Structured CoT Data Construction for Efficient Mathematical Reasoning

Anonymous ACL submission

Abstract

In this paper, we propose the Structured CoT Data Construction Method (SCDC), a simple yet practical approach that strictly constrains the model's reasoning output to four concise and fixed segments: (1) problem analysis, (2) selection of relevant formulas and theorems, (3) substitution of data into computations, and (4) final answer verification. We reorganize the GSM8K training set by reformatting it into this four-step standardized reasoning paradigm, thereby constructing structured CoT supervision data. Extensive experiments on multiple mathematical benchmark datasets, including GSM8K, AQuA, SVAMP, and MATH500, demonstrate that our method reduces the average reasoning text length by up to 90.0%, and improves reasoning efficiency by a factor of 6.1 to 9.8, while maintaining or even surpassing the original model's reasoning accuracy. The experimental results confirm that structured reasoning data construction is a promising approach for achieving efficient reasoning in large language models, offering a practical and viable alternative to unconstrained free-form CoT generation. Our code and dataset will be released in `https://anonymous.4open.science/r/SCDC-Method/README.md`

1 Introduction

Large Language Models (LLMs) have revolutionized AI, demonstrating strong capabilities in tasks such as question answering and code generation (Brown et al., 2020; OpenAI et al., 2023). Chain-of-Thought (CoT) reasoning (Wei et al., 2022) guides models to generate intermediate steps, significantly improving performance on multi-step arithmetic, commonsense, and symbolic reasoning (Kojima et al., 2022; Wang et al., 2022). This has spurred reasoning-specialized models like OpenAI o1 (OpenAI, 2024), DeepSeek-R1 (DeepSeek-AI et al., 2025), and Qwen3 (Yang et al., 2025a), which achieve state-of-the-art results by generating long CoT sequences.

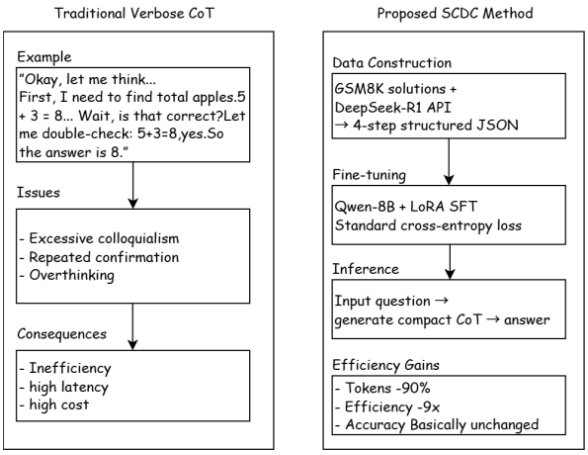

Figure 1: Verbose CoT (left) vs. SCDC (right). SCDC reduces token usage by 90% and improves efficiency by 9× while preserving accuracy.

However, CoT sequences often contain substantial redundant tokens—colloquial preambles ("Okay, let me think"), self-reflective phrases, and socially oriented statements (Jehu-Appiah, 2026). Such verbosity increases computational cost, latency, and resource waste, which is critical for real-time applications and edge deployment (Aggarwal and Welleck, 2025). Recent test-time scaling studies (Muennighoff et al., 2025) show that longer sequences correlate with accuracy only when tokens carry semantic value; meaningless tokens break this correlation. Distilled models (Fan et al., 2025) inherit verbosity but lack pruning ability.

Existing CoT compression methods include prompt optimization (Xu et al., 2025; Aytes et al., 2025), hard budget truncation, redundancy pruning (Kang et al., 2024; Li et al., 2025), and distillation (Jiang et al., 2026). However, these methods are limited in their ability to sufficiently reduce tokens, and often leave behind residual redundancy. Recent structured CoT designs (Li et al., 2023; Yang et al., 2025b) focus on accuracy rather than efficiency. Thus, integrating structured reasoning with precise

redundancy pruning remains an open challenge.

We propose Structured CoT Data Construction (SCDC). The core idea is that mathematical problem-solving can be distilled into a fixed four-step framework: (1) problem analysis, (2) formula selection, (3) substitution and calculation, (4) final answer verification. We restructure the original CoT chains from GSM8K (Cobbe et al., 2021) into this four-step template and fine-tune a base model (Qwen3-8B) (Yang et al., 2025a). Unlike prompt engineering, our supervised fine-tuning directly reshapes generation preference; unlike hard length constraints, we preserve logical coherence.

We evaluate on GSM8K, AQuA (Ling et al., 2017), SVAMP (Patel et al., 2021), and Math500 (Hendrycks et al., 2021), using Qwen3-8B, DeepSeek-R1-7B (DeepSeek-AI et al., 2025), and Mistral-8B (Jiang et al., 2023). SCDC compresses average CoT length by up to 90%, reduces inference time by 81%, while maintaining competitive accuracy (e.g., only 1.3% drop on GSM8K). Controlled experiments confirm gains come from fine-tuned structured generation, not prompt differences. Ablation studies show the substitution step has the largest impact. Extension experiments on DeepSeek-R1-7B and Mistral-8B validate generalization.

To sum up, our key contributions are:

1. We propose SCDC, a four-step standardized CoT paradigm that drastically reduces redundancy while preserving reasoning accuracy.

2. Experimentally, we provide extensive quantitative evidence across four datasets and three model families, 90% token reduction, 6.1–9.8× efficiency gain, accuracy maintained.

3. Ablation and prompt controls quantify each module's role and rule out prompt-only effects.

4. Low-cost deployment for low-latency, resource-constrained environments, and a reference for future research.

2 Related Work

2.1 Efficient Reasoning and Length Control in LLMs

Inference efficiency has become critical for LLM deployment. Existing techniques fall into four categories.

Xu et al. (2025) compress CoT into latent space.Propose Sketch of Thought (SoT) (Aytes et al., 2025) with dynamic reasoning paradigms, achieving up to 84% token reduction.

Post-hoc compression and hard budget impose a strict token limit. The S1 method (Muennighoff et al., 2025) truncates or injects "Wait" tokens, but truncation harms interpretability and continuation produces repetition.

Redundancy pruning removes low-information steps. Kang et al. (2024) propose C3oT, a compressor that maps long CoT to short ones with over 50% compression. Li et al. (2025) explore step-level pruning based on step entropy.

Distillation trains student models on compressed paths.DRP (Jiang et al., 2026), combining inference pruning with fine-tuning.

Another direction is test-time scaling (Snell et al., 2024; Wu et al., 2024) and reinforcement learning based length control (e.g., LCPO (Aggarwal and Welleck, 2025)). These add computation for accuracy, whereas we reduce baseline cost without sacrificing accuracy.

2.2 Structured, Template-Based, and Retrieval-Augmented Chain of Thought

Instead of quantitative reduction, qualitative restructuring of CoT can improve both efficiency and correctness. (Li et al., 2023) propose Structured CoT (SCoT) using programmatic structures (sequence, branch, loop) for code generation, reducing ambiguity. (Yang et al., 2025b) propose ReasonFlux, a hierarchical framework with about 500 high-level thought templates, trained via reinforcement learning to decompose complex problems.

Positioning of our work.Unlike C3oT, DRP (pruning after generation), LCPO (scalar budget optimization without structural improvement), or S1 (test-time prompts), SCDC builds a fixed four-step structure into the training data before fine-tuning, reshaping the model's intrinsic generation behavior via SFT. Unlike SCoT and ReasonFlux (aimed at accuracy), our goal is reasoning efficiency. Experiments show structured reasoning preserves and sometimes improves accuracy.

In summary, SCDC offers a simple, reproducible, model-agnostic baseline for token-efficient mathematical reasoning.

3 Method

3.1 Problem Definition and the Verbose Trap

We define a mathematical reasoning problem as a natural language query expecting a CoT and a final answer. Standard CoT often contains redun-

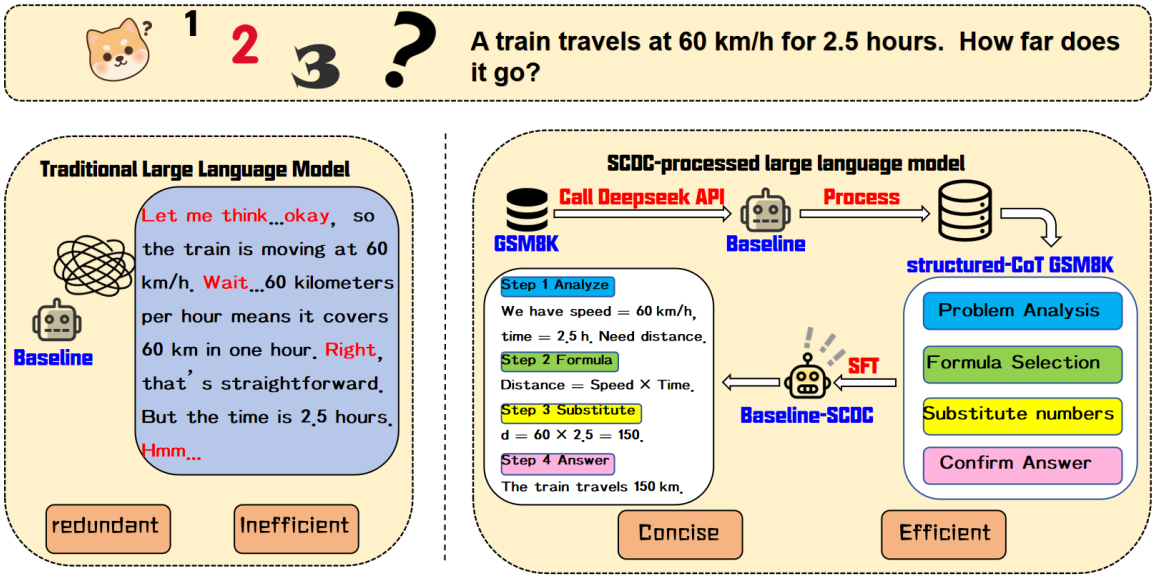

Figure 2: Comparison of existing large models (left) and our SCDC framework (right) on the same reasoning problem. SCDC restructures training data into a four-step format (Analysis → Formula → Substitution → Answer) and fine-tunes the base model to generate compact reasoning chains.

dant fillers, repetitions, and self-verifications—the **verbose trap**. Our goal: generate a concise yet logically complete CoT retaining only essential steps. We do not modify model architecture or loss; instead, we restructure training data into a minimal structured format and fine-tune with standard supervised fine-tuning (SFT) for efficient, concise reasoning.

3.2 Four Step Normative Reasoning Paradigm

For mathematical problem solving, we design a fixed domain specific reasoning template consisting of four sequential steps:

1.Problem analysis: Identify the known conditions, unknown quantities, constraints, and the goal to be solved.

2.Available formulas and theorems: List the mathematical principles, equations, and theorems relevant to the problem.

3.Data substitution and computation: Substitute the specific numerical values into the selected formulas and perform algebraic and arithmetic derivations.

4.Final answer: Output the result in a concise, standalone form.

This four step paradigm draws on Polya's problem solving framework while aligning with the natural decomposition logic of elementary and middle school mathematics problems. Ablation experiments (Section 4.4.3) verify that each step plays a distinct role in the reasoning process; removing any substep breaks logical completeness.

3.3 Structured Training Data Construction – SCDC

The core innovation of this section is the Structured CoT Data Construction Method (SCDC), which transforms a conventional training dataset into a standardized four step structured format. The complete procedure is as follows.

3.3.1 Source Data Selection

We use the GSM8K training set (Cobbe et al., 2021) to construct our structured CoT dataset for three reasons. First, its grade-school math problems require 2-5 steps, fitting our four-step paradigm (analysis, formula, substitution, answer). Second, GSM8K is a standard benchmark, enabling fair comparison with mainstream methods. Third, its 7,473 problems balance diversity and annotation cost. Clean answer formats also simplify evaluation. We validate SCDC on GSM8K then test on out-of-domain datasets (AQuA, SVAMP, Math500).

3.3.2 Rewriting Construction via the DeepSeek-R1 API

We construct the structured CoT dataset by calling the DeepSeek-R1 large language model API. For each problem in the GSM8K training set, we input the original question and the ground-truth answer

Algorithm 1 Structured CoT Data Construction (SCDC)

Input: GSM8K training set $D = \{(q_i, a_i, a2_i)\}$ (q_i: question, a_i: original answer, $a2_i$: secondary answer)
Output: Structured dataset $D' = \{(q_i, cot_i, a_i, a2_i)\}$ in JSON
Define four-step template T and prompt P with T and JSON spec
for each $(question, answer)$ in D **do**
  $response \leftarrow$ call_DeepSeek_API($P(question, answer)$)
  if $response$ valid **then**
    extract cot from JSON (strip "'json if needed)
    $D' \leftarrow D' \cup \{(question, cot, answer)\}$
  end if
end for
return D'

into the API, together with a tailored prompt. The full prompt template is provided in Appendix A; the core constraints are as follows:

Strict four-step format:

The output must strictly follow the fixed format:

1.Problem analysis

2.Available formulas and theorems

3.Data substitution and computation

4.Final answer

Correct and complete reasoning: Each reasoning step must be logically correct and sufficiently detailed to fully support the final answer; no critical information may be omitted.

3.3.3 Rewriting Process and Quality Control

The original questions and answers of the GSM8K dataset are batch-fed into the rewriting model, which outputs structured CoT text. After the first round of rewriting, a random 5% of samples are manually inspected. Common error types include:

1.Mixing the problem analysis and formula steps together;

2.Omitting the data substitution and computation step entirely;

3.Residual colloquial transition phrases;

4.Incorrect order of reasoning steps.

If the error rate for a batch exceeds 5%, the prompt is refined and the batch is regenerated. After at most two rounds of iterative correction, the final dataset achieves a 100% pass rate on manual spot checks.

3.3.4 Supervised fine-tuning with Structured Data

After constructing the structured CoT dataset, we perform standard SFT on a base LLM. The training objective is: given an input question, maximize the log likelihood of the model generating the structured reasoning process and the final answer.

The model is trained to predict the entire sequence in an autoregressive manner.

• Loss Function

We adopt the standard causal language modeling objective, cross-entropy loss over the target tokens. Let the sequence length be L, the target tokens be $y_1, \ldots, y_L$, and the model's predicted probability be $p(y_t \mid y_{<t}, q)$. The loss is:

$$\mathcal{L} = -\frac{1}{L} \sum_{t=1}^{L} \log p(y_t \mid y_{<t}, q)$$

We do not use additional strategies such as loss weighting or token masking; all tokens in the CoT and the answer share the same loss weight.

• Base Model

We use Qwen3-8B as the base model and apply LoRA low rank fine-tuning ((Hu et al., 2022)). Training updates only the LoRA adapter parameters, effectively reducing GPU memory usage and training time. The fine tuned model is uniformly named SCDC. During inference, the model requires no custom prompts; it autonomously outputs the standard fourstep reasoning content relying on the structured generation paradigm learned during training.

3.4 Inference Experimental Setup

After fine-tuning, we evaluate models on test sets without special prompts or post-processing.

For each test question, we input text using the model's tokenizer dialogue template. No system prompt is needed, as the model learned the four-step structured paradigm during training.

• Two-Stage Generation Strategy The model generates in two stages:

CoT generation: Outputs the standardized four-step reasoning (analysis $\rightarrow$ formula $\rightarrow$ substitution $\rightarrow$ answer). Stops at end-of-sequence or max length (4,096 tokens for standard test sets, 8,192 for harder ones). Greedy decoding ensures deterministic outputs.

Answer generation: Outputs the final answer.

• Token and Time Statistics For each sample, we record CoT token count, answer token count,

CoT generation time, and answer generation time.

• **Answer Extraction and Evaluation** We use the qwen2.5-math-evaluation toolkit (Team, 2024):

Normalization: Extract valid answer (numbers, expressions, boxed answers).

Cleaning: Remove punctuation, units, extraneous text; output standardized format.

Equivalence matching: Compare with ground truth using mathematical equivalence (floating-point rounding, equivalent fractions, etc.).

This fully automated pipeline is applied uniformly, ensuring fair and reproducible results.

3.5 Validity Principle of Structured Data Construction – The Sufficiency Hypothesis

SCDC removes redundant tokens, retaining only the minimal sufficient reasoning path. For any math problem, a shortest logically necessary sequence exists. Human solutions exceed this length; our four-step template approximates the lower bound.

Two observations support this: 90.0% compression with near-unchanged accuracy, proving removed tokens are redundant. Ablation (Sec. 4.4.3): removing the substitution step drops accuracy to random, it is essential; other steps matter less.

4 Experiments

4.1 Baselines

To validate SCDC, we compare against several baseline categories: standard CoT with different token budgets, prompt-only compression, four-step ablation variants, and cross-architecture models.

• **Standard Chain of Thought**

Base model Qwen3-8B uses a generic CoT prompt (Kojima et al., 2022). On GSM8K we set max token limits 512, 1024, 2048, 4096; other datasets default to 4096.

• **Prompt only compression baselines**

Two zero-shot prompts on GSM8K: standard CoT instruction vs. a structured four-step instruction (no fine-tuning). This isolates the effect of fine-tuning from prompt design.

• **Ablation variants**

Four SCDC ablations on GSM8K, each removing one reasoning step: analysis, formulas, substitution, or answer. Same hyperparameters as full SCDC.

• **Cross architecture baseline models**

We fine-tune SCDC on two other base models: DeepSeek-R1-7B (reasoning-specialized) and Mistral-8B (general-purpose). Both original and SCDC versions are evaluated on GSM8K.

• **Reference to external literature**

We cite two recent compression works for comparison:

- MixReasoning (Lu et al., 2025): adjusts reasoning depth per step, mixing detailed and concise inference to shorten CoT without accuracy loss.

- Step Entropy (Li et al., 2025): uses step entropy to identify redundant steps; a two-stage training (SFT+GRPO) prunes up to 80% of low-entropy steps.

Due to hardware and base model differences (both are also based on Qwen3-8B), we directly cite their published results (marked with * in Table 1). Table 1 also marks the reduction in tokens and the improvement in efficiency factors.

4.2 Experimental Models and Datasets

Base model: Qwen3-8B. SCDC fine-tunes it on the structured CoT dataset (Section 3). For cross-model generalization, we also fine-tune DeepSeek-R1-7B and Mistral-8B.

Four mathematical reasoning benchmarks:

- **GSM8K**: grade-school math word problems.
- **SVAMP**: similar to GSM8K but with more numeric distractors and variant templates.
- **AQuA**: algebraic and logical word problems.
- **Math500**: subset of MATH competition problems.

4.3 Evaluation Metrics

We use five metrics to measure the accuracy-efficiency trade-off:

- **Accuracy (ACC)**: proportion of correctly answered questions after normalization.
- **Average CoT length (Token)**: average number of tokens in the generated reasoning chain. The less represents the more concise.
- **Average inference latency (Latency)**: wall-time (seconds) for CoT generation.
- **Token efficiency (TE)**: defined as Yan et al. (2025):

$$TE = \frac{ACC \times 100}{Length}$$

Higher TE indicates better information utilization per token.

(a) GSM8K and AQuA

Method	Model	GSM8K				AQuA			
		Acc	Tokens	Lat	TE	Acc	Tokens	Lat	TE
Original (MixReasoning baseline)	Qwen3-8B	**95.83**	2239.00	–	4.3	–	–	–	–
MixReasoning*	Qwen3-8B	95.62	1217.00$^{\downarrow 45.6\%}$	–	7.9$^{\times 1.83}$	–	–	–	–
Original (Step Entropy baseline)	Qwen3-8B	94.46	3053.67	–	3.1	–	–	–	–
Step Entropy*	Qwen3-8B	94.39	2557.47$^{\downarrow 20.0\%}$	–	3.7$^{\times 1.19}$	–	–	–	–
Original (SCDC baseline)	Qwen3-8B	93.78	2192.73	53.77	4.3	**83.46**	3178.47	83.96	2.6
SCDC(Ours)	Qwen3-8B	92.49	**218.97**$^{\downarrow 90.0\%}$	**10.27**	**42.2**$^{\times 9.81}$	81.89	**348.43**	**16.12**	**23.5**

(b) SVAMP and MATH500

Method	Model	SVAMP				MATH500			
		Acc	Tokens	Lat	TE	Acc	Tokens	Lat	TE
Original (MixReasoning baseline)	Qwen3-8B	–	–	–	–	**93.20**	5192.00	–	1.8
MixReasoning*	Qwen3-8B	–	–	–	–	93.13	3531.00$^{\downarrow 32.0\%}$	–	2.6$^{\times 1.4}$
Original (Step Entropy baseline)	Qwen3-8B	–	–	–	–	91.37	7138.49	–	1.3
Step Entropy*	Qwen3-8B	–	–	–	–	91.13	5209.24$^{\downarrow 27.0\%}$	–	1.8$^{\times 1.4}$
Original (SCDC baseline)	Qwen3-8B	**95.70**	1698.26	42.38	5.6	76.00	3265.33	82.0	2.3
SCDC(Ours)	Qwen3-8B	94.60	**179.37**	**8.58**	**52.7**	78.80	**560.76**$^{\downarrow 82.8\%}$	**26.7**	**14.1**$^{\times 6.13}$

Table 1: Main results on GSM8K, AQuA, SVAMP, and MATH500. TE = ACC × 100 / Tokens. * marks results from prior work. Red superscripts indicate token reduction (↓) and TE improvement (×) relative to each method's own baseline.

4.4 Experimental Results

4.4.1 Main Results

Table 1(a) presents main results on four datasets (accuracy, tokens, latency, TE). Results with * are from prior work; absolute numbers are not fully comparable due to setting differences.

Table 1(b) reports relative efficiency improvements (token compression ratio and efficiency factor). SCDC significantly outperforms MixReasoning and Step Entropy, validating structured step-wise supervision for efficiency.

Compared to the base model, SCDC reduces tokens by 82–90% across all four datasets with only minor accuracy changes. Token efficiency (TE) improves by 6–10×. While MixReasoning and Step Entropy achieve only 20.0%–45.6% reduction and 1.3×–1.4× improvement. This confirms SCDC's superior effectiveness over post-hoc reduction methods.

4.4.2 Effect of Token Budget on Base Model

Table 2 analyzes the impact of different token budgets on the reasoning performance of the Qwen3-8B base model on the GSM8K test set, with our method provided for comparison. The table shows how the base model's accuracy, CoT length, inference time, and efficiency vary under different maximum token limits, thereby reflecting the trade off between reasoning completeness and computational cost in standard CoT generation.

To analyze the impact of the reasoning budget, we evaluate the base model under different maximum generation lengths: 512, 1024, 2048, and 4096 tokens, as shown in Table 2.

Increasing the budget from 512 to 4096 improves accuracy by 26.2% but increases token cost 4×, reducing TE by 67.4%. SCDC achieves comparable accuracy (92.49%) with 90% fewer tokens, decoupling reasoning quality from length.

4.4.3 Step-wise Ablation Analysis

Table 3 ablates each reasoning step in the structured pipeline. Removing different steps causes varying degradation, confirming their distinct contributions.

Removing Step-3 (Numerical Substitution) causes the largest accuracy drop: 92.49% → 42.91% (49.58%), showing it bridges abstract reasoning and computation. Removing Step 1 (Analy-

Model	Budget	Accuracy (%)	Tokens	Latency (s)	TE
Qwen3-8B	512	67.55	511.73	12.55	13.2
Qwen3-8B	1024	77.71	1009.48	26.88	7.7
Qwen3-8B	2048	87.11	1696.35	44.30	5.1
Qwen3-8B	4096	**93.78**	2192.73	53.77	4.3
SCDC(Ours)	4096	92.49	**218.97**	**10.27**	**42.2**

Table 2: Impact of token budget on Qwen3-8B (GSM8K). Accuracy, CoT length, latency, and efficiency vary with max tokens, revealing the trade-off between reasoning completeness and cost. Our method is shown for comparison.

Model	Ablation Step	Accuracy (%)	Tokens	Latency (s)	TE
Qwen3-8B (Original)	–	**93.78**	2192.73	53.77	4.3
Qwen3-8B	Step 1 (Analyze)	86.13	160.73	7.68	**53.6**
Qwen3-8B	Step 2 (Formulas)	91.81	173.42	8.10	52.9
Qwen3-8B	Step 3 (Substitute)	42.91	**137.13**	**6.45**	31.3
Qwen3-8B	Step 4 (Answer)	90.75	194.75	9.46	46.6
Qwen3-8B (SCDC)	–	92.49	218.97	10.27	42.2

Table 3: Ablation analysis of four reasoning steps on GSM8K (Step 1–4 removal). Performance measured by accuracy, CoT length, latency, and TE.

sis) drops to 86.13% (6.36%), Step 2 (Formula) to 91.81% (0.68%). Step 4 (Answer) only formats and verifies; its removal yields 90.75% (1.74%), since the model can often output the answer directly from Step 3.

Some ablations use fewer tokens (160–195 vs. 218.97 for full SCDC), but this does not guarantee better efficiency when accuracy is considered. For example, removing Step-1 reduces tokens but also accuracy, lowering overall efficiency. Overly compressed paths may lack semantic grounding.

While removing Step-2 yields high TE (52.9), its accuracy drop (1.97%) is less acceptable than the balanced trade-off of full SCDC. Removing Step-4 causes a larger accuracy drop (>3%), making it also suboptimal.

We choose full SCDC as the final configuration because it provides the best overall accuracy-efficiency balance.

4.4.4 Prompt Engineering vs. Fine-tuning and Generalization

Prompt-based methods achieve only limited efficiency (Table-4). SCDC surpasses the best prompt by 77% additional token reduction and 4.3× TE gain, proving that fine-tuning, not just prompt design, drives compression.

Cross-model evaluation shows SCDC generalizes well. On DeepSeek-R1-7B (strong reasoning), SCDC compresses tokens by 68.7%, improves TE by 3.0×, while maintaining accuracy. On Mistral-8B (weak reasoning), token compression reaches 30.5%, TE improves by 1.9×, and accuracy increases by 14.86%. Thus SCDC is a learned structural constraint that proactively compresses reasoning traces across model families, improving efficiency transferably.

5 Discussion

5.1 Mechanism Analysis: Why SCDC Works

Why does a fixed four-step structure achieve 87.8% token compression with near-zero accuracy loss? Three mechanisms explain this.

• **Eliminating semantic redundancy via fixed structure** Traditional CoT data contain cognitive fillers, repetitions, and self-verifications. Rewriting all samples into a strict four-step paradigm removes such noise at the source, preventing the model from learning verbosity—unlike post-hoc methods that cannot erase an already ingrained preference for long outputs.

• **Forming an implicit inductive bias for minimal complete reasoning** The four steps mirror Polya's problem-solving framework. Through extensive structured training, the model internalizes a termination logic: completing the four steps signals a closed reasoning loop. This inductive bias curbs

Model	Method	Accuracy (%)	Tokens	Latency (s)	TE
Qwen3-8B	–	**93.78**	2192.73	53.77	4.3
Qwen3-8B	Basic Prompt	88.32	1582.19	40.35	5.6
Qwen3-8B	Structured Prompt	92.04	933.42	23.44	9.9
Qwen3-8B	SCDC	92.49	218.97	10.27	42.2
DeepSeek-R1-7B	–	87.72	573.32	10.68	15.3
DeepSeek-R1-7B	SCDC	83.40	179.53	6.33	**46.5**
Mistral-8B	–	42.99	212.60	**4.06**	20.2
Mistral-8B	SCDC	57.85	**147.75**	2.82	39.2

Table 4: Comparison of prompt engineering vs. fine-tuning, and cross-model generalization. SCDC is compared against standard and structured prompts; also applied to DeepSeek-R1-7B and Mistral-8B to assess transferability.

overthinking and redundant verification, whereas standard CoT lacks an explicit stopping constraint.

• **Decoupling accuracy from reasoning length** Standard CoT ties accuracy to length (Table 2). SCDC breaks this coupling: on GSM8K it achieves 92.49% accuracy with only 200 tokens, while the base model needs >2,000 tokens. Consequently, token efficiency improves by an order of magnitude.

• **Corroboration with ablation results** Ablations (Table 3) confirm the mechanisms. Removing Step-3 collapses accuracy to near random, proving its indispensability. Removing Step-1 or Step-2 reduces accuracy but retains high TE due to implicit compensation. The full four-step template optimally balances accuracy and efficiency, precisely eliminating meaningless redundancy while preserving core logical dependencies.

5.2 Comparison with Existing Compression Methods

MixReasoning and Step Entropy are post-hoc reduction methods that prune tokens after training, relying on inference-time heuristics such as reasoning path mixing or entropy-based step filtering. These approaches achieve only 20.0%–45.6% token compression and 1.3×–1.4× TE improvements, as they do not modify the underlying training distribution. In contrast, SCDC eliminates redundancy at the source by restructuring training data before fine-tuning—specifically through structured stepwise supervision and content substitution. This proactive strategy yields 82%–90% token reduction and 6×–10× TE gains across four datasets, with negligible accuracy change . Although absolute numbers are not fully comparable across different settings due to variations in base models and evaluation protocols, the relative efficiency advantage of SCDC is substantial . This confirms that source-level redundancy elimination is a fundamentally more effective paradigm than post-hoc pruning for improving token efficiency

6 Conclusion

This paper proposed the Structured CoT Data Construction method (SCDC), which standardizes mathematical reasoning into a fixed four step template: problem analysis → formulas/theorems → numerical substitution → answer output. By enforcing this fixed step structure, SCDC eliminates extraneous tokens that typically arise from unstructured reasoning chains. Fine-tuning models on the restructured structured data enables them to generate highly concise CoT reasoning traces while preserving logical completeness. We conducted extensive experiments on four evaluation datasets and three base models. The results show that SCDC consistently achieves over 80% token compression and substantial TE improvement with only minor fluctuations in accuracy. These efficiency gains cannot be achieved by prompt engineering alone. Moreover, the compressed traces reduce the computational overhead during both training and inference without sacrificing logical correctness. Ablation studies further validate the critical role of each reasoning step, with numerical substitution being especially essential. Moreover, SCDC exhibits excellent cross model generalization, adapting robustly to both strong reasoning base models and lightweight models without native CoT generation capability, confirming its architecture agnostic nature.

7 Limitations

Currently, SCDC is designed specifically for elementary and middle school mathematical reasoning. Without re-customization, the fixed four step template is less effective for problems with complex solution structures (e.g., MATH500, AIME24) and does not readily generalize to other reasoning tasks such as commonsense reasoning or code generation.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 159 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Chenrui Fan, Ming Liu, Yixin Cao, and Lei Chen. 2025. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Saurabh Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Rodney Jehu-Appiah. 2026. Trivial vocabulary bans improve llm reasoning more than deep linguistic constraints. *arXiv preprint arXiv:2604.02699*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 8b. *arXiv preprint arXiv:2401.04088*.

Yiwei Jiang, David Li, and Francis Ferraro. 2026. Drp: Distilled reasoning pruning with skill-aware step decomposition for efficient large reasoning models. *arXiv preprint arXiv:2505.13975*.

Yu Kang, Zhendong Wang, Xin Wang, and Dong Li. 2024. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *arXiv preprint arXiv:2412.11664*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Jia Li, Yongmin Li, Ge Li, and Zhi Jin. 2023. Structured chain-of-thought prompting for code generation. *arXiv preprint arXiv:2305.06599*.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. 2025. Making slow thinking faster: Compressing llm chain-of-thought via step entropy. *arXiv preprint arXiv:2508.03346*. Accepted by ICLR 2026.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Haiquan Lu, Gongfan Fang, Xinyin Ma, Qi Li, and Xinchao Wang. 2025. Mixreasoning: Switching modes to think. *arXiv preprint arXiv:2510.06052*. Preprint.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Tatsunori B. Hashimoto, and Christopher Re. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2024. Learning to reason with llms. `https://openai.com/index/learning-to-reason-with-llms/`. OpenAI o1 blog post.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin,

Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 386 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Test-time scaling laws for language models. *arXiv preprint arXiv:2410.10813*.

Qwen Team. 2024. Qwen2.5-math evaluation harness. `https://github.com/QwenLM/Qwen2.5-Math/tree/main/evaluation`. GitHub repository.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Menghua Wu, Weijie Zhu, Qi Zhu, Yanjun Ma, Jiayi Zhang, Hong Yu, and Jian Ma. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference. *arXiv preprint arXiv:2408.11869*.

Silei Xu, Wen Xie, Ling Zhao, and Peng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

Jianzhi Yan, Yue Zhang, Xiaoyu Chen, Yifei Wang, Shiyang Li, Jia Liu, and Wei Zhang. 2025. From long to lean: Performance-aware and adaptive chain-of-thought compression via multi-round refinement. *arXiv preprint arXiv:2509.22144*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guoyin Wang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 41 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025b. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*.

A Appendix

A.1 Prompt Template for Data Construction

The following prompt was used to call the DeepSeek-R1 API for rewriting each GSM8K training example into the structured four-step format.

Compression Prompt (for each round):

```
You are an expert in mathematical
problems.
Please generate a four-step
chain-of-thought based on the
following question and answer.
Each step must be correct and
sufficient to solve the problem:
1. Analyze the problem:
2. Formulas or theorems that may be
used:
3. Substitute the data:
4. Give the answer:(State at the end:
"The final answer is xx (where xx
represents a pure number).")
Please produce a JSON object
containing only the fields:
"question": "Consistent with the
original answer",
"cot": "Complete four-step
chain-of-thought, each step clearly
solvable",
"answer": "Consistent with the
original answer"
question: question
original answer: answer
Ensure that each step of the CoT is
complete and strictly accurate,
enabling the derivation of the
answer.
```

A.2 Visualization charts of each data table

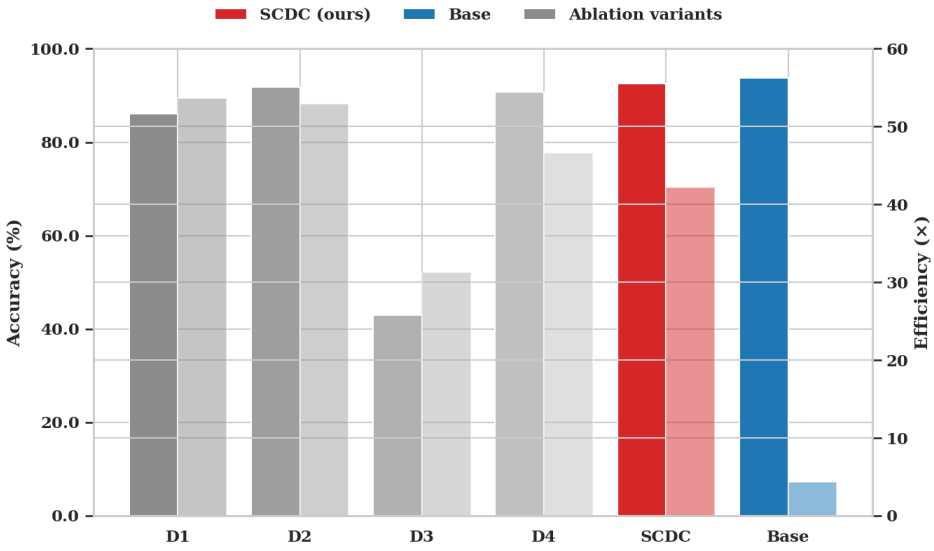

Figure 3: Ablation on GSM8K: removing Step 3 (Substitution) drops accuracy to 42.91%; full SCDC best balances accuracy and efficiency.

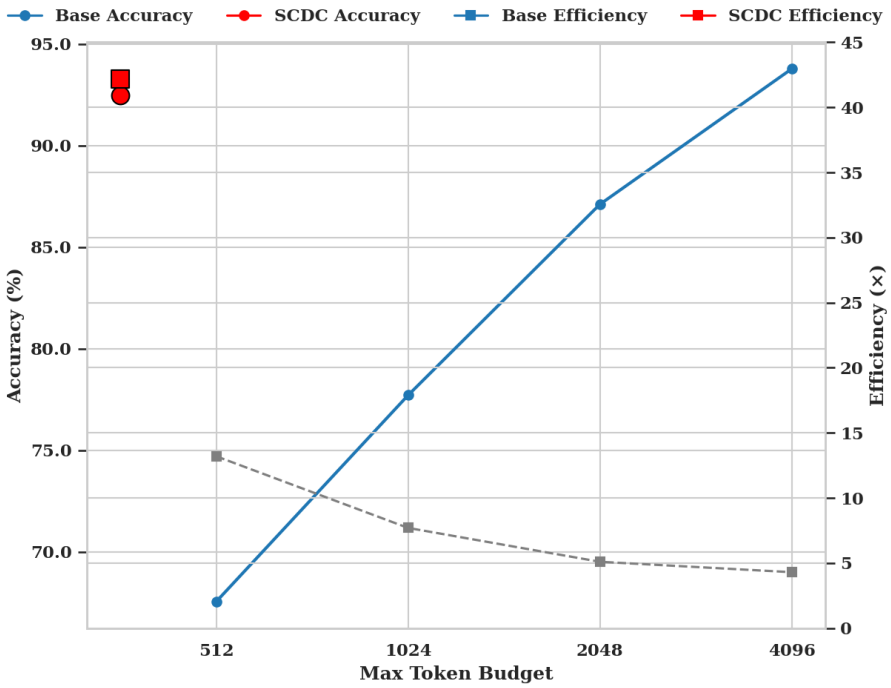

Figure 4: Token budget trade-off: higher budget improves accuracy but lowers TE. SCDC achieves 92.5% accuracy with 90% fewer tokens.

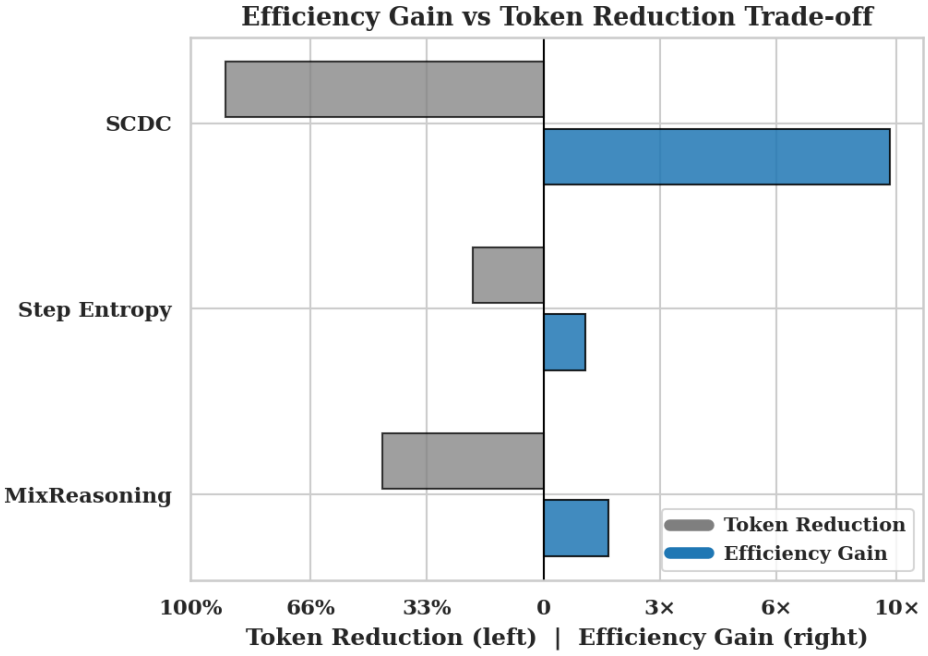

Figure 5: Token reduction and efficiency gain on GSM8K and MATH500. SCDC achieves 90.0% token reduction and 9.81× efficiency gain, outperforming TokenSkip and MACC.

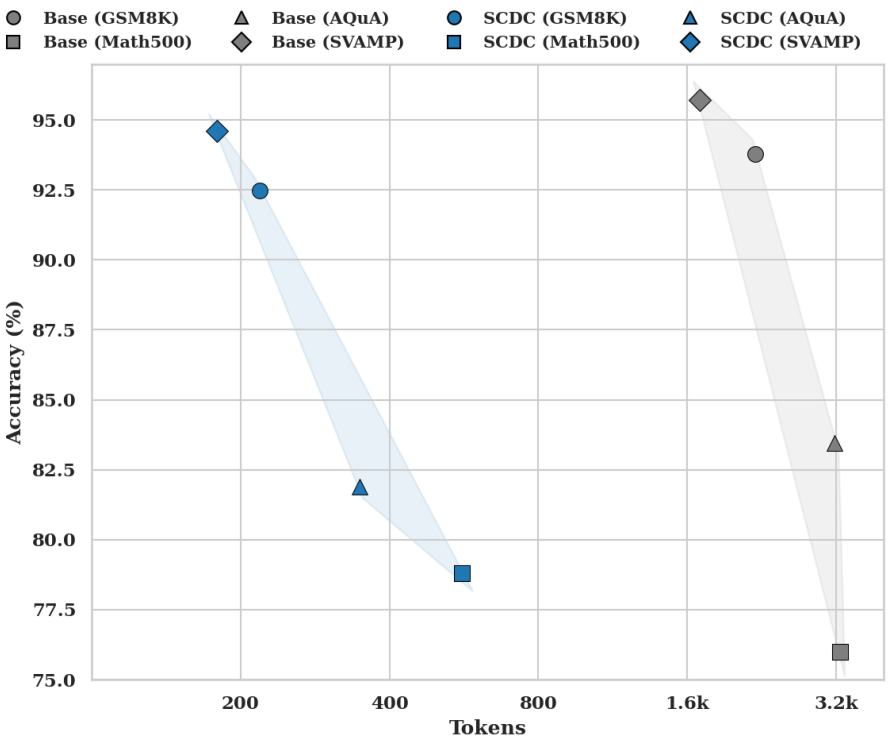

Figure 6: Accuracy vs. token efficiency on four datasets. SCDC achieves higher TE than the base model with competitive accuracy.

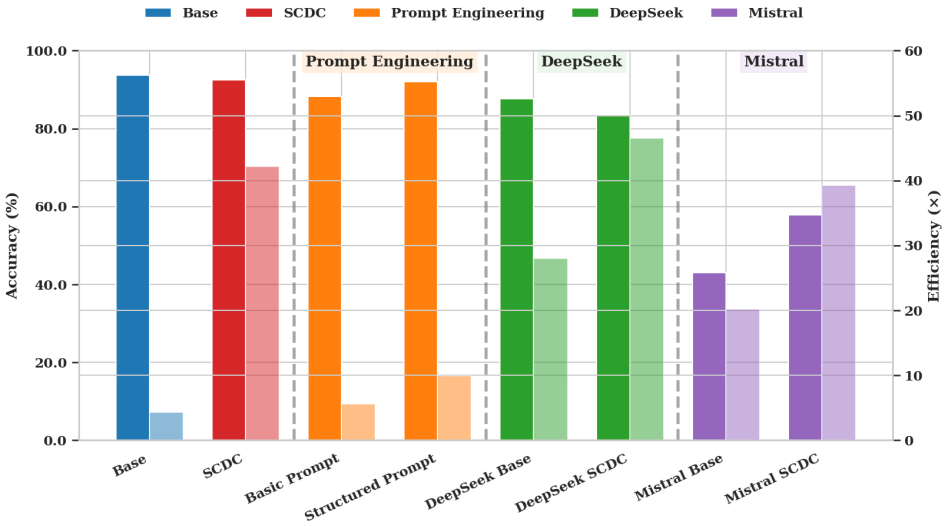

Figure 7: Prompt engineering vs. fine-tuning and cross-model generalization on GSM8K. SCDC (fine-tuned) achieves TE 42.2, outperforming basic prompt (5.6) and structured prompt (9.9). It generalizes to DeepSeek-R1-7B (1.7× TE) and Mistral-8B (1.9× TE).