

# VCORE: Variance-Controlled Optimization-based Reweighting for Chain-of-Thought Supervision

Xuan Gong<sup>1</sup>, Senmiao Wang<sup>2</sup>, Hanbo Huang<sup>1</sup>, Ruoyu Sun<sup>2</sup>, Shiyu Liang<sup>1\*</sup>

<sup>1</sup> Shanghai Jiao Tong University, <sup>2</sup> Chinese University of Hong Kong (Shenzhen)

{gongxuan0610, hhuang417, lsy18602808513}@sjtu.edu.cn,  
senmiaowang1@link.cuhk.edu.cn, ruoyus@illinois.edu

## Abstract

Supervised fine-tuning (SFT) on long chain-of-thought (CoT) trajectories has emerged as a crucial technique for enhancing the reasoning abilities of large language models (LLMs). However, the standard cross-entropy loss treats all tokens equally, ignoring their heterogeneous contributions across a reasoning trajectory. This uniform treatment leads to misallocated supervision and weak generalization, especially in complex, long-form reasoning tasks. To address this, we introduce **V**ariance-**C**ontrolled **O**ptimization-based **R**eweighting (VCORE), a principled framework that reformulates CoT supervision as a constrained optimization problem. By adopting an optimization-theoretic perspective, VCORE enables a principled and adaptive allocation of supervision across tokens, thereby aligning the training objective more closely with the goal of robust reasoning generalization. Empirical evaluations demonstrate that VCORE achieves the strongest overall average performance, with especially clear gains on lower-capacity models. Across both in-domain and out-of-domain settings, VCORE achieves substantial performance gains on mathematical and coding benchmarks, using models from the Qwen3 series (4B, 8B, 32B) and LLaMA-3.1-8B-Instruct. Moreover, we show that VCORE serves as a more effective initialization for subsequent reinforcement learning, establishing a stronger foundation for advancing the reasoning capabilities of LLMs.<sup>1</sup>

## 1 Introduction

Recent advances in LLMs have highlighted the impressive benefit of long chain-of-thought (CoT) for enhancing the reasoning capabilities. Recent LLMs, such as OpenAI-o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025a), Kimi k1.5 (Team et al., 2025a), and Qwen3 series (Yang et al., 2025),

\*Corresponding author

<sup>1</sup>The code will be released at <https://github.com/coder-gx/VCORE>.

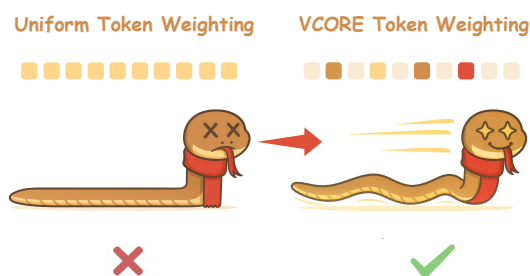


Figure 1: **Overview of VCORE.** Compared to the standard cross-entropy loss, VCORE approaches long-CoT SFT from an optimization perspective and adjusts token weights according to their gradient utility, thereby enabling more effective use of supervision signals and improving generalization.

pursue this direction by scaling CoT lengths and show strong reasoning performance on tasks such as challenging math problem solving and code generation benchmarks.

Beyond reinforcement learning (RL) and test-time methods, long CoT supervised fine-tuning (SFT) has been increasingly adopted by AI labs and companies (Team, 2025; Muennighoff et al., 2025; Xu et al., 2025; Labs, 2025; Ye et al., 2025). Compared with these two paradigms, long-CoT SFT typically distills reasoning traces from teacher models or curated datasets, offering a more straightforward yet effective route to improving reasoning. Nevertheless, most existing long-CoT SFT works emphasize engineering implementations and data recipes, leaving substantial blanks on the *optimization-algorithm progress*.

Unfortunately, long-CoT SFT is particularly susceptible to supervision noise, which can lead to misallocated supervision and degraded generalization (Luo et al., 2025; Lobo et al., 2025). To a large extent, this susceptibility can be attributed to the convention of uniform token weighting in the cross-entropy loss across lengthy reasoning traces (e.g. exceeding 1k tokens). A growing body of evidence shows that: (1) not all intermediate tokens

are equally worth learning from (Choi et al., 2025; Li et al., 2025); (2) spurious or unfaithful tokens may corrupt learning signals (Chen et al., 2025b; Turpin et al., 2023; Zhou et al., 2024). These findings motivate *breaking the limitation of uniform token weighting* in long-CoT supervision.

The above discussion leads to a central question:

*Can we design a long CoT supervision algorithm that reweights tokens more effectively through an optimization-based approach?*

In this paper, we answer this question by proposing **VCORE** (Figure 1): a variance-controlled, optimization-based reweighting framework for CoT supervision. VCORE departs from prior heuristics and formulates token reweighting as a constrained optimization problem: for each trajectory, compute the distribution over token positions that maximizes expected loss descent under a single SGD step, subject to a KL constraint for stability. This yields a closed-form *Gibbs distribution* over token-wise gradient utilities, directly grounded in first-order descent dynamics. To efficiently estimate utilities, VCORE introduces a lightweight *one-backward probing trick* that requires only one backward pass and forward-mode perturbations per batch. To ensure stable updates, it further rescales the weights using a principled variance control coefficient  $\alpha$ , which matches the update variance to that of uniform weighting. This end-to-end formulation identifies high-impact tokens from the training signal itself, *without relying on teacher guidance, confidence thresholds or entropy filters*. Our contributions are summarized as follows:

- We cast CoT supervision as a constrained optimization problem over token weights, grounded in the dynamics of SGD. This formulation yields a closed-form Gibbs distribution that allocates weight based on token-level gradient utility.
- We propose **VCORE**, a unified framework that combines optimization-derived token reweighting with variance-controlled scaling in a single efficient pipeline.
- VCORE outperforms baselines on math and code benchmarks, with particularly strong gains on lower-capacity models, improving both in-domain and out-of-domain performance.
- VCORE further strengthens RL fine-tuning by providing a more effective initialization, yielding higher post-RL performance.

## 2 Related Work

*Due to space constraints, we present a concise overview here; the complete related work is provided in Appendix B.*

**SFT for Reasoning.** Beyond chain-of-thought (CoT) prompting (Wei et al., 2022), supervised fine-tuning (SFT) on *long* CoT traces has emerged as a simple and effective way to imbue LLMs with slow, multi-step reasoning, often via distillation from stronger teachers or curated rationales (Team, 2025; Muennighoff et al., 2025; Xu et al., 2025; Labs, 2025). Compared to alternatives that require on-policy rollouts, long-CoT SFT is operationally lightweight, attains strong generalization, and frequently serves as an initialization for subsequent RL-based improvement (Yeo et al., 2025).

**Token Reweighting in SFT.** Some recent studies modify SFT by adopting token-level loss reweighting. Wu et al. (2025) reinterprets standard SFT as a policy-gradient-style update with an implicit  $1/\pi_\theta$  importance factor that over-weights low-probability tokens; it therefore introduces Dynamic Fine-Tuning (DFT), which rectifies the update by multiplying per-token loss by the model’s probability for the target token. Qin and Springenberg (2025) shows that SFT on curated/filtered data optimizes a lower bound to an RL objective; accordingly, it proposes Importance-weighted SFT (iw-SFT) which tightens that bound by importance-weighting the SFT log-likelihood relative to a reference or current policy.

Our method (VCORE) is also an SFT-stage token-level reweighting, yet it is *conceptually distinct* from DFT and iw-SFT. Whereas DFT and iw-SFT are *RL-motivated*, VCORE is *optimization-driven*: we formulate the choice of token weights as maximizing the first-order loss decrease of a single SGD step subject to a KL constraint. This yields a closed-form Gibbs weighting over tokens. We further introduce an explicit variance-control mechanism that matches the update variance of the reweighted supervision to that under uniform weighting, stabilizing training on long CoT sequences. Empirically, VCORE delivers stronger reasoning generalization over DFT and iw-SFT within the same SFT protocol.

## 3 Preliminaries

**Notations.** Let  $\mathcal{V}$  be a finite vocabulary, and let  $\mathcal{V}^*$  denote the set of all finite-length sequences over  $\mathcal{V}$ .

Each training instance  $(x, y)$  consists of an input prompt  $x$  and a target sequence  $y = (y_1, y_2, \dots) \in \mathcal{V}^*$  of length  $|y|$ , typically encoding a reasoning trajectory distilled from a teacher model via CoT supervision. The goal is to train a student model  $p_\theta$ , parameterized by  $\theta$ , to imitate these trajectories through next-token prediction. The language model  $p_\theta$  defines a conditional distribution over tokens, with the sequence likelihood factorized autoregressively as  $p_\theta(y|x) = \prod_{t=1}^{|y|} p_\theta(y_t|x, y_{<t})$ , where  $y_{<t} = (y_1, \dots, y_{t-1})$  denotes the prefix up to position  $t - 1$ . Each term is computed by applying a softmax to the output logits, yielding a next-token probability distribution.

**Naive CoT Supervision.** We train the model  $p_\theta$  by minimizing the expected per-token log-loss under the true data distribution  $\mathcal{P}$ , defined as  $\mathcal{L}(\theta) = \mathbb{E}_{(x,y) \sim \mathcal{P}} \left[ \frac{-\log p_\theta(y|x)}{|y|} \right]$ . In practice, we optimize this objective over a finite training dataset  $\mathcal{D}$  by iteratively sampling mini-batches  $\mathcal{B} \subset \mathcal{D}$  and applying gradient-based updates of the form  $\theta^+ \leftarrow \theta - \mathcal{T}(\nabla_\theta \hat{\mathcal{L}}_{\mathcal{B}}(\theta))$ , where  $\mathcal{T}$  denotes a generic update operator (e.g., SGD or Adam) that acts on the loss gradient and, when applicable, historical statistics. The mini-batch gradient takes the following form:

$$\begin{aligned} \nabla_\theta \hat{\mathcal{L}}_{\mathcal{B}}(\theta) &= \sum_{(x,y) \in \mathcal{B}} \left[ -\frac{1}{|y|} \nabla_\theta \log p_\theta(y|x) \right] \\ &\triangleq \sum_{(x,y) \in \mathcal{B}} \left[ \sum_{t \geq 1} \underbrace{(1/|y|)}_{\text{uniform}} \cdot \nabla_\theta \ell_t(\theta; x, y) \right]. \end{aligned}$$

Here,  $\ell_t(\theta; x, y) = -\log p_\theta(y_t | x, y_{<t})$  denotes the token-level prediction loss at position  $t$ . The resulting gradient update corresponds to a uniform average over all next-token prediction tasks, implicitly treating each token as equally informative and equally reliable. This choice is not arbitrary: under the autoregressive factorization, uniform weighting yields an unbiased estimator of the population loss gradient  $\nabla_\theta \mathcal{L}(\theta)$ , making it a natural default.

**Limitations of Uniform Weighting.** (1) *Not all tokens are equally worth learning from.* Uniform weighting spreads supervision evenly across positions, regardless of how much signal each token provides. Many next-token predictions are either trivially easy or hopelessly ambiguous—both yield gradients with little learning value. This misallocation wastes updates and slows convergence. (2)

*Spurious tokens corrupt learning signals.* Auto-distilled CoTs often contain hallucinated or misaligned tokens. Uniform weighting treats these as equally important, allowing noise to dominate gradients and impair generalization.

**Toward Adaptive Token Weighting.** A natural remedy to these limitations is to adapt token-wise supervision to the input and model state. Let  $q_t(x, y, \theta)$  denote a distribution over positions  $t = 1, \dots, |y|$ , conditioned on the input prompt  $x$ , target sequence  $y$  and current model parameters  $\theta$ . Unlike the uniform weighting, adaptive  $q$  concentrates learning on salient or uncertain positions while suppressing redundant or noisy ones. The reweighted gradient over a mini-batch  $\mathcal{B}$  becomes:

$$\nabla_\theta \hat{\mathcal{L}}_{\mathcal{B}}(\theta; q) = \sum_{(x,y) \in \mathcal{B}} \left[ \sum_{t \geq 1} \underbrace{q_t(x, y)}_{\text{adaptive}} \nabla_\theta \ell_t(\theta; x, y) \right]$$

with the update  $\theta^+(q) \leftarrow \theta - \mathcal{T}(\nabla_\theta \hat{\mathcal{L}}_{\mathcal{B}}(\theta; q))$ , where  $\mathcal{T}$  denotes a generic update operator.

**Problem: How Should We Choose  $q$ ?** If only a subset of tokens meaningfully contributes to learning, then uniform supervision fails to account for their differing impact. This raises a fundamental question of credit assignment: where should gradients go? The goal of adaptive weighting is to allocate supervision to the most impactful positions under the current model. This requires selecting  $q(t | x, y, \theta)$  based on local gradient utility, rather than adhering to a fixed prior. To avoid instability from overly sharp focus, we constrain  $q$  to remain close to uniform:

$$\min_q \mathcal{L}(\theta^+(q)) \quad \text{s.t.} \quad \text{KL}(q(\cdot | x, y, \theta) \| u) \leq \delta, \quad (1)$$

where  $u(t) = 1/|y|$  and  $\delta$  bounds deviation. This formulation balances targeted supervision with stability, enabling fine-grained, model- and input-aware token selection.

## 4 Method

### 4.1 Optimal Reweighting under SGD

In this subsection, we derive the optimal reweighting distribution  $q^*(t | x, y, \theta)$  that maximizes the loss decrease after a single-step SGD update. By applying a first-order Taylor approximation to the SFT objective, we fortunately derive a closed-form solution for the optimal token-wise weights, enhancing interpretability and enabling efficient algorithmic implementation.

**Step 1: First-order Taylor Expansion.** Consider an SGD step with learning rate  $\eta$ , using the reweighted gradient:  $\theta^+ = \theta - \eta \nabla_{\theta} \hat{\mathcal{L}}_{\mathcal{B}}(\theta; q)$ . For small  $\eta$ , the population loss at the updated parameters admits a first-order approximation:

$$\begin{aligned} \mathcal{L}(\theta^+) - \mathcal{L}(\theta) &= -\eta \langle \nabla \mathcal{L}(\theta), \nabla \hat{\mathcal{L}}_{\mathcal{B}}(\theta; q) \rangle + \mathcal{O}(\eta^2) \\ &= -\eta \sum_{(x,y) \in \mathcal{B}} \sum_{t \geq 1} \left[ q_t(x, y) \cdot s_t(x, y, \theta) \right] + \mathcal{O}(\eta^2) \end{aligned}$$

where we define the per-token gradient utility

$$s_t(x, y, \theta) \triangleq \langle \nabla_{\theta} \mathcal{L}(\theta), \nabla_{\theta} \ell_t(\theta; x, y) \rangle. \quad (2)$$

This quantity captures the alignment between the global descent direction and the gradient induced by supervising token  $y_t$ . Tokens with higher  $s_t$  contribute more to reducing the population loss and should be prioritized.

**Step 2: Optimal Adaptive Weighting.** Maximizing the descent in the first-order expansion reduces to a constrained optimization over the probability simplex  $\Delta$  at each training instance  $(x, y)$ :

$$\begin{aligned} \max_{q \in \Delta} \sum_{t \geq 1} q(t) \cdot s_t(x, y) \quad \text{s.t.} \quad \text{KL}(q \parallel u) \leq \delta \\ \implies q^*(t \mid x, y, \theta) = \frac{\exp(\tau s_t(x, y, \theta))}{\sum_{j \geq 1} \exp(\tau s_j(x, y, \theta))}, \end{aligned}$$

where  $\tau > 0$  is a temperature set by the constraint. This standard exponential tilting problem admits a unique *closed-form solution*: a Gibbs distribution over token-level gradient utilities. As  $\tau \rightarrow 0$ ,  $q^*$  recovers the uniform prior; as  $\tau \rightarrow \infty$ , it concentrates on the highest-utility tokens.

**Step 3: A One-Backward Trick for Estimating  $s_t$ .** Estimating the token utility  $s_t(x, y, \theta)$  in Equation (2) naively requires one backward pass per token, computationally infeasible for long sequences. We introduce a non-trivial and highly efficient solution: a *one-backward forward-mode probing trick* that recovers all  $s_t$  values using just a single backward step and one forward step for  $|y|$  token evaluations. Specifically, we draw an independent mini-batch  $\mathcal{B}' \sim \mathcal{P}$ , compute the descent direction  $\nabla_{\theta} \mathcal{L}_{\mathcal{B}'}(\theta; u)$  under uniform token weights, and measure the change in the token-wise loss after a small perturbation in that direction. This yields an unbiased estimator:

$$\begin{aligned} \lim_{\epsilon \rightarrow 0} \frac{\mathbb{E}_{\mathcal{B}'} [\ell_t(\theta; x, y) - \ell_t(\theta - \epsilon \nabla_{\theta} \mathcal{L}_{\mathcal{B}'}(\theta; u); x, y)]}{\epsilon} \\ = \langle \nabla_{\theta} \mathcal{L}(\theta), \nabla_{\theta} \ell_t(\theta; x, y) \rangle = s_t(x, y, \theta). \end{aligned}$$

This construction reduces the cost of estimating all  $s_t$  values from  $|y|$  backward passes to just one backward (to compute the descent direction from  $\mathcal{B}'$ ) and one forward pass (to evaluate perturbed token losses). It requires no second-order gradients, no backward hooks and no additional model queries. By leveraging the directional derivative structure of  $s_t$ , this probing trick makes adaptive token weighting both scalable and plug-and-play in standard training pipelines.

**Beyond Heuristics: Token Weighting as Optimization, Not Guesswork.** Some prior works rank tokens by confidence (Wu et al., 2025), importance (Qin and Springenberg, 2025) or by their estimated influence on outcome correctness (Lin et al., 2024), typically without leveraging training-time gradient information. In contrast, we take an optimization perspective and derive a closed-form optimal token-weighting distribution, moving beyond heuristic rules. Crucially, our method identifies the most impactful tokens directly from the training signal *without relying on teacher guidance or manually tuned thresholds*. A detailed theoretical analysis demonstrating its advantages over uniform weighting is provided in Appendix A.

## 4.2 VCORE: Variance-Controlled Optimization-based REweighting

We introduce **VCORE**, a Variance-Controlled Optimization-based REweighting framework for CoT supervision, which is shown in Algorithm 1. Building on Section 4.1, which presented the Gibbs-form reweighting  $q^*(t \mid x, y, \theta) \propto \exp(\tau s_t)$  and a one-backward token-utility estimation trick, we now add a variance-normalization mechanism to control the update variance at each training step. Concretely, we rescale the parameter update by an adaptive coefficient  $\alpha$  in order to align the variance of the reweighted supervision with that of the uniform supervision. This trick helps stabilize the training dynamics while allowing the learning signal to concentrate on informative tokens. Moreover, it introduces no extra architectural changes.

**Variance-Controlled Descent Scaling.** The Gibbs reweighting  $q^*(t \mid x, y, \theta) \propto \exp(\tau s_t)$  focuses learning on informative tokens, but it also changes the variability of the update. To make this explicit, we define the variance of the (per-batch) update under reweighted supervision *before* any scaling,  $\mathcal{V}_q \triangleq \text{Var}[\sum_t q_t s_t]$ , and the variance under uniform supervision by,  $\mathcal{V}_u \triangleq \text{Var}[\sum_t s_t / |y|]$ . We then introduce an adaptive coefficient  $\alpha$  to con-

---

**Algorithm 1** VCore Algorithm

---

**Require:** Dataset  $\mathcal{D} = \{(x, y)\}$ , model parameters  $\theta$ , learning rate  $\eta$ , temperature  $\tau$ , probing scale  $\epsilon$

**Ensure:** Updated parameters  $\theta^+$

- 1: **for** each batch  $\mathcal{B}$  in  $\mathcal{D}$  **do**
  - 2:   Draw a random batch  $\mathcal{B}'$  from  $\mathcal{D}$
  - 3:   Compute descent direction  $\nabla_{\theta} \mathcal{L}_{\mathcal{B}'}(\theta; u)$  under uniform token weights
  - 4:   Estimate token  $t$  score  $s_t(x, y, \theta)$  by  $\lim_{\epsilon \rightarrow 0} \frac{\mathbb{E}_{\mathcal{B}'}[\ell_t(\theta; x, y) - \ell_t(\theta - \epsilon \nabla_{\theta} \mathcal{L}_{\mathcal{B}'}(\theta; u); x, y)]}{\epsilon}$
  - 5:   Get weights distribution  $q^*(t \mid x, y, \theta)$  by  $\frac{\exp(\tau s_t(x, y, \theta))}{\sum_j \exp(\tau s_j(x, y, \theta))}$
  - 6:   Compute  $\alpha$  via a uniform/reweighted loss ratio
  - 7:    $\theta \leftarrow \theta - \eta \mathbb{E}_{(x, y), t \sim q^*} [\nabla_{\theta} (\alpha \ell_t(\theta; x, y))]$
  - 8: **end for**
  - 9: Get the final trained model parameters  $\theta^+ = \theta$
- 

trol the update magnitude, choosing  $\alpha$  to match the variance of reweighted updates with that of uniform weighting:

$$\alpha^2 \mathcal{V}_q = \text{Var} \left[ \alpha \sum_t q_t s_t \right] \approx \text{Var} \left[ \sum_t \frac{s_t}{|y|} \right] = \mathcal{V}_u.$$

This yields a choice:  $\alpha = \sqrt{\mathcal{V}_u / \mathcal{V}_q}$ . If the scaling coefficient  $\alpha$  is too small, gradient steps shrink and training slows down; if  $\alpha$  is too large, especially under sharply peaked weights, stochastic gradients become highly variable and convergence degrades.

*Intuition.* If token utilities  $s_t$  are uncorrelated with constant variance  $\sigma^2$ , then  $\mathcal{V}_u \approx \sigma^2 / |y|$ . Under a highly peaked Gibbs weighting (e.g.,  $q_t$  concentrates on one token),  $\mathcal{V}_q^{(0)} \approx \sigma^2$ , leading to  $\alpha = 1 / \sqrt{|y|}$ . Thus, when  $q$  is sharp or sequences are long, aggressive reweighting would amplify variance and  $\alpha$  must shrink to stabilize training; when  $q$  is balanced,  $\alpha \approx 1$  and the full descent step is recovered without loss of stability.

**Summary: A New Perspective on CoT Supervision.** (1) *Optimization-Derived Weighting.* Instead of heuristic reweighting, our framework derives  $q^*(t)$  as the unique solution to an optimization problem that maximizes population loss reduction under a single SGD step with a KL constraint. This yields a closed-form Gibbs distribution over token-level gradient utilities, grounding token supervision in first-order descent rather than heuristics. (2)

*Variance-Controlled Stabilization.* Although  $q^*$  is theoretically optimal for maximizing descent, its practical effectiveness depends on the stability of the updates it induces. Heavy-tailed utilities can cause the resulting weights to become sharply concentrated, amplifying gradient variance and destabilizing training. Our VCore framework addresses this by introducing a principled variance-controlled scaling coefficient  $\alpha$  that matches the variance of reweighted updates to that of uniform supervision. This preserves the adaptivity of Gibbs weighting while ensuring stable and efficient learning without heuristic clipping or ad-hoc thresholds.

## 5 Experiments

In this section, we investigate and answer the following research questions:

**RQ1:** Can VCore achieve better generalization than uniform and heuristic reweighting methods?

**RQ2:** Which parts of VCore are essential for improving reasoning and stability?

**RQ3:** What are the practical implications and limitations of VCore in CoT training?

### 5.1 Experimental Setups

**Models and Supervised Tasks.** We study long CoT supervision on two domains (math and coding), using Qwen3 models (4B, 8B, 32B) (Yang et al., 2025) and LLaMA3.1-8B-Instruct (AI@Meta, 2024). Domain-specific training data are curated from recent high-quality sources: OpenMathReasoning (Moshkov et al., 2025) and the C++ subset of OpenCodeReasoning (NVIDIA, 2025). We retain only CoT instances generated by DeepSeek-R1 (Guo et al., 2025a), applying automatic filtering to ensure trajectory correctness. For Qwen3 models, we sample 3.2k examples per domain and the average CoT length is 3155.01 (math) and 2861.25 (code); for LLaMA, we use 32k examples per domain and the average CoT length is 3007.79 (math) and 2805.43 (code). Both datasets ensure that all methods reach convergence. Further preprocessing details and CoT length statistics are provided in Appendix C.1.

**Evaluation Benchmarks.** We evaluate each model under both *in-domain* and *out-of-domain* generalization settings. **In-domain** benchmarks include the union of AIME 2024 and AIME 2025 (AIME) (Art of Problem Solving Community, 2025) and math subset of Olympiad-Bench (Olympiad) (He et al., 2024) for math,

Model	Method	Math				Code				Avg.	
		AIME	Olympiad	RBench	SGPQA-1k	LCB	OJBench	RBench	SGPQA-1k	ID	OOD
LLaMA3.1 8B Instruct	Original	3.33	17.80	21.76	18.60	9.67	1.29	21.76	18.60	8.02	<b>20.18</b>
	SFT	3.33	23.59	4.02	12.10	9.29	0.86	5.21	9.50	9.27	7.71
	DFT	3.33	18.84	6.22	13.90	0.00	0.00	3.47	5.10	5.54	7.17
	iw-SFT	5.00	22.55	2.83	11.60	9.00	0.86	3.38	6.60	<u>9.35</u>	6.10
	Random	1.67	22.55	7.95	15.10	6.16	3.02	4.75	8.50	8.35	9.07
	VCORE	6.67	25.96	3.56	12.80	10.33	2.58	6.22	15.80	<b>11.38</b>	<u>9.59</u>
Qwen3 4B	Original	38.33	60.09	23.13	29.10	22.56	3.02	23.13	29.10	31.00	26.12
	SFT	46.67	61.72	30.71	30.50	24.74	2.16	28.34	31.80	<u>33.82</u>	30.34
	DFT	35.00	62.91	34.00	32.40	26.45	5.60	30.16	33.40	32.49	<u>32.49</u>
	iw-SFT	40.00	62.31	28.79	31.00	24.83	2.59	29.34	32.60	32.43	30.43
	Random	38.33	60.98	31.35	30.50	25.69	0.86	29.34	31.30	31.46	30.62
	VCORE	48.33	66.17	34.64	34.30	25.97	3.88	30.25	32.30	<b>36.09</b>	<b>32.87</b>
Qwen3 8B	Original	43.33	60.09	24.86	34.30	25.21	4.31	24.86	34.30	33.24	29.58
	SFT	48.33	63.80	35.47	34.00	26.07	4.74	34.92	37.90	<u>35.74</u>	35.07
	DFT	41.67	65.13	38.67	37.90	29.38	6.03	35.10	37.50	35.55	<b>37.29</b>
	iw-SFT	45.00	61.72	35.37	34.60	27.20	5.60	35.47	35.90	34.88	<u>35.34</u>
	Random	43.33	61.57	35.47	36.20	25.59	4.31	33.46	35.90	33.70	35.26
	VCORE	45.00	64.84	35.01	36.80	28.63	4.74	33.82	35.40	<b>35.80</b>	35.26
Qwen3 32B	Original	43.33	64.99	41.13	43.90	27.30	5.60	41.13	43.90	35.30	42.52
	SFT	38.33	63.80	46.16	39.80	31.94	9.91	48.54	40.60	35.99	43.78
	DFT	43.33	67.66	54.48	46.50	37.82	10.34	49.91	47.40	<u>39.79</u>	<b>49.57</b>
	iw-SFT	40.00	63.65	43.33	39.40	31.94	6.90	45.43	42.40	35.62	42.64
	Random	40.00	62.31	45.80	40.30	35.55	9.91	47.71	45.50	36.94	44.83
	VCORE	51.67	68.55	49.91	45.00	35.45	9.48	45.70	43.10	<b>41.29</b>	<u>45.93</u>

Table 1: **Main Results.** Accuracy (%) of different methods on **in-domain** (AIME, Olympiad, LCB, OJBench) and **out-of-domain** (RBench, SGPQA-1k) benchmarks across Math and Code. Best and second-best are shown in **bold** and underline, respectively.

OJBench (**OJBench**) (Wang et al., 2025) and LiveCodeBench(v6) (**LCB**) (Jain et al., 2025) for coding. For **out-of-domain** evaluation, we use R-Bench-T (**RBench**) (Guo et al., 2025b) and a 1k-sample subset of SuperGPQA (**SGPQA-1k**) (Team et al., 2025b). We run inference using vLLM v1 (Kwon et al., 2023) with a maximum generation length of 8192 tokens. For all the benchmarks, we use greedy decoding and report Pass@1 as the metric. See Appendix C.3 for more detailed settings.

**Baselines.** We compare against two recent reweighting-based methods: **DFT** (Wu et al., 2025) and **iw-SFT** (Qin and Springenberg, 2025). We also include the **Original** model (without any fine-tuning) and standard **SFT**, which uses the full CoT supervision signal. In the **Random** baseline, 80% of CoT tokens are randomly dropped during training to simulate partial supervision.

**Implementation Details.** All the training experiments are conducted on four RTX PRO 6000 Blackwell GPUs using the LLaMA-Factory framework (Zheng et al., 2024), with a batch size of 32. We apply one epoch of LoRA fine-tuning using the AdamW optimizer with a cosine learning

rate schedule. The learning rate is set to  $2e-5$  for Qwen3 models and  $2e-4$  for LLaMA3.1-8B-Instruct. The LoRA rank is set to 8 for Qwen3 and 64 for LLaMA, with the LoRA alpha fixed at twice the rank in both cases. In the algorithmic setting of VCORE, for each parameter update we randomly select a batch  $B'$  ( $|B'| = 32$ ) from the training set. The hyperparameters  $\epsilon$  and  $\tau$  are tuned specifically for each model. Complete training configurations are provided in Appendix C.2.

## 5.2 Main Results (RQ1)

**Obs 1: VCORE achieves the highest average score<sup>2</sup> (31.03).** It outperforms SFT (28.97), DFT (29.99), iw-SFT (28.35) and Random (28.78) (see Table 1). VCORE achieves strong supervised reasoning while also preserving cross-domain generalization, mitigating the degradation often caused by long CoT supervision. Moreover, scaling from Qwen-8B to Qwen-32B amplifies gains over the baseline, increasing from +4.12 to +4.70, highlighting improved effectiveness with larger models.

<sup>2</sup>The average of the ID and OOD ‘‘Avg.’’ columns across the four models in Table 1.

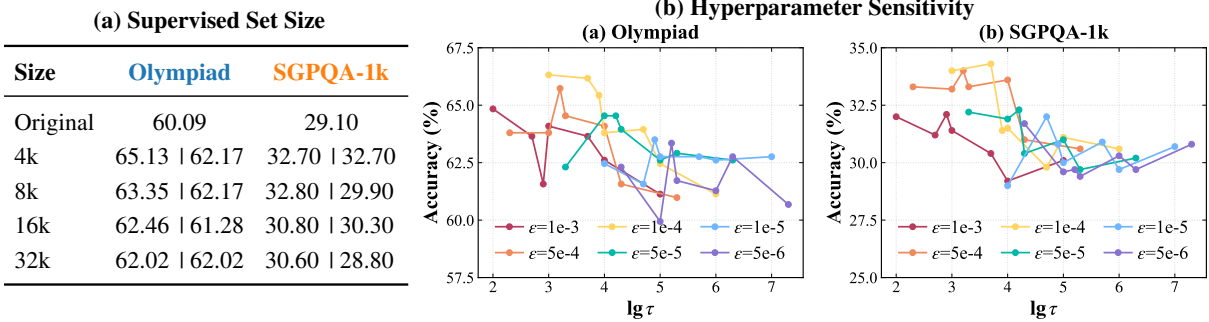


Figure 2: **Component Analysis and Ablation.** (a) Impact of supervised set size on in-domain (Olympiad) and out-of-domain (SGPQA-1k) accuracy for VCORE|DFT; (b) Hyperparameters: reweighting temperature  $\tau$  and probing scale  $\epsilon$ . All results use Qwen3-4B. Metrics are accuracy (%) on Olympiad (in-domain) and SGPQA-1k (out-of-domain).

**Obs 2: VCORE consistently improves performance on lower-capacity models.** VCORE does not always outperform the baselines and this may stem from the strength of the underlying SFT objective: VCORE reweights tokens by population loss, yielding limited gains when models already exhibit strong CoT reasoning or when CoT supervision is misaligned with the target task. Consequently, VCORE achieves larger gains on weaker models or more challenging datasets, where it better approximates DeepSeek-style supervision. As shown in Table 1, for LLaMA-Instruct, the scores increase from 5.54/7.17 (DFT) to 11.38/9.59 (VCORE), while for Qwen-4B, they rise from 32.49/32.49 to 36.09/32.87. Table 2 shows results for additional weaker models, with Qwen3-1.7B using the Qwen3-4B training setup and Mistral-7B-Instruct-v0.3 using the Llama-3.1-8B-Instruct setup from the main experiments.

Method	Qwen3-1.7B		Mistral-7B-Instruct	
	Olympiad	SGPQA-1k	Olympiad	SGPQA-1k
Original	53.41	21.10	3.41	5.10
SFT	51.34	17.50	9.50	7.80
DFT	51.78	<b>21.80</b>	4.45	3.90
VCORE	<b>55.64</b>	21.00	<b>9.94</b>	<b>8.70</b>

Table 2: Performance on Weaker Models

### 5.3 Component Analysis and Ablations (RQ2)

**Obs 3: VCORE outperforms DFT across training set scales (Figure 2(a)).** As the supervised set grows from 4k to 32k, VCORE consistently outperforms DFT on both Olympiad and SGPQA-1k, demonstrating robust generalization under increasing training set size. At larger scales, overall performance slightly declines as additional reasoning traces with diverse quality, styles, and implicit assumptions are aggregated, amplifying distribution mismatch and weakening the effective supervision

signal. Despite this shift, VCORE maintains a clear advantage and consistently improves over the base model across all dataset sizes.

**Obs 4: VCORE is robust to optimization-derived reweighting hyperparameters (Figure 2(b)).** We analyze two key hyperparameters in VCORE: the reweighting temperature  $\tau$ , which controls the shape of the supervision distribution, and the probing scale  $\epsilon$ , which governs token utility estimation accuracy. Across both Olympiad and SGPQA-1k with Qwen3-4B, VCORE remains stable over a wide range of  $\tau$  and  $\epsilon$ , exhibiting only mild performance variation. Performance peaks at moderate settings (e.g.,  $\epsilon \approx 1e-4$ ,  $\lg \tau \approx 4$ ), while extreme values degrade accuracy due to overly uniform or overly concentrated reweighting.

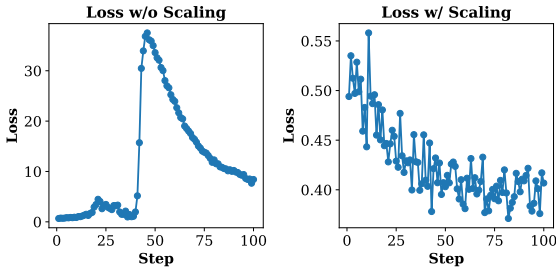


Figure 3: **Loss Scaling.** Loss curves of Qwen3-4B on the math domain with and without loss scaling ( $\epsilon = 1e-4$ ,  $\tau = 5e3$ ).

**Obs 5: Variance control is essential for stable optimization under sharp reweighting (Figure 3).** Under sharply peaked reweighting, the population loss frequently exhibits transient spikes during training before reconverging. This exposes a fundamental trade-off: while sharp reweighting improves supervision focus, it amplifies gradient variance and destabilizes optimization. VCORE addresses this issue by introducing an adaptive descent scaling factor that aligns update variance with uniform supervision, resulting in smooth, stable,

Method	bs=16	bs=64	lr=1e-5	lr=1e-4
SFT	40.28	40.99	39.98	39.08
DFT	40.71	42.17	41.76	38.08
VCORE	<b>42.48</b>	<b>42.67</b>	<b>42.84</b>	<b>41.87</b>

Table 3: Performance across batch sizes and learning rates.

and reproducible convergence. This confirms that variance control is not optional but essential to safely realize the benefits of optimization-aligned supervision.

**Obs 6: VCore is insensitive to learning rate and batch size (Table 3).** We vary the learning rate and batch size on Qwen3-4B using math-domain data under the same training setup as the main experiments. Models are evaluated on the same four mathematics reasoning benchmarks, reporting average accuracy. As shown in Table 3, VCore consistently achieves strong performance across hyperparameter settings.

#### 5.4 Discussions: Practical Implications and Limitations (RQ3)

Method	Olympiad		SGPQA-1k	
	Before RL	After RL	Before RL	After RL
DFT	62.91	65.13	32.40	28.30
VCORE	<b>66.17</b>	<b>67.51</b>	<b>34.30</b>	<b>33.90</b>

(a) Results on Qwen3-4B.

Method	Olympiad		SGPQA-1k	
	Before RL	After RL	Before RL	After RL
DFT	<b>65.13</b>	67.95	<b>37.90</b>	38.00
VCORE	64.84	<b>68.84</b>	36.80	<b>38.60</b>

(b) Results on Qwen3-8B.

Table 4: **RL Results.** Performance before and after RL testing on Olympiad and SGPQA-1k using DFT and VCore. We highlight the best performance in **bold**.

**Obs 7: VCore offers a more capable foundation model to support reasoning tasks in reinforcement learning.** To assess the downstream reasoning capabilities of different post-SFT models, we select the DFT and VCore variants of Qwen-4B and Qwen-8B and fine-tune them using GRPO (Shao et al., 2024) for 200 steps on a subset of the BigMath (Albalak et al., 2025) dataset. Evaluation is performed on Olympiad and SGPQA-1k, following the same setup as in Section 5.1. Further details of the RL setup are presented in Appendix D. As shown in Table 4, VCore achieves clear performance gains over DFT after RL training, despite starting from a slightly lower baseline. This suggests that VCore initialization may provide

a higher RL performance ceiling. One possible explanation is that DFT tends to reduce generation entropy, which can constrain policy exploration during RL and limit the discovery of improved reasoning trajectories, indicating an additional advantage of VCore for reasoning generalization.

Method	GSM8K	MATH500
Original	95.15	90.4
SFT	93.63 (-1.52)	92.2 (+1.8)
DFT	94.69 (-0.46)	92.6 (+2.2)
iw-SFT	93.18 (-1.97)	90.8 (+0.4)
Random	93.56 (-1.59)	92.4 (+2.0)
VCORE	94.16 (-0.99)	92.8 (+2.4)

Table 5: Performance comparison of Qwen3-8B with different SFT strategies on GSM8K and MATH500. Numbers in parentheses indicate the performance change relative to *Original*.

**Obs 8: Long CoT SFT methods such as VCore may exhibit performance degradation when the task is relatively simple.** To assess reasoning ability under different difficulty levels, we employ two mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2023). GSM8K consists of grade-school arithmetic problems requiring only short reasoning chains, whereas MATH500 contains competition-level problems demanding deeper multi-step reasoning. Using the finetuned Qwen-8B model under the same evaluation settings as Section 5.1, we observe from Table 5 that long CoT SFT slightly degrades performance on GSM8K but consistently improves results on MATH500. This indicates that long CoT supervision primarily benefits reasoning-intensive tasks. These trends align with prior findings (Stechly et al., 2025; Gema et al., 2025): (1) CoT reasoning is sensitive to prompt structure and may overcomplicate simple problems; (2) longer reasoning chains accumulate errors and increase susceptibility to minor arithmetic mistakes.

## 6 Conclusion

In this paper, we present an in-depth investigation into improving the reasoning capabilities of LLMs through long CoT supervised fine-tuning. We introduce VCore, a variance-controlled, optimization-based reweighting framework. Going beyond heuristic token-weighting methods, we formulate VCore as an optimization problem that identifies the optimal token importance distribution by maximizing expected loss descent under SGD. Experiments on mathematical and coding benchmarks demonstrate that VCore significantly enhances

reasoning performance, particularly on complex tasks. These results bridge the gap between heuristic SFT practices and optimization-theoretic principles, offering a principled path toward building more generalizable reasoning models.

## Limitations

Our study is subject to computational and time constraints, which restricted the training corpus to the OpenMathReasoning and OpenCodeReasoning datasets, both derived from long CoT annotations generated by DeepSeek. We have not yet explored more diverse datasets or long CoT data produced by other state-of-the-art reasoning models, which could potentially reveal different generalization behaviors. We consider this an important avenue for future research.

There is also a potential failure mode where reweighting overemphasizes spurious patterns that leak information about the final answer. While rare, this mode can amplify dataset artifacts or annotation bias. Addressing this may require integrating additional regularization (e.g., dropout masking, answer prefix control) into the utility computation.

## Acknowledgments

This work was supported by the National Key R&D Program of China (2025YFF0516900 & 2025YFF0516904), NSFC U25B2039 and National Natural Science Foundation of China (No. 62306179); NSFC (No. 12326608), the Hetao Shenzhen–Hong Kong Science and Technology Innovation Cooperation Zone Project (No. HZQSW-KCCYB-2024016).

## References

AI@Meta. 2024. [Llama 3 model card](#).

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. 2025. [Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models](#). *Preprint*, arXiv:2502.17387.

Art of Problem Solving Community. 2025. Aime problems and solutions. [https://artofproblemsolving.com/wiki/index.php/AIME\\_Problems\\_and\\_Solutions](https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions).

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human

preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, and 1 others. 2025b. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*.

Daewon Choi, Jimin Lee, Jihoon Tack, Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Bhavana Ganesh, Jinwoo Shin, Aram Galstyan, and Sra-van Babu Bodapati. 2025. Think clearly: Improving reasoning via redundant token pruning. *arXiv preprint arXiv:2507.08806*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, Jacob Goldman-Wetzler, Kit Fraser-Taliente, Henry Sleight, Linda Petrini, Julian Michael, Beatrice Alex, Pasquale Minervini, Yanda Chen, Joe Benton, and Ethan Perez. 2025. [Inverse scaling in test-time compute](#). *Preprint*, arXiv:2507.14417.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Meng-Hao Guo, Jiajun Xu, Yi Zhang, Jiayi Song, Haoyang Peng, Yi-Xuan Deng, Xinzhi Dong, Kiyohiro Nakayama, Zhengyang Geng, Chen Wang, and 1 others. 2025b. R-bench: Graduate-level multidisciplinary benchmarks for llm & mllm complex reasoning evaluation. *arXiv preprint arXiv:2505.02018*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. [Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems](#). *Preprint*, arXiv:2402.14008.

- Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.
- Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. 2025. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. [Live-codebench: Holistic and contamination free evaluation of large language models for code](#). In *The Thirteenth International Conference on Learning Representations*.
- Kaixuan Ji, Jiafan He, and Quanquan Gu. 2024. Reinforcement learning from human feedback with active queries. *arXiv preprint arXiv:2402.09401*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Bespoke Labs. 2025. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. [www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation](http://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation). Accessed: 2025-01-22.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.
- Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. 2025. Compressing chain-of-thought in llms via step entropy. *arXiv preprint arXiv:2508.03346*.
- Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. 2023. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Lin, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. 2024. Critical tokens matter: Token-level contrastive estimation enhances llm’s reasoning capability. *arXiv preprint arXiv:2411.19943*.
- Elita Lobo, Chirag Agarwal, and Himabindu Lakkaraju. 2025. [On the impact of fine-tuning on chain-of-thought reasoning](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11679–11698, Albuquerque, New Mexico. Association for Computational Linguistics.
- Renjie Luo, Jiaxi Li, Chen Huang, and Wei Lu. 2025. [Through the valley: Path to effective long cot training for small language models](#). *Preprint*, arXiv:2506.07712.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.
- Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. 2025. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.
- NVIDIA. 2025. [Opencodereasoning-2](https://huggingface.co/datasets/nvidia/OpenCodeReasoning-2). <https://huggingface.co/datasets/nvidia/OpenCodeReasoning-2>. Accessed: 2025-08-03.
- 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.
- Chongli Qin and Jost Tobias Springenberg. 2025. Supervised fine tuning on curated data is reinforcement learning (and can be improved). *arXiv preprint arXiv:2507.12856*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2025. Chain of thoughtlessness? an analysis of cot in planning. *Preprint*, arXiv:2405.04776.
- Kimi Team, Angang Du, Bofei Gao, Bawei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025a. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- M-A-P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, Kang Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixing Deng, Shuyue Guo, Shian Jia, Sichao Jiang, Yiyao Liao, Rui Li, Qinrui Li, and 76 others. 2025b. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *Preprint*, arXiv:2502.14739.
- NovaSky Team. 2025. Sky-t1: Train your own o1 preview model within \$450. <https://novasky-ai.github.io/posts/sky-t1>. Accessed: 2025-01-09.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Zhexu Wang, Yiping Liu, Yejie Wang, Wenyang He, Bofei Gao, Muxi Diao, Yanxu Chen, Kelin Fu, Flood Sung, Zhilin Yang, Tianyu Liu, and Weiran Xu. 2025. Ojbench: A competition level code benchmark for large language models. *Preprint*, arXiv:2506.16395.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilya Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*.
- Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, and 1 others. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.
- Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.
- Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. 2025. On the generalization of sft: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*.
- Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*.
- Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, and 1 others. 2025. Redstar: Does scaling long-cot data unlock better slow-reasoning systems? *arXiv preprint arXiv:2501.11284*.
- 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 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.
- Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, and 1 others. 2025a. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingen Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, and 1 others. 2023. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyang Luo, Zhangchi Feng, and Yongqiang Ma. 2024. [Llamafactory: Unified efficient fine-tuning of 100+ language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. 2024. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances in Neural Information Processing Systems*, 37:123846–123910.

## A Theory

The theorem below shows that the Gibbs-form distribution  $q^*$  used in our method achieves a strictly larger first-order loss decrease than uniform weighting whenever token-level gradient utilities are not all identical. Uniform sampling is optimal only in the degenerate case where every token contributes exactly the same utility—an unrealistic setting for chain-of-thought supervision, where informative and uninformative steps naturally coexist. Thus, whenever utility varies across tokens (the norm in CoT data), adaptive weighting is strictly better.

### Theorem 1 (Strictly improves over uniform)

Fix  $(x, y)$  and  $\theta$ , and let  $u(t) = 1/|y|$ . For any  $\delta > 0$ , let  $q^*(\cdot | x, y, \theta)$  be the solution to

$$\max_{q \in \Delta} \sum_t q(t) s_t(x, y, \theta) \quad \text{s.t.} \quad \text{KL}(q \| u) \leq \delta,$$

i.e., the Gibbs-form  $q^*(t) \propto u(t) \exp(\tau s_t)$  with  $\tau > 0$  chosen to satisfy the KL constraint. If  $\{s_t\}_t$  are not all equal, then

$$\sum_t q^*(t) s_t > \sum_t u(t) s_t.$$

Consequently, in the first-order loss expansion,

$$\begin{aligned} \mathcal{L}(\theta^+) - \mathcal{L}(\theta) &= -\eta \sum_{(x,y) \in \mathcal{B}} \sum_t q(t) s_t(x, y, \theta) + O(\eta^2), \end{aligned}$$

the update with  $q^*$  yields a strictly larger decrease than the update with  $u$  for any mini-batch containing at least one instance with non-constant  $\{s_t\}$ . Equality holds only if  $\delta = 0$  or all  $s_t$  are identical.

**Proof 1** For  $\tau \geq 0$  define the tilted family

$$q_\tau(t) \triangleq \frac{u(t) e^{\tau s_t}}{Z(\tau)}, \quad Z(\tau) \triangleq \sum_j u(j) e^{\tau s_j},$$

so  $q_0 = u$  and  $q_\tau$  matches the Gibbs form. Let  $\phi(\tau) \triangleq \log Z(\tau)$ . Standard calculations give

$$\phi'(\tau) = \sum_t q_\tau(t) s_t = \mathbb{E}_{q_\tau}[s_t],$$

$$\phi''(\tau) = \text{Var}_{q_\tau}(s_t) \geq 0,$$

with strict inequality for all  $\tau$  whenever  $\{s_t\}$  are not all equal. Hence  $\phi'(\tau)$  is strictly increasing on  $(0, \infty)$  and

$$\begin{aligned}
& \mathbb{E}_{q_\tau}[s_t] - \mathbb{E}_u[s_t] \\
&= \phi'(\tau) - \phi'(0) \\
&= \int_0^\tau \phi''(t) dt \\
&= \int_0^\tau \text{Var}_{q_t}(s_t) dt > 0, \quad \text{for } \tau > 0. \quad (3)
\end{aligned}$$

Next, define  $\Psi(\tau) \triangleq \text{KL}(q_\tau \| u)$ . Using  $\text{KL}(q_\tau \| u) = \tau \mathbb{E}_{q_\tau}[s_t] - \phi(\tau)$  and  $\phi'(\tau) = \mathbb{E}_{q_\tau}[s_t]$ , we obtain

$$\Psi'(\tau) = \tau \frac{d}{d\tau} \mathbb{E}_{q_\tau}[s_t] = \tau \text{Var}_{q_\tau}(s_t) \geq 0,$$

which is  $> 0$  for  $\tau > 0$  when  $\{s_t\}$  are not all equal. Thus  $\Psi$  is strictly increasing on  $(0, \infty)$ , implying that for every  $\delta > 0$  there is a unique  $\tau^* > 0$  with  $\Psi(\tau^*) = \delta$ , and the KL-constrained optimum is  $q^* = q_{\tau^*}$ .

Combining uniqueness with (3) at  $\tau = \tau^*$  gives  $\sum_t q^*(t) s_t > \sum_t u(t) s_t$  whenever  $\{s_t\}$  are not all equal. Substituting into the first-order loss expansion yields the claimed strict improvement for any mini-batch containing at least one such instance. If  $\delta = 0$  or all  $s_t$  are identical, then  $\tau^* = 0$  and  $q^* = u$ , hence equality.

## B Related Work

**The rise of test-time scaling and long CoT.** Recent advances in LLMs have highlighted the importance of *test-time scaling* (Snell et al., 2024; Welleck et al., 2024; Zhang et al., 2025a), i.e., improving reasoning performance by allocating more inference-time compute rather than solely scaling model size or training resources. A practical instance of this idea is *long chain-of-thought* (*long CoT*): LLMs allocate more compute to generate longer reasoning chains at inference. Recent LLMs—OpenAI-o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025a), Kimi k1.5 (Team et al., 2025a)—pursue this direction by scaling CoT lengths and show strong reasoning performance on tasks such as challenging math problem solving and code generation benchmarks. Compared with short CoT, long CoT enables deeper reasoning, more extensive exploration, and feasible reflection (Chen et al., 2025a), thereby supporting more complex reasoning tasks.

**Enhancing LLM reasoning ability.** While CoT prompting (Wei et al., 2022) can improve reasoning

by explicitly encouraging models to articulate intermediate steps in natural language, it still struggles on complex reasoning problems. Consequently, recent work has focused on strengthening models’ intrinsic reasoning via post-training or test-time methods:

- **RL for reasoning.** Reinforcement learning has emerged as an effective post-training stage for eliciting multi-step reasoning in LLMs (Ouyang et al., 2022; Lightman et al., 2023; Guo et al., 2025a; Wen et al., 2025; Hou et al., 2025). On the algorithmic side, most work adopts policy-gradient methods such as REINFORCE (Williams, 1992), PPO (Schulman et al., 2017) and LLM-tailored variants such as ReMax and (Li et al., 2023), GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), etc. In parallel, some preference optimization (PO) methods (e.g., DPO (Rafailov et al., 2023), KTO (Ethayarajh et al., 2024), IPO (Azar et al., 2024), CPO (Xu et al., 2024)) optimize supervised objectives built from pairwise comparisons or binary accept/reject signals, avoiding on-policy rollouts. Orthogonal to the optimization algorithm, we categorize methods by reward source: (1) *Outcome-reward* RL, which directly optimizes final answers; a prominent subfamily is RL with verifiable rewards (RLVR) for math/coding, where unit tests or checkers define the reward (Guo et al., 2025a; Lambert et al., 2024); and (2) *Process-reward* RL, which performs step-level credit assignment via process reward models (PRMs) to shape the reasoning trajectory and improve reasoning faithfulness (Lightman et al., 2023; Wang et al., 2023; Zhang et al., 2025b).
- **SFT for reasoning:** However, adopting RL may sometimes be cumbersome—sensitive to hyperparameters (Henderson et al., 2018; Zheng et al., 2023), resource-intensive (Schulman et al., 2017), and costly in terms of training data collection (Ji et al., 2024; Wang et al., 2023). In comparison, a more straightforward way for enhancing reasoning ability is to distill long reasoning traces into LLMs via SFT. Recent works show that training on curated long CoT data—either distilled from stronger teachers or manually constructed—can endow models with robust slow-thinking behaviors (Team, 2025; Muennighoff et al., 2025; Xu

et al., 2025; Labs, 2025; Ye et al., 2025). Compared to SFT on short CoT, SFT on long CoT offers some practical benefits: a higher performance ceiling, better generalization, and larger downstream gains when used to initialize RL (Yeo et al., 2025).

- **Test-time methods:** A complementary line of work improves reasoning *during inference*. Approaches roughly fall into categories, including but not limited to: (i) prompting-based strategies that encourage stepwise thinking or decomposition—zero-/few-shot chain-of-thought (Wei et al., 2022), least-to-most prompting (Zhou et al., 2022), and ReAct-style reasoning–acting (Yao et al., 2023b); (ii) sampling-and-aggregation schemes that draw multiple rationales and then vote or rerank, e.g., self-consistency (Wang et al., 2022); (iii) search/planning over intermediate states, such as tree-structured exploration (Yao et al., 2023a); and (iv) self-reflection and debate that iteratively critique and revise candidate chains (Madaan et al., 2023). In practice, such test-time strategies are often combined and are complementary to SFT and RL for eliciting multi-step reasoning.

We remark that our method falls into the *SFT for reasoning* category and our work focuses on modifying the SFT phase itself to endow models with stronger generalization ability on reasoning tasks.

**Token-Level Reweighting in SFT** To retain the simplicity of SFT yet benefit from RL-induced reasoning improvements, recent studies modify the SFT objective to narrow its gap with RL. Specifically, Dynamic Fine-Tuning (DFT) (Wu et al., 2025) and importance-weighted SFT (iw-SFT) (Qin and Springenberg, 2025) pursue this via token-level loss reweighting. DFT reinterprets standard SFT as a biased policy update that over-concentrates on low-probability tokens; accordingly, it neutralizes that bias by rescaling the token-level loss. iw-SFT shows that SFT on curated/filtered data optimizes a lower bound to an RL objective; accordingly, it tightens that bound with explicit importance weights relative to a reference or current policy. Since our method can be regarded as a hard/sparse version of token-level loss reweighting, we include DFT and iw-SFT as our baselines.

## C Experimental Details of Main Results

### C.1 Dataset Curation for supervised tasks

For CoT supervised fine-tuning, we use two datasets: OpenMathReasoning<sup>3</sup> and OpenCodeReasoning<sup>4</sup>, which correspond to the mathematics and coding domains, respectively. We summarize the details of sampling and processing procedure as follows:

**OpenMathReasoning.** From the 3.2M samples in the cot split, we extract a subset that satisfies the following conditions: `problem_type = "has_answer_extracted"` and `generation_model = "DeepSeek-R1"`. We then use `math_verify`<sup>5</sup> to rigorously check whether the answers in the generated CoT content are equivalent to the expected answers. Based on this verified subset, we randomly sample 3,200 examples for training the Qwen3 series and 32,000 examples for training LLaMA3.1-8B-Instruct. To adapt the data for CoT training, we augment the original questions with new prompts designed for CoT reasoning. The specific format is as follows:

#### Math CoT Training Data Template

You are a helpful and accurate assistant for solving the following math problem:  
{ORIGINAL QUESTION}

Please reason step by step, and put the correct answer within `\boxed{ }` at last.

**OpenCodeReasoning.** From the 942K samples in the cpp split, we extract the subset that satisfies the condition `judgement = "right"`. Based on this subset, we randomly sample 3,200 examples for training the Qwen3 series and 32,000 examples for training LLaMA3.1-8B-Instruct. For the training data templates, we take inspiration from the prompt templates in `open-r1/codeforces`<sup>6</sup> and design our own training templates as follows:

<sup>3</sup>Hugging Face: OpenMathReasoning

<sup>4</sup>Hugging Face: OpenCodeReasoning-2

<sup>5</sup><https://github.com/huggingface/Math-Verify>

<sup>6</sup><https://huggingface.co/datasets/open-r1/codeforces>

Hyperparameter	Qwen3	LLaMA3.1-8B-Instruct
batch size	32	32
learning rate	2e-5	2e-4
training steps	100	1000
LoRA target	all	all
LoRA rank / alpha	8 / 16	64 / 128
LoRA dropout	0.10	0.05
lr schedule	cosine	cosine
warmup ratio	0.10	0.05
optimizer	AdamW	AdamW
seed	42	42
data type	bf16	bf16
cutoff length	16384	16384

Table 6: Hyperparameter settings for Qwen3 and LLaMA3.1-8B-Instruct.

### Code CoT Training Data Template

You are an expert competitive programmer. You will be given a problem statement, test case constraints, and example test inputs and outputs.

Please reason step by step about the solution (that must respect memory and time limits), then provide a complete implementation in c++17.

Your solution must read input from standard input (cin) and write output to standard output (cout).

Do not include any debug prints or additional output.

Put your final solution within a single code block:

```
```cpp
<your code here>
```
```

# Problem  
[{ORIGINAL QUESTION}](#)

Now solve the problem and return the code.

## C.2 Training Details

All baselines and our method are implemented on top of the LLaMA-Factory<sup>7</sup> framework. Most hyperparameters are shared across these methods, as summarized in Table 6.

<sup>7</sup><https://github.com/hiyouga/LLaMA-Factory>

**Hyperparameters of VCore** As shown in Figure 2, we conduct a small-scale grid search over  $\epsilon \in \{1e-4, 1e-5\}$  and  $\tau \cdot \epsilon \in \{0.5, 0.8\}$ , resulting in four configurations. We train each model under these settings and report the best-performing one in Table 7.

| Model                       | Math       |                       | Code       |                       |
|-----------------------------|------------|-----------------------|------------|-----------------------|
|                             | $\epsilon$ | $\tau \cdot \epsilon$ | $\epsilon$ | $\tau \cdot \epsilon$ |
| <b>LLaMA3.1-8B Instruct</b> | 1e-5       | 0.8                   | 1e-5       | 0.8                   |
| <b>Qwen3-4B</b>             | 1e-4       | 0.5                   | 1e-4       | 0.8                   |
| <b>Qwen3-8B</b>             | 1e-5       | 0.5                   | 1e-5       | 0.5                   |
| <b>Qwen3-32B</b>            | 1e-4       | 0.8                   | 1e-4       | 0.8                   |

Table 7: Selected hyperparameters ( $\epsilon, \tau \cdot \epsilon$ ) for each model and domain.

**The implementation details of Method Random.** In the **Random** baseline method, we retain only 20% of the original supervision tokens in total. The supervision on the final answer tokens (those inside `\boxed{ . . . }`) is always preserved. Once these tokens are fixed, we randomly sample from the remaining supervision tokens such that the overall proportion of preserved tokens (answer plus non-answer) amounts to 20%. All other tokens are excluded from the loss.

The purpose of designing this algorithm is to investigate the effect of sparse supervision on SFT. The method essentially performs a discrete binary weighting of supervision tokens, thereby allowing us to ablate and examine how reducing the amount of supervision signal influences the training dynamics and overall performance.

## C.3 Evaluation Details

We select two groups of benchmarks, each consisting of two domain-specific and two comprehensive tasks, resulting in a total of six benchmarks to thoroughly evaluate the generalization ability of different methods. Detailed information for each benchmark is provided in Table 8.

Specifically, for computational efficiency, we adopt a 1,000-sample i.i.d. subset from the SuperGPQA, referred to as SGPQA-1k. We ensure that its distribution of different disciplines remains consistent with the original dataset. Figure 4 illustrates the distribution before and after downsampling.

For OJBench, evaluation is performed using its original problem format. For other non-multiple-choice benchmarks, we adopt the same instruction templates as used in the corresponding training domain during inference. For multiple-choice bench-

| Benchmark | Size | Year         | Task Type          | Metric | Subset              | Source   |
|-----------|------|--------------|--------------------|--------|---------------------|--|
| AIME      | 60   | 2024<br>2025 | Open-ended QA      | Acc@1  | Full                | AoPS 2024 I AoPS 2024 II<br>AoPS 2025 I AoPS 2025 II |
| Olympiad  | 674  | 2024         | Open-ended QA      | Acc@1  | OE_TO_maths_en_COMP | HF (OlympiadBench)                                   |
| LCB (v6)  | 1055 | 2024         | Code generation    | Pass@1 | release_v6          | HF (LiveCodeBench)                                   |
| OJBench   | 232  | 2025         | Code generation    | Pass@1 | Full                | HF (OJBench)   |
| RBench    | 1094 | 2025         | Multiple-choice QA | Acc@1  | rbench-t_en         | HF (R-Bench)   |
| SGPQA-1k  | 1000 | 2025         | Multiple-choice QA | Acc@1  | 1k samples          | HF (SuperGPQA)                                       |

Table 8: Benchmarks used for evaluation.

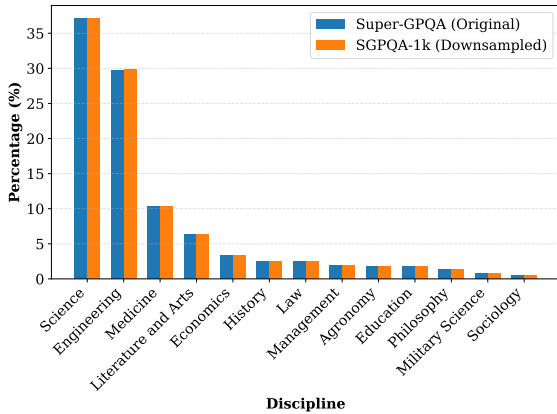


Figure 4: Construction of GPQA-1k.

marks, including the two comprehensive benchmarks, we use the template provided below. All evaluations are conducted on  $8 \times$  NVIDIA RTX PRO 6000 Blackwell GPUs.

We evaluate the models using the vLLM<sup>8</sup> framework. The detailed hyperparameter settings for inference are provided in Table 9.

We extract the predicted answers from model outputs using regular expression matching on the `\boxed{}` format. For AIME and Olympiad, we further apply `math_verify` to ensure answer equivalence. For OJBench<sup>9</sup>, LiveCodeBench<sup>10</sup>, we adopt the official repositories for evaluation. For two comprehensive benchmarks, we directly verify answers using exact match.

<sup>8</sup><https://github.com/vllm-project/vllm/releases/tag/v0.9.2>

<sup>9</sup><https://github.com/He-Ren/OJBench>

<sup>10</sup><https://github.com/LiveCodeBench/LiveCodeBench>

### Multiple-Choice Benchmark Evaluation Template

You are a helpful and accurate assistant for solving the following multiple-choice question:

{ORIGINAL QUESTION}

Options are:

(A): {OPTION A}

(B): {OPTION B}

...

Please reason step by step, and put the correct answer within `\boxed{}` at last.

| Hyperparameter  | Value |
|-----------------|-------|
| temperature     | 0     |
| top_p           | 1.0   |
| top_k           | -1    |
| batch_size      | 512   |
| VLLM_USE_V1     | True  |
| seed            | 42    |
| enable_thinking | True  |
| max_new_tokens  | 8192  |

Table 9: Inference hyperparameters used in evaluation.

## D Experimental Details of Reinforcement Learning

We conduct the reinforcement learning experiments using the VERL<sup>11</sup> framework with the GRPO(Shao et al., 2024) algorithm. For training, we randomly sample 16,800 examples from the BigMath<sup>12</sup> dataset, which is the largest open-source dataset of high-quality mathematical problems, curated specifically for RL training in LLMs. We per-

<sup>11</sup><https://github.com/volcengine/verl>

<sup>12</sup><https://huggingface.co/datasets/SynthLabsAI/Big-Math-RL-Verified>

form additional full-parameter RL training on the Qwen3-4B and Qwen3-8B models obtained from the main experiments. To ensure comparability, we use identical hyperparameter configurations across model sizes and initialization strategies (DFT and VCORE). The detailed hyperparameter settings are provided in Table 10.

| Hyperparameter               | Value      |
|------------------------------|------------|
| algorithm                    | GRPO       |
| train_batch_size             | 128        |
| max_prompt_length            | 4096       |
| max_response_length          | 8192       |
| total_epochs                 | 2          |
| n_gpus_per_node              | 8          |
| learning rate                | 5e-6       |
| lr_warmup_steps_ratio        | 0.1        |
| warmup_style                 | cosine     |
| ppo_mini_batch_size          | 32         |
| ppo_micro_batch_size_per_gpu | 2          |
| entropy_coeff                | 0.001      |
| use_kl_loss                  | True       |
| kl_loss_coef                 | 0.02       |
| kl_loss_type                 | low_var_kl |
| use_kl_in_reward             | False      |
| rollout.n                    | 4          |
| rollout.max_model_len        | 8192       |

Table 10: Hyperparameters used in RL training.

## E Experimental Details of GSM8K and MATH500 Evaluation

We evaluate our models on the test split of GSM8K<sup>13</sup> and MATH500<sup>14</sup>. The evaluation is conducted using vLLM, and all other testing configurations remain identical to those described in Table 9.

## F Comparison with RL Approaches

We compare our method against the two most widely recognized RL approaches in the community (Shao et al., 2024; Yu et al., 2025). We train Qwen3-4B on the same math datasets used in our main results. For GRPO, we strictly follow the RL hyperparameters in Section 5.4 reported in the paper. Since the dataset in this setting is relatively small, we disable the filtering mechanism for DAPO. We set clip\_ratio\_low=0.2 and clip\_ratio\_high=0.28, while keeping all other RL-related hyperparameters identical to those used in GRPO.

<sup>13</sup><https://huggingface.co/datasets/openai/gsm8k>

<sup>14</sup><https://github.com/openai/prm800k>

As shown in the Table 11, our method consistently outperforms these baselines, demonstrating its effectiveness.

| Method | AIME         | Olympiad     | RBench       | SGPQA-1k     |
|--------|--------------|--------------|--------------|--------------|
| GRPO   | 40.00        | 63.80        | 25.05        | 32.10        |
| DAPO   | 43.33        | 65.88        | 28.61        | 34.10        |
| VCORE  | <b>48.33</b> | <b>66.17</b> | <b>34.64</b> | <b>34.30</b> |

Table 11: Comparing VCORE with RL methods (GRPO, DAPO).

## G Computational Overhead Analysis

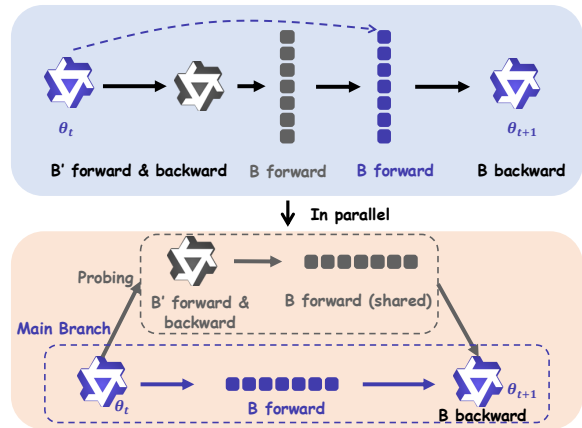


Figure 5: Computational Overhead Analysis

The primary computational overhead in VCORE stems from the additional forward and backward passes on the batch  $B'$ , which are required to estimate the population loss. As illustrated in Figure 5, the total computational cost of VCORE under parallel settings ( $VCORE_{parallel}$ ) comprises operations on both  $B'$  and  $B$  (i.e.,  $B' + B$  forward/backward passes), whereas SFT involves only  $B$ . Consequently, in the regime where  $|B'| \ll |B|$ , the additional overhead incurred by VCORE becomes negligible, rendering its efficiency comparable to that of standard SFT.

Table 12 presents the average wall-clock time per training step, along with the additional computational cost incurred by the  $B'$  branch. We compare VCORE against vanilla SFT across different sizes of  $B'$ . All experiments are conducted on 4 NVIDIA RTX 5880 GPUs with the same setting of Qwen-4B on math domain in the main results.

| Method                    | $ \mathcal{B}' $ | Per Step (s)  | Performance(Avg.) |
|---------------------------|------------------|---------------|-------------------|
| SFT                       | –                | 18.30         | 42.40             |
| VCORE <sub>8 GPUs</sub>   | 4                | 26.36 (+44%)  | 46.35             |
| VCORE <sub>parallel</sub> | 4                | 32.12 (+75%)  | 46.35             |
| VCORE <sub>parallel</sub> | 8                | 35.32 (+93%)  | 45.07             |
| VCORE <sub>parallel</sub> | 16               | 40.74 (+123%) | 46.79             |
| VCORE <sub>parallel</sub> | 32               | 52.91 (+189%) | 45.86             |
| VCORE                     | 32               | 57.54 (+215%) | 45.86             |

Table 12: Wall-clock time per training step and additional computational overhead. VCORE<sub>parallel</sub> computes  $\ell_t(\theta; x, y)$  in a parallel branch process on the same 4 GPUs, while VCORE<sub>8 GPUs</sub> runs the branch process on another 4 GPUs, both of which eliminate one forward pass in the main training process and reduce the overall step time.

## H Optimizer Scope and Robustness

### H.1 From the SGD Derivation to AdamW Implementation

We clarify that the derivation in Sec 4.1 is exact for a single-step SGD update and is presented in that form for clarity of exposition. The same reasoning admits a local extension to a broad class of coordinate-wise first-order optimizer updates of the form

$$\theta^+(q) = \theta - \eta d_k(g_{\mathcal{B}}(q)), \quad g_{\mathcal{B}}(q) \triangleq \nabla_{\theta} \hat{\mathcal{L}}_{\mathcal{B}}(\theta; q),$$

where  $d_k(\cdot)$  is an update map at iteration  $k$ , parametrized by optimizer state (e.g., momentum and second-moment buffers) carried over from previous iterations. SGD corresponds to  $d_k(g) = g$ , while Adam/AdamW corresponds to a preconditioned first-order direction. For AdamW, the decoupled weight-decay term is additive and independent of  $q$ . Since  $q$  is chosen to maximize the first-order decrease of the loss, such  $q$ -independent terms do not affect the KL-constrained optimization. We therefore present the Adam/AdamW case as a local first-order extension of the SGD analysis, not as an exact optimizer-agnostic optimality result. The algorithmic pipeline itself remains unchanged across optimizers.

Throughout this section, we assume that the update map  $d_k$  acts coordinate-wise on the gradient and is  $C^1$  in its gradient argument at  $g_u$ , so that its Jacobian (defined below) is diagonal and therefore symmetric. We also assume bounded token gradients,  $\sup_{t,x,y} \|\nabla_{\theta} \ell_t(\theta; x, y)\|_2 \leq G$ , as is standard in local first-order analyses.

**Step 1: First-order loss expansion.** For small  $\eta$ , a first-order Taylor expansion of the population

loss yields

$$\begin{aligned} L(\theta^+(q)) - L(\theta) \\ = -\eta \langle \nabla L(\theta), d_k(g_{\mathcal{B}}(q)) \rangle + O(\eta^2). \end{aligned}$$

**Step 2: KL-constrained linear objective.** Let  $u$  denote the uniform token weighting and define

$$g_u \triangleq g_{\mathcal{B}}(u), \quad \Delta g(q) \triangleq g_{\mathcal{B}}(q) - g_u.$$

Under the KL constraint  $\text{KL}(q||u) \leq \delta$ ,  $q$  remains close to uniform. Because

$$\begin{aligned} \Delta g(q) \\ = \sum_{(x,y) \in \mathcal{B}} \sum_t (q_t(x,y) - u_t(x,y)) \nabla_{\theta} \ell_t(\theta; x, y), \end{aligned}$$

the bounded-gradient assumption together with Pinsker’s inequality implies

$$\|\Delta g(q)\|_2 = O(\sqrt{\delta}),$$

where the implicit constant absorbs the batch size and the gradient bound  $G$ . Under the smoothness assumption above, a first-order linearization of  $d_k$  around  $g_u$  gives

$$d_k(g_{\mathcal{B}}(q)) = d_k(g_u) + J_k \Delta g(q) + O(\|\Delta g(q)\|_2^2),$$

where the Jacobian is evaluated at the uniform-weight gradient,

$$J_k \triangleq \left. \frac{\partial d_k(g)}{\partial g} \right|_{g=g_u} \in \mathbb{R}^{p \times p}.$$

Substituting this into the loss expansion yields

$$\begin{aligned} L(\theta^+(q)) - L(\theta) \\ = C(\theta) - \eta \langle \nabla L(\theta), J_k \Delta g(q) \rangle \\ + O(\eta^2) + O(\eta\delta), \end{aligned}$$

where  $C(\theta)$  is independent of  $q$ , and the  $O(\eta\delta)$  term arises from multiplying the step size  $\eta$  with the  $O(\|\Delta g(q)\|_2^2) = O(\delta)$  Taylor remainder on  $d_k$ . Dropping higher-order terms, the  $q$ -dependent first-order decrease remains linear in  $q$ . Therefore, the KL-constrained maximization retains the same form as in Sec. 4.1 and admits the same Gibbs-form solution, with the optimizer-aligned token utility

$$\tilde{s}_t(x, y, \theta) = \langle J_k^{\top} \nabla L(\theta), \nabla_{\theta} \ell_t(\theta; x, y) \rangle.$$

Concretely, the optimal reweighting is

$$q^*(t | x, y, \theta) = \frac{\exp(\tau \tilde{s}_t(x, y, \theta))}{\sum_j \exp(\tau \tilde{s}_j(x, y, \theta))}.$$

**Step 3: One-backward probing.** To estimate  $\tilde{s}_t$ , we draw an independent probing batch  $\mathcal{B}'$ , i.i.d. from the data distribution and independent of the main batch  $\mathcal{B}$ , and compute its uniform-weight gradient

$$g_{\mathcal{B}'}(u) \triangleq \nabla_{\theta} \hat{\mathcal{L}}_{\mathcal{B}'}(\theta; u), \mathbb{E}_{\mathcal{B}'}[g_{\mathcal{B}'}(u)] = \nabla L(\theta).$$

Define the probing direction

$$v_k \triangleq J_k g_{\mathcal{B}'}(u).$$

By the directional-derivative identity, for any smooth token loss and small  $\varepsilon$ ,

$$\begin{aligned} \ell_t(\theta - \varepsilon v_k; x, y) \\ = \ell_t(\theta; x, y) - \varepsilon \langle v_k, \nabla_{\theta} \ell_t(\theta; x, y) \rangle + O(\varepsilon^2). \end{aligned}$$

Taking expectation over  $\mathcal{B}'$  and invoking the symmetry  $J_k^{\top} = J_k$  from the coordinate-wise assumption, we obtain

$$\begin{aligned} \lim_{\varepsilon \rightarrow 0} \mathbb{E}_{\mathcal{B}'} \left[ \frac{\ell_t(\theta; x, y) - \ell_t(\theta - \varepsilon v_k; x, y)}{\varepsilon} \right] \\ = \langle J_k \nabla L(\theta), \nabla_{\theta} \ell_t \rangle \\ = \langle J_k^{\top} \nabla L(\theta), \nabla_{\theta} \ell_t \rangle \\ = \tilde{s}_t(x, y, \theta). \end{aligned}$$

Hence the one-backward probing scheme yields an unbiased estimator of  $\tilde{s}_t(x, y, \theta)$ , where the expectation is taken over the probing batch  $\mathcal{B}'$  with  $\theta$ , the optimizer state, and  $\mathcal{B}$  held fixed.

**Explicit Jacobian for Adam.** As an instance of the general first-order framework above, we present the Jacobian  $J_k$  for the Adam update map explicitly. For a gradient input  $g$ , the Adam direction at iteration  $k$  is defined element-wise as

$$d_k(g) = \hat{m}_k(g) \oslash (\sqrt{\hat{v}_k(g)} + \epsilon_{\text{adam}}),$$

where  $\oslash$  denotes element-wise division,  $\sqrt{\cdot}$  is taken element-wise, and  $\epsilon_{\text{adam}} > 0$  is the standard stabilization constant. The first- and second-moment estimates are

$$m_k(g) = \beta_1 m_{k-1} + (1 - \beta_1)g,$$

$$v_k(g) = \beta_2 v_{k-1} + (1 - \beta_2)(g \odot g),$$

where  $\odot$  denotes element-wise multiplication, and  $m_{k-1}, v_{k-1}$  are fixed buffers from the previous iteration (independent of the current gradient  $g$ ). The bias-corrected moments are

$$\hat{m}_k(g) = \frac{m_k(g)}{1 - \beta_1^k}, \quad \hat{v}_k(g) = \frac{v_k(g)}{1 - \beta_2^k}.$$

Since all operations are element-wise and  $\epsilon_{\text{adam}} > 0$ , the map  $d_k : \mathbb{R}^p \rightarrow \mathbb{R}^p$  is smooth in  $g$ , and its Jacobian is diagonal:

$$J_k = \text{diag}(J_{k,1}, \dots, J_{k,p}),$$

where, for each coordinate  $i$ , evaluated at  $g = g_u$ ,

$$J_{k,i} = \frac{c_1}{r_i} - \frac{c_2 g_{u,i} \hat{m}_{k,i}(g_u)}{r_i^2 \sqrt{\hat{v}_{k,i}(g_u)}},$$

with

$$c_1 \triangleq \frac{1 - \beta_1}{1 - \beta_1^k}, \quad c_2 \triangleq \frac{1 - \beta_2}{1 - \beta_2^k},$$

$$r_i \triangleq \sqrt{\hat{v}_{k,i}(g_u)} + \epsilon_{\text{adam}}.$$

For AdamW, the decoupled weight decay contributes only an additive  $q$ -independent term to the update and therefore does not change the KL-constrained reweighting problem above.

## H.2 Empirical Comparison Between SGD and AdamW

To assess optimizer sensitivity, we additionally compare AdamW with plain SGD under the same training configuration. We keep the batch size, number of training steps, LoRA settings, and learning-rate schedule identical across VCore and the baseline methods. For SGD, we disable momentum and weight decay. The comparison is performed on the same Qwen3-4B math setting as in the main experiments. The results are shown at Table 13. Although the absolute values differ due to optimizer dynamics, VCore consistently outperforms SFT and DFT under both optimizers. The relative ordering remains stable across all four benchmarks and the overall average. These results indicate that the empirical advantage of VCore is preserved across the two optimizers in this setting.

## I Case Study

We conducted several case studies to analyze the CoT behaviors generated by different methods. In the table 14, we illustrate the qualitative differences in reasoning behaviors across methods, and we observe a clear shift in how VCore organizes its chains. SFT produces structured, template-driven reasoning, while DFT often yields verbose narrative traces. In contrast, VCore generates exploratory and self-corrective chains with frequent

| Optimizer | Method       | AIME         | Olympiad     | RBench       | SGPQA-1k     | Avg.         |
|-----------|--------------|--------------|--------------|--------------|--------------|--------------|
| AdamW     | SFT          | 46.67        | 61.72        | 30.71        | 30.50        | 42.40        |
|           | DFT          | 35.00        | 62.91        | 34.00        | 32.40        | 41.08        |
|           | <b>VCORE</b> | <b>48.33</b> | <b>66.17</b> | <b>34.64</b> | <b>34.30</b> | <b>45.86</b> |
| SGD       | SFT          | 40.00        | 60.98        | 23.40        | 30.40        | 38.69        |
|           | DFT          | 40.00        | 60.83        | 24.04        | 27.70        | 38.14        |
|           | <b>VCORE</b> | <b>43.33</b> | <b>61.72</b> | <b>24.77</b> | <b>30.60</b> | <b>40.11</b> |

Table 13: Comparison between AdamW and plain SGD under the same training configuration on the Qwen3-4B math setting. Avg. denotes the average over the four reported benchmarks.

| Method       | Example CoT Behaviors  | Characteristics   |
|--------------|--|---|
| <b>SFT</b>   | “Let me think about small cases first.”, “Now let’s move to the $n = 2$ case.”, “Let’s verify this pattern.”, “Now generalize. . .”                                  | Template-like reasoning; structured and explanatory               |
| <b>DFT</b>   | “Wait, but that’s not possible. . . Actually maybe I should mark the diagonal. . . But the diagonal doesn’t work. . . let me re-evaluate the $4 \times 4$ case. . .” | Narrative filler; continuously adding details                     |
| <b>VCORE</b> | “Wait, that might be wrong.”, “But that contradicts what I said earlier.”, “Maybe $k = 2n$ ?”, “No, that seems too large.”   | Highly exploratory; trial-and-error with frequent self-correction |

Table 14: Qualitative comparison of example chain-of-thought (CoT) behaviors produced by different training methods.

hypothesis checks. This behavior is also consistent with the design of VCORE, which emphasizes token-level utility weighting and naturally promotes more flexible and self-corrective reasoning patterns. This indicates that VCORE not only improves accuracy but also promotes more flexible and adaptive reasoning dynamics