From Long to Lean: Performance-aware and Adaptive Chain-of-Thought Compression via Multi-round Refinement

Anonymous ACL submission

Abstract

Chain-of-Thought (CoT) reasoning improves performance on complex tasks but introduces significant inference latency due to its verbosity. In this work, we propose Multiround Adaptive Chain-of-Thought Compression (**MACC**), a framework that leverages the *token elasticity phenomenon*—where overly small token budgets may paradoxically increase output length—to progressively compress CoTs via multiround refinement. This adaptive strategy allows MACC to dynamically determine the optimal compression depth for each input. Our method achieves an average accuracy improvement of 5.6% over state-of-the-art baselines, while also reducing CoT length by an average of 47 tokens and significantly lowering latency. Furthermore, we show that **test-time performance**—accuracy and token length—can be reliably predicted using interpretable features like perplexity and compression rate **on training set**. Evaluated across different models, our method enables efficient model selection and forecasting without repeated fine-tuning, demonstrating that CoT compression is both effective and predictable. Our code will be released to facilitate reproducibility and future research in CoT compression.

1 Introduction

Chain-of-Thought (CoT) reasoning significantly enhances the performance of large language models (LLMs) on complex tasks by decomposing questions into intermediate steps and reasoning sequentially (Nye et al., 2021; Wei et al., 2023; Kojima et al., 2023). Recent models such as OpenAI-o1 (OpenAI et al., 2024) and DeepSeek-R1 (DeepSeek-AI et al., 2025) demonstrate that Test-Time Scaling (TTS)—increasing CoT length during inference—can further boost reasoning accuracy (Snell et al., 2024; Tian et al., 2025; Yang et al., 2025). Nevertheless, longer CoTs substantially increase inference latency and memory usage due to larger key-value caches and the quadratic complexity of attention for sequence length (Dao et al., 2022; Liu et al., 2024a; SageScribe, 2025). To address the inefficiency of lengthy CoT reasoning, recent work has proposed a range of compression strategies. Token-level methods (TokenSkip (Han et al., 2025), C3oT (Kang et al., 2024)) prune redundant steps and use fine-tuning to preserve performance under compression (Jang et al., 2024; Liu et al., 2024b; Cui et al., 2025; Shen et al., 2025a; Yu et al., 2024). Prompt-based approaches (CoD (Xu et al., 2025), SoT (Aytes et al., 2025), TALE-EP (Han et al., 2025)) guide concise reasoning via routing prompts, minimal templates, or difficulty-aware designs (Yan et al., 2025; Zhang et al., 2025; Sui et al., 2025). Reward-based methods (O1-Pruner (Luo et al., 2025), DAST (Shen et al., 2025b), IBPO (Xu et al., 2025)) optimize reasoning length via reinforcement learning or preference modeling (Qu et al., 2025b; Yeo et al., 2025; Chen et al., 2025).

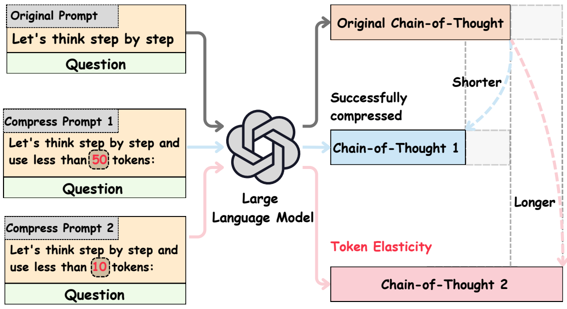

Figure 1: **Visualization of the *Token Elasticity* phenomenon**. As the prompt-specified token budget decreases, the actual token cost initially declines but eventually rebounds when the budget becomes too small.

However, prior approaches **lack fine-grained adaptability in managing the trade-off between compression and accuracy across diverse reasoning inputs**. Pruning- and prompt-based methods typically apply uniform compression, ignoring input-specific reasoning complexity, while reward-driven strategies optimize global preferences without instance-level control. For example, TokenSkip performs static token pruning and often degrades performance under tight budgets (Xia et al., 2025); CoD uses fixed prompts without controlling reasoning depth per instance (Xu et al., 2025); and TALE, though budget-aware, compresses in a single pass without adapting to input difficulty (Han et al., 2025). These methods lack the adaptive refinement needed to balance efficiency and accuracy in a controllable, input-sensitive manner.

To address these issues, we proposed Multi-round Adaptive Chain-of-Thought Compression, a framework grounded in the observed phenomenon of *token elasticity*-as shown in Figure 1—where overly aggressive compression may paradoxically increase token usage due to degraded generation quality. Our framework consists of three main components: (1) Chain-of-thought generation, (2) Multi-round progressive compression, and (3) Multitask fine-tuning. Given a question, we first prompt a model to generate a full reasoning trace, which is then progressively compressed through multiple rounds using compressor models. Each round removes redundant or verbose steps while preserving essential information, with dynamic control over compression ratios to adapt granularity. The final compressed CoTs are used to fine-tune models for efficient inference. Moreover, we proposed ***Performance Estimation Hypothesis*: test-time performance of the compressed CoT can be estimated *before fine-tuning*, based solely on a small set of interpretable features derived from the training set**—including compression rate, perplexity, original model training set accuracy, and average training set CoT length. We train lightweight regression models to predict both the downstream accuracy and token efficiency of the target model on the test set, enabling early-stage compression strategy selection without costly retraining. This predictive capacity makes our framework both efficient and performance-aware.

To sum up, our key contributions are:

1. We propose MACC, a multi-round compression framework that adaptively shortens reasoning chains while preserving essential information.

2. MACC achieves 5.6% higher accuracy, reduces reasoning by 47 tokens on average, and lowers latency, while supporting efficient model selection via interpretable metrics.

3. We propose ***Performance Estimation Hypothesis*** and demonstrate that fine-tuned performance can be predicted from interpretable features on the training set, enabling efficient strategy selection.

2 Related Work

2.1 LLM Reasoning and Token Cost

Recent advances in LLM reasoning techniques, particularly CoT prompting and its extensions such as self-consistency and tree-structured reasoning, have significantly enhanced complex problem-solving capabilities (Wei et al., 2023; Wang et al., 2023; Yao et al., 2023; Zhou et al., 2023). A variety of techniques have been proposed to enhance LLM reasoning. Chen et al. (2024) frame reasoning as latent distribution sampling optimized via variational methods, while Ho et al. (2023) leverages LLMs as reasoning teachers to distill knowledge into smaller models. But at the cost of substantially increased token consumption and computational overhead (Wang et al., 2024; Chiang and yi Lee, 2024; Bhargava et al., 2024). To improve efficiency, Li et al. 2021 propose a multi-hop filtering method to discard irrelevant reasoning, but it is limited to traditional neural networks and does not generalize to LLMs. Zheng et al. (2023) enhance inference speed via response length prediction and scheduling, yet their method operates only at the scheduling level without reducing token usage. Hao et al. (2024) lowers token cost by replacing decoded text with continuous latent tokens.

2.2 Chain-of-Thought Compression

To improve LLM inference efficiency, recent work explores compressing CoT reasoning while preserving answer correctness. These approaches can be broadly categorized into three paradigms (Liu et al., 2025; Qu et al., 2025a). First, Token-level compression methods, such as TokenSkip (Han et al., 2025) and C3oT (Kang et al., 2024), prune redundant tokens or steps and use supervised fine-tuning to maintain accuracy under varying compression ratios (Jang et al., 2024; Liu et al., 2024b). Second, Prompt design and sketch-based approaches, including CoD (Xu et al., 2025), SoT (Aytes et al., 2025), and TALE-EP (Han et al., 2025), guide concise reasoning using routing prompts, minimalist structures, or token-aware templates (Yan et al., 2025; Zhang et al., 2025; Sui et al., 2025). Third, Reward-based and preference optimization methods, such as O1-Pruner (Luo et al., 2025), DAST (Shen et al., 2025b), and IBPO (Xu et al., 2025), leverage reinforcement learning or preference objectives to balance length and accuracy during gen-

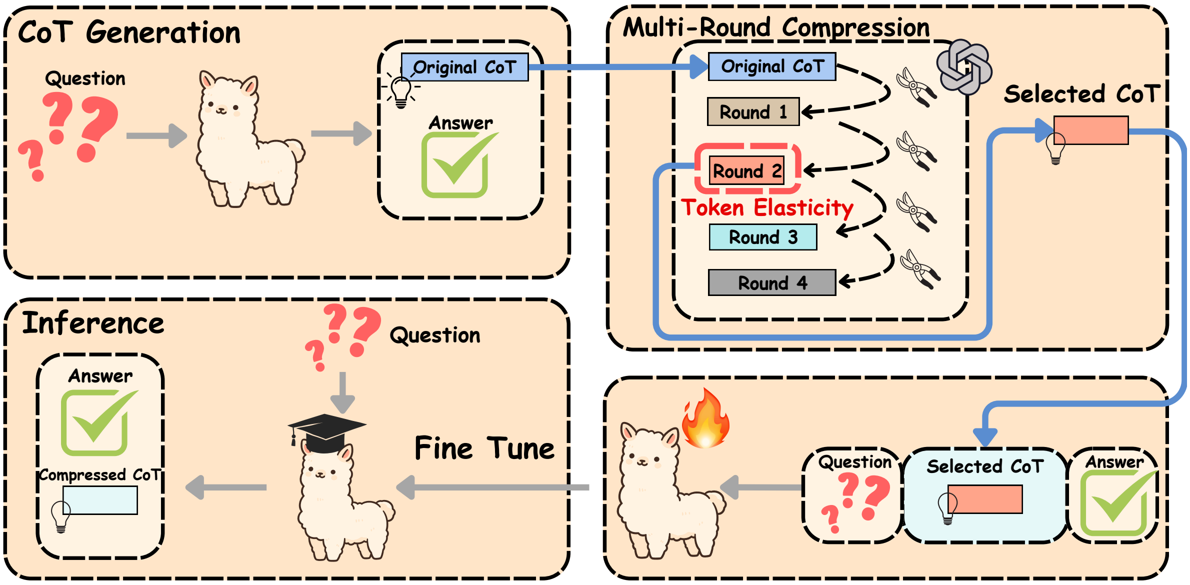

Figure 2: **Overview of MACC framework**. Given an input question, model first generates a full reasoning trace (CoT). The CoT is then progressively compressed through multiple rounds using a compressor model to remove redundancy while retaining essential reasoning content. The resulting compressed CoTs are used to fine-tune a smaller target model for efficient inference.

eration (Qu et al., 2025b).

While effective, most existing methods apply static or globally optimized strategies, lacking adaptability to instance-specific reasoning complexity. We address this gap through multiround adaptive compression guided by token elasticity.

3 Method

3.1 Token Elasticity Phenomenon

Recent studies have identified the *Token Elasticity* phenomenon in LLMs (Han et al., 2025), where overly tight token budgets can lead to unexpected increases in output length due to compensatory and redundant generation. This reveals a nonlinear relationship between token constraints and actual model behavior. Motivated by this, we adopt multi-round progressive compression strategy that gradually tightens the CoT length over several steps. This allows the model to adapt more smoothly, avoiding abrupt information loss and mitigating the adverse effects of over-compression.

3.2 CoT Generation

Let x denote the task input, and let D_{train} be the training dataset and $\mathcal{P}$ be the prompt template. The initial CoT C_0 is generated by the target model $\mathcal{S}$ conditioned on x, $\mathcal{P}$, and parameters $\theta_{\mathcal{S}}$ learned from D_{train}:

$$r_0 = \mathcal{S}(x \mid \mathcal{P},\, \theta_{\mathcal{S}}(D_{\text{train}})) \tag{1}$$

This initial CoT serves as the uncompressed sequence and is iteratively refined into shorter, semantically equivalent versions.

3.3 Multi-Round Progressive Compression

We then iteratively apply a sequence of N compression process $\{f_1, f_2, \ldots, f_N\}$, implemented via an API-based compressor model, to produce a series of compressed CoTs:

$$r_i = f_i(r_{i-1} \mid \mathcal{P}_{\text{compress}}), \quad \text{for } i = 1, 2, \ldots, N \tag{2}$$

Each f_i operates over the previous CoT C_{i-1} and is guided by a fixed compression prompt $\mathcal{P}_{\text{compress}}$. This design enables the gradual reduction of token length while attempting to preserve the correctness and reasoning validity of the original CoT.

To quantify the effect of compression at each stage, we define the compression rate at round i as:

$$\text{CR}_i = \frac{|r_i|_{\text{tok}}}{|r_0|_{\text{tok}}} \tag{3}$$

where tok denotes the number of tokens of the CoT sequence.

Our objective is to adaptively determine the maximum achievable compression rate for each input-specific CoT r, while preserving its reasoning validity. Instead of predefining a fixed target length or compression ratio, we propose a progressive compression framework that iteratively explores the compressibility of r over multiple rounds. In each compression round i, a shorter variant r_i is generated. The process terminates once the length of the newly generated chain $\text{len}_{\text{tok}}(r_i)$ exceeds that of the previous round $\text{len}_{\text{tok}}(r_{i-1})$, indicating that further compression leads to redundancy or loss of fidelity. In such cases, r_{i-1} is selected as the maximally compressed yet valid chain r^*. Formally, the maximally compressed yet valid chain r^* is selected as:

$$\begin{aligned} r^* = \arg\min_{r_j} \quad & |r_j|_{\text{tok}} \\ \text{subject to} \quad & |r_j|_{\text{tok}} < |r_{j-1}|_{\text{tok}} \end{aligned} \tag{4}$$

where $|r_j|_{\text{tok}}$ denotes the tokenized length of r_j. The selected r^* is subsequently used to fine-tune the target model. This adaptive criterion ensures compression proceeds only when meaningful token reduction is achieved, avoiding redundancy or semantic loss, and eliminating the need for manual compression targets. The process of the entire framework is shown in Algorithm 1

3.4 Multi-Task Fine-Tune

After obtaining the compressed rationale r^* via the multi-round progressive compression framework described in Section 3.2, we employ a multi-task fine-tuning strategy to train the target model. We unify training on both original and compressed CoT by prepending a special token `<compress>`, denoted as $\boldsymbol{t_c}$ in the following format, signals the model to reason based on a concise chain of thought. Each training sample is thus formatted as:

$$\mathcal{Q}\ \text{[EOS]}\ \boldsymbol{t_c}\text{[EOS] Compressed CoT}\ r^*$$

where $\langle \mathcal{Q}, \mathcal{A} \rangle$ indicates the $\langle$question, answer$\rangle$ pair. Formally, given a question $\boldsymbol{x}$, compression token $\boldsymbol{t_c}$, and the output sequence $\boldsymbol{y} = \{y_i\}_{i=1}^{l}$, which includes the compressed CoT r^* and the answer $\boldsymbol{a}$, we fine-tunes the target LLM $\mathcal{S}$, enabling it to perform chain-of-thought in a compressed pattern by minimizing

$$\mathcal{L} = \sum_{i=1}^{l} \log P\left(y_i \mid \boldsymbol{x}, \boldsymbol{t_c}, \boldsymbol{y}_{<i}; \boldsymbol{\theta}_{\mathcal{S}}\right) \tag{5}$$

Algorithm 1: MACC: Multi-Round Adaptive Chain-of-Thought Compression for Dataset Construction

```
Input: Training set $D = \{x_j\}_{j=1}^{N}$, target
       model $\mathcal{S}$, compressor model $\mathcal{C}$,
       initial prompt $\mathcal{P}$, compression
       prompt $\mathcal{P}_{\text{compress}}$, max rounds $T$
Output: Compressed training set
        $D' = \{(x_j, r_j^*)\}_{j=1}^{N}$
Initialize $D' \leftarrow \emptyset$;
foreach $x_j \in D$ do
    $r_0 \leftarrow \mathcal{S}(\mathcal{P}(x_j))$;
    $r^* \leftarrow r_0$;
    for $i = 1$ to $T$ do
        $r_i \leftarrow \mathcal{C}(\mathcal{P}_{\text{compress}}(x_j, r_{i-1}))$;
        if $\text{len}_{tok}(r_i) > \text{len}_{tok}(r_{i-1})$ then
            break;
        $r^* \leftarrow r_i$;
    Add $(x_j, r^*)$ to $D'$;
return $D'$
```

where $\boldsymbol{y} = \left\{c_1^*, \cdots, c_{m'}^*, a_1, \cdots, a_t\right\}$. To retain the reasoning capabilities of LLMs, we include a fraction of original CoT trajectories in the training data, without setting $\boldsymbol{t_c}$.

3.5 Inference

MACC performs inference via autoregressive decoding. Given a question x and a compression token $\boldsymbol{t_c}$, the input prompt follows the fine-tuning format: $\mathcal{Q}$ [EOS] $\boldsymbol{t_c}$ [EOS]. The LLM $\mathcal{S}$ then generates the output sequence $\hat{y}$ step by step:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}^*} \sum_{j=1}^{l'} \log P\left(y_j \mid \boldsymbol{x}, \boldsymbol{t_c}, \boldsymbol{y}_{<j}; \boldsymbol{\theta}_{\mathcal{S}}\right)$$

where $\hat{y} = \{\hat{c}_1, \ldots, \hat{c}_{m''}, \hat{a}_1, \ldots, \hat{a}_{t'}\}$ represents the generated output sequence, consisting of CoT tokens $\hat{c}$ and final answer tokens $\hat{a}$. The training and inference workflow of MACC is illustrated in Figure 2.

3.6 Performance Estimation Hypothesis

Empirical observations suggest that the downstream performance of compressed CoT reasoning—measured by fine-tuned accuracy and CoT length—is correlated with interpretable features such as compression rate and perplexity. Based on this, we hypothesize that compressed performance can be predicted prior to fine-tuning:

Given a compressor model $\mathcal{C}$, a target model $\mathcal{S}$, and a training set $\mathcal{D}_{train}$, we define a feature vector $\boldsymbol{x}$ that encodes compression-related statistics, including the compression rate, perplexity, original CoT length, accuracy of both compressor and target model (the answer accuracy on training set). The downstream performance $\boldsymbol{y} = [\mathrm{Acc}, \mathrm{Len}]$ on test set $\mathcal{D}_{test}$ can be estimated as follows:

$$P(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{y}) \cdot P(\boldsymbol{y})}{P(\boldsymbol{x})} \tag{6}$$

In practice, we approximate the posterior using Bayesian regression (e.g., Bayesian Ridge), yielding predictive distributions:

$$\boldsymbol{y} \sim \mathcal{N}\left(\begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix}, \begin{bmatrix}\sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2\end{bmatrix}\right) \tag{7}$$

where $\mu(\cdot)$ and $\sigma^2(\cdot)$ denote the posterior mean and variance conditioned on the compression features.

This probabilistic formulation enables principled estimation of post-compression performance, supporting early-stage strategy selection without costly full fine-tuning. In practice, we implement this with Bayesian ridge regression, which provides both predictive means and uncertainty estimates. The effectiveness of this hypothesis is empirically validated in Section 4.4.2.

4 Experiments

4.1 Baseline

To benchmark the effectiveness of our proposed compression framework, we compare against two recent and representative methods for efficient CoT reasoning:

- **TokenSkip** (Xia et al., 2025). Compresses chain-of-thought by pruning low-importance tokens and fine-tuning the model to generate concise rationales based on a target compression ratio.
- **TALE** (Han et al., 2025). TALE controls CoT length by estimating token budgets from problem complexity, enabling efficient inference with minimal accuracy loss.
- **Prompt**. Following Xia et al. (2025), we guides the LLM to shorten its CoT output by incorporating explicit instructions into the prompt. For example, the input may include a directive such as: "Please reduce 50% of the words in your Chain-of-Thought reasoning."

4.2 Models and Datasets

We evaluate the MACC framework using diverse models and two math benchmarks: GSM8K and MATH. `GPT-4o-mini` serves as the compressor for its strong reasoning and efficiency (OpenAI, 2024), while models include LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2024), fine-tuned with compressed rationales via our multi-task strategy in Section 3.4. For MATH, we evaluate on a 500-example subset (MATH-500) from Lightman et al. (2023), which reliably reflects full-benchmark performance.

Evaluation Metrics

We evaluate MACC using four key metrics to comprehensively assess reasoning performance and efficiency: accuracy (percentage of correctly answered questions), average CoT token count (to quantify reasoning verbosity), inference latency, and **Token Efficiency**.

We define Token Efficiency a composite metric defined as:

$$\text{Token Efficiency} = \frac{Acc}{Length} \times 100$$

where Acc is the accuracy and $Length$ is the average number of tokens in the generated Chain-of-Thought. This metric reflects the trade-off between accuracy and token usage—higher values indicate more efficient reasoning. Together, these metrics offer a comprehensive view of both the performance and cost-effectiveness of each method.

Implementation Details

We adopt LoRA (Hu et al., 2021), an efficient and reproducible approach that has been widely verified as effective in LLM fine-tuning, to train our models. The rank r is set to 8, and the scaling parameter α is set to 16. MACC is characterized by its low training cost, with training taking -1.5 hours for the 7B model. During inference, the maximum number of tokens is set to 512 for GSM8K and 1024 for MATH. We implement our training process using the LLaMA-Factory (Zheng et al., 2024) library

4.3 Results

4.3.1 Main Result

Table 1 presents a comprehensive comparison of five CoT compression methods—Original, Prompt, TokenSkip, TALE, and our proposed **MACC**—on

Methods	**Model**	**GSM8K**				**MATH-500**			
		Acc. ↑	Tokens ↓	Lat. (s) ↓	TE. ↑	Acc. ↑	Tokens ↓	Lat. (s) ↓	TE. ↑
ORIGINAL	LLaMA-3.1-8B	86.2	213.17	1.33	40.44	48.6	502.60	6.83	9.67
	Qwen2.5-7B	91.4	297.83	1.96	30.69	71.4	574.85	6.65	12.42
	Qwen2.5-3B	83.7	314.87	1.99	26.58	61.6	578.51	5.90	10.65
PROMPT	LLaMA-3.1-8B	76.9	136.48	1.08	56.35	37.6	335.92	3.78	11.19
	Qwen2.5-7B	82.7	175.83	1.12	47.03	49.1	355.47	3.45	13.81
	Qwen2.5-3B	71.3	185.22	1.28	38.49	42.0	423.88	3.98	9.91
TOKENSKIP	LLaMA-3.1-8B	78.2	113.05	0.86	69.17	40.2	292.17	3.53	13.76
	Qwen2.5-7B	86.0	151.44	0.89	56.79	52.8	330.8	3.12	15.96
	Qwen2.5-3B	74.4	**170.55**	**1.02**	**43.62**	44.2	396.29	3.74	11.15
TALE	LLaMA-3.1-8B	78.5	139.63	0.88	56.22	-	-	-	-
MACC (OURS)	LLaMA-3.1-8B	**81.1**	**88.57**	**0.75**	**91.57**	**44.0**	**198.04**	**2.05**	**22.22**
	Qwen2.5-7B	**86.2**	**148.76**	**0.87**	**57.94**	**58.4**	**254.89**	**2.02**	**22.91**
	Qwen2.5-3B	**80.5**	216.25	1.33	37.22	**54.0**	**265.80**	**2.20**	**20.32**

Table 1: **Performance comparison on GSM8K and MATH-500 using three base models across five CoT compression methods**: Original, Prompt-only, TokenSkip, TALE, and our proposed MACC. Metrics include Accuracy, Token count, Latency (s), and Token Efficiency (Accuracy per token, scaled by 100). **Bold** values indicate the best results under each setting.

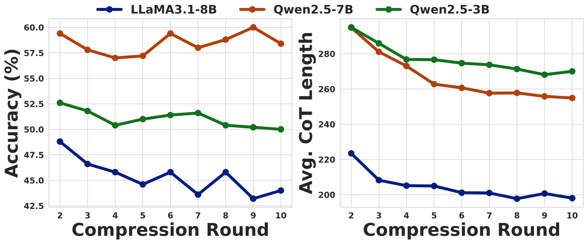

Figure 3: **Accuracy (left) and average CoT length (right) across compression rounds for three Models**. Larger models tend to retain higher accuracy under aggressive compression.

GSM8K and MATH-500, using three instruction-tuned models: LLaMA-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-3B-Instruct. MACC consistently achieves the best trade-off between accuracy and efficiency across all settings.

On GSM8K, MACC improves accuracy over TokenSkip by +2.9 points on LLaMA-3.1-8B-Instruct and +0.2 points on Qwen2.5-7B-Instruct, while reducing the average CoT length by over 20%. This leads to a clear gain in Token Efficiency, as fewer tokens are used without compromising performance. On MATH-500, which involves more complex reasoning, MACC continues to outperform TokenSkip, achieving +3.8 and +5.6 point gains in accuracy on LLaMA and Qwen2.5-7B, respectively. It also reduces inference latency (from 3.53s to 2.05s on LLaMA), highlighting its practicality for efficient reasoning.

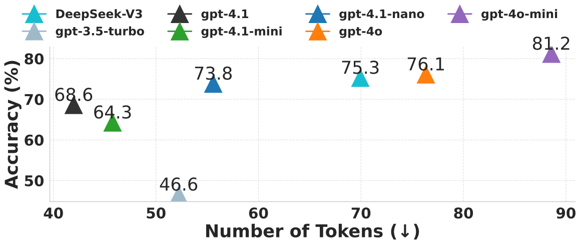

Figure 4: **Comparison of compressed CoT performance across different compressors on llama3.1-8B-Instruct after 5 round**. Each point represents the accuracy and average CoT length achieved by different compressors. Models toward the upper-left indicate better trade-offs between efficiency and accuracy.

Even with the smaller Qwen2.5-3B model, MACC shows consistent improvements, confirming its robustness across different model capacities. In contrast, Prompt performs worse than TokenSkip in both accuracy and efficiency, leading to the lowest Token Efficiency overall. These results demonstrate that MACC effectively compresses CoT while preserving reasoning quality and reducing computational cost.

4.3.2 Effect of Compression Rounds

Figure 3 shows how fine-tuned accuracy and average CoT length evolve as the number of compression rounds increases under the MACC framework. As expected, the average length of the reasoning chains steadily decreases across rounds, demon-

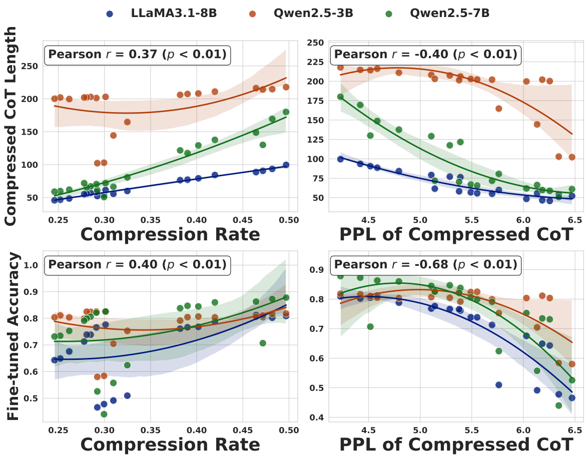

Figure 5: **Effect of compression rate and perplexity on compressed CoT length and fine-tuned accuracy across different models.**. Each subplot shows the relationship between a compression feature and a target metric, with model-specific quadratic fits.

strating that MACC's progressive strategy effectively eliminates redundant content while preserving the information necessary for reasoning.

The effect on accuracy, however, varies with model scale. Larger models like Qwen2.5-7B are more robust, maintaining high accuracy even with shorter CoTs. In contrast, smaller models suffer greater performance drops under aggressive compression, likely due to limited capacity to recover from incomplete rationales.

These results support the design of MACC's adaptive stopping mechanism, which halts compression once further reduction harms accuracy. They also suggest that compression depth should be tailored to the model's capacity, avoiding overcompression. Full results are provided in Table 8, Table 9, and Table 10 in Appendix A.

4.3.3 Effect of Different Compressor Models

Next, we investigate how the choice of compressor model affects the quality of CoT compression, using LLaMA-3.1-8B-Instruct as the target model and GSM8K as the evaluation benchmark. As shown in Figure 4, different compressors exhibit distinct trade-offs between compressed rationale length and fine-tuned accuracy across multiple compression rounds.

High-capacity compressors such as `GPT-4o` and `GPT-4o-mini` maintain high accuracy while significantly reducing CoT length, showing strong ability to preserve essential reasoning under compression. In contrast, lower-capacity models like `GPT-4.1-nano` and `GPT-3.5-turbo` cause greater accuracy drops, indicating weaker semantic fidelity and limited robustness in preserving logical consistency.

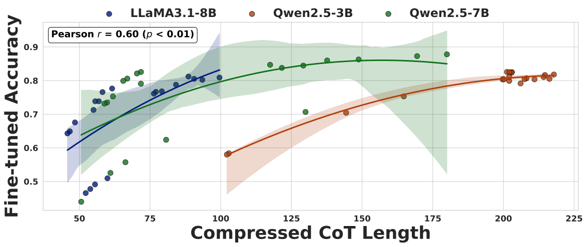

Figure 6: **Relationship between compressed CoT length and fine-tuned accuracy across models**. Each point denotes a sample colored by target model. Longer compressed CoTs yield higher accuracy, suggesting the need to preserve key reasoning steps.

Overall, our results highlight that higher-capacity compressors tend to produce more compact yet informative rationales, enabling better fine-tuning outcomes. These findings underscore the importance of selecting an appropriate compressor model in multi-round compression pipelines, especially when targeting smaller or more sensitive student models.

4.4 Estimating Compressed CoT Effectiveness

Given the significant impact of compressor choice on the quality of CoT reasoning, it becomes increasingly important to assess, a priori, how a target model will perform when fine-tuned on compressed rationales. Instead of relying on exhaustive training and evaluation for every possible compression strategy, we investigate whether the downstream accuracy of the target model can be effectively predicted in advance. To this end, we explore a lightweight performance estimation framework conditioned on both the chosen compressor and the architecture of the target model. Specifically, we aim to estimate fine-tuned accuracy using a set of interpretable and readily available features extracted from compressed CoTs—such as token length, perplexity, and compression ratio. This approach enables efficient compression strategy selection without incurring the full cost of model retraining, and offers a practical pathway toward scalable and adaptive CoT compression.

Based on *Performance Estimation Hypothesis*, we model their relationship in a probabilistic manner in Section 4.4.2.

4.4.1 Analyse of Features

To better understand the factors influencing post-compression performance, we analyze how interpretable features correlate with both the average

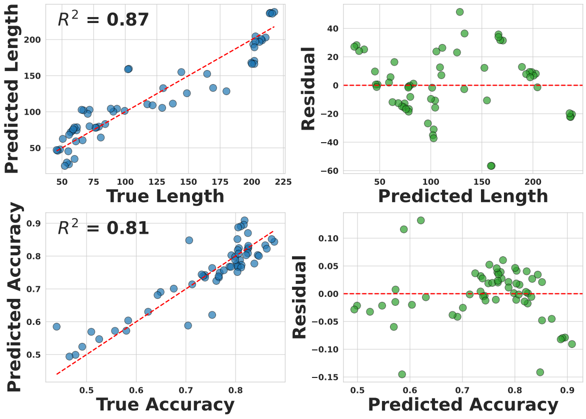

Figure 7: **Bayesian Ridge regression results for predicting compressed CoT performance using features obtained from training set.** Top row shows predictions and residuals for CoT length after compression; bottom row for fine-tuned accuracy. Predictions are based on training-set features before fine-tuning, demonstrating strong alignment with ground truth.

CoT length and the fine-tuned accuracy. Figure 5 show that compression rate and perplexity are moderately correlated with the resulting CoT length ($r = 0.37$ and $r = -0.40$, respectively), serving as a proxy for reasoning verbosity.

Figure 6 illustrates the relationship between the length of compressed CoT sequences and the downstream accuracy of fine-tuned models. A clear positive correlation is observed: longer compressed CoTs tend to yield higher fine-tuned accuracy. This highlights the need to preserve essential reasoning during compression, as over-truncation harms fidelity and performance. The results support the core design of the MACC framework, which adaptively determines compression depth to balance brevity and correctness. Additionally, the figure reveals that higher-capacity models achieve better accuracy at comparable CoT lengths, suggesting an interaction between model capacity and robustness to compression.

4.4.2 Evaluating Predictability of Compressed CoT Effectiveness

To validate the *Performance Estimation Hypothesis*, we test whether fine-tuned accuracy can be predicted from interpretable features before training. We train regression models that take compression-related statistics—such as compression rate, perplexity of compressed CoT, original CoT length, and Compressor Accuracy—as inputs to estimate the downstream performance.

We experiment with both random forest and Bayesian ridge regressors under 5-fold cross-validation. As shown in Figure 7, the predicted accuracies closely align with the true values, with low residuals across samples. This indicates that compressed CoT effectiveness is highly predictable using lightweight feature sets.

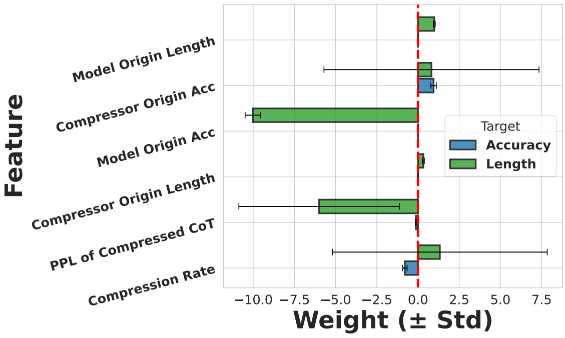

Figure 8: **Bayesian Ridge regression weights for predicting fine-tuned accuracy and compressed CoT length using features extracted from the *training set*.** Bars show feature importance (mean ± std), reflecting key factors for downstream performance.

Figure 8 shows that original CoT accuracy and length are the strongest predictors of fine-tuned accuracy, followed by compressed CoT perplexity. Notably, the compression rate itself contributes the least predictive signal, suggesting that surface-level reduction is less indicative of reasoning quality compared to semantic coherence or input-specific difficulty. These findings highlight the value of model- and CoT-aware features for estimating compression quality.

These results support our hypothesis in Section 3.6: training-set features reliably predict performance, enabling efficient compressor selection without fine-tuning.

5 Conclusion

This paper presents **MACC**, a novel framework for adaptive and performance-aware compression of CoT reasoning. By leveraging the token elasticity phenomenon and multi-round refinement, MACC substantially reduces the length of reasoning chains with only minimal loss in accuracy. Extensive experiments across models and benchmarks demonstrate that MACC consistently outperforms prior approaches in terms of efficiency, accuracy, and latency. Furthermore, we show that key metrics such as post-compression accuracy and token usage can be reliably predicted using interpretable features. This enables informed compressor selection and efficient deployment, improving the scalability of CoT-based inference.

Limitations

While MACC achieves substantial gains in compression efficiency and reasoning accuracy, it has several limitations. The reliance on external compressors (e.g., GPT-4o-mini) introduces potential model bias and limits applicability in low-resource settings. The multi-round process, while adaptive, adds preprocessing latency that may affect deployment speed. Additionally, compression prompts are task-agnostic, which may hinder performance on domains requiring structured reasoning. Lastly, our performance estimation relies on a limited feature set, which may not generalize well to unseen model-task combinations.

References

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

Aman Bhargava, Cameron Witkowski, Shi-Zhuo Looi, and Matt Thomson. 2024. What's the magic word? a control theory of llm prompting. *Preprint*, arXiv:2310.04444.

Haolin Chen, Yihao Feng, Zuxin Liu, Weiran Yao, Akshara Prabhakar, Shelby Heinecke, Ricky Ho, Phil Mui, Silvio Savarese, Caiming Xiong, and Huan Wang. 2024. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *Preprint*, arXiv:2411.04282.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

Cheng-Han Chiang and Hung yi Lee. 2024. Over-reasoning and redundant calculation of large language models. *Preprint*, arXiv:2401.11467.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, Suhang Wang, Yue Xing, Jiliang Tang, and Qi He. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *Preprint*, arXiv:2502.13260.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Preprint*, arXiv:2205.14135.

DeepSeek-AI, Daya Guo, and Dejian Yang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Aaron Grattafiori, Abhimanyu Dubey, and Abhinav Jauhri. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. *Preprint*, arXiv:2412.18547.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. *Preprint*, arXiv:2212.10071.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Joonwon Jang, Jaehee Kim, Wonbin Kweon, and Hwanjo Yu. 2024. Verbosity-aware rationale reduction: Effective reduction of redundant rationale via principled criteria. *Preprint*, arXiv:2412.21006.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2024. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *Preprint*, arXiv:2412.11664.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Chenliang Li, Bin Bi, Ming Yan, Wei Wang, and Songfang Huang. 2021. Addressing semantic drift in generative question answering with auxiliary extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 942–947, Online. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. 2024a. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *Preprint*, arXiv:2409.10516.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024b. Can language models learn to skip steps? *Preprint*, arXiv:2411.01855.

Yue Liu, Jiaying Wu, Yufei He, Hongcheng Gao, Hongyu Chen, Baolong Bi, Jiaheng Zhang, Zhiqi Huang, and Bryan Hooi. 2025. Efficient inference for large reasoning models: A survey. *Preprint*, arXiv:2503.23077.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *Preprint*, arXiv:2501.12570.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *Preprint*, arXiv:2112.00114.

OpenAI, :, Aaron Jaech, and Adam Kalai. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2024. Gpt-4o mini: Advancing cost-efficient intelligence. Accessed: 2025-05-07.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-Sheng Hua, Bowen Zhou, and Yu Cheng. 2025a. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *Preprint*, arXiv:2503.21614.

Yuxiao Qu, Matthew Y. R. Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. 2025b. Optimizing test-time compute via meta reinforcement fine-tuning. *Preprint*, arXiv:2503.07572.

AI SageScribe. 2025. Kv cache: Optimizing transformer inference.

Xuan Shen, Yizhou Wang, Xiangxi Shi, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. 2025a. Efficient reasoning with hidden thinking. *Preprint*, arXiv:2501.19201.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025b. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *Preprint*, arXiv:2503.04472.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Yuan Sui, Yufei He, Tri Cao, Simeng Han, and Bryan Hooi. 2025. Meta-reasoner: Dynamic guidance for optimized inference-time reasoning in large language models. *Preprint*, arXiv:2502.19918.

Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yunjie Ji, Yiping Peng, Han Zhao, and Xiangang Li. 2025. Think twice: Enhancing llm reasoning by scaling multi-round test-time thinking. *Preprint*, arXiv:2503.19855.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024. Reasoning in token economies: Budget-aware evaluation of llm reasoning strategies. *Preprint*, arXiv:2406.06461.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

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

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *Preprint*, arXiv:2502.18600.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. 2025. Inftythink: Breaking the length limits of long-context reasoning in large language models. *Preprint*, arXiv:2503.06692.

An Yang, Baosong Yang, and Binyuan Hui. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2025. Towards thinking-optimal scaling of test-time compute for llm reasoning. *Preprint*, arXiv:2502.18080.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *Preprint*, arXiv:2502.03373.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *Preprint*, arXiv:2407.06023.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. 2025. Lightthinker: Thinking step-by-step compression. *Preprint*, arXiv:2502.15589.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 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)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.

Zangwei Zheng, Xiaozhe Ren, Fuzhao Xue, Yang Luo, Xin Jiang, and Yang You. 2023. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline. *Preprint*, arXiv:2305.13144.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

A Appendix

A.1 Correlation Analysis Between Compression Features and Downstream Performance

Tables 2 and 3 report the Pearson correlation coefficients between various compression-related features (e.g., perplexity, compression rate, original CoT length) and two key downstream metrics: accuracy and average CoT length after fine-tuning.

Specifically, Table 2 shows correlations with post-finetuning accuracy, while Table 3 focuses on average CoT length. Across models, compressed perplexity and compression rate exhibit strong correlations with performance outcomes, validating their predictive utility. Notably, compressed perplexity tends to negatively correlate with accuracy and positively with output length, reinforcing its role as a proxy for semantic loss during compression. These results support the feasibility of estimating performance outcomes based on interpretable, compression-time statistics.

Table 2: Pearson correlation between compression-related features and fine-tuned accuracy.

Model	Feature	n	Pearson r	p-value
LLaMA-3.1-8B	CR	20	0.54	1.30×10^{-2}
LLaMA-3.1-8B	PPL	20	-0.81	1.30×10^{-5}
LLaMA-3.1-8B	Len	20	0.65	1.80×10^{-3}
Qwen2.5-3B	CR	20	0.21	3.90×10^{-1}
Qwen2.5-3B	PPL	20	-0.56	1.10×10^{-2}
Qwen2.5-3B	Len	20	0.98	1.20×10^{-14}
Qwen2.5-7B	CR	20	0.42	6.50×10^{-2}
Qwen2.5-7B	PPL	20	-0.70	6.50×10^{-4}
Qwen2.5-7B	Len	20	0.63	2.80×10^{-3}

Table 3: Pearson correlation between compression-related features and compressed CoT length.

Model	Feature	n	Pearson r	p-value
LLaMA-3.1-8B	CR	20	0.99	5.50×10^{-16}
LLaMA-3.1-8B	PPL	20	-0.93	1.90×10^{-9}
LLaMA-3.1-8B	Len	20	1.00	3.50×10^{-20}
Qwen2.5-3B	CR	20	0.37	1.10×10^{-1}
Qwen2.5-3B	PPL	20	-0.67	1.30×10^{-3}
Qwen2.5-3B	Len	20	1.00	9.00×10^{-21}
Qwen2.5-7B	CR	20	0.96	5.00×10^{-11}
Qwen2.5-7B	PPL	20	-0.91	3.60×10^{-8}
Qwen2.5-7B	Len	20	0.99	3.80×10^{-17}

A.2 Prompt Templates

We provide the prompt templates used for both initial CoT generation and subsequent compression rounds.

Initial CoT Generation Prompt:

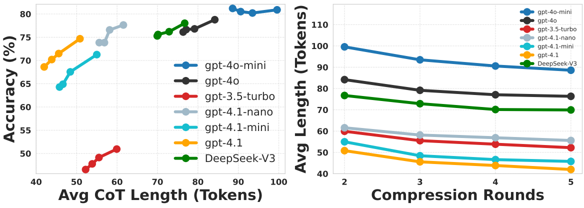

Figure 9: Bayesian Ridge regression weights for predicting accuracy and CoT length using features obtained from **training set**. Bars show mean feature importance with standard deviation.

```
 Please reason step by step, and put your
final answer within boxed.

QUESTION:\n<Here is Question>\n
```

Compression Prompt (for each round):

```
 You have a question now:\n
QUESTION:\n<Here is Question>\n
THOUGHT PROCESS: <Here is Original CoT>\n
ANSWER:\n<Here is Final Answer>\n
Now you need to simplify the THOUGHT PROCESS
and retain the key information needed to
solve the question.
And do not add additional information that
is not included in the original THOUGHT
PROCESS.\n
SIMPLIFIED THOUGHT PROCESS:
```

A.3 Example of Multi-Round CoT Compression

Table 4 and 5 presents a case study illustrating the progressive compression behavior of different compressors (GPT-4o-mini, GPT-3.5-turbo) on the same arithmetic question. Each row represents the reasoning trace generated at a specific compression round.

We observe that higher-capacity models (e.g., GPT-4o-mini) tend to produce more concise yet faithful compressed outputs in fewer rounds, while smaller models require more rounds to converge. Across all compressors, the CoT length gradually shrinks while preserving the correct final answer, which validates the effectiveness of our multi-round adaptive compression mechanism.

A.4 Detailed Results of Compression and Fine-tuning

Tables 8, 9, and 10 provide the full experimental results for all combinations of Model and compressor under different compression rounds. Each row shows the original Chain-of-Thought (CoT) length and accuracy, the compressed output's perplexity and token length, and the downstream accuracy

Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?	
GPT-4o-mini	**Round 0 (Original):** *Let's break it down step by step: 1. Natalia sold clips to 48 of her friends in April. 2. In May, she sold half as many clips as she did in April. To find half of 48, we divide 48 by 2: 48 ÷ 2 = 24. So, Natalia sold 24 clips in May. 3. To find the total number of clips she sold in April and May, we add the number of clips she sold in each month: 48 (April) + 24 (May) = 72. Therefore, Natalia sold 72 clips altogether in April and May.* $\boxed{72}$ **Round 1:** *1. Natalia sold 48 clips in April. 2. In May, she sold half of that: 48 ÷ 2 = 24. 3. Total clips sold in April and May: 48 + 24 = 72.* $\boxed{72}$ **Round 2:** *1. Natalia sold 48 clips in April. 2. In May, she sold 24 clips (half of April). 3. Total sold: 48 + 24 = 72.* $\boxed{72}$ **Round 3:** *1. Natalia sold 48 clips in April. 2. In May, she sold 24 clips (half of April). 3. Total sold: 48 + 24 = 72.* **Round 4:** *1. Natalia sold 48 clips in April. 2. In May, she sold 24 clips (half of April). 3. Total sold: 48 + 24 = 72.* **Round 5:** *1. Natalia sold 48 clips in April. 2. In May, she sold 24 clips (half of April). 3. Total sold: 48 + 24 = 72.*
GPT-3.5-turbo	**Round 0 (Original):** *Let's break it down step by step: 1. Natalia sold clips to 48 of her friends in April. 2. In May, she sold half as many clips as she did in April. To find half of 48, we divide 48 by 2:48 ÷ 2 = 24 3. So, Natalia sold 24 clips in May. 4. To find the total number of clips she sold in April and May, we add the number of clips she sold in each month: 48 (April) + 24 (May) = 72. Therefore, Natalia sold 72 clips altogether in April and May.* $\boxed{72}$ **Round 1:** *Natalia sold 48 clips in April and half as many, which is 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.* **Round 2:** *Natalia sold 48 clips in April and half as many, 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.* **Round 3:** *Natalia sold 48 clips in April and half as many, 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.* **Round 4:** *Natalia sold 48 clips in April and half as many, 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.* **Round 5:** *Natalia sold 48 clips in April and half as many, 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.*
DeepSeek-V3	**Round 0 (Original):** *Let's break it down step by step: 1. Natalia sold clips to 48 of her friends in April. 2. In May, she sold half as many clips as she did in April. To find half of 48, we divide 48 by 2:48 ÷ 2 = 24 3. So, Natalia sold 24 clips in May. 4. To find the total number of clips she sold in April and May, we add the number of clips she sold in each month: 48 (April) + 24 (May) = 72. Therefore, Natalia sold 72 clips altogether in April and May.* $\boxed{72}$ **Round 1:** *1. April clips sold: 48 2. May clips sold: 48 ÷ 2 = 24. 3. Total clips sold: 48 + 24 = 72* $\boxed{72}$ **Round 2:** *1. April: 48 clips 2. May: Half of April → 24 clips 3. Total:* $\boxed{72}$ **Round 3:** *1. April: 48 clips 2. May: Half of April → 24 clips 3. Total:* $\boxed{72}$ **Round 4:** *1. April: 48 clips 2. May: Half of April → 24 clips 3. Total:* $\boxed{72}$ **Round 5:** *1. April: 48 clips 2. May: Half of April → 24 clips 3. Total:* $\boxed{72}$
GPT-4o	**Round 0 (Original):** *Let's break it down step by step: 1. Natalia sold clips to 48 of her friends in April. 2. In May, she sold half as many clips as she did in April. To find half of 48, we divide 48 by 2:48 ÷ 2 = 24 3. So, Natalia sold 24 clips in May. 4. To find the total number of clips she sold in April and May, we add the number of clips she sold in each month: 48 (April) + 24 (May) = 72. Therefore, Natalia sold 72 clips altogether in April and May.* $\boxed{72}$ **Round 1:** *1. Natalia sold 48 clips in April. 2. She sold half as many in May: (48 div 2 = 24). 3. Total clips sold in April and May: (48 + 24 = 72).* $\boxed{72}$ **Round 2:** *Natalia sold 48 clips in April and half as many, 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.* **Round 3:** *Natalia sold 48 clips in April and half as many, 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.* **Round 4:** *Natalia sold 48 clips in April and half as many, 24 clips, in May, totaling* $\boxed{72}$ *clips sold altogether in April and May.*

Table 4: Example of Multi-Round CoT Compression results on **GSM8K** across different compressors. Each row shows the reasoning chain under different compression rounds. The selected CoT is marked in blue

Question: What is the sum of the values of x that satisfy the equation $x^2-5x+5=9$?	
GPT-4o-mini	**Round 0 (Original):** *Step 1: Write down the given equation The given equation is $x^2-5x+5=9$. Step 2: Subtract 9 from both sides of the equation to set it equal to zero Subtracting 9 from both sides gives $x^2-5x-4=0$. Step 3: Use the quadratic formula to find the solutions for x The quadratic formula is $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$, where $a=1$, $b=-5$, and $c=-4$. Step 4: Plug the values of a, b, and c into the quadratic formula Plugging in the values gives $x=\frac{-(-5)\pm\sqrt{(-5)^2-4(1)(-4)}}{2(1)}$. Step 5: Simplify the expression under the square root Simplifying the expression gives $x=\frac{5\pm\sqrt{25+16}}{2}$. Step 6: Continue simplifying the expression under the square root Continuing to simplify gives $x=\frac{5\pm\sqrt{41}}{2}$. Step 7: The sum of the values of x is the sum of the two solutions The sum of the values of x is $\frac{5+\sqrt{41}}{2}+\frac{5-\sqrt{41}}{2}$. Step 8: Simplify the sum of the two solutions Simplifying the sum gives $\frac{10}{2}$. Step 9: Simplify the fraction Simplifying the fraction gives 5. The final answer is: $\boxed{5}$*
	Round 1: *Step 1: Write down the given equation The given equation is $x^2-5x+5=9$. Step 2: Set the equation to zero Subtracting 9 from both sides gives $x^2-5x-4=0$. Step 3: Use the quadratic formula Using the quadratic formula $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, and $c=-4$. Step 4: Substitute values into the formula Substituting gives $x=\frac{5\pm\sqrt{(-5)^2-4(1)(-4)}}{2(1)}$. Step 5: Simplify the expression This simplifies to $x=\frac{5\pm\sqrt{25+16}}{2}=\frac{5\pm\sqrt{41}}{2}$. Step 6: Find the sum of the solutions The sum of the solutions is $\frac{5+\sqrt{41}}{2}+\frac{5-\sqrt{41}}{2}=\frac{10}{2}=5$. The final answer is: $\boxed{5}$*
	Round 2: *Step 1: Write down the given equation The equation is $x^2-5x+5=9$. Step 2: Set the equation to zero This leads to $x^2-5x-4=0$. Step 3: Use the quadratic formula Using $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, $c=-4$. Step 4: Substitute values into the formula This results in $x=\frac{5\pm\sqrt{41}}{2}$. Step 5: Find the sum of the solutions The sum is $\frac{5+\sqrt{41}}{2}+\frac{5-\sqrt{41}}{2}=5$. Final answer: $\boxed{5}$*
	Round 3: *Step 1: Write down the given equation The equation is $x^2-5x+5=9$. Step 2: Set the equation to zero This leads to $x^2-5x-4=0$. Step 3: Use the quadratic formula Using $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, $c=-4$. Step 4: Substitute values into the formula This results in $x=\frac{5\pm\sqrt{41}}{2}$. Step 5: Find the sum of the solutions The sum is $\frac{5+\sqrt{41}}{2}+\frac{5-\sqrt{41}}{2}=5$. Final answer: $\boxed{5}$*
	Round 4: *Step 1: Write down the given equation The equation is $x^2-5x+5=9$. Step 2: Set the equation to zero This leads to $x^2-5x-4=0$. Step 3: Use the quadratic formula Using $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, $c=-4$. Step 4: Substitute values into the formula This results in $x=\frac{5\pm\sqrt{41}}{2}$. Step 5: Find the sum of the solutions The sum is 5. Final answer: $\boxed{5}$*
	Round 5: *Step 1: Write down the given equation The equation is $x^2-5x+5=9$. Step 2: Set the equation to zero This leads to $x^2-5x-4=0$. Step 3: Use the quadratic formula Using $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, $c=-4$. Step 4: Substitute values into the formula This results in $x=\frac{5\pm\sqrt{41}}{2}$. Step 5: Find the sum of the solutions The sum is 5. Final answer: $\boxed{5}$*
	Round 6: *Step 1: Write down the given equation The equation is $x^2-5x+5=9$. Step 2: Set the equation to zero This leads to $x^2-5x-4=0$. Step 3: Use the quadratic formula Using $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, $c=-4$. Step 4: Substitute values into the formula This results in $x=\frac{5\pm\sqrt{41}}{2}$. Step 5: Find the sum of the solutions The sum is 5. Final answer: $\boxed{5}$*
	Round 7: *Step 1: Write down the given equation The equation is $x^2-5x+5=9$. Step 2: Set the equation to zero This leads to $x^2-5x-4=0$. Step 3: Use the quadratic formula Using $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, $c=-4$. Step 4: Substitute values into the formula This results in $x=\frac{5\pm\sqrt{41}}{2}$. Step 5: Find the sum of the solutions The sum is 5. Final answer: $\boxed{5}$*
	Round 8: *Step 1: Write down the given equation The equation is $x^2-5x+5=9$. Step 2: Set the equation to zero Subtracting 9 gives $x^2-5x-4=0$. Step 3: Use the quadratic formula Using $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ with $a=1$, $b=-5$, $c=-4$. Step 4: Substitute values into the formula This gives $x=\frac{5\pm\sqrt{25+16}}{2}=\frac{5\pm\sqrt{41}}{2}$. Step 5: Find the sum of the solutions The sum is $\frac{5+\sqrt{41}}{2}+\frac{5-\sqrt{41}}{2}=\frac{10}{2}=5$. The final answer is: $\boxed{5}$*

Table 5: Example of Multi-Round CoT Compression results on **MATH** across different compressors. Each row shows the reasoning chain under different compression rounds. The selected CoT is marked in blue

after fine-tuning the Model.

These detailed tables validate our key findings:

- Increasing compression rounds leads to more compact reasoning traces but also higher perplexity.
- Compressors like GPT-4o and GPT-4o-mini consistently preserve semantic integrity better under aggressive compression, leading to superior fine-tuned accuracy.
- Lower-capacity compressors (e.g., GPT-3.5-turbo) experience sharper performance degradation under deeper compression.

We also observe that while the average CoT length drops by over 60% in many cases, the fine-tuned accuracy retains over 90% of its original value when using a well-matched compressor.

Table 8 reports the full results of multi-round compression and model fine-tuning across different compressors and Models. For each configuration, we list the number of compression rounds, the original CoT accuracy and length, the resulting compression rate, perplexity of the compressed CoT, and the fine-tuned model accuracy.

These results demonstrate the trade-off between compression depth and downstream performance. While deeper compression rounds reduce CoT length, they also tend to increase perplexity and reduce fine-tuned accuracy, especially under low-capacity compressors. Notably, models compressed by GPT-4o or GPT-4o-mini consistently outperform others in both efficiency and accuracy retention.

A.5 Training Hyperparameters

Table 6 lists the hyperparameters used for Model fine-tuning across different datasets. We use the LLaMA-Factory framework for fine-tuning, which is distributed under the Apache License 2.0. Our use complies with the terms and conditions of this license.

A.6 Performance Estimation Setup

We use Bayesian Ridge regression as our default performance estimator. All features are normalized to zero mean and unit variance. We train one model per Model using 5-fold cross-validation with 80/20 train/test split.

For comparison, we also evaluate Random Forest regression with 100 trees, which shows similar but less interpretable results.

Parameter	**Value**
LoRA rank	8
LoRA alpha	16
Learning rate	2×10^{-5}
Batch size	32
Epochs	3
Max sequence length	512 (GSM8K), 1024 (MATH-500)
Precision	bfloat16
Optimizer	AdamW
Scheduler	Cosine with warmup

Table 6: Fine-tuning hyperparameters for Models.

We report the average R^2 across folds for each model and target in Table 7.

Model	R^2 (Accuracy)	R^2 (CoT Length)
LLaMA3.1-8B	0.81	0.87
Qwen2.5-7B	0.78	0.91
Qwen2.5-3B	0.73	0.89

Table 7: Prediction performance of Bayesian Ridge on held-out data.

Table 8: MACC compression results for Model: **LLaMA-3.1-8B**. Each row shows the result of multi-round compression using a specific compressor.

Model	**Compressor**	**Rounds**	**Compressor Acc**	**Original Acc**	**Original Len**	**Compression Rate**	**PPL**	**Compressed Len**	**Finetuned Acc**
LLaMA3.1-8B	GPT-3.5-turbo	2	86.1	0.840	147.46	0.325	5.762	59.92	0.509
LLaMA3.1-8B	GPT-3.5-turbo	3	86.1	0.840	147.46	0.310	6.133	55.53	0.491
LLaMA3.1-8B	GPT-3.5-turbo	4	86.1	0.840	147.46	0.300	6.343	53.85	0.478
LLaMA3.1-8B	GPT-3.5-turbo	5	86.1	0.840	147.46	0.292	6.471	52.22	0.466
LLaMA3.1-8B	GPT-4.1-mini	2	86.1	0.949	190.29	0.278	5.696	54.99	0.713
LLaMA3.1-8B	GPT-4.1-mini	3	86.1	0.949	190.29	0.262	6.029	48.43	0.676
LLaMA3.1-8B	GPT-4.1-mini	4	86.1	0.949	190.29	0.252	6.183	46.61	0.649
LLaMA3.1-8B	GPT-4.1-mini	5	86.1	0.949	190.29	0.246	6.255	45.77	0.643
LLaMA3.1-8B	GPT-4.1-nano	2	86.1	0.905	252.29	0.301	5.141	61.57	0.776
LLaMA3.1-8B	GPT-4.1-nano	3	86.1	0.905	252.29	0.291	5.377	58.17	0.766
LLaMA3.1-8B	GPT-4.1-nano	4	86.1	0.905	252.29	0.285	5.490	56.88	0.738
LLaMA3.1-8B	GPT-4.1-nano	5	86.1	0.905	252.29	0.281	5.552	55.60	0.738
LLaMA3.1-8B	GPT-4o	2	86.1	0.953	273.57	0.420	4.793	84.17	0.788
LLaMA3.1-8B	GPT-4o	3	86.1	0.953	273.57	0.402	5.107	79.16	0.768
LLaMA3.1-8B	GPT-4o	4	86.1	0.953	273.57	0.390	5.287	77.08	0.766
LLaMA3.1-8B	GPT-4o	5	86.1	0.953	273.57	0.382	5.389	76.34	0.761
LLaMA3.1-8B	GPT-4o-mini	2	86.1	0.922	330.37	0.497	4.227	99.57	0.809
LLaMA3.1-8B	GPT-4o-mini	3	86.1	0.922	330.37	0.482	4.419	93.48	0.802
LLaMA3.1-8B	GPT-4o-mini	4	86.1	0.922	330.37	0.472	4.517	90.57	0.805
LLaMA3.1-8B	GPT-4o-mini	5	86.1	0.922	330.37	0.464	4.584	88.58	0.812

Table 9: MACC compression results for Model: **Qwen2.5-3B**.

Model	**Compressor**	**Rounds**	**Compressor Acc**	**Original Acc**	**Original Len**	**Compression Rate**	**PPL**	**Compressed Len**	**Finetuned Acc**
Qwen2.5-3B	GPT-3.5-turbo	2	83.7	0.840	147.46	0.325	5.762	164.75	0.753
Qwen2.5-3B	GPT-3.5-turbo	3	83.7	0.840	147.46	0.310	6.133	144.35	0.704
Qwen2.5-3B	GPT-3.5-turbo	4	83.7	0.840	147.46	0.300	6.343	102.87	0.584
Qwen2.5-3B	GPT-3.5-turbo	5	83.7	0.840	147.46	0.292	6.471	102.17	0.580
Qwen2.5-3B	GPT-4.1-mini	2	83.7	0.949	190.29	0.278	5.696	202.04	0.799
Qwen2.5-3B	GPT-4.1-mini	3	83.7	0.949	190.29	0.262	6.029	199.71	0.804
Qwen2.5-3B	GPT-4.1-mini	4	83.7	0.949	190.29	0.252	6.183	202.05	0.811
Qwen2.5-3B	GPT-4.1-mini	5	83.7	0.949	190.29	0.246	6.255	200.17	0.804
Qwen2.5-3B	GPT-4.1-nano	2	83.7	0.905	252.29	0.301	5.141	203.00	0.825
Qwen2.5-3B	GPT-4.1-nano	3	83.7	0.905	252.29	0.291	5.377	201.23	0.826
Qwen2.5-3B	GPT-4.1-nano	4	83.7	0.905	252.29	0.285	5.490	202.89	0.824
Qwen2.5-3B	GPT-4.1-nano	5	83.7	0.905	252.29	0.281	5.552	202.15	0.825
Qwen2.5-3B	GPT-4o	2	83.7	0.953	273.57	0.420	4.793	210.98	0.804
Qwen2.5-3B	GPT-4o	3	83.7	0.953	273.57	0.402	5.107	208.16	0.807
Qwen2.5-3B	GPT-4o	4	83.7	0.953	273.57	0.390	5.287	207.44	0.804
Qwen2.5-3B	GPT-4o	5	83.7	0.953	273.57	0.382	5.389	206.06	0.792
Qwen2.5-3B	GPT-4o-mini	2	83.7	0.922	330.37	0.497	4.227	217.85	0.818
Qwen2.5-3B	GPT-4o-mini	3	83.7	0.922	330.37	0.482	4.419	214.69	0.817
Qwen2.5-3B	GPT-4o-mini	4	83.7	0.922	330.37	0.472	4.517	214.21	0.810
Qwen2.5-3B	GPT-4o-mini	5	83.7	0.922	330.37	0.464	4.584	216.25	0.805

Table 10: MACC compression results for Model: **Qwen2.5-7B**.

Model	**Compressor**	**Rounds**	**Compressor Acc**	**Original Acc**	**Original Len**	**Compression Rate**	**PPL**	**Compressed Len**	**Finetuned Acc**
Qwen2.5-7B	GPT-3.5-turbo	2	91.4	0.840	147.46	0.325	5.762	80.68	0.624
Qwen2.5-7B	GPT-3.5-turbo	3	91.4	0.840	147.46	0.310	6.133	66.28	0.557
Qwen2.5-7B	GPT-3.5-turbo	4	91.4	0.840	147.46	0.300	6.343	50.64	0.440
Qwen2.5-7B	GPT-3.5-turbo	5	91.4	0.840	147.46	0.292	6.471	60.97	0.525
Qwen2.5-7B	GPT-4.1-mini	2	91.4	0.949	190.29	0.278	5.696	71.76	0.791
Qwen2.5-7B	GPT-4.1-mini	3	91.4	0.949	190.29	0.262	6.029	61.87	0.753
Qwen2.5-7B	GPT-4.1-mini	4	91.4	0.949	190.29	0.252	6.183	59.74	0.735
Qwen2.5-7B	GPT-4.1-mini	5	91.4	0.949	190.29	0.246	6.255	58.86	0.732
Qwen2.5-7B	GPT-4.1-nano	2	91.4	0.905	252.29	0.301	5.141	71.77	0.826
Qwen2.5-7B	GPT-4.1-nano	3	91.4	0.905	252.29	0.291	5.377	70.35	0.821
Qwen2.5-7B	GPT-4.1-nano	4	91.4	0.905	252.29	0.285	5.490	66.94	0.806
Qwen2.5-7B	GPT-4.1-nano	5	91.4	0.905	252.29	0.281	5.552	65.50	0.799
Qwen2.5-7B	GPT-4o	2	91.4	0.953	273.57	0.420	4.793	137.63	0.860
Qwen2.5-7B	GPT-4o	3	91.4	0.953	273.57	0.402	5.107	129.21	0.845
Qwen2.5-7B	GPT-4o	4	91.4	0.953	273.57	0.390	5.287	117.46	0.847
Qwen2.5-7B	GPT-4o	5	91.4	0.953	273.57	0.382	5.389	121.62	0.838
Qwen2.5-7B	GPT-4o-mini	2	91.4	0.922	330.37	0.497	4.227	180.00	0.878
Qwen2.5-7B	GPT-4o-mini	3	91.4	0.922	330.37	0.482	4.419	169.46	0.873
Qwen2.5-7B	GPT-4o-mini	4	91.4	0.922	330.37	0.472	4.517	129.99	0.707
Qwen2.5-7B	GPT-4o-mini	5	91.4	0.922	330.37	0.464	4.584	148.76	0.863