gradient}}. \tag{8}$$

Term A represents the standard REINFORCE gradient, where $\hat{A}$ serves as the group relative advantage, analogous to GRPO. Term B is a unique dynamic term introduced by CoT-Flow, originating from the dependence of $\log p(\boldsymbol{y}|\mathcal{I}_i)$ on the parameter θ in the definition of $v(s_i)$. This term captures the evolutionary direction of the flow field itself.

3.2.3 Flow Gradient Estimation

We derive a tractable gradient estimator for the flow dynamics term. By applying the law of total probability and utilizing a single-sample Monte Carlo approximation, we transform the cumulative flow objective into a differentiable loss. Detailed mathematical proofs are in Appendix C. The final normalized gradient ∇_θTerm B is formulated as:

$$\mathcal{M} \cdot \left(\underbrace{\log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})}_{\text{Answer Generation}} + \underbrace{\sum_{k=1}^{T} \frac{k-1}{T} \log \pi_\theta(s_k)}_{\text{Time-Weighted CoT}}\right). \tag{9}$$

This formulation introduces two critical mechanisms for stable training:

Emergence of Time Weighting. The term $\frac{k-1}{T}$ naturally emerges from exchanging the order of summation in the flow trajectory. This creates a quadratic dependency which we normalize by T to match the scale of standard RL gradients. Intuitively, this assigns higher weights to later tokens in the reasoning chain, reflecting their more direct impact on the final answer certainty.

Soft Quality Gate for Variance Reduction. Relying on a raw single-sample estimate introduces high variance, potentially reinforcing low-quality paths. To mitigate this, we introduce the trajectory-level importance gate $\mathcal{M}$. Leveraging the group-relative policy optimization (GRPO) framework, we compute a dynamic baseline $\mu = \frac{1}{G}\sum_{i=1}^{G} \log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}_i)$ from a group of sampled trajectories. The gate is defined as:

$$\mathcal{M} = \text{ReLU}\left(\log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}) - \mu\right). \tag{10}$$

This acts as a high-pass filter where gradients are backpropagated only when the reasoning path $\boldsymbol{s}$ yields a posterior answer probability superior to the model's current average ($\mathcal{M} > 0$). This strictly prevents the reinforcement of below-average noisy paths and stabilizes the dense reward signal in open-ended generation.

4 Experiments

4.1 Experimental Setup

Datasets and Models. We conduct evaluations on seven challenging reasoning benchmarks: AIME24, AIME25, AMC23 and Math-500 for high-difficulty mathematics competitions; TheoremQA (Chen et al., 2023), WebInstruct (Ma et al., 2025b) and GPQA-Diamond (Rein et al., 2023) for graduate-level scientific QA. We employed the Qwen3 series (Yang et al., 2025) as backbone models to assess scalability. The LLM is trained on DeepMath-103K (He et al., 2025; Liu et al., 2025c).

Baselines. For the RL setting, we compare CoT-Flow-RL against GRPO (Shao et al., 2024), which represents the state-of-the-art outcome-based sparse reward method, and VeriFree (Zhou et al., 2025), a representative method featuring verifier-free rewards.

4.2 Implementation Details

For the test-time greedy flow decoding, at each decoding step, we set top-$p = 0.95$ to construct the candidate vocabulary $\mathcal{V}_\tau$. For the flow-based reinforcement learning, we construct a training set by randomly subsampling 5,000 instances from the DeepMath dataset. The training process utilize a learning rate of $1\text{e}{-6}$. During the rollout phase, we generate 8 responses per prompt with a temperature of $T = 1.0$ to encourage exploration. The group relative advantage is computed within a batch of 16 prompts. For all baselines and RL-tuned models, we set the sampling parameters to temperature $T = 0.6$, top-$p = 0.95$, and top-$k = 20$, with a maximum generation length of 8192 tokens.

4.3 Train-Free Greedy Flow Decoding

We first investigate the capability of CoT-Flow as a *train-free* decoding strategy.

Base Model	Method	Math Reasoning				General Task		
		AIME24	AIME25	AMC23	Math-500	GPQA-D	TheoremQA	WebInstruct
Qwen3-1.7B	Standard CoT	24.2	22.3	61.1	76.9	27.9	50.9	70.0
	CoT-Flow	**28.3**	**25.4**	**63.1**	**77.9**	**33.6**	**54.1**	**73.8**
Qwen3-4B	Standard CoT	40.8	26.5	75.9	78.5	44.1	55.2	74.3
	CoT-Flow	**56.7**	**30.8**	**84.2**	**82.1**	**44.6**	**61.1**	**77.5**
Qwen3-8B	Standard CoT	38.1	23.5	68.9	87.0	39.9	62.6	**64.7**
	CoT-Flow	**50.2**	**29.6**	**80.3**	**91.5**	**42.8**	**67.4**	63.8
Qwen3-14B	Standard CoT	44.2	29.0	79.5	90.8	55.2	69.1	83.5
	CoT-Flow	**50.8**	**34.0**	**85.0**	**93.5**	**55.9**	**72.2**	**85.2**
Qwen3-32B	Standard CoT	43.1	32.5	77.5	91.5	**57.7**	**73.0**	**84.1**
	CoT-Flow	**54.6**	**33.3**	**83.4**	**94.0**	56.2	70.6	83.5

Table 1: The pass@1 accuracy (%) of Standard CoT and CoT-Flow greedy decoding on seven diverse reasoning benchmarks. We categorize the benchmarks into Math Reasoning (AIME, AMC, Math) and General Task (GPQA, TheoremQA, WebInstruct) to demonstrate the model's capabilities across different domains.

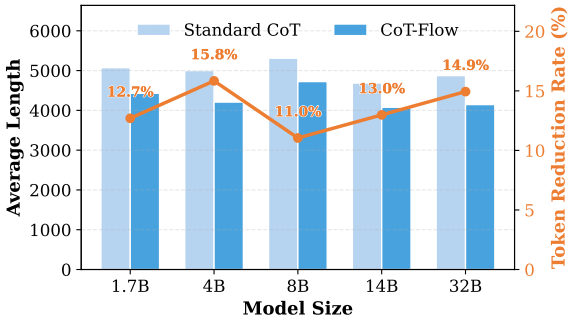


Figure 4: Comparison of token consumption for Standard CoT and our CoT-Flow method. CoT-Flow maintains comparable verbosity while improving accuracy.

Method	8K	16K	32K
Stan. CoT	42.4 (4382)	62.5 (4576)	**68.8** (5417)
CoT-Flow	**48.4** (3654)	**66.3** (3482)	67.9 (3500)

Table 2: Average accuracy (%) and response length (tokens) across different reasoning budgets on benchmarks AIME24 and GPQA-Diamond.

Accuracy Improvements. Table 1 summarizes the performance across nine benchmarks. We observe that CoT-Flow consistently surpasses standard decoding across all model scales. The advantages are particularly pronounced on complex mathematical reasoning tasks that require rigorous logic. For instance, on the Qwen3-4B model, CoT-Flow boosts accuracy on AIME24 from 40.8% to 56.7% (+15.9%). These substantial improvements indicate that the flow metric effectively steers the model away from plausible but erroneous reasoning paths often traversed by standard sampling.

Inference Efficiency. Crucially, these accuracy gains do not come at the cost of verbosity. As illustrated in Figure 4, CoT-Flow significantly reduces the average inference length compared to Standard CoT across all model sizes. For Qwen3-4B and Qwen3-32B, the average token consumption is reduced by more than 15% and 14%. We further analyze the reasoning dynamics under varying computational constraints (8K, 16K, 32K tokens) using Qwen3-4B. Table 2 illustrates that as the reasoning budget increases, Standard CoT (Stan. CoT) generates significantly longer chains (from 4,382 to 5,417 tokens) with diminishing returns in accuracy. In contrast, CoT-Flow converges to a stable trajectory length (∼3,500 tokens) regardless of the upper limit. This suggests that our method identifies the intrinsic complexity of the problem, rectifying the path to its necessary length rather than exploiting available budget for redundant computation.

4.4 Flow-Based Reinforcement Learning

We further evaluate the efficacy of CoT-Flow when integrated into the reinforcement learning loop. The core comparison lies between the sparse outcome signals of GRPO, the verifier-free signals of VeriFree, and the flow-derived dense signals of CoT-Flow. As visualized in Figure 5, CoT-Flow in reinforcement learning establishes a superior Pareto frontier compared to baselines. On high-difficulty benchmarks like AIME24 and AIME25, CoT-Flow demonstrates robust convergence to higher accuracy levels. Crucially, these gains are achieved with strictly shorter reasoning paths. While GRPO tends to generate verbose chains, the

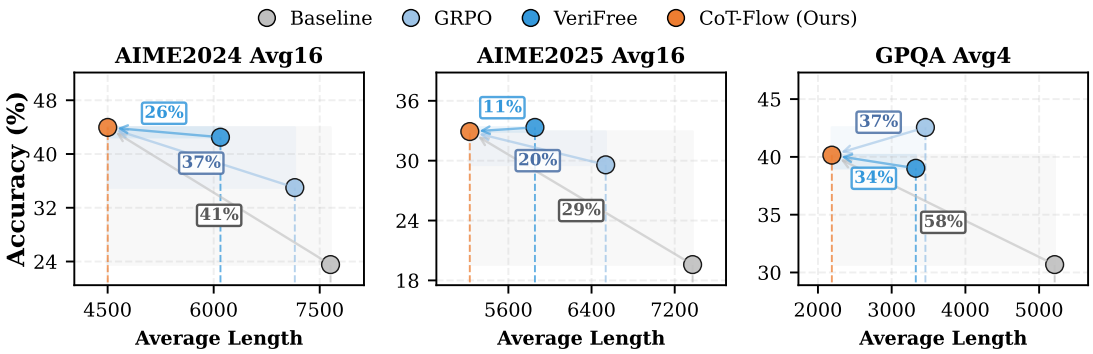


Figure 5: Pareto frontier analysis of reasoning efficiency. The model accuracy (%) and computational cost (average token length) across datasets are presented.

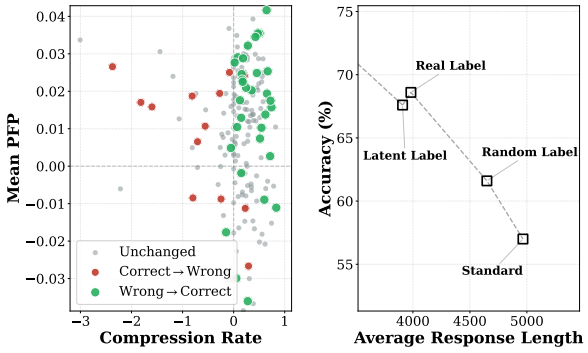


Figure 6: Analysis of inference dynamics. Left: The relationship between compression rate and mean PFP. Green points (Wrong → Correct) cluster in high-PFP regions (Upper Right). Right: Impact of posterior estimation quality. The trend confirms that more accurate posterior approximations yield superior flow guidance, with Latent Labels achieving performance comparable to ground truth.

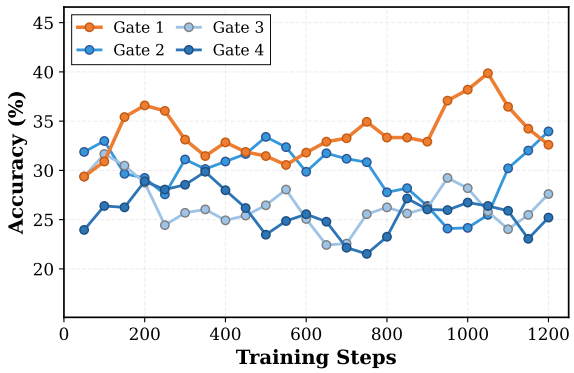


Figure 7: Ablation study on quality gate $\mathcal{M}$. We compare four gating variants on AIME2024 (Qwen3-1.7B). Gate 1 (Ours): ReLU($\Delta \log p$); Gate 2: Binary $\mathbb{I}(\Delta \log p > 0)$; Gate 3: Ratio $p/\bar{p}$; Gate 4: Absolute $p(\boldsymbol{y}|\boldsymbol{s})$.

dense reward of CoT-Flow naturally penalizes redundant steps that yield negligible information gain. This results in a simultaneous improvement in both accuracy and efficiency, validating the effectiveness of the flow-based dense supervision.

5 Further Analysis

Trajectory Rectification and Error Correction. Figure 6 visualizes the intrinsic mechanism driving the efficiency of CoT-Flow on DeepMath. CoT-Flow effectively rectifies the reasoning process into a geodesic path. Crucially, this compression is not lossy but corrective. As shown in the left panel, instances where the model corrects an initially wrong answer cluster in regions of moderate compression. This suggests that CoT-Flow effectively identifies and prunes erroneous branches that previously misled the model.

Robustness to Posterior Approximation. A core premise of test-time CoT-Flow is the estimation of velocity v without access to ground truth $\boldsymbol{y}$. As illustrated in the right panel of Figure 6, we observe a strong correlation: as the accuracy of the posterior estimation improves (quantified by the metric detailed in Appendix A.2), the performance gain of CoT-Flow increases monotonically. Remarkably, using *Latent Labels* often yields results comparable to, or even surpassing, those obtained with Gold Labels.

Impact of Quality Gating Strategy. We justify the design of our soft quality gate $\mathcal{M} = \text{ReLU}(\log p - \mu)$ by comparing it against three variants: Binary Gate (step function), Ratio Gate (linear scale), and Absolute Gate (raw probability). As shown in Figure 7, Gate 4 (Absolute) fails to converge, confirming that relative improvement (baseline subtraction) is essential for variance reduction in open-ended reasoning. Gate 3 fails to

stabilize due to the exponential nature of the ratio formulation, which induces numerical overflow and gradient explosion. While Gate 2 (Binary) incorporates the baseline, its performance plateaus early because it discards the magnitude of the improvement, treating marginally better paths the same as significantly better ones. Our soft relative gate yields the most robust convergence.

6 Related Work

6.1 Reinforcement Learning

Reinforcement Learning (RL) has emerged as a central approach for eliciting the reasoning capabilities of LLMs. Outcome-based methods, exemplified by GRPO (Shao et al., 2024), estimate policy gradients via group-relative advantages, eliminating the need for value networks. Variants like DAPO (Yu et al., 2025a) and λ-GRPO (Parthasarathi et al., 2025) further refine this by introducing dynamic sampling and learnable token preferences, while Scaf-GRPO (Zhang et al., 2025b) employs scaffolded prompting to mitigate the cold-start problem. Other approaches explore instance-adaptive budgets (Park et al., 2025) or mirror descent optimization (Wang et al., 2025c).

Nevertheless, outcome-based supervision suffers from sparse reward signals. To address this, Verifier-Free approaches seek intrinsic dense signals without expensive human annotations. Methods like VeriFree (Zhou et al., 2025) and NOVER (Liu et al., 2025b) utilize answer consistency or reasoning perplexity as rewards. Others leverage posterior regularization (Yu et al., 2025b; Fan et al., 2025), Jensen-enhanced bounds (Tang et al., 2025), or bandit-based allocation (Wang et al., 2025a) to stabilize training. Unlike these heuristic proxies, our CoT-Flow derives dense rewards directly from the transport theory of probabilistic flow.

6.2 Efficient and Hybrid Reasoning

Balancing reasoning depth with inference latency is a critical challenge. Hybrid strategies dynamically switch between concise and detailed reasoning paths. HybridCoT (Luo et al., 2025) interleaves text and latent reasoning, while TokenSkip (Xia et al., 2025) and C3oT (Kang et al., 2025) accelerate generation by selectively skipping redundant tokens or distilling lengthy chains. Other works explore continuous or latent reasoning representations. SoftCoT (Xu et al., 2025b) employs soft thought markers, and CoT-Valve (Ma et al., 2025a) identifies latent directions controlling reasoning length. KAPPA (Li et al., 2025) utilizes information-theoretic metrics for pruning. While effective, these methods often require specialized architectures or separate training stages. In contrast, CoT-Flow achieves efficiency endogenously via greedy flow rectification at test time.

6.3 Flow Dynamics and Guided Decoding

Our framework connects reasoning optimization with continuous flow dynamics. Theoretical works like FlowRL (Zhu et al., 2025) and Cognitive Flow (Matos et al., 2025) model reasoning as state transitions, while MixIE (Sanyal et al., 2025) and context-aware modeling (Yao et al., 2025) refine distribution estimation. Crucially, our velocity formulation ($v \propto \log p_{\text{post}} - \log p_{\text{prior}}$) shares theoretical underpinnings with Contrastive Decoding (Li et al., 2023) and Classifier-Free Guidance (Ho and Salimans, 2022). These paradigms amplify generation quality by contrasting a conditional distribution against an unconditional baseline. While typically applied in diffusion models for manifold constraints (Chung et al., 2025; Mirbeygi and Beigy, 2025) or controllable text generation (Huang et al., 2025; Cheng et al., 2024), CoT-Flow reinterprets this mechanism through the lens of Rectified Flow. Other theoretical frameworks (Ton et al., 2025; Wang et al., 2025b) view reasoning through the lens of information gain or optimization. Meanwhile, (Liu et al., 2025a) frames the generation of reasoning steps as an implicit gradient descent process during test time.

7 Conclusion

In this work, we propose **CoT-Flow**, a principled framework that conceptualizes LLM reasoning as a continuous probabilistic flow. By introducing the probabilistic flow progress metric, we bridge the gap between discrete token generation and continuous optimal transport theory, providing a granular measure of reasoning utility. Our dual-optimization approach, greedy flow decoding for train-free inference refinement and flow-based reinforcement learning for verifier-free dense supervision, effectively addresses the critical bottlenecks of inference inefficiency and feedback sparsity in open-domain reasoning. Empirical evaluations across challenging benchmarks demonstrate that CoT-Flow consistently achieves superior accuracy with significantly reduced computational overhead.

Limitations

Despite the promising results, our work has limitations that suggest avenues for future research. First, our velocity estimation relies on prompt-based posterior approximation. While effective, this heuristic is bounded by the zero-shot performance of the base model. We believe that developing more rigorous, trained posterior estimators could significantly enhance the precision of the flow guidance. Second, regarding RL efficiency and integration, our current experiments are conducted in an on-policy setting. Future work could extend CoT-Flow to off-policy frameworks to improve sample efficiency.

Acknowledgement

This research has been supported by the China National Key R&D Program (Grant No. 2023YFB3307201), Natural Science Foundation of Beijing (Grant No. L241020), and Putuo District (Shanghai) FuturistAI Lab Foundation (Grant No. QH2024-03-001).

References

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7889–7901.

Emily Cheng, Marco Baroni, and Carmen Amo Alonso. 2024. Linearly controlled language generation with performative guarantees. *CoRR*, abs/2405.15454.

Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. 2025. CFG++: manifold-constrained classifier free guidance for diffusion models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.

Lishui Fan, Yu Zhang, Mouxiang Chen, and Zhongxin Liu. 2025. Posterior-grpo: Rewarding reasoning processes in code generation. *CoRR*, abs/2508.05170.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, and 37 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *CoRR*, abs/2504.11456.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598.

Yingbing Huang, Deming Chen, and Abhishek K. Umrawal. 2025. JAM: controllable and responsible text generation via causal reasoning and latent vector manipulation. *CoRR*, abs/2502.20684.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24312–24320.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Sophie Li, Nicholas Huang, Nayan Saxena, Nina Luo, Vincent Lin, Kevin Zhu, and Sunishchal Dev. 2025. Inference-time chain-of-thought pruning with latent informativeness signals. *CoRR*, abs/2511.00699.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 12286–12312.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Junnan Liu, Hongwei Liu, Songyang Zhang, and Kai Chen. 2025a. Rectifying llm thought from lens of optimization. *Preprint*, arXiv:2512.01925.

Wei Liu, Siya Qi, Xinyu Wang, Chen Qian, Yali Du, and Yulan He. 2025b. NOVER: incentive training for language models via verifier-free reinforcement learning. *CoRR*, abs/2505.16022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2023. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Zihe Liu, Jiashun Liu, Yancheng He, Weixun Wang, Jiaheng Liu, Ling Pan, Xinyu Hu, Shaopan Xiong,

Ju Huang, Jian Hu, Shengyi Huang, Siran Yang, Jiamang Wang, Wenbo Su, and Bo Zheng. 2025c. Part I: tricks or traps? A deep dive into RL for LLM reasoning. *CoRR*, abs/2508.08221.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *CoRR*, abs/2505.11896.

Haotian Luo, Haiying He, Yibo Wang, Jinluan Yang, Rui Liu, Naiqiang Tan, Xiaochun Cao, Dacheng Tao, and Li Shen. 2025. Ada-r1: Hybrid-cot via bi-level adaptive reasoning optimization. *Preprint*, arXiv:2504.21659.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025a. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 6025–6035.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. 2025b. General-reasoner: Advancing LLM reasoning across all domains. *CoRR*, abs/2505.14652.

José Matos, Catarina Silva, and Hugo Goncalo Oliveira. 2025. Cognitive flow: An LLM-automated framework for quantifying reasoning distillation. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 596–616, Hanoi, Vietnam. Association for Computational Linguistics.

Mohaddeseh Mirbeygi and Hamid Beigy. 2025. Prompt guided diffusion for controllable text generation. In *Proceedings of the Tenth Workshop on Noisy and User-generated Text*, pages 78–84, Albuquerque, New Mexico, USA. Association for Computational Linguistics.

Young-Jin Park, Kristjan H. Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. 2025. Know what you don't know: Uncertainty calibration of process reward models. *CoRR*, abs/2506.09338.

Prasanna Parthasarathi, Mathieu Reymond, Boxing Chen, Yufei Cui, and Sarath Chandar. 2025. Grpo-λ: Credit assignment improves LLM reasoning. *CoRR*, abs/2510.00194.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Soumya Sanyal, Tianyi Xiao, and Xiang Ren. 2025. Mixing inference-time experts for enhancing LLM reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21246–21260, Suzhou, China. Association for Computational Linguistics.

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

Yunhao Tang, Sid Wang, Lovish Madaan, and Rémi Munos. 2025. Beyond verifiable rewards: Scaling reinforcement learning for language models to unverifiable data. *Preprint*, arXiv:2503.19618.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Jean-Francois Ton, Muhammad Faaiz Taufiq, and Yang Liu. 2025. Understanding chain-of-thought in llms through information theory. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arik. 2025a. Dynscaling: Efficient verifier-free inference scaling via dynamic and integrated sampling. *CoRR*, abs/2506.16043.

Jingyao Wang, Wenwen Qiang, Zeen Song, Changwen Zheng, and Hui Xiong. 2025b. Learning to think: Information-theoretic reinforcement fine-tuning for llms. *CoRR*, abs/2505.10425.

Mingzhi Wang, Chengdong Ma, Qizhi Chen, Linjian Meng, Yang Han, Jiancong Xiao, Zhaowei Zhang, Jing Huo, Weijie J. Su, and Yaodong Yang. 2025c. Magnetic preference optimization: Achieving last-iterate convergence for language model alignment. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *CoRR*, abs/2502.12067.

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, Shuohang Wang, Weijian Xu, Jianfeng Gao, and Weizhu Chen.

2025a. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *CoRR*, abs/2504.21233.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025b. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 23336–23351. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

Yueyang Yao, Jiajun Li, Xingyuan Dai, Mengmeng Zhang, Xiaoyan Gong, Fei-Yue Wang, and Yisheng Lv. 2025. Context-aware probabilistic modeling with LLM for multimodal time series forecasting. *CoRR*, abs/2505.10774.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025a. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. 2025b. RLPR: extrapolating RLVR to general domains without verifiers. *CoRR*, abs/2506.18254.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025a. Adaptthink: Reasoning models can learn when to think. *CoRR*, abs/2505.13417.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. 2025b. Scaf-grpo: Scaffolded group relative policy optimization for enhancing LLM reasoning. *CoRR*, abs/2510.19807.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. 2025. Reinforcing general reasoning without verifiers. *CoRR*, abs/2505.21493.

Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, Qizheng Zhang, Lin Chen, Fanghao Shao, Bo Xue, Yunchong Song, Zhenjie Yang, Ganqu Cui, Ning Ding, Jianfeng Gao, and 4 others. 2025. Flowrl: Matching reward distributions for LLM reasoning. *CoRR*, abs/2509.15207.

A Theoretical and Empirical Justification for Posterior Estimation

A.1 Theoretical Justification for Prompt-based Posterior Estimation

A core challenge in the CoT-Flow framework lies in accurately estimating the posterior probability of the current reasoning step conditioned on the ground-truth answer, denoted as $p(s_i|\mathcal{I}_{i-1}, \boldsymbol{y})$. Theoretically, the ground-truth posterior can be approximated via a frequentist Monte Carlo approach. This involves sampling a large number of complete reasoning trajectories $\mathcal{S} = \{S^{(1)}, \ldots, S^{(N)}\}$ starting from the current state $\mathcal{I}_{i-1}$ and calculating the proportion of trajectories that successfully lead to the correct answer $\boldsymbol{y}$:

$$p(s_i|\mathcal{I}_{i-1}, \boldsymbol{y}) \approx \frac{\sum_{k=1}^{N} \mathbb{I}(S^{(k)} \vdash \boldsymbol{y})}{N}, \qquad (11)$$

where $\mathbb{I}(\cdot)$ is the indicator function and $S^{(k)} \vdash \boldsymbol{y}$ denotes that trajectory $S^{(k)}$ concludes with answer $\boldsymbol{y}$. However, this method is computationally prohibitive for online decoding due to the necessity of massive sampling.

From a modeling perspective, one could train a dedicated conditional model via supervised learning on (Context, CoT, Answer) triplets. To strictly adhere to the causal dependency of $p(S|\boldsymbol{x}, \boldsymbol{y})$, the self-attention mechanism of such a model implies a specific masking pattern, where the token s_i can attend to the context $\boldsymbol{x}$, the answer $\boldsymbol{y}$, and preceding reasoning steps $s_{<i}$. This ideal attention structure is illustrated in the left panel of Figure 8. However, training a separate posterior model introduces significant overhead. We argue that a sufficiently pretrained autoregressive LLM can inherently function as a posterior estimator through *In-Context Posterior Approximation*, eliminating the need for additional training.

By restructuring the input sequence to place the answer $\boldsymbol{y}$ within the prefix (i.e., $\boldsymbol{x} \to \boldsymbol{y} \to S$), we force the Transformer to compute the representation of s_i conditioned on $\boldsymbol{y}$. As visualized in the right panel of Figure 8, the attention connectivity for the reasoning tokens S in our CoT-Flow strategy is mathematically identical to the ideal target model within the causal masking constraint. The discrepancy lies in the blind spot (depicted in gray) where the answer tokens $\boldsymbol{y}$ cannot attend to the subsequent reasoning S. Crucially, while the attention mask allows for the correct information flow, a discrepancy remains in the positional embeddings. To mitigate this positional mismatch and strictly prevent the model from treating the inserted answer as a trivial leakage, we employ a specific prompt: *"It is crucial that your reasoning appears natural and self-derived. Do not, under any circumstances, state or imply that you were given the ground truth answer."*

A.2 Comparative Analysis with Standard Zero-Shot Prompting

While our prompt-based posterior estimation (Section A) utilizes a specific template to approximate the target distribution, a natural alternative is the widely used zero-shot trigger, *"Let's think step by step"* (Kojima et al., 2022). In this section, we analyze why this standard prompting strategy is insufficient for calculating the flow velocity $v(s_i)$ and justify the necessity of our Latent Label design.

Fundamentally, the velocity in CoT-Flow is defined as the log-likelihood ratio between a goal-conditioned posterior and an unconditioned prior: $v(s_i) = \log p_{post}(s_i|\mathcal{I}_{i-1}) - \log p_{prior}(s_i|\mathcal{I}_{i-1})$. The efficacy of this metric relies on the *divergence* between these two distributions.

The standard zero-shot prompt operates as a *strong prior* rather than a posterior approximation. While it encourages chain-of-thought generation, it does so in a forward-looking manner similar to the base model's intrinsic exploration tendency. Consequently, if we employ *"Let's think step by step"* as the proxy for π_{post}, the resulting distribution $\pi_{zero-shot}$ remains semantically and structurally close to the prior π_{prior}. This similarity leads to a vanishing velocity field ($v \approx 0$), failing to provide the distinctive directional guidance required to rectify the reasoning path.

In contrast, our Latent Label strategy (Figure 9) structurally mimics the generation process of a model that *has observed* the answer. By injecting the placeholder structure ($x \to$ Answer $\to s$), we induce a "pseudo-oracle" state. Although the explicit answer is absent (latent), this structural constraint forces the model to adopt a verification-like stance, creating a significant informational divergence from the standard prior. This divergence manifests as a high-magnitude velocity vector that effectively highlights key logical transformations while suppressing generic or redundant tokens common in standard zero-shot reasoning.

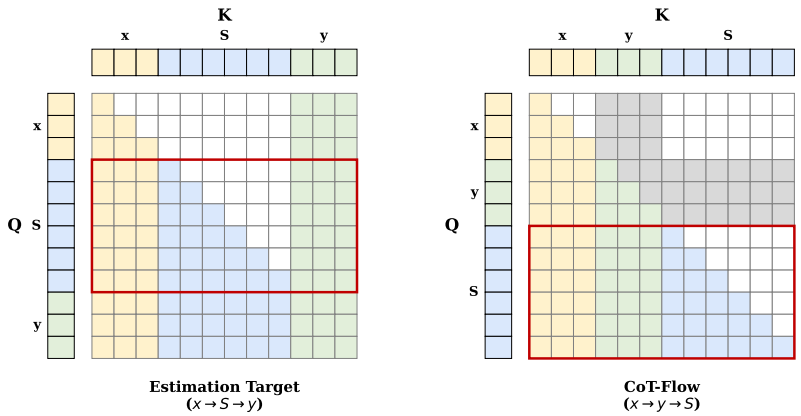


Figure 8: Comparison of attention masks for posterior estimation. Left (Estimation Target): The ideal attention pattern for modeling $p(S|\boldsymbol{x}, \boldsymbol{y})$, where the reasoning steps S can attend to both the context $\boldsymbol{x}$ and the future answer $\boldsymbol{y}$. Right (CoT-Flow): Our proposed approximation using prompt engineering ($\boldsymbol{x} \to \boldsymbol{y} \to S$). The red box highlights the estimation target. The gray areas indicate attention connections present in the target but lost in our approximation (i.e., $\boldsymbol{y}$ cannot attend to S).

B Extended Empirical Results

B.1 Computational Cost and Efficiency Analysis

Inference Latency vs. Throughput. While CoT-Flow introduces a per-step computational overhead by calculating the difference between posterior and prior logits ($\log \pi_{\text{post}} - \log \pi_{\text{prior}}$), the end-to-end latency for generating a single sample is comparable to, or even lower than, Standard CoT. This is attributed to the significantly shorter reasoning paths generated by our method. As shown in Table 3, the single-sample latency (batch size = 1) demonstrates that the reduction in decoding steps effectively offsets the per-step overhead.

However, a distinction must be made between strict single-sample latency and total computational cost in high-throughput scenarios. When processing large batches, the theoretical lower bound for total FLOPs approaches $2\times$ that of standard decoding, as our method evaluates both π_{post} and π_{prior}. Empirically, in our current implementation (using 2 NVIDIA A100 GPUs with a tensor parallel size of 2), the observed time cost overhead is approximately $2.2\times$ to $3.3\times$ (Table 3). This discrepancy arises because our dual-forward pass framework currently lacks advanced system-level optimizations. Bridging this gap through system-algorithm co-design remains a promising direction for future work.

Training Efficiency. In the reinforcement learning setting, CoT-Flow-RL introduces virtually zero additional computational overhead during training.

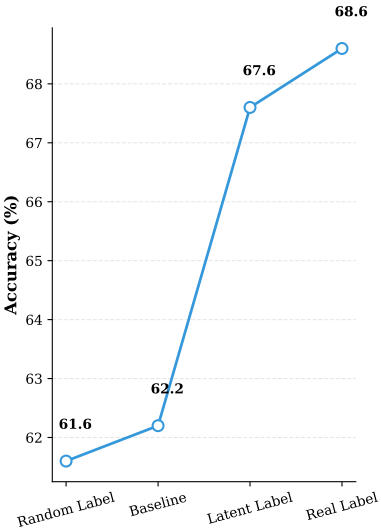
Method	Latency (s) ↓		Total Time (s) ↓	
	AIME24	AMC23	AIME24	AMC23
Model: Qwen3-1.7B				
Standard CoT	41.5	28.5	**992**	**872**
CoT-Flow	**37.1**	**24.8**	3322	2897
Model: Qwen3-32B				
Standard CoT	186.6	131.9	**4027**	**3654**
CoT-Flow	**174.7**	**110.1**	9056	7864

Table 3: Comparison of computational efficiency between Standard CoT and CoT-Flow. *Latency* refers to the end-to-end time for a single-sample inference (batch size = 1), whereas *Total Time* represents the overall throughput cost (batch size = 16).

The calculation of the dense reward, $\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})$, utilizes a prefill operation that merely appends the ground-truth answer. In standard GRPO, a similar prefill is already required to compute the old policy probabilities (π_{old}) for the importance sampling ratio. Furthermore, because CoT-Flow inherently encourages more concise reasoning paths, the rollout phase—typically the primary bottleneck in RL—is significantly accelerated. As detailed in Table 4, CoT-Flow-RL reaches specific training milestones substantially faster than baseline methods.

B.2 Robustness Across Model Architectures

To verify the generalizability of our flow-guided decoding, we evaluate CoT-Flow across diverse LLM architectures beyond the Qwen3 series. As shown in Table 5, CoT-Flow achieves accuracy gains on most benchmarks while consistently maintaining



Method	Prompt Template
Random Label	Note: You have been provided with the ground truth answer: {{random.randint(0, 1000)}}. Your task is to generate a step-by-step reasoning process (Chain-of-Thought) that logically arrives at the conclusion. It is crucial that your reasoning appears natural and self-derived. Do not, under any circumstances, state or imply that you were given the ground truth answer.
Baseline	Note: Your task is to generate a step-by-step reasoning process (Chain-of-Thought) that logically arrives at the conclusion.
Latent Label	Note: You have been provided with the ground truth answer. Your task is to generate a step-by-step reasoning process (Chain-of-Thought) that logically arrives at the conclusion.
Real Label	Note: You have been provided with the ground truth answer: {{ground_truth}}. Your task is to generate a step-by-step reasoning process (Chain-of-Thought) that logically arrives at the conclusion. It is crucial that your reasoning appears natural and self-derived. Do not, under any circumstances, state or imply that you were given the ground truth answer.

Figure 9: Impact of posterior prompt quality on CoT-Flow performance. We compare the reasoning accuracy when using different prompt templates to approximate the posterior distribution π_{post}. The results exhibit a monotonic increase in accuracy as the prompt becomes more informative, confirming that our *Latent Label* strategy effectively elicits a pseudo-oracle guidance comparable to the *Real Label* upper bound.

Progress	CoT-Flow (Ours)	VeriFree	GRPO
25%	**13h 09m**	14h 15m	15h 05m
50%	**24h 20m**	25h 46m	29h 20m
75%	**33h 51m**	36h 37m	44h 14m
100%	**42h 17m**	46h 53m	62h 36m

Table 4: Cumulative wall-clock time to reach training milestones for Qwen3-1.7B on DeepMath (4× A100 GPUs). CoT-Flow consistently reduces total training time by minimizing generation lengths during rollout.

shorter reasoning paths across different model architectures such as Phi-4-mini-reasoning (Xu et al., 2025a), QwQ (Team, 2025), and GLM-Z1-32B-0414 (GLM et al., 2024), which confirms its strong generalization ability.

B.3 Robustness of Latent Label Prompting

In the train-free setting, substituting the unobserved ground truth $\boldsymbol{y}$ with a latent label $\boldsymbol{y}'$ acts as a heuristic approximation. By conditioning on a greedy implicit estimate $\boldsymbol{y}'$, CoT-Flow prevents the model's reasoning trajectory from diffusing across multiple distinct semantic modes, prioritizing logical convergence over early correctness. Furthermore, to verify that our improvements are resilient to prompt engineering, we test multiple semantic variations of the latent label prompt. As shown in Table 6, the accuracy remains consistently superior to the baseline, confirming that the performance gain stems from the structural conditioning rather than specific phrasing.

Method	AIME24	AIME25	GPQA
Model: Phi-4-mini-reasoning			
Std. CoT	31.7 (7042)	25.0 (7037)	26.2 (6557)
CoT-Flow	**32.5** (6666)	**26.6** (6957)	**33.0** (6066)
Model: QwQ			
Std. CoT	38.7 (7395)	**26.6** (7444)	46.8 (5595)
CoT-Flow	**43.3** (7075)	24.1 (7333)	**50.7** (3998)
Model: GLM-Z1-32B-0414			
Std. CoT	58.5 (6024)	39.7 (6508)	**55.4** (4753)
CoT-Flow	**60.4** (5700)	**46.6** (6214)	53.6 (3955)

Table 5: Performance across different model families. Values indicate accuracy (%) with average generation length in parentheses.

B.4 Ablation on Velocity Formulation: Posterior-Only Decoding

To further validate the theoretical motivation of CoT-Flow, we investigate the necessity of the velocity formulation defined in Eq. (3): $v(s_i) = \log p(s_i|\mathcal{I}_{i-1}, \text{Prompt}_{\text{post}}) - \log p(s_i|\mathcal{I}_{i-1})$. A natural question arises: does the performance gain stem primarily from the *contrast* between posterior and prior (the flow velocity), or simply from the *guidance* provided by the posterior prompt itself? To answer this, we introduce a posterior-only baseline, where the decoding objective is to maximize the raw posterior probability:

$$s_i^* = \arg\max_{s_i} \log p(s_i|\mathcal{I}_{i-1}, \text{Prompt}_{\text{post}}). \quad (12)$$

Method	Prompt Phrasing	Accuracy (%)
Baseline	(Standard CoT without posterior conditioning)	56.9
Ours (Var. 1)	*You have already known the ground truth answer.*	**67.6**
Ours (Var. 2)	*You are fully aware of the ground truth answer and the correct solution.*	66.6
Ours (Var. 3)	*Given that you have access to the true answer.*	66.0

Table 6: Robustness analysis of the latent label prompt on AIME24. Varying the instruction phrasing yields stable improvements over the baseline, indicating low sensitivity to prompt selection.

We compare Standard CoT, CoT-Flow, and Post-Only across four representative benchmarks: AIME 2024, AIME 2025, AMC 23, and GPQA. The results are summarized in Table 7. As observed in Table 7, Post-Only decoding consistently outperforms the Standard CoT baseline. This indicates that our proposed posterior prompt template effectively elicits a latent label that guides the model toward better reasoning. However, CoT-Flow frequently surpasses Post-Only, particularly on the most challenging benchmarks like AIME 2024 (e.g., +10.5% over Post-Only on Qwen3-4B).

This result validates our hypothesis. Simply following the posterior distribution is insufficient, as it may still assign high probability to trivial or safe tokens that do not contribute to solving the problem. By subtracting the prior (Standard CoT probability), CoT-Flow explicitly selects tokens with high *velocity*, those that contribute most to the *change* in certainty, thereby rectifying the path more effectively than posterior guidance alone.

B.5 Extended Analysis on Pass@k Scaling

In this section, we provide a comprehensive analysis of the scaling properties of CoT-Flow across varying sampling budgets (k). Figures 10 and 11 illustrate the Pass@k performance curves for Qwen3-8B and Qwen3-32B, respectively. A consistent phenomenon is observed across all model scales: CoT-Flow (CFG) significantly outperforms the Baseline and Post-Only methods at low sample budgets (e.g., $k = 1, 2$), but the gap narrows or reverses as k increases (e.g., $k = 16$). This behavior is theoretically expected and highlights the distinct operational mechanism of our method.

Concentration of Probability Mass. Standard CoT sampling operates on the raw high-entropy distribution of the language model. This random walk nature allows for diverse exploration; given a sufficiently large budget ($k \to \infty$), the model is likely to stumble upon the correct reasoning path simply via broad coverage. However, this comes at the cost of high redundancy and error rates in single-pass generation. In contrast, CoT-Flow employs a greedy flow decoding strategy ($v = \log p_{\text{post}} - \log p_{\text{prior}}$). This difference-of-logits operation acts as a *contrastive filter*, aggressively suppressing generic, low-information, or plausible-but-incorrect tokens while amplifying tokens that specifically contribute to the likelihood of the answer. Geometrically, this rectifies the reasoning flow, forcing the probability mass to concentrate around the geodesic path.

Distillation into Pass@1. Consequently, CoT-Flow effectively distills the model's reasoning capability into the top-ranked trajectory. It optimizes for precision rather than coverage. By pruning the branching factor of the reasoning tree, CoT-Flow ensures that the single most probable path is highly accurate, thereby achieving superior efficiency. However, this sharpening of the distribution inherently reduces generation diversity. At high k, the model tends to generate topologically similar paths, yielding diminishing returns compared to the baseline, which benefits from the wisdom of crowds effect in high-variance sampling. In practical deployment scenarios, where inference latency and compute costs are critical constraints, the performance at low k (especially Pass@1) is the dominant metric. CoT-Flow's ability to maximize utility per sample makes it an ideal solution for resource-constrained reasoning.

Model	Method	Math Reasoning			General Task
		AIME24	AIME25	AMC23	GPQA
Qwen3-1.7B	Standard CoT	24.2	22.3	61.1	27.9
	Post-Only	25.6	22.5	**64.7**	30.7
	CoT-Flow	**28.3**	**25.4**	63.1	**33.6**
Qwen3-4B	Standard CoT	40.8	26.5	75.9	44.1
	Post-Only	46.2	**33.1**	82.2	**45.7**
	CoT-Flow	**56.7**	30.8	**84.2**	44.6
Qwen3-8B	Standard CoT	38.1	23.5	68.9	39.9
	Post-Only	40.2	27.9	74.7	36.6
	CoT-Flow	**50.2**	**29.6**	**80.3**	**42.8**
Qwen3-14B	Standard CoT	44.2	29.0	79.5	55.2
	Post-Only	**51.5**	33.8	81.9	53.3
	CoT-Flow	50.8	**34.0**	**85.0**	**55.9**
Qwen3-32B	Standard CoT	43.1	32.5	77.5	57.7
	Post-Only	45.8	**33.5**	**83.9**	**57.8**
	CoT-Flow	**54.6**	33.3	83.4	56.2

Table 7: Ablation study comparing Standard CoT, Posterior-Only decoding, and CoT-Flow (Contrastive). Values represent Pass@1 accuracy (%). While Post-Only generally improves over the Standard baseline, CoT-Flow achieves superior performance on challenging reasoning tasks (e.g., AIME 2024), confirming the importance of the velocity metric in filtering high-likelihood but low-information tokens.

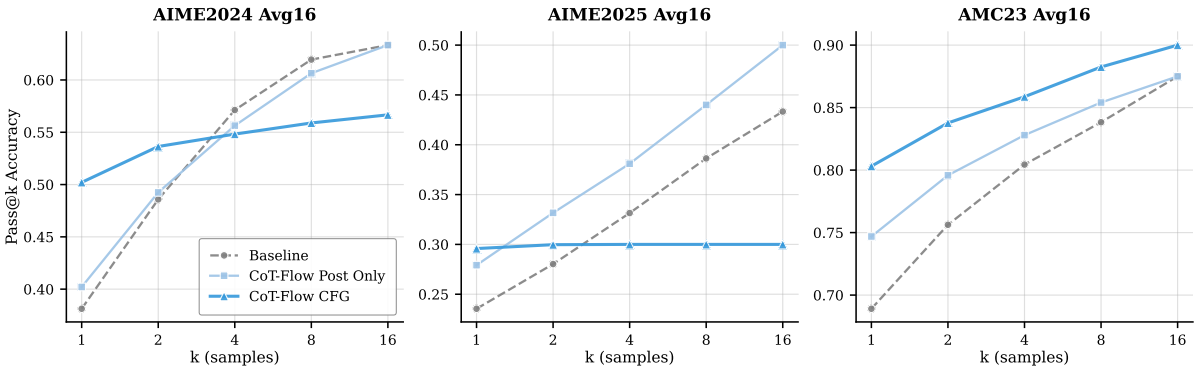


Figure 10: Pass@k scaling curve for Qwen3-8B on AIME 2024, AIME 2025, and AMC 23.

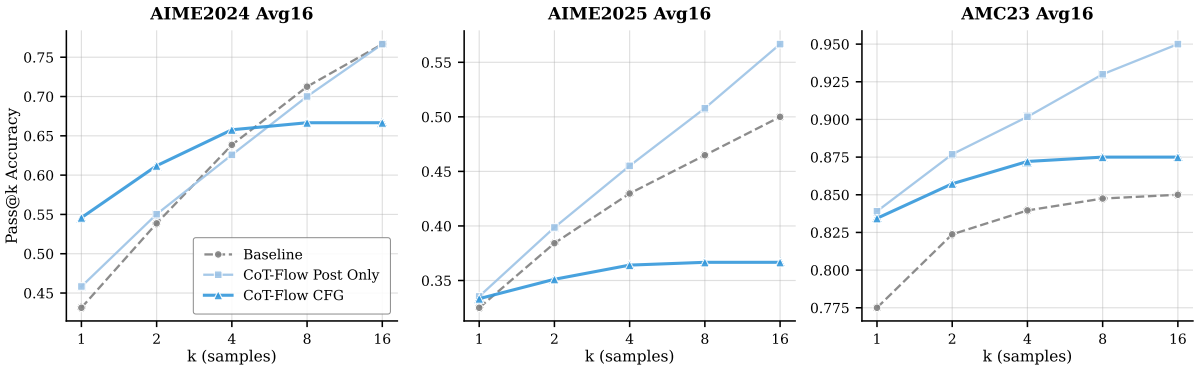


Figure 11: Pass@k scaling curve for Qwen3-32B on AIME 2024, AIME 2025, and AMC 23.

C Detailed Mathematical Derivation of Flow-based RL Objectives

This appendix provides a rigorous step-by-step derivation of the global reward and the decomposition of its gradient, specifically focusing on the emergence of the time-weighted Flow Gradient (Term B).

C.1 Orthogonal Gradient Decomposition

When optimizing $\mathcal{J}(\theta) = \mathbb{E}_{\boldsymbol{s}\sim\pi_\theta(\cdot|\boldsymbol{x})}[R_{\text{global}}(\theta)]$, the gradient involves two components due to the dependence of R_{global} on θ. Applying the identity $\nabla_\theta \mathbb{E}_{\pi_\theta}[f(\theta)] = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta \cdot f(\theta) + \nabla_\theta f(\theta)]$ yields:

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{\boldsymbol{s}}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_t|\mathcal{I}_{t-1})\right) \hat{R}_{\text{global}}\right] + \mathbb{E}_{\boldsymbol{s}}\left[\sum_{i=1}^{T} \nabla_\theta v(s_i)\right]. \tag{13}$$

The scaler global reward $\hat{R}_{\text{global}}$ is defined as the accumulation of Probabilistic Flow Progress (PFP) along a complete reasoning trajectory $\boldsymbol{s} = (s_1, \ldots, s_T)$. Using the definition $v(s_i) = \log p_\theta(\boldsymbol{y}|\mathcal{I}_i) - \text{sg}[\log p_\theta(\boldsymbol{y}|\mathcal{I}_{i-1})]$, we analyze the total reward:

$$\hat{R}_{\text{global}} = \sum_{i=1}^{T} v(s_i) = \sum_{i=1}^{T} \Big(\log p_\theta(\boldsymbol{y}|\mathcal{I}_i) - \text{sg}[\log p_\theta(\boldsymbol{y}|\mathcal{I}_{i-1})]\Big) = \log p_\theta(\boldsymbol{y}|\mathcal{I}_T) - \log p_\theta(\boldsymbol{y}|\mathcal{I}_0), \tag{14}$$

where $\mathcal{I}_T = (\boldsymbol{x}, \boldsymbol{s})$ and $\mathcal{I}_0 = \boldsymbol{x}$. Using group relative normalization (e.g., GRPO), the constant $\log p_\theta(\boldsymbol{y}|\boldsymbol{x})$ is canceled out, leaving the final answer likelihood normalized as $\hat{A}$.

C.2 Step-by-Step Derivation of Term B

By the definition of $v(s_i)$ in Eq. (7), the term $\nabla_\theta v(s_i)$ is:

$$\nabla_\theta v(s_i) = \nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i) - \nabla_\theta \text{sg}[\log p_\theta(\boldsymbol{y}|\mathcal{I}_{i-1})] = \nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i). \tag{15}$$

The stop-gradient operation effectively eliminates the baseline gradient, ensuring that the flow field is optimized solely based on the improvement at each step. To expand $\sum_{i=1}^{T} \nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i)$, we first examine the single-step term. Using the law of total probability over all possible future trajectories $\boldsymbol{s}_{i+1:T}$, we obtain:

$$p_\theta(\boldsymbol{y}|\mathcal{I}_i) = \sum_{\boldsymbol{s}_{i+1:T}} \pi_\theta(\boldsymbol{s}_{i+1:T}|\mathcal{I}_i) \cdot p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}). \tag{16}$$

Applying the log-derivative trick $\nabla_\theta \log f = \frac{\nabla_\theta f}{f}$ yields:

$$\begin{aligned}\nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i) &= \frac{1}{p_\theta(\boldsymbol{y}|\mathcal{I}_i)} \sum_{\boldsymbol{s}_{i+1:T}} \nabla_\theta \left(\pi_\theta(\boldsymbol{s}_{i+1:T}|\mathcal{I}_i) p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})\right) \\ &= \frac{1}{p_\theta(\boldsymbol{y}|\mathcal{I}_i)} \sum_{\boldsymbol{s}_{i+1:T}} \left(p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}) \nabla_\theta \pi_\theta(\boldsymbol{s}_{i+1:T}|\mathcal{I}_i) + \pi_\theta(\boldsymbol{s}_{i+1:T}|\mathcal{I}_i) \nabla_\theta p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})\right).\end{aligned} \tag{17}$$

By converting the sum back to an expectation $\mathbb{E}_{\pi_\theta}$, we derive:

$$\nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i) = \mathbb{E}_{\boldsymbol{s}_{i+1:T}}\left[\frac{p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})}{p_\theta(\boldsymbol{y}|\mathcal{I}_i)} \left(\nabla_\theta \log \pi_\theta(\boldsymbol{s}_{i+1:T}|\mathcal{I}_i) + \nabla_\theta \log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})\right)\right]. \tag{18}$$

Using the importance weight $M_i = \frac{p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})}{p_\theta(\boldsymbol{y}|\mathcal{I}_i)}$ and a single-sample Monte Carlo approximation, we obtain:

$$\nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i) \approx M_i \left(\sum_{k=i+1}^{T} \nabla_\theta \log \pi_\theta(s_k|\mathcal{I}_{k-1}) + \nabla_\theta \log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})\right). \tag{19}$$

C.3 Summation Reordering and Final Form

Substituting the single-step gradient into the global summation $\sum_{i=1}^{T} \nabla_\theta \log p_\theta(\boldsymbol{y}|\mathcal{I}_i)$ requires careful treatment of the importance weight $M_i = p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})/p_\theta(\boldsymbol{y}|\mathcal{I}_i)$. Under a naive single-sample Monte Carlo (MC) estimation where $p_\theta(\boldsymbol{y}|\mathcal{I}_i) \approx p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})$, the weight M_i simplifies to 1. However, this approximation exhibits high variance. Specifically, when the sampled trajectory $\boldsymbol{s}$ is of low quality (i.e., $p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})$ is significantly lower than the model's average capability), $M_i \approx 1$ would erroneously reinforce suboptimal or incorrect reasoning paths by assigning them full gradient weight.

To mitigate this, we replace the unstable M_i with a trajectory-level soft quality gate $\mathcal{M}$, which is a function of the global answer likelihood $p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s})$. Since $\mathcal{M}$ is independent of the summation index i, it can be factored out of the global sum:

$$\nabla_\theta \text{Term B} \approx \mathcal{M} \left[\sum_{i=1}^{T} \sum_{k=i+1}^{T} \nabla_\theta \log \pi_\theta(s_k|\mathcal{I}_{k-1}) + \sum_{i=1}^{T} \nabla_\theta \log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}) \right]. \tag{20}$$

By exchanging the order of the double summation $\sum_{i=1}^{T} \sum_{k=i+1}^{T} = \sum_{k=1}^{T} \sum_{i=1}^{k-1}$, the first part becomes:

$$\sum_{k=1}^{T} \left(\sum_{i=1}^{k-1} 1 \right) \nabla_\theta \log \pi_\theta(s_k|\mathcal{I}_{k-1}) = \sum_{k=1}^{T} (k-1) \nabla_\theta \log \pi_\theta(s_k|\mathcal{I}_{k-1}). \tag{21}$$

The second part, being independent of i, simply accumulates T times. Combining these and normalizing by $1/T$ to match the $O(T)$ scale of Term A, we arrive at the final Flow Objective:

$$\mathcal{L}_{\text{Flow}} = \mathcal{M} \left(\sum_{k=1}^{T} \frac{k-1}{T} \log \pi_\theta(s_k|\mathcal{I}_{k-1}) + \log p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}) \right). \tag{22}$$

This derivation shows that the $(k-1)/T$ weighting naturally rewards tokens that appear later in the chain, as they reduce the marginal uncertainty $p(\boldsymbol{y}|\mathcal{I}_i)$ more directly.