EntroCoT: Enhancing Chain-of-Thought via Adaptive Entropy-Guided Segmentation

Anonymous ACL submission

Abstract

Chain-of-Thought (CoT) prompting has significantly enhanced the mathematical reasoning capabilities of Large Language Models. We find existing fine-tuning datasets frequently suffer from the "answer right but reasoning wrong" probelm, where correct final answers are derived from hallucinated, redundant, or logically invalid intermediate steps. This paper proposes **EntroCoT**, a unified framework for automatically identifying and refining low-quality CoT supervision traces. EntroCoT first proposes an entropy-based mechanism to segment the reasoning trace into multiple steps at uncertain junctures, and then introduces a Monte Carlo rollout-based mechanism to evaluate the marginal contribution of each step. By accurately filtering deceptive reasoning samples, EntroCoT constructs a high-quality dataset where every intermediate step in each reasoning trace facilitates the final answer. Extensive experiments on mathematical benchmarks demonstrate that fine-tuning on the subset constructed by EntroCoT consistently outperforms the baselines of full-dataset supervision. Our code is available in "software" appendix.

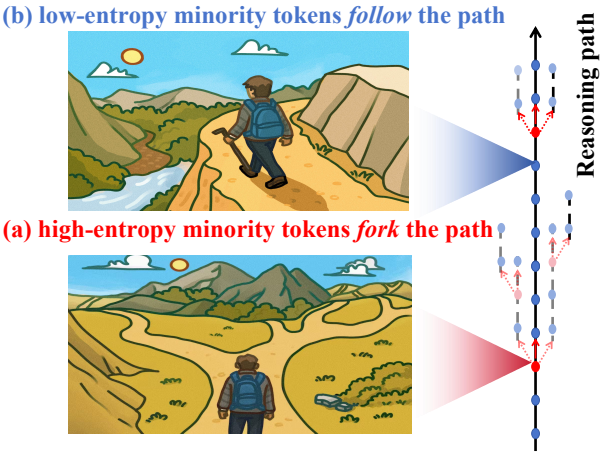
1 Introduction

Large language models (LLMs) have recently demonstrated strong performance on complex mathematical reasoning tasks, a success largely attributed to the adoption of *chain-of-thought* (CoT) prompting (Wei et al., 2022). By explicitly decomposing a problem into intermediate reasoning steps, CoT enables models to emulate human-like multi-step reasoning and has become a standard technique for improving accuracy on solving mathematical problem. Nowadays, many recent fine-tuning datasets include explicit CoT annotations, either written by humans or synthetically generated by models (Zhao et al., 2025).

Existing CoT training paradigms often rely on final answers as the sole quality metric, which neglect the logical integrity within CoT traces. We observe that there are many samples with correct final answer but logically redundant or wrong reasoning steps (Lyu et al., 2023), especially in large-scale synthetic datasets (Luo et al., 2025). For instance, a reasoning trace may erroneously apply the geometric mean formula to a question requiring an arithmetic mean; if the input values are identical, this flawed derivation will coincidentally yield the correct final result. More examples can be found in Appendix C. This "answer right but reasoning wrong" issue forces models to mimic logically flawed patterns and ultimately impairs their ability to solve difficult mathematical tasks (Xu et al., 2025). Empirically, we also find that selective training on filtered subsets yields superior performance, indicating many CoT data is counterproductive. Consequently, establishing a precise definition of CoT trace quality and rectifying logical errors within those traces are critical for effective model tuning (Manakul et al., 2023).

To address this issue, we propose a unified framework called **EntroCoT** for the automated identification and correction of misleading CoT traces. Our approach is grounded in the observation that high-entropy tokens often signal the emergence of logical inconsistencies (Wang et al., 2025a), representing critical forks where the model's reasoning becomes uncertain and multiple logical paths diverge. As illustrated in Figure 1, we conceptualize high-entropy tokens as critical forks where model's reasoning trace is most uncertain and multiple possible logical paths diverge. After partitioning the trace at these pivotal junctures, our framework systematically evaluates the marginal contribution of each segment to the final answer and adaptively filters those deceptive samples. Finally, we guarantee that each step in the reliable CoT trace consistently promotes the correct solution, thereby securing a high-fidelity reasoning process.

Guided by this philosophy, EntroCoT incorpo-



Q: There are 15 candies. If we give away 8 candies, how many candies are left?
A: Well, let's figure out this problem step by step first. We know the total number of candies is 15, but doing 15 minus 8 directly [...]

Figure 1: In chain-of-thought reasoning, high-entropy tokens function as forks introducing alternative reasoning branches, whereas low-entropy tokens proceed along the established path.

rates three stages to assess and filter reasoning traces. In the first stage, we compute token-level entropy across the generated CoT trace using a strong teacher model, leveraging high-entropy tokens as "logical anchors" to partition the trace into discrete segments at critical junctures where the trace is most uncertain. In the second stage, we introduce a Monte Carlo rollout-based prefix evaluation mechanism to quantify the marginal contribution of each segment to final answer. Finally, the filter process begins: should a segment yield a negative impact on performance, the framework automatically filters the corresponding misleading reasoning path(sample). This pipeline is repeated until each sample in the dataset has been filtered, effectively eliminating samples containing deceptive intermediate steps. In this way, EntroCoT enables a scalable alignment between training data and high-fidelity reasoning, yielding models that exhibit both conclusion accuracy and logical integrity.

Extensive experiments across various benchmarks, base models, and training datasets demonstrate that EntroCoT consistently improves the average accuracy by 2.71% for Llama-3.1-8B and 5.17% for Qwen2.5-Math-1.5B compared to vanilla training on full dataset for Numinamath. We also conduct ablation studies to validate the design details of our entropy-guided segmentation.

This paper makes the following contributions.

- We identify and formalize a critical bottleneck of existing CoT fine-tuning traces, where the final answer is correct but the intermediate CoT steps are wrong. We demonstrate that such deceptive traces impair model performance.
- We propose EntroCoT, a unified framework to assess the quality of CoT data and filter misleading traces. EntroCoT features an entropy-based segmentation mechanism and is compatible to existing fine-tuning workflows.
- We conduct extensive evaluation showing EntroCoT can effectively improve fine-tuning accuracy compared to full-dataset supervision.

2 Related Work

The advancement of Large Language Models (LLMs) has been significantly driven by the Chain-of-Thought (CoT) reasoning paradigm. As models scale, ensuring the quality of these reasoning chains has become paramount for effective model distillation. Our work draws inspiration from and builds upon recent developments in CoT distillation, entropy-based reasoning analysis, and process-level supervision.

2.1 CoT Distillation and Data Synthesis

Distilling reasoning capabilities from strong teacher models (e.g., OpenAI o1, GPT-4) to smaller student models is a widely adopted strategy. However, the reliability of the teacher's intermediate reasoning steps remains a challenge. Zhao et al. proposed *PromptCoT* (Zhao et al., 2025), which synthesizes Olympiad-level problems by mimicking the rationale generation process of human experts, highlighting the importance of high-quality reasoning paths for data generation. Similarly, Xiang et al. introduced *Meta-CoT* (Xiang et al., 2025), which models the "meta-reasoning" process to supervise the generation of synthetic data, aiming to emulate System 2 thinking. While these methods focus on *generating* new data, our work addresses the challenge of *verifying* and *filtering* existing long-CoT datasets (such as OpenR1), which often contain correct final answers but hallucinated or logically flawed intermediate steps.

2.2 Entropy-based Reasoning Analysis

Entropy, as a measure of uncertainty in next-token prediction, has emerged as a powerful tool for analyzing the internal dynamics of LLM reasoning.

Cheng et al. (Cheng et al., 2025) demonstrated that high-entropy tokens often correlate with exploratory behaviors, such as pivotal turning points or self-corrections within a reasoning chain. Leveraging this property, Li et al. (Li et al., 2025) proposed using step entropy to identify and remove redundant, low-information steps, thereby compressing CoT without sacrificing accuracy. In parallel, Wang et al. introduced *R1-Compress* (Wang et al., 2025b), which combines chunk compression with search strategies to optimize long reasoning chains. Different from these works which utilize entropy primarily for *compression* or *exploration*, we propose to use the spatial distribution of high-entropy points (across the beginning, middle, and end of the reasoning process) as a signal for *structural segmentation*. We hypothesize that these high-entropy points represent logical junctions that require verification.

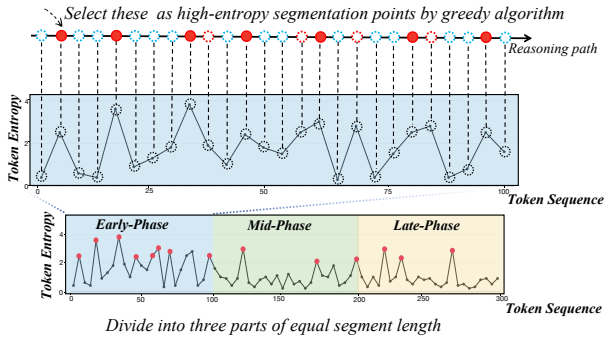


Figure 2: Token-level entropy is computed along the CoT reasoning path, and the sequence is evenly divided into early, middle, and late segments by token order. Based on the distribution of high-entropy positions within each segment, a greedy algorithm selects spatially dispersed high-entropy positions as segmentation points (red dots) for subsequent construction of multiple prompts.

2.3 Structured Process Supervision

To mitigate the "reasoning gap" where a model arrives at the correct answer through erroneous steps, researchers have turned to fine-grained process supervision. Zhang et al. (Zhang et al., 2025) provided a comprehensive analysis of Process Reward Models (PRMs) in mathematical reasoning, emphasizing the necessity of step-by-step verification. Luo et al. proposed *DLCoT* (Luo et al., 2025), a framework that deconstructs long CoT into segments for simplified optimization and error correction. Furthermore, Xu et al. addressed the issue of "Thought Leaps" in *Mind the Gap* (Xu et al., 2025), proposing methods to bridge disconnected reasoning steps during fine-tuning. Our approach aligns with the philosophy of process supervision but introduces a novel, computation-efficient verification mechanism. Instead of training a heavy PRM, we utilize a smaller model to "rollout" from entropy-determined breakpoints. By monitoring the non-decreasing trend of accuracy across segments, we can identify reliable reasoning paths and selectively employ strong models to repair fractured logic, thus constructing a high-quality dataset for robust distillation.

3 Method

While the responses generated by teacher models are typically correct in the final output, they may contain redundant, inconsistent, or locally erroneous reasoning steps. This section introduces an automated method for filtering reasoning data (e.g., Chain-of-Thought, CoT). The core objective of this approach is to identify responses that exhibit *stable positive guidance* for student models at the intermediate reasoning level.

The process is structured into two main stages: (1) Token-level entropy-based response segmentation; (2) Prefix evaluation using Monte Carlo rollouts. The overall algorithmic flow is depicted in Algorithm 1.

3.1 Entropy-Guided CoT Segmentation

Given a sample $(x, y, \mathcal{T})$, where x represents the input question, y is the final answer, and $\mathcal{T} = (t_1, \ldots, t_L)$ is the complete reasoning sequence generated by the teacher model, the following procedure is applied.

First, we compute the token-level entropy for each token in the sequence $\mathcal{T}$, using the teacher model $\mathcal{M}_T$ conditioned on the input x:

$$H_i = -\sum_{v \in \mathcal{V}} p_{\mathcal{M}_T}(v \mid x, t_{<i}) \log p_{\mathcal{M}_T}(v \mid x, t_{<i}),$$

where $\mathcal{V}$ denotes the model's vocabulary. The K highest entropy positions are identified, forming the index set $\mathcal{H} = \{h_1, \ldots, h_K\}$.

We use Figure 2 to illustrate the specific CoT segmentation procedure. We first divide the sequence $\mathcal{T}$ is into three regions: the beginning, middle, and end. Then we count the number of high-entropy positions in each region, yielding the values (r_1, r_2, r_3), where $r_1 + r_2 + r_3 = K$.

Given the target number of segments N, the number of segments for each region is determined based

on the proportion of high-entropy positions in each region:

$$s_i = \left\lfloor N \cdot \frac{r_i}{r_1 + r_2 + r_3} \right\rfloor, \quad i \in \{1, 2, 3\},$$

with a normalization step to ensure that $s_1 + s_2 + s_3 = N$. This allocation strategy ensures that regions with higher uncertainty are divided into more reasoning sub-segments.

For the i-th region, let $\mathcal{H}_i$ be the set of high-entropy positions, and let s_i sub-segments be allocated to the region. The segmenting points are chosen greedily, starting from the smallest and largest high-entropy positions, and then iteratively selecting the position that maximizes the sum of distances to the current set of splitting points:

$$h^\star = \arg \max_{h \in \mathcal{H}_i \setminus \mathcal{S}_i^{(m-1)}} \sum_{h' \in \mathcal{S}_i^{(m-1)}} |h - h'|.$$

This procedure continues until $|\mathcal{S}_i^{(m)}| = s_i - 1$. By using a simple greedy algorithm, this strategy prevents the excessive concentration of high-entropy split points in local regions.

The final sequence of cutting points across all regions is merged and sorted to obtain the segmented reasoning sequence:

$$\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_N).$$

These segments are then used to construct prefix-conditioned prompts for the subsequent stage, where they undergo stability evaluation using Monte Carlo rollouts.

3.2 Monte Carlo Rollout-based Prefix Evaluation

A key property of high-quality responses is that their reasoning prefixes should gradually reduce uncertainty and increase the probability of generating the correct answer. We quantitatively evaluate this property using Monte Carlo rollouts.

For the segmented CoT, we construct a series of prefix-conditioned prompts:

$$\mathcal{P}_k = \text{Concat}(x, \mathcal{T}_1, \ldots, \mathcal{T}_k), \quad k = 1, \ldots, N-1,$$

removing the final reasoning segment to avoid potential leakage of the answer information.

For each prefix $\mathcal{P}_k$, we perform R independent generations using a lightweight model $\mathcal{M}_g$ and estimate the probability of generating the correct answer through Monte Carlo:

$$\hat{a}_k \approx \mathbb{P}_{\mathcal{M}_g}(y \mid \mathcal{P}_k) = \frac{1}{R} \sum_{r=1}^{R} \mathbb{I}\left(\mathcal{M}_g(\mathcal{P}_k^{(r)}) = y\right),$$

where $\mathbb{I}$ is the indicator function.

If the estimated probability sequence satisfies:

$$\hat{a}_1 \leq \hat{a}_2 \leq \cdots \leq \hat{a}_{N-1},$$

it indicates that each additional CoT segment enhances the probability of the model reaching the correct answer. We classify such samples as *reliable data*, and they are retained for distillation training.

Algorithm 1 Entropy-Guided Progressive CoT Filtering

Require: Dataset $\mathcal{D}$, teacher model $\mathcal{M}_T$, rollout model $\mathcal{M}_g$
Ensure: Reliable dataset $\mathcal{D}_{\text{rel}}$, deferred dataset $\mathcal{D}_{\text{def}}$

```
for all (x, y, T) ∈ D do
    Compute token-level entropy {H_i} using M_T
    Select top-K high-entropy positions H
    Divide T into early, middle, and late regions
    Allocate segment numbers proportional to region-wise entropy counts
    Greedily select dispersed split points and segment T into (T_1, ..., T_N)
    for k = 1 to N − 1 do
        Construct prefix prompt P_k = Concat(x, T_1, ..., T_k)
        Estimate correctness probability â_k via R Monte Carlo rollouts of M_g
    end for
    if â_1 ≤ â_2 ≤ ··· ≤ â_{N−1} then
        Add sample to D_rel
    else
        Add sample to D_def
    end if
end for
```

4 Experiments

4.1 Datasets and Evaluation Metrics

We employ a diverse suite of benchmarks categorized by difficulty: (1) **GSM8K** (Cobbe et al., 2021) is a dataset consisting of grade-school level

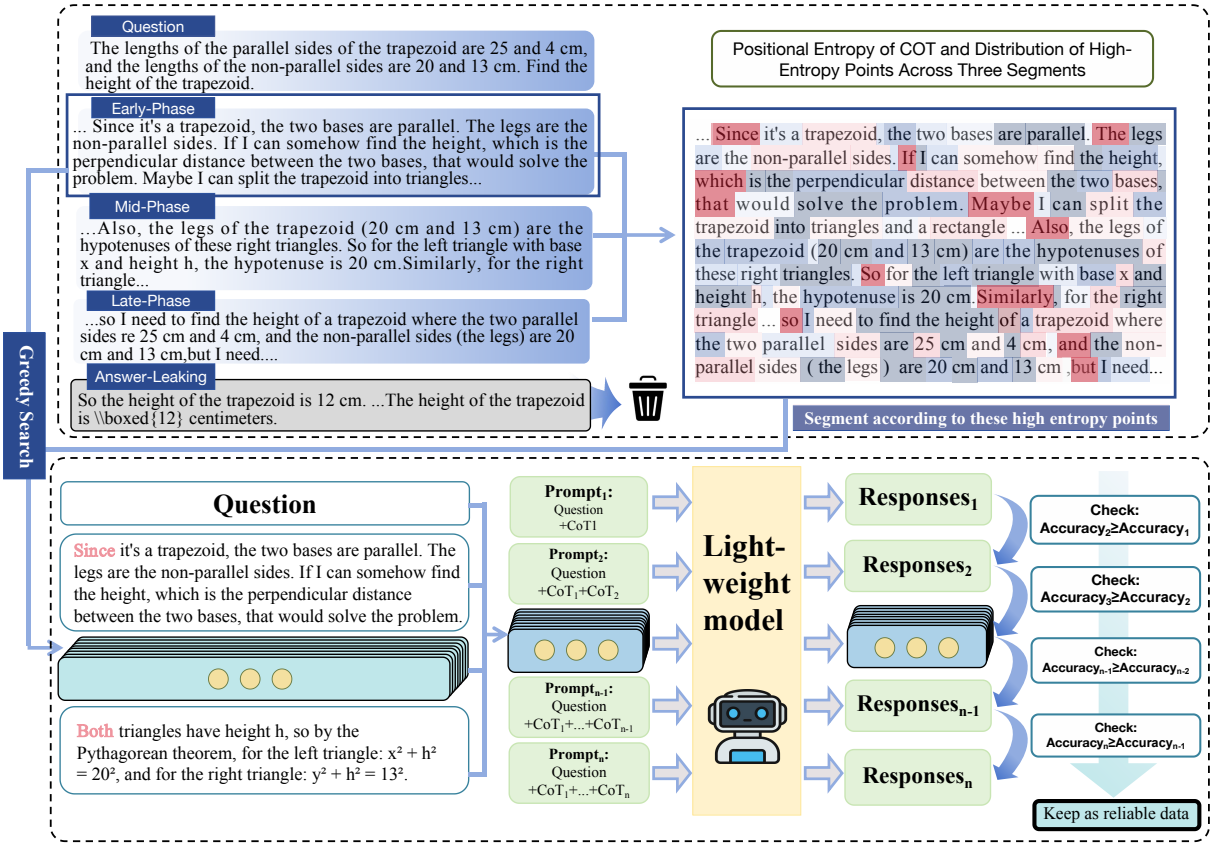


Figure 3: Token-level entropy is computed over the CoT, and high-entropy positions are analyzed across early, middle, and late segments. Segmentation points are adaptively selected via greedy search to match the segment-wise entropy distribution, partitioning the CoT into sub-segments. The question combined with progressively accumulated CoT segments is used to prompt a lightweight model for multiple rollouts; the last segment is excluded to avoid answer leakage. Samples whose CoT segments' accuracy remains non-decreasing are retained as reliable data.

math problems that require logical reasoning, testing a model's ability to solve elementary-level math problems; (2) **MATH-500** (Lightman et al., 2023) is a curated subset of 500 representative high-school math problems, serving to assess a model's ability to handle more advanced mathematical reasoning; (3) **Gaokao** (Zhang et al., 2023) is a benchmark that includes mathematics problems from the Chinese National College Entrance Examination, designed to evaluate high school graduates' comprehensive mathematical proficiency. These three datasets serve as basic-level benchmarks. For advanced competition-level assessment, we use (4) **AMC23** (Mathematical Association of America, 2023), which comprises challenging problems from the 2023 American Mathematics Competitions; (5) **MathOdyssey** (Fang et al., 2025), a dataset spanning high-school to early undergraduate difficulty that emphasizes multi-step derivations and conceptual understanding; and (6) **OlympiadBench** (He et al., 2024), a collection of olympiad-level problems with non-standard formats requiring creative problem-solving strategies. We use exact match accuracy as the primary metric for evaluating performance, determined by comparing the predicted final answer, enclosed by \boxed, with the ground-truth answer. All evaluations are conducted using the vLLM inference backend under identical generation parameters: zero-shot prompting with greedy decoding. To ensure statistical stability, we report the average exact-match accuracy across five independent runs for each model-dataset pair. Further implementation details are provided in Appendix B.2.

4.2 Baselines

We evaluate our method by establishing a series of baselines for comparison. The most important baseline is Direct-SFT. In addition, to disentangle the contribution of every design decision in Entro-CoT, we conduct the ablation experiments. Below we describe several baselines that progressively isolate (i) the impact of entropy-based segmentation and (ii) the importance of the greedy dispersion

Dataset	Size	Method	Basic Level			Competition Level			Avg.
			GSM8K	MATH	GaoKao	Odyssey	Olympiad	AMC23	
Meta-Llama3.1-8B									
MetaMathQA	395k	Direct SFT	**77.03**	33.80	23.64	7.46	6.22	7.50	25.94
	332k	EntroCoT-random	74.83	31.40	23.90	6.68	5.93	5.00	24.62
	358k	EntroCoT-w/o-greedy	75.01	32.60	23.22	6.78	**6.84**	8.50	25.49
	344k	EntroCoT-full	76.89	**35.80**	**27.01**	**7.97**	6.81	**15.00**	**28.25**
NuminaMath	859k	Direct SFT	72.10	37.20	32.73	**20.82**	13.04	19.00	32.48
	515k	EntroCoT-random	71.34	39.24	36.67	19.69	12.86	19.00	33.13
	395k	EntroCoT-w/o-greedy	70.96	39.80	38.96	17.48	12.00	17.50	32.78
	480k	EntroCoT-full	**76.00**	**41.20**	**40.00**	19.54	**14.37**	**20.00**	**35.19**
Qwen2.5-Math-1.5B									
MetaMathQA	395k	Direct SFT	48.60	33.84	33.72	17.12	10.28	7.50	25.18
	332k	EntroCoT-random	45.40	33.92	35.58	16.45	**12.12**	8.50	25.33
	358k	EntroCoT-w/o-greedy	47.43	34.44	35.12	16.20	10.67	8.50	25.39
	344k	EntroCoT-full	**50.19**	**34.56**	**37.35**	**17.23**	11.14	**15.00**	**27.58**
NuminaMath	859k	Direct SFT	70.90	54.64	46.07	21.44	19.73	32.50	40.88
	515k	EntroCoT-random	71.01	52.12	44.21	22.67	21.48	32.50	40.67
	395k	EntroCoT-w/o-greedy	73.09	48.20	40.52	20.31	18.07	35.00	39.20
	480k	EntroCoT-full	**74.65**	**59.60**	**48.80**	**23.40**	**24.35**	**45.50**	**46.05**

Table 1: Main results (%) on mathematical benchmarks. MATH, GaoKao, Odyssey, and Olympiad correspond to the MATH500, GaoKao2023EN, MathOdyssey, and OlympiadBenchEN benchmarks, respectively. Bold marks the best score per dataset. Avg. is derived by calculating the average accuracy of the six benchmarks.

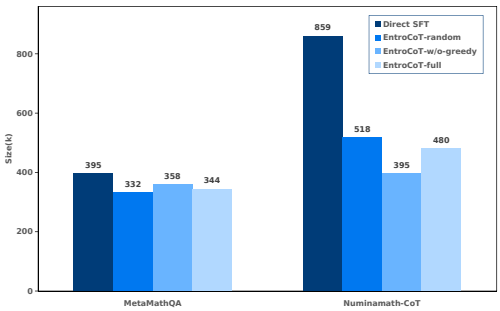


Figure 4: Dataset sizes for MetaMathQA and NuminaMath under different training strategies. Direct SFT denotes standard supervised fine-tuning, while EntroCoT variants differ in sampling and greedy constraints.

heuristic. (1) **Direct-SFT**: The primary reference that fine-tunes on the complete MetaMathQA-395 k or NuminaMath-859 k dataset without any sample removal or rewriting. This replicates the conventional distillation pipeline where every trace is treated as gold supervision. (2) **EntroCoT-random**: We preserve the entire segmental rollout pipeline (§3.2) but replace entropy-guided segmentation with random cutting points. High-entropy tokens are still computed, yet they are ignored when constructing $(\mathcal{T}_1, \ldots, \mathcal{T}_N)$. This ablation tests whether token-level uncertainty genuinely marks logical fault lines or merely acts as a spurious feature correlated with sequence position. (3) **EntroCoT-w/o-greedy**: We retain the entropy computation and the early/middle/late ternary split, but inside each third we *randomly* choose $s_i - 1$ segmentation points instead of applying the greedy max-sum-of-distances rule. The resulting segments therefore still concentrate around high-entropy regions, yet may cluster adjacent high-uncertainty tokens. Comparing with EntroCoT-full quantifies the benefit of spatially *dispersing* segmentation anchors so that the lightweight model receives CoT which contains relatively more complete reasoning steps. (4) **EntroCoT-full**: Our complete method: entropy-guided segmentation with greedy dispersion, rollout-based verification, and training *only* on the samples that satisfy the reliable criterion.

4.3 Implementation Details

We empirically evaluate the Entropy-Guided Progressive CoT Filtering (EntroCoT) framework as a data-centric procedure for enhancing mathematical reasoning. Our experiments aim to determine whether filtering unreliable reasoning traces can outperform full-dataset training and to identify the essential components of our framework. To evaluate the generality and effectiveness of our proposed approach, we perform supervised fine-tuning (SFT)

experiments using MetaMathQA (Yu et al., 2023) and NuminaMath-CoT (Li et al., 2024) datasets on representative base models. Specifically, we select Llama-3.1-8B (Touvron et al., 2023) as a large-scale model of llama series, and Qwen2.5-Math-1.5B (Yang et al., 2024) as a model of qwen series for finetune. For each dataset, we apply our method to partition the samples into three disjoint buckets: a reliable set of behaviorally beneficial CoTs, a reject set of logically harmful traces, and an all-zero set of extremely difficult questions or harmful traces. Specifically, we use Qwen3-4B-Instruct (Yang et al., 2025) as the lightweight model for rollout and the number of rollouts per round is set to $r = 8$. Moreover, the number of segments is set to $k = 5$. After applying our method on datasets, we get a total of $m = 480{,}313$ reliable samples for Numinamath-CoT, while for MetaMathQA (Yu et al., 2023), we get a total of $n = 344{,}405$ reliable samples. In addition, Deepseek-R1 (Guo et al., 2025) is selected for the entropy calculation. To ensure comparability, we maintain a unified experimental configuration across all SFT variants in the same set of experiments. The detailed training and rollout settings, including learning rates, training epochs and so on, are provided in Appendix A.

4.4 Main Results

Table 1 presents our main experimental results across all benchmarks, base models, and training datasets. We observe several consistent patterns that highlight the effectiveness of our Entropy-Guided Progressive CoT Filtering (EntroCoT) approach.

Filtering via EntroCoT consistently outperforms full-dataset training. Despite discarding 45% of NuminaMath-CoT and 13% of MetaMathQA, the "EPCoT-full" models surpass the Direct-SFT baseline on almost all benchmarks. The largest absolute gains appear on the most challenging splits: Llama-3.1-8B improves +7.50% on AMC23 when training on MetaMathQA; Qwen2.5-Math-1.5B gains +13.00% on AMC23 and +4.62% on Olympaid. Averaged across all six datasets, EntroCoT yields +2.31% for Llama-3.1-8B and +2.40% for Qwen2.5-Math-1.5B when training on MetaMathQA and yields +2.71% for Llama-3.1-8B and +5.17% for Qwen2.5-Math-1.5B when training on Numinamath-CoT, mirroring the prior observation that higher-quality supervision disproportionately benefits competition-level reasoning. The outsized gains on long-chain problems stem from two factors. First, competition-grade questions require significantly more intermediate steps, so a single "early-error, late-correction" sequence constitutes an extended source of incorrect gradient updates; EntroCoT's monotonicity test rejects such entire chains, removing the longest and most frequent negative signals. Second, the lightweight rollout verifier is already near-chance on these difficult items, so any contaminated prefix causes a sharp drop in estimated correctness, ensuring that almost all flawed long derivations are filtered out. Consequently, the retained set is enriched for clean, multi-step symbolic derivations, and the student model observes a higher proportion of valid long-range reasoning patterns during SFT, which directly translates into larger accuracy improvements on the hardest benchmarks.

Entropy-guided segmentation is the key driver. EPCoT-random incurs consistent and non-negligible degradations across almost all benchmarks: on the Llama3.1-8B it drops accuracy by 2.06% on GSM8K, 4.40% on MATH-500, 3.11% on GaoKao2023EN, and margins of 1.29%, 0.88% and 10.00% on the competition-level Odyssey, OlympiadBenchEN and AMC23 benchmarks, respectively on MetaMathQA; similar trends appear for Qwen2.5-Math-1.5B, where average losses reach 5.24% on basic tasks and 5.53% on competition problems when training on Numinamath. These systematic deficits, summarised in an overall -5.38% average, fall below the EPCoT-full and demonstrate that random segmentation is actively harmful rather than merely ineffective. The root cause is the inability of random cuts to disentangle erroneous segments from subsequent correct ones: cut points frequently land within mixed-quality spans, producing prefixes that begin with a hallucinated claim but end with a valid derivation. The lightweight rollout model can still exploit the correct tail signal to reach the right answer, so estimated accuracy does not decline and the monotonicity criterion fails to reject the flawed prefix. As a result, early-stage mistakes survive filtering, are included in training, and measurably degrade generalisation performance.

Greedy spatial dispersion is indispensable. EPCoT-w/o-greedy exhibits consistent performance degradation across all test suites. On MetaMathQA, the average accuracy of Llama-3.1-8B drops from 28.25% to 25.49%, while Qwen2.5-Math-1.5B falls from 27.58% to 25.39%. When training on the larger NuminaMath-CoT dataset,

the two models exhibit further decrements of 2.41% and 6.85%, respectively, demonstrating that the performance degradation caused by removing greedy dispersion becomes more pronounced as the dataset size increases. This deterioration is attributable to the clustering of high-entropy cut points within a narrow 50-token window, which reduces the aggregate token coverage of adjacent segments and limits the behavioural diversity presented to the rollout verifier. Consequently, the estimated accuracy curves exhibit higher variance, and harmful reasoning fragments are less likely to be detected and removed. The greedy max-sum-of-distance rule mitigates this effect by maximising inter-cut spacing, thereby ensuring that each segment contains a rich and non-redundant portion of the reasoning chain and preserving the discriminative power of the filter.

5 Discussion

Filtering generalises beyond mathematics. Although we benchmark on math, the pipeline makes no domain-specific assumptions. When the same entropy-cut/rollout strategy is applied, it is also applicable to problems with rich reasoning processes such as synthetic chemical mechanism problems and physical problems. The key requirement is a verifiable final answer; domains such as open-ended creative writing or legal argumentation would need an external value function instead of exact-match grading, but the entropy signature itself remains informative.

Rejected samples can be recovered. For samples that fail the monotonicity check, they can not be discarded immediately, but can be attempted to recover their inference traces. Let the first position where rollout accuracy $\hat{a}_k > \hat{a}_{k+1}$ occur at index $k^\star$. We regenerate candidate continuations starting from this segment using a stronger model $\mathcal{M}_R$ (e.g., GPT series), and only keep the candidates where the final answer is correct. The prefix evaluation process from Section 3.2 is then repeated for the recovered samples. If any candidate satisfies the monotonicity condition, the repaired sample is added to the reliable dataset; otherwise, the sample is placed into a deferred dataset for more complex recovery strategies or manual analysis. The code related is also provided in "software" appendix.

Recovered-rejected pairs can be served as DPO treasure trove. The traces that are filtered out—along with their refined counterparts—naturally form paired preference data: the original (high-entropy, misleading) chain serves as the "rejected" response, while the entropy-guided repaired version becomes the "chosen" one. Feeding these pairs directly into Direct Preference Optimization (DPO) without any relabelling can enhance the model's capabilities at the reasoning step level, effectively recycling waste compute into an extra alignment signal. This observation suggests that future pipelines should *store* rejected traces rather than delete them, turning the expensive filtering stage into a dual-purpose generator of both clean SFT data and cheap preference data for RLHF-style training.

6 Conclusion

We present **EntroCoT**, an entropy-guided pipeline that automatically filters "answer-right-but-reasoning-wrong" traces from large-scale CoT datasets. By segmenting each reasoning chain at its highest-uncertainty tokens and validating every segment with lightweight Monte-Carlo rollouts, we retain only samples whose intermediate steps monotonically increase the probability of the correct answer. Across six mathematical benchmarks and two base models, training on this reliable subset systematically outperforms full-dataset supervision, such as yielding average gains of **+2.71 %** on Llama-3.1-8B and **+5.17 %** on Qwen2.5-Math-1.5B while reducing training compute by up to **45 %** on Numinamath. The entropy-based spatial dispersion heuristic is shown to be indispensable: ablating it collapses verifier diversity and erases the entire accuracy advantage. These results demonstrate that *reasoning quality*, not data quantity, is the decisive factor for eliciting robust mathematical generalization in distilled LLMs. EntroCoT can be dropped into any existing CoT fine-tuning workflow without architectural changes, offering a scalable path toward models that not only answer correctly but also reason correctly.

Limitations

First, the overall pipeline is computationally expensive: for every candidate reasoning path we first perform a full forward pass with a strong model to calculate token-level entropy; subsequently, each trace is split into five segments and evaluated with eight independent Monte-Carlo rollouts(8 per segment × 5 segments by default).

Second, EntroCoT's rollout verifier relies on an exact final answer to estimate segment-wise accuracy; it therefore cannot handle proof verification, creative writing, legal argument, or any task where the "correct" conclusion is subjective or undefined. Extending the framework to tasks with only external, human-graded or debate-based validation signals is left to future work.

References

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with Exploration: An Entropy Perspective. *arXiv preprint arXiv:2506.14758*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. 2025. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *Scientific Data*, 12(1):1392.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

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*.

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*.

Yijia Luo, Yulin Song, Xingyao Zhang, Jiaheng Liu, Weixun Wang, GengRu Chen, Wenbo Su, and Bo Zheng. 2025. Deconstructing long chain-of-thought: A structured reasoning optimization framework for long cot distillation. *arXiv preprint arXiv:2503.16385*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025a. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Yibo Wang, Haotian Luo, Huanjin Yao, Tiansheng Huang, Rui Liu, Naiqiang Tan, Jiaxing Huang, Kai Zhang, Dacheng Tao, and Li Shen. 2025b. R1-Compress: Long Chain-of-Thought Compression via Chunk Compression and Search. *arXiv preprint arXiv:2505.16838*.

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.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, and 1 others. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *arXiv preprint arXiv:2501.04682*.

Haolei Xu, Yuchen Yan, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Shengpei Jiang, Kaitao Song, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025. Mind the gap: Bridging thought leap for improved chain-of-thought tuning. *arXiv preprint arXiv:2505.14684*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The Lessons of Developing Process Reward Models in Mathematical Reasoning. *arXiv preprint arXiv:2501.07301*.

Xueliang Zhao, Wei Wu, Jian Guan, and Lingpeng Kong. 2025. Promptcot: Synthesizing olympiad-level problems for mathematical reasoning in large language models. *arXiv preprint arXiv:2503.02324*.

A Training and Rollout Details

A.1 Training Details

We utilize Pai-Megatron as the SFT training framework. The initial learning rate is set to 1×10^{-5} with a warm-up ratio of 0.1, and cosine scheduling is applied to gradually decay the learning rate to 1×10^{-6}. We apply sequence packing for training, while the sequence length is set to 32768 tokens, with a global batch size of 64 and a micro batch size as 1. For the parallel strategy, when training Llama-3.1-8B, we set tensor-parallel as 4 and pipeline-parallel as 2, and for Qwen2.5-Math-1.5B, we set both tensor-parallel and pipeline-parallel as 1. Llama-3.1-8B is trained for 3 epochs on MetaMathQA and NuminaMath-CoT while Qwen2.5-Math-1.5B is trained for 2 epochs on MetaMathQA and NuminaMath-CoT. All SFT experiments are conducted on **8× NVIDIA H20 GPUs**. For the training template, we adopt the following default style template.

Training Template

System: You are a helpful AI assistant, who always ready to help user.

User:
`<question text>`

Assistant:
`<answer text>`

A.2 Rollout Details

We use Qwen3-4B-Instruct as the lightweight model for rollout. In detail, the max new tokens is set as 8192 while top-p and top-k are set as 0.8 and 20 respectively. In addition, repetition penalty is set as 1.1. To accelerate, we use **96× NVIDIA H20 GPUs** to perform Monte Carlo rollout in parallel. Moreover, we use sglang as the inference engine.

A.3 Entropy Calculation Details

We utilize DeepSeek-R1 as the model for entropy calculation. Specifically, four replication of DeepSeek-R1 is deployed on **64× NVIDIA H20 GPUs** while each of them is deployed on **16× NVIDIA H20 GPUs**. In addition, the tensor-parallel, expert-parallel and data parallel are set as 16, 16 and 8 respectively. Moreover, the parameter of logprobs is set as 5 to derive the top-5 highest values of log probabilities for entropy calculation.

B Evaluation

B.1 Evaluation Prompt

We extract final answers by extracting the content included in `\boxed{}` and applying normalization rules (e.g., trimming spaces, canonicalizing fractions). Each problem is wrapped in a template that explicitly encourages step-by-step reasoning and the template is referenced to the selected benchmark. For example, the template for AMC23 is as following:

Prompt Template

You are an expert mathematician specializing in competition-level mathematics. Solve the given AIME problem step by step and provide your final answer.

Guidelines:

- AIME answers are always integers from 0 to 999
- Show your complete reasoning process
- Provide the final answer as an integer enclosed in `\boxed{}`
- Be precise and rigorous in your mathematical reasoning

Problem:
<problem>

Please solve this problem step by step. Remember that the answer must be an integer from 0 to 999. Provide your final answer in `\boxed{}` format.

This evaluation pipeline is kept identical across all methods to ensure a fair comparison of different strategies.

B.2 Evaluation Settings

The maximum token limit for generation was set to 4096. It is noteworthy that, although we set the decoding temperature parameter to zero, vLLM(inference engine we use for evaluation) outputs still exhibited some randomness. For answer extraction and comparison, we employed the Math-Verify tool. Given limitations of math-verify in handling complex expressions, responses failing initial verification were subsequently validated using DeepSeek-R1. All model evaluations were performed using **1× NVIDIA H20 GPU**.

C Case Study

C.1 Reasoning and Annotation Errors

Cases of the samples containing reasoning or annotation errors are provided. In particular, some errors arise from arithmetic mistakes or semantic misinterpretations, while others stem from the inappropriate selection of problem-solving strategies. In the following case studies, we explicitly identify the *error onset* and analyze how the subsequent reasoning is affected.

Case Study 1: Zinc Intake Problem

Problem:
Jerry takes two antacids that weigh 2 grams each and contain 5% zinc by weight. He then takes x smaller antacids that weigh 1 gram each and contain 15% zinc. The total amount of zinc consumed is 650 milligrams. What is the value of x?

Answer from Dataset:

To solve this problem, we need to determine the value of x, which represents the number of smaller antacids that Jerry takes.
Let's break down the information given:
Weight of each larger antacid: 2 grams
Zinc content of each larger antacid: 5%
Weight of each smaller antacid: 1 gram
Zinc content of each smaller antacid: 15%
Total amount of zinc consumed: 650 milligrams
To find the amount of zinc from the larger antacids, we multiply the weight of each antacid by the zinc content:

```
2 grams × 5% = 0.1 grams of zinc
```

To find the amount of zinc from the smaller antacids, we multiply the weight of each antacid by the zinc content:
1 gram $\times$ 15% = 0.15 grams of zinc
Now, we can set up the equation based on the total amount of zinc consumed:
0.1 grams $+x \times 0.15$ grams = 650 milligrams
Let's convert the units to grams:
0.1 $+0.15x = 0.65$ grams
Let's solve for x:
0.15x = 0.65 - 0.1
0.15x = 0.55
To isolate x, we divide both sides of the equation by 0.15:
x = 0.55 / 0.15
x = 3.67
The value of x is approximately 3.67. Since we cannot have a fraction of an antacid, we can conclude that Jerry takes 3 smaller antacids. The value of x is 3.

Error Analysis:
The error occurs in the computation of zinc contributed by the larger antacids. While 2 g $\times 5\% = 0.1$ g is correct *per antacid*, the dataset fails to account for the fact that Jerry consumes two such antacids.

The correct zinc contribution from the larger antacids is therefore:

$$2 \times 2 \text{ g} \times 5\% = 0.2 \text{ g}.$$

After correction, the total zinc equation becomes:

$$0.2 + 0.15x = 0.65.$$

Solving yields:

$$0.15x = 0.45 \quad \Rightarrow \quad x = 3.$$

Data source: METAMATHQA.

Case Study 2: Meeting Attendance Table

Problem:
Each person marks the table with an X on the day(s) he or she would not be able to attend a meeting. On which day can the most people attend the meeting?

	Mon	Tues	Wed	Thurs	Fri
Anna	X		X		
Bill		X		X	X
Carl	X	X		X	X

Answer from Dataset:

```
We can count the number of X's for each
day to see which day has the most people
unavailable.
```

On Monday, there is 1 X. On Tuesday, there are 2 X's. On Wednesday, there are 0 X's. On Thursday, there are 2 X's. On Friday, there are 1 X. Therefore, the day with the most people available is `\\boxed{Wednesday}`.The answer is: Wednesday.

Error Analysis:
The error occurs at the very first reasoning step. The problem asks for the day on which the most people *can* attend the meeting. However, the reasoning path instead focuses on counting the number of people who *cannot* attend.
This semantic inversion fundamentally alters the task objective. By optimizing for maximum unavailability rather than maximum availability, the sample's reasoning path proceeds in the wrong direction from the outset, rendering the entire solution invalid regardless of any subsequent calculations.

Data source: METAMATHQA.

Case Study 3: Dune Buggy Average Speed

Problem:
Conner rides a dune buggy in the desert. On flat sand, it travels at 60 miles per hour. On downhill slopes, it travels 12 miles per hour faster than on flat sand. On uphill slopes, it travels 18 miles per hour slower than on flat sand. If Conner rides his dune buggy one-third of the time on flat sand, one-third of the time on uphill slopes, and one-third of the time on downhill slopes, what is his average speed in miles per hour?

Answer from Dataset:

To find the average speed, we need to consider the speeds at which Conner rides on each type of terrain and the time spent on each.
On flat sand, the dune buggy rides at 60 mph.
On downhill slopes, it rides at 60 mph + 12 mph = 72 mph.
On uphill slopes, it rides at 60 mph - 18 mph = 42 mph.

```
Since Conner rides one-third of the time on each type of terrain, we can calculate the average speed by taking the harmonic mean of the three speeds.
```

The harmonic mean is more appropriate than the arithmetic mean for average speed because it takes into account the time spent at each speed, rather than just the distances. The formula for the harmonic mean (H) of three numbers (a, b, c) is given by:
$H = 3/(1/a + 1/b + 1/c)$
Plugging in the values for a, b, and c:
$H = 3/(1/60 + 1/72 + 1/42)$
First, find the common denominator for the fractions, which is the least common multiple (LCM) of 60, 72, and 42. The LCM of 60, 72, and 42 is 360.
Now convert the fractions:
$1/60 = 6/360$
$1/72 = 5/360$
$1/42 = 8.57142857/360(approximately)$
Now add the fractions: $6/360 + 5/360 + 8.57142857/360 = 19.57142857/360$ Now take the reciprocal to find the harmonic mean:
$H = 3/(19.57142857/360)$
$H = 3 * (360/19.57142857)$
$H \approx 3 * 18.39795918$
$H \approx 55.19387755$
Conner's average speed is approximately \\boxed{58} miles per hour.

Error Analysis:
The reasoning becomes incorrect at the boxed step. The problem specifies that Conner spends an equal fraction of *time* on each terrain type, in which case the average speed should be computed using the *arithmetic mean*, not the harmonic mean. The harmonic mean is only appropriate when equal *distances* are traveled at different speeds. This incorrect choice of averaging strategy leads to an erroneous final answer. Since Conner spends equal time on each terrain type (one-third each), the average speed is the arithmetic mean of the three speeds.
Given:

- Flat sand: 60 mph
- Downhill: $60 + 12 = 72$ mph
- Uphill: $60 - 18 = 42$ mph

The average speed is:

$$v_{\text{avg}} = \frac{60 + 72 + 42}{3} = \frac{174}{3} = 58 \text{ mph}$$

Data source: NUMINAMATH-COT.

D Use of AI

We use LLM to help polish the sentences in the paper and correct grammatical errors.