FORMAT-ADAPTER: Optimize Training-Free Test-Time Scaling by Adapting Suitable Format

Anonymous ACL submission

Abstract

Test-time scaling (TTS) has emerged as a practical way to improve LLM reasoning by allocating more inference-time computation, especially through parallel scaling. However, most existing TTS pipelines scale compute within fixed reasoning formats, limiting performance. In this work, we propose that TTS should scale not only the number of samples, but also the reasoning formats. We first introduce an error measurement for multi-sample reasoning that enables comparing TTS behaviors across formats. Based on this, we propose FORMAT-ADAPTER, a training-free, format-adaptive TTS method that automatically generates candidate reasoning formats and selects the most suitable ones under a test-time budget by minimizing the proposed error measurement. We evaluate FORMAT-ADAPTER on mathematics and general reasoning benchmarks. Under the same TTS budget, FORMAT-ADAPTER yields consistent gains over existing baselines, achieving an average relative improvement of 2.2% and establishing a new state of the art (SOTA) for training-free parallel-scaling TTS.

1 Introduction

Large Language Models (LLMs) have shown impressive progress on reasoning-intensive benchmarks, yet their performance still hinges on how much computation allocated at training time (Hestness et al., 2017; Ghorbani et al., 2022; Hoffmann et al., 2022). Recent evidence suggests that, under a fixed LLM, increasing inference-time compute (a.k.a. **test-time scaling, TTS**) can be a more efficient path to better reasoning than merely scaling parameters, especially when compute is allocated adaptively to prompt difficulty (Snell et al., 2025; Wu et al., 2025; Agarwal et al., 2025). This trend is also reflected in frontier reasoning systems, where models are often evaluated under maximal test-time compute settings (OpenAI et al., 2024a; Yang et al., 2025a; Guo et al., 2025).

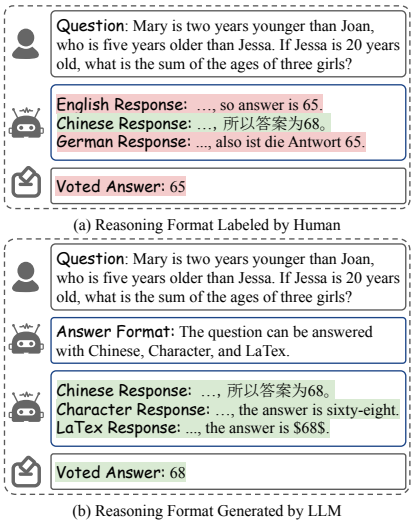

Figure 1: TTS with a fixed format (a) versus our format-adaptive TTS (b). The red parts denote incorrect answers and the green parts denote correct ones. Prior TTS commonly samples multiple solutions under a human-chosen, fixed reasoning format, which could be unsuitable to the query. Our method generates and selects suitable reasoning formats under a test-time budget, achieving better performance.

TTS improves reasoning by expending more computation during inference. Such methods typically include two axes: *parallel scaling* (sampling multiple candidate solutions and aggregating/choosing the best) (Wang et al., 2023; Zhou et al., 2024; Bi et al., 2025) and *sequential scaling* (allocating more steps/tokens for deliberation, revision, or verification) (Wu et al., 2025; Muennighoff et al., 2025; Zeng et al., 2025). A dominant and simple instantiation is to sample multiple responses and then vote or score to pick the final answer (Wang et al., 2023; Yao et al., 2023; Besta et al., 2024), often referred to as best-of-N sampling. Such multi-sample inference is widely adopted because it can reduce variance caused by stochastic decoding and minor prompt/parameter perturbations, where the same question could yield inconsistent outputs (Wang et al., 2022). In this paper, we mainly study the parallel scaling.

However, a key limitation of existing TTS pipelines is that they usually scale computation within a fixed reasoning format[1]. That is, the system repeats the same format template and only varies decoding randomness, sample count, or reasoning length. Recent studies show that the effectiveness of TTS is highly sensitive to prompt strategy, model type, and problem difficulty. Even sophisticated prompting could scale worse than simpler strategies as the sample budget increases (Liu et al., 2025; Agarwal et al., 2025; Snell et al., 2025). This suggests that format choice is not a cosmetic detail but a central factor that shapes the solution distribution explored by TTS, and a mismatched format can waste substantial test-time compute.

Therefore, in this paper, we revisit TTS from a new angle: **to optimize TTS, we should scale not only the number of samples/steps, but also the format that governs what the model can express and explore**. We first prove that multi-sample inference under a single fixed format primarily improves robustness that reduces variance, whereas adapting multiple suitable formats can further improve capability by expanding the explored reasoning modes and reducing systematic errors induced by a mismatched format. Then, based on this perspective, we propose FORMAT-ADAPTER, a format-adaptive TTS method that optimizes inference-time performance by generating and selecting suitable reasoning formats without additional training.

To evaluate the effectiveness of FORMAT-ADAPTER, we study two mainstream reasoning domains: math and reasoning. Experiments show that, under the same TTS budget, FORMAT-ADAPTER improves performance by 2.2% on average over previous works, demonstrating new SOTA results on the parallel scaling TTS, which highlights the necessity of format selection in TTS.

Our contributions are as follows:

- From the perspective of optimizing TTS, we analyze why adapting multiple formats can outperform scaling compute within a single format.
- We propose FORMAT-ADAPTER, which generates and selects suitable reasoning formats using LLMs to improve TTS effectiveness.
- Experiments show FORMAT-ADAPTER yields 2.2% improvement on average over previous methods, demonstrating new SOTA results on the parallel scaling TTS.

[1]In this paper, we define a *reasoning format* as the prompt-and-output schema that specifies *how an LLM externalizes intermediate reasoning and the final answer*.

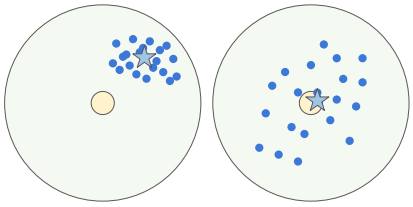

Figure 2: The comparison between single-format TTS (left) and multi-format TTS (right) under the same parallel test-time compute budget. The yellow ◯ denotes the correct answer, the blue • denotes different predictions, and the blue ☆ denotes the average prediction. Single-format TTS produces answers that all cluster around a specific response, averaging which mainly just reduces the variance. Multi-format TTS yields more diverse answers, making their average ends up closer to the correct answer of the given query.

2 Preliminaries

In this section, based on the error analysis, we derive the TTS errors under a single format and under multiple formats, thereby demonstrating that **single-format TTS can only improve robustness, whereas multi-format TTS is necessary to enhance the reasoning capability**. All proofs of this section are present in Appendix A.

2.1 Error of Single Reasoning Format

First, we discuss the error of the general ensemble method (Sagi and Rokach, 2018), since a dominant instantiation of parallel TTS is to generate multiple responses and then aggregate, which can be regarded as an ensemble method. In this paper, we use the error function $L(x, y)$ as follows:

$$L(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases} \tag{1}$$

The function L represents whether the prediction result matches the correct answer exactly. We define $D = \{(x_i, y_i)\}_{|D|}$ as the experimental dataset, $\{\phi_i\}_m$ as the set of m predictors (corresponding to m generated answers under a fixed test-time budget), and $\bar{\phi} = \texttt{avg}(\phi_{i_m})$ as the ensemble predictors. Proved by Wood et al. (2024), the error of ensemble learning with m predictors on the dataset D can be expressed as the error over the dataset minus the divergence among the individual predictors, that is:

$$\begin{aligned} \mathbb{E}_D\left[L(\bar{\phi}, y)\right] = \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_D\left[L\left(\phi_i, y\right)\right] \\ - \mathbb{E}_D\left[\frac{1}{m}\sum_{i=1}^{m} L\left(\phi_i, \bar{\phi}\right)\right] \end{aligned} \tag{2}$$

Then we discuss the error of generating multiple responses using LLMs under single-format TTS. For a model employing a single reasoning format, we assume the used format is $\mathtt{f}$ and the model is ϕ. When generating multiple answers, we can regard each predictor as applying a perturbation (e.g., temperature, top_p) to the model inherent performance under this fixed format, expressed as $\phi_i = \phi \circ \mathtt{f} + \delta_i$, where δ_i denotes the perturbation. It can be derived that generating one single answer using ϕ_i can be presented as:

$$\mathbb{E}_D\left[L(\phi_i, y)\right] = \mathbb{E}_D\left[L\left(\phi \circ \mathtt{f} + \delta_i, y\right)\right] \quad (3)$$

We assume an ideal scenario where the average of all predictors represents the inherent performance of the model under the fixed format, i.e., $\lim_{m\to\infty} \bar{\phi} = \phi \circ \mathtt{f}$. It can be proven that the error in generating multiple answers using a single reasoning format satisfies:

$$\mathbb{E}_D\left[L(\bar{\phi}, y)\right] = \mathbb{E}_D\left[L\left(\phi \circ \mathtt{f}, y\right)\right] \quad (4)$$

It can be seen that, compared with Equation 3, scaling test-time computation by generating multiple answers under a fixed format can eliminate the perturbation δ, enhancing robustness. However, when using the single fixed format f, Equation 4 is determined by $\phi \circ \mathtt{f}$, showing that further performance improvement is constrained by the inherent capability and systematic errors induced by the model-format pairing during TTS.

2.2 Error of Multiple Reasoning Format

In the following, we discuss the error of using multiple reasoning formats in TTS and why it can outperform single-format TTS under the same test-time budget. During inference, we employ multiple formats $\{f_i\}_m$ with one single model ϕ, so we can assume the predictors to be $\phi_i = \phi \circ \mathtt{f}_i + \delta_i$. It can be proved that the reasoning error follows:

$$\begin{aligned}\mathbb{E}_D\left[L(\bar{\phi}, y)\right] &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_D\left[L(\phi \circ \mathtt{f}_i, y)\right] \\ &\quad - \mathbb{E}_D[\frac{1}{m}\sum_{i=1}^{m} L(\phi \circ \mathtt{f}_i, \bar{\phi})]\end{aligned} \quad (5)$$

In Equation 5, the first term denotes the average error over all format-conditioned predictions and correct answers, and the second term measures the divergence induced by different reasoning formats. It can be observed that, even with the same model and the same test-time compute budget, we can combine different reasoning formats to minimize errors, thereby improving performance, as shown in the right part of Figure 2.

3 Methodology

This section introduces FORMAT-ADAPTER, a training-free TTS method that improves inference-time reasoning by adapting reasoning formats under a given test-time compute budget. Unlike conventional TTS pipelines that repeatedly scale computation within a fixed reasoning format, FORMAT-ADAPTER additionally scales over the format space by generating and selecting suitable reasoning formats. An illustration of our method is shown in Figure 3. The prompts we used are provided in Appendix B.1. We also discuss the efficiency of FORMAT-ADAPTER in Appendix C.1.

3.1 Format Generation

This step generates candidate reasoning formats to support format-adaptive TTS, ensuring both the relevance and diversity of the explored formats. Relevance means that the generated reasoning formats are appropriate for the given task, so that test-time compute is not wasted on mismatched ones. Diversity demands that the generated reasoning formats be varied, enabling models to explore multiple reasoning modes beyond stochastic decoding variations within a limited scope during TTS.

To ensure relevance, an example is provided during generation to help LLMs learn how to produce task-relevant reasoning formats. To ensure diversity, we design the instruction to ask LLMs to generate reasoning formats across multiple categories, where each category consists of multiple formats. For instance, as shown in Figure 3, *Natural Language* is the reasoning format category, while *English* and *Chinese* are the reasoning formats of this category. In summary, the input includes the task definition and an example of the format generation, while the output consists of multiple reasoning formats to be used for subsequent TTS.

3.2 Answer Generation

This step performs multi-sample inference at test time to generate candidate answers under each generated reasoning format. First, the instruction is rewritten according to each reasoning format to ensure that answer generation follows the given prompt-and-output schema. We take the original

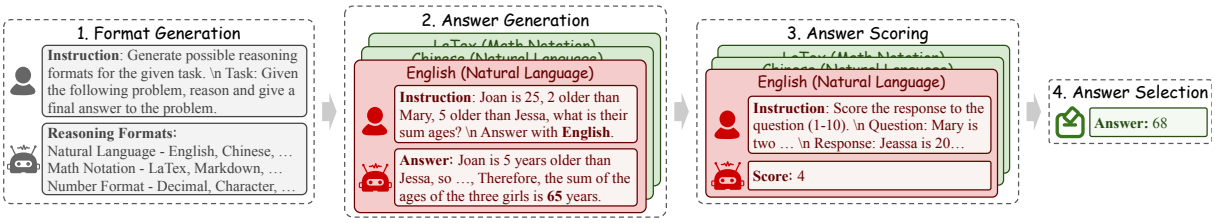

Figure 3: The pipeline of FORMAT-ADAPTER, which includes four steps: *(i) Format Generation*: Generate candidate reasoning formats; *(ii) Answer Generation*: Perform budgeted multi-sample inference under each format to obtain candidate answers; *(iii) Answer Scoring*: Use LLMs to score the generated answer; *(iv) Answer Selection*: Select the final answer using Equation 5. Red and green parts represent the reasoning formats leading to incorrect and correct answers during TTS, respectively.

instruction of the task and the reasoning format as input and ask LLMs to output the rewritten instruction based on the reasoning format. Then, the rewritten instruction is used to generate different reasoning answers for the given user question. Concretely, we apply zero-shot inference by inputting the rewritten instruction and the user question, and we allocate test-time compute by sampling one or multiple answers per format, consistent with standard parallel-scaling TTS.

3.3 Answer Scoring

After obtaining candidate answers across different reasoning formats, we aim to select suitable formats that minimize the test-time error objective in Equation 5. However, the error between the prediction and the ground-truth answer, $L(\phi \circ f_i, y)$, is difficult to compute at test time as the ground truth is unavailable. Therefore, we use an LLM-based scorer to estimate the probability that each predicted answer is correct, which serves as a proxy for the unknown loss under the TTS setting. Following previous works (Zheng et al., 2023; Yao et al., 2023; Zhou et al., 2025), we input the user question and the predicted answer, outputting a score from 1 to 10 to represent the degree of probability that the answer is correct. To ensure that there is the same scale between the first term and the second term of Equation 5 during the calculation, we divide the rating by 10. We also evaluate other score methods in Appendix D.8, which requires higher computation while demonstrates better performance during inference.

3.4 Answer Selection

Based on the predicted answers and corresponding scores under different reasoning formats, we select the final answer according to Equation 5 in a budget-aware TTS manner. Specifically, given the dataset D and the model ϕ, we hope to find suitable reasoning formats to minimize the error that satisfies the following:

$$\{\mathtt{f}_i\}_n = \underset{\{\mathtt{f}_i\}_n \subseteq \{\mathtt{f}_i\}_m}{\arg\min} \; \mathbb{E}_D \left[L(\bar{q}, y) \right] \tag{6}$$

In Equation 5, the first term can be directly calculated by averaging the scores obtained in §3.3. The second term requires calculating the average difference between all results and the average result, where we take the average prediction $\bar{\phi}(x)$ as the answer that appears most frequently among all outcomes. Considering computational efficiency under a test-time budget, we adopt a greedy algorithm to select formats: we add each format $\mathtt{f}_i$ one by one to the selected results. If the value of Equation 5 decreases, we retain $\mathtt{f}_i$. Otherwise, we remove $\mathtt{f}_i$. Due to the inherent scoring errors of LLMs, we do not directly use the answer with the highest score within the selected set. Instead, consistent with standard multi-sample TTS aggregation, we choose the most frequently occurring answer among the selected results as the final answer, ensuring high quality while keeping consistency.

4 Experiment

4.1 Experimental Setup

Datasets To validate the effectiveness of FORMAT-ADAPTER, we conduct our experiments on four mainstream reasoning datasets including: GSM8K (Cobbe et al., 2021), MATH (Saxton et al., 2019), ARC-Challenge (Yadav et al., 2019), and GPQA (Rein et al., 2024). To fully verify the effectiveness while reducing inference cost, we employ subsets of certain datasets. Specifically, for GSM8K and ARC-Challenge, we sample 256 questions that are not well solved by the current LLMs, referred to as GSM8K-Hard and ARC-Challenge-Hard, respectively. For MATH, we utilize MATH500 (Lightman et al., 2024),

which samples 500 questions from the original dataset. For GPQA, we employ the original test set, comprising 448 questions. We use Exact Match (EM) as the evaluation metric for all datasets.

Models We conduct the experiments with Llama-3.1-8B/70B-Instruct (Llama-3.1) (Dubey et al., 2024), Qwen3-8B (Yang et al., 2025a), and `gpt-5-nano-2024-11-20` (`gpt-5-nano`) (OpenAI et al., 2024b). Llama-3.1 and Qwen3 are mainstream and high-performing open-source LLMs. `gpt-5-nano` represents powerful close-source LLMs in terms of reasoning capabilities.

Baselines To better reflect the effectiveness of FORMAT-ADAPTER, we compare it with current mainstream parallel scaling TTS methods, including Self-Consistency (Wang et al., 2023), Tree-of-Thought (Yao et al., 2023), DTV (Zhou et al., 2024), Self-Certainty (Kang et al., 2025), Shortest MV (Zeng et al., 2025), and Forest-of-Thought (Bi et al., 2025). Appendix B.2 introduces the above baselines. We compare FORMAT-ADAPTER with more baselines in Appendix D.1.

Implement Details Following Qin et al. (2023), we evaluate FORMAT-ADAPTER in a zero-shot manner. We set the temperature as 0.5 and top_p as 0.9. The sample number is consistent with the number of generated formats, which is shown in Appendix B.3. Considering that the optimal parameters vary across different methods, the parameters for each baseline are set to the optimal values adopted in their respective papers. All our experiments are adapted on two A100-80G, which costs one hour on average for each setting. All implementations are based on Transformers (Wolf et al., 2020) and vLLM (Kwon et al., 2023).

4.2 Main Experiment

The main experimental results are shown in Table 1. FORMAT-ADAPTER achieves 2.2% relative performance improvement on average over the best baseline in each setting and establishes new SOTA results on the parallel scaling TTS methods, demonstrating that adapting reasoning formats makes TTS more effective. We also report the average performance of five runs in Appendix D.4. Besides, from Table 1, we can also observe that:

Dataset Compared with Self-Consistency, our method yields more pronounced performance gains on simpler datasets such as GSM8K and ARC-Challenge than on more complex datasets (4.8% vs. 10.1%). This suggests that LLMs tend to produce more consistent answers on simple problems, where adjusting the reasoning format is necessary to prevent LLMs from being anchored to a particular answer, enhancing reasoning capability.

Model Compared with Self-Consistency, our method yields more pronounced performance gains on Llama-3.1 than on more advanced models (3.4% vs. 11.4%). On the one hand, this is because more advanced models can already solve the given problems well, so there are fewer bad cases and thus smaller room for improvement. On the other hand, this indicates that weaker models are more likely to generate consistent answers, and therefore rely on changes to the reasoning format to produce more diverse outcomes during inference.

Oracle A noticeable gap remains between Vote and Oracle, which can be attributed to the fact that the scoring quality of LLMs is suboptimal, making it challenging to reliably evaluate correctness at test time. Besides, for choice-based datasets such as ARC-Challenge and GPQA, LLMs could output the correct choice with imperfect reasoning, which can be penalized by score-based selection, thereby widening the Vote-Oracle gap.

4.3 Ablation Study

To demonstrate the effectiveness of each step of FORMAT-ADAPTER, we adapt the ablation experiment. The results are shown in Table 2, where removing any step degrades performance, validating the necessity of each component for making inference-time computation more effective. Furthermore, we can observe that: *(i)* Removing the Select step causes the largest drop, suggesting that under a fixed test-time budget, only a small subset of generated formats tends to yield correct solutions for many questions, showing that selecting suitable formats is critical; *(ii)* The degradation is more pronounced for smaller models, implying that small-scale models produce fewer formats capable of reaching correct answers, and thus rely more on Rewrite and Select for benefit.

4.4 Analysis

In this section, we conduct analytical experiments to better understand how FORMAT-ADAPTER improves reasoning under TTS and to guide parameter choices. We also provide a case study to further illustrate how FORMAT-ADAPTER improves performance in Appendix C.2.

Model	Method	GSM8K-Hard		MATH500		ARC-C-Hard		GPQA	
		Vote	Oracle	Vote	Oracle	Vote	Oracle	Vote	Oracle
Llama-3.1-8B	Single	23.0	–	47.8	–	16.8	–	28.1	–
	Self-Consistency	36.7	55.1	51.4	63.0	18.4	23.4	30.8	59.2
	Tree-of-Thought	43.0	53.9	52.8	56.8	24.2	33.8	32.8	46.7
	DTV	51.6	56.2	55.4	56.4	39.1	42.6	32.6	50.0
	Self-Certainty	49.8	68.9	59.0	71.2	48.5	69.7	33.4	64.3
	Shortest MV	48.1	66.0	58.2	70.0	44.2	65.0	33.0	70.0
	Forest-of-Thought	53.6	88.0	59.2	73.4	56.2	89.5	33.6	91.9
	FORMAT-ADAPTER	**54.7**	**89.8**	**60.4**	**75.0**	**57.4**	**91.4**	**34.3**	**93.8**
Llama-3.1-70B	Single	66.0	–	63.4	–	68.0	–	43.1	–
	Self-Consistency	70.3	77.3	64.4	72.8	69.1	69.9	46.2	66.5
	Tree-of-Thought	71.5	77.6	67.2	75.2	70.7	72.3	48.0	73.2
	DTV	71.7	84.3	65.8	81.8	69.9	73.8	50.2	75.9
	Self-Certainty	74.3	88.0	69.2	81.0	71.1	83.5	49.8	79.5
	Shortest MV	73.0	86.2	69.4	80.0	70.9	82.0	49.0	77.0
	Forest-of-Thought	74.6	92.9	69.6	83.6	71.3	86.9	50.4	94.4
	FORMAT-ADAPTER	**76.2**	**94.9**	**71.0**	**85.4**	**72.8**	**88.7**	**51.5**	**96.4**
Qwen3-8B	Single	78.3	–	76.2	–	83.2	–	42.4	–
	Self-Consistency	79.1	87.5	77.0	83.5	84.0	90.4	45.3	64.7
	ToT	80.3	88.9	77.9	84.7	84.8	91.9	47.6	68.9
	DTV	80.9	91.4	78.5	86.2	85.7	94.1	49.1	71.8
	Self-Certainty	81.6	95.0	78.6	87.9	86.3	96.4	50.9	82.3
	Shortest MV	81.0	93.2	78.2	86.0	86.0	95.0	50.4	79.6
	Forest-of-Thought	81.8	95.3	78.8	88.0	86.5	96.6	51.1	95.2
	FORMAT-ADAPTER	**83.5**	**97.3**	**80.4**	**89.8**	**88.3**	**98.6**	**52.2**	**97.2**
`gpt-5-nano`	Single	80.0	–	78.2	–	85.6	–	70.4	–
	Self-Consistency	81.4	90.5	79.0	86.8	86.7	93.8	71.8	76.2
	FORMAT-ADAPTER	**84.2**	**97.7**	**81.2**	**90.6**	**89.0**	**99.1**	**73.0**	**97.9**

Table 1: EM of FORMAT-ADAPTER and baselines under different settings. The best results of each setting are marked in **bold**. Oracle denotes whether the sampled answers contain the gold answer. Single denotes generating one single answer. Due to the limitations of computing resources for test-time scaling, we only compare the performance of FORMAT-ADAPTER with Self-Consistency on `gpt-5-nano`.

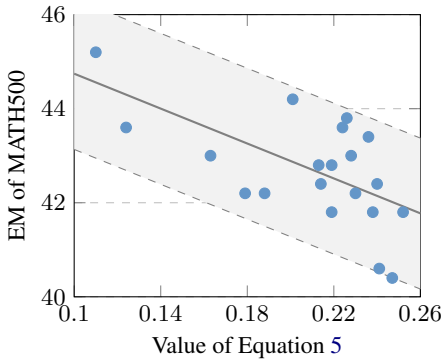

Figure 4: The performance on MATH with different formats using Llama-3.1-8B. Different blue • denotes the result using different formats, where the formats are randomly sampled from those generated by FORMAT-ADAPTER. The correlation coefficient is -0.652.

4.4.1 Reasoning Error

To show that Equation 5 effectively captures test-time reasoning error under multiple formats, we analyze MATH and correlate model performance with the error values computed by Equation 5 using different reasoning formats in Figure 4. The figure shows that: *(i)* As the value of Equation 5 increases, the reasoning performance consistently decreases, indicating that the metric reflects progressively larger reasoning errors at test time; *(ii)* The error values concentrate around 0.22, suggesting that most formats yield similar (but suboptimal) results compared to the best formats, which highlights the necessity of selecting suitable reasoning formats to avoid systematic errors induced by mismatched formats under TTS.

4.4.2 Reasoning Format Category

To examine how different reasoning format categories contribute to FORMAT-ADAPTER, we analyze the average improvement achieved when restricting FORMAT-ADAPTER to specific format categories in Figure 5. We also list the most suitable reasoning format for each task in Appendix B.3. From Figure 5, we can see that: *(i)* For a single reasoning format category, the improvement depends on the diversity of answers induced by formats in that category. Categories with low im-

Model	Method	GSM8K-Hard	MATH500	ARC-C-Hard	GPQA
Llama-3.1-8B	FORMAT-ADAPTER	54.7	56.8	57.4	33.5
	- Rewrite	$51.6_{(-3.1)}$	$53.6_{(-3.2)}$	$52.0_{(-5.4)}$	$32.4_{(-1.1)}$
	- Select	$49.2_{(-5.5)}$	$47.8_{(-9.0)}$	$54.7_{(-2.7)}$	$25.4_{(-8.1)}$
	- Score	$53.9_{(-0.8)}$	$54.2_{(-2.6)}$	$52.7_{(-4.7)}$	$30.1_{(-3.4)}$
Llama-3.1-70B	FORMAT-ADAPTER	76.2	70.4	71.5	51.0
	- Rewrite	$75.4_{(-0.8)}$	$69.6_{(-0.8)}$	$68.8_{(-2.7)}$	$47.1_{(-3.9)}$
	- Select	$75.0_{(-1.2)}$	$68.6_{(-1.8)}$	$66.4_{(-5.1)}$	$44.2_{(-6.8)}$
	- Score	$73.8_{(-2.4)}$	$70.2_{(-0.2)}$	$69.9_{(-1.6)}$	$47.3_{(-3.7)}$

Table 2: The ablation study results under: *(i)* Rewrite: Generate answers without rewriting instructions; *(ii)* Select: Vote the answer from the responses with the highest score; *(iii)* Score: Set all answers with the same score of 1.0.

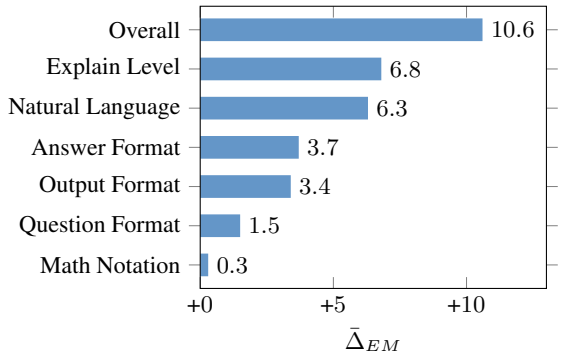

Figure 5: The average improvement brought by FORMAT-ADAPTER with different reasoning categories having more than four formats. $\bar{\Delta}_{EM}$ denotes the average improvement compared to Self-Consistency. Overall denotes using all reasoning categories.

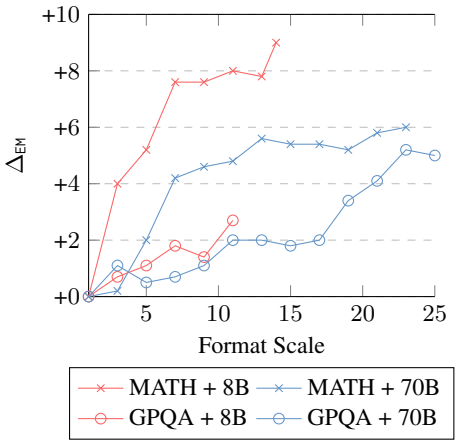

Figure 6: The average performance improvement brought by FORMAT-ADAPTER with different format scales on MATH and GPQA using Llama-3.1. Δ_{EM} denotes the EM improvement compared with the inference result using the single format.

provements (e.g., Math Notation) tend to produce similar answers across formats, behaving similarly to single-format Self-Consistency. In contrast, categories with higher improvements (e.g., Explain Level) yield more diverse solutions, increasing the chance of including a correct candidate under a fixed test-time budget; *(ii)* Even the best single category underperforms when using all categories (Overall), indicating that different questions benefit from different reasoning modes, and combining multiple categories better realizes the capability gains of TTS beyond variance reduction.

4.4.3 Reasoning Format Scale

Considering practical inference budgets and to better understand how the reasoning format scale affects performance, we evaluate the performance of FORMAT-ADAPTER under different reasoning format scales during inference. Figure 6 shows that performance improves as the number of formats increases, supporting the necessity of scaling formats to make TTS more effective. Notably, even with a small number of formats (< 5), FORMAT-ADAPTER still yields significant gains, demonstrating the effectiveness of our method under the limited test-time computing scenario.

Moreover, Figure 6 shows that the improvement trend typically increases quickly, saturates, and then rises again. This can be explained by two complementary effects of TTS: *(i)* In the early stage, the main bottleneck is stochastic inconsistency, where increasing the number of formats improves robustness via variance reduction, similar to multi-sample inference under a fixed format; *(ii)* After sufficient scaling, the bottleneck shifts to format mismatch, where adding more formats increases the probability of including a suitable reasoning mode for the question, reducing systematic errors and yielding further capability improvements.

4.4.4 Format Selection Ratio

To better understand how FORMAT-ADAPTER allocates test-time compute across formats, we compute the ratio of formats selected by FORMAT-ADAPTER and the ratio of formats that contain a correct answer per category (Oracle), as shown in Figure 7. We can observe that: *(i)* The selection ratio of FORMAT-ADAPTER follows a trend similar to Oracle, indicating that FORMAT-ADAPTER

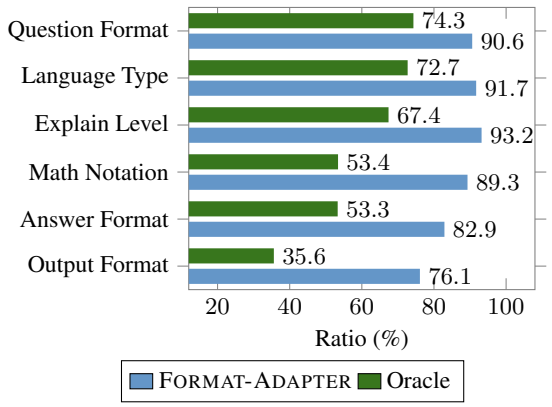

Figure 7: The average ratio over all datasets of each reasoning format category that is selected by FORMAT-ADAPTER (blue) and that contains the format that can solve the question correctly (Oracle, green).

tends to allocate compute to appropriate formats that are more likely to contain correct answers; *(ii)* Compared to the average Oracle performance of FORMAT-ADAPTER in Table 1 (89.4%), the best single format still lags by 15.1%, suggesting that no single fixed format is universally optimal and that multiple formats are necessary for effective TTS; *(iii)* FORMAT-ADAPTER selects a relatively high proportion ($> 70\%$) for each category, implying that LLM-based scoring tends to assign overly high scores, leading to selecting many categories that do not contain correct answers, which motivates future improvements on test-time scoring.

5 Related Works

Current research show that TTS is an effective method to improve LLMs performance (Snell et al., 2025; Wu et al., 2025; Agarwal et al., 2025; OpenAI et al., 2024a). Most existing TTS methods scale compute along (i) *parallel scaling*, which samples multiple candidate solutions and aggregates or selects a final answer (Wang et al., 2023; Brown et al., 2024; Yao et al., 2023; Besta et al., 2024), and many of these can be interpreted within the Minimum Bayes Risk (MBR) decoding framework (Bertsch et al., 2023; Kamigaito et al., 2025; Wood et al., 2024; Heineman et al., 2024); and (ii) *sequential scaling*, which spends more steps for deliberation, revision, or verification (Zeng et al., 2025). Recent work further studies *compute-optimal* and *adaptive* TTS, showing that the optimal allocation of test-time compute depends on prompt difficulty and that scaling behavior is highly sensitive to prompting strategy and model type (Snell et al., 2025; Wu et al., 2025; Agarwal et al., 2025; Liu et al., 2025). Very recent extensions explore training-free test-time scaling beyond single-model best-of-N, such as multi-model routing and agentic scaling (Geuter and Kornhardt, 2025; Song et al., 2025; Zhu et al., 2025), as well as efficiency-oriented interventions including verifier-based thinking compression and minimal test-time intervention (Lin et al., 2025; Yang et al., 2025b), and LLM-guided search for optimization problems (Li et al., 2025; Romera-Paredes et al., 2024).

Despite this progress, a key limitation is that most training-free TTS methods scale computation within a fixed, human-chosen reasoning format, varying only in decoding randomness, sample count, or reasoning length. Since TTS effectiveness is highly sensitive to formatting choices, a mismatched format can systematically bias the solution distribution and waste substantial test-time computation (Liu et al., 2025; Agarwal et al., 2025; Snell et al., 2025). Therefore, in this paper, we prove that using a single format can only enhance robustness, while adapting multiple formats can enhance inference ability. Based on this, we present FORMAT-ADAPTER to improve training-free TTS by generating and selecting suitable reasoning formats at inference time, aiming to enhance not only robustness but also capability by expanding the explored reasoning formats.

6 Conclusion

In this paper, we revisit TTS for LLM reasoning and argue that effective TTS should scale not only inference compute but also the reasoning format. We propose FORMAT-ADAPTER, a format-adaptive TTS framework that allocates a fixed test-time budget across multiple reasoning formats. First, we analyze that scaling compute within a single fixed format mainly improves robustness by reducing variance, whereas adapting multiple suitable formats can further improve capability by expanding the explored reasoning modes. Building on this perspective, FORMAT-ADAPTER leverages LLMs to generate and select reasoning formats under a given TTS budget, enabling the system to spend inference-time compute on suitable formats. Experiments on math and commonsense reasoning demonstrate that, under the same TTS budget, FORMAT-ADAPTER consistently outperforms prior TTS baselines, achieving an average improvement of 2.2% compared with existing methods, which demonstrates new SOTA results on the training-free parallel-scaling TTS, showing its effectiveness.

Limitations

Current research mainly focuses on training-free TTS, while how to adapt the motivation of this paper to the post-training is still under discovery.

Ethics Statement

All datasets and models used in this paper are publicly available, and our usage follows their licenses and terms. We have employed the LLM tools for coding and polishing.

References

Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. 2025. The art of scaling test-time compute for large language models. *Preprint*, arXiv:2512.02008.

Amanda Bertsch, Alex Xie, Graham Neubig, and Matthew Gormley. 2023. It's MBR all the way down: Modern generation techniques through the lens of minimum Bayes risk. In *Proceedings of the Big Picture Workshop*, pages 108–122, Singapore. Association for Computational Linguistics.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2025. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In *Forty-second International Conference on Machine Learning*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 516 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jonathan Geuter and Gregor Kornhardt. 2025. Robon: Routed online best-of-n for test-time scaling with multiple llms. *Preprint*, arXiv:2512.05542.

Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. 2022. Scaling laws for neural machine translation. In *International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruisong Zhang, Shirong Ma, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

David Heineman, Yao Dou, and Wei Xu. 2024. Improving minimum Bayes risk decoding with multi-prompt. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22525–22545, Miami, Florida, USA. Association for Computational Linguistics.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep learning scaling is predictable, empirically. *Preprint*, arXiv:1712.00409.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

Hidetaka Kamigaito, Hiroyuki Deguchi, Yusuke Sakai, Katsuhiko Hayashi, and Taro Watanabe. 2025. Diversity explains inference scaling laws: Through a case study of minimum Bayes risk decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29060–29094, Vienna, Austria. Association for Computational Linguistics.

Zhewei Kang, Xuandong Zhao, and Dawn Song. 2025. Scalable best-of-n selection for large language models via self-certainty. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Dong Li, Xujiang Zhao, Linlin Yu, Yanchi Liu, Wei Cheng, Zhengzhang Chen, Zhong Chen, Feng Chen, Chen Zhao, and Haifeng Chen. 2025. SolverLLM:

Leveraging test-time scaling for optimization problem via LLM-guided search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Weizhe Lin, Xing Li, Zhiyuan Yang, Xiaojin Fu, Hui-Ling Zhen, Yaoyuan Wang, Xianzhi Yu, Wulong Liu, Xiaosong Li, and Mingxuan Yuan. 2025. Trimr: Verifier-based training-free thinking compression for efficient test-time scaling. *Preprint*, arXiv:2505.17155.

Yexiang Liu, Zekun Li, Zhi Fang, Nan Xu, Ran He, and Tieniu Tan. 2025. Rethinking the role of prompting strategies in LLM test-time scaling: A perspective of probability theory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27962–27994, Vienna, Austria. Association for Computational Linguistics.

Xianzhen Luo, Qingfu Zhu, Zhiming Zhang, Libo Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Python is not always the best choice: Embracing multilingual program of thoughts. *Preprint*, arXiv:2402.10691.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, Suzhou, China. Association for Computational Linguistics.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. 2002. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024a. Openai o1 system card. *Preprint*, arXiv:2412.16720.

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 262 others. 2024b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2695–2709, Singapore. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco Zhernov, Heiko Ruiz, and 1 others. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–477.

Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

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.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Zhende Song, Shengji Tang, Peng Ye, Jiayuan Fan, Lei Bai, Tao Chen, and Wanli Ouyang. 2025. Ctts: Collective test-time scaling. *Preprint*, arXiv:2508.03333.

Xuezhi Wang, Haohan Wang, and Diyi Yang. 2022. Measure and improve robustness in NLP models: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4569–4586, Seattle, United States. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Danny Wood, Tingting Mu, Andrew M. Webb, Henry W. J. Reeve, Mikel Luján, and Gavin Brown. 2024. A unified theory of diversity in ensemble learning. *J. Mach. Learn. Res.*, 24(1).

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2019. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2578–2589, Hong Kong, China. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhen Yang, Mingyang Zhang, Feng Chen, Ganggui Ding, Liang Hou, Xin Tao, Pengfei Wan, and Ying-Cong Chen. 2025b. Less is more: Improving llm reasoning with minimal test-time intervention. *Preprint*, arXiv:2510.13940.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. 2025. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4651–4665, Vienna, Austria. Association for Computational Linguistics.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Baoxin Wang, Dayong Wu, Qingfu Zhu, and Wanxiang Che. 2024. Flextaf: Enhancing table reasoning with flexible tabular formats. *Preprint*, arXiv:2408.08841.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, Chao Cao, Hanqi Jiang, Hanxu Chen, Yiwei Li, Junhao Chen, Huawen Hu, Yihen Liu, Huaqin Zhao, Shaochen Xu, and 59 others. 2024. Evaluation of openai o1: Opportunities and challenges of agi. *Preprint*, arXiv:2409.18486.

Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. 2024. Don't trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*.

Yilun Zhou, Austin Xu, PeiFeng Wang, Caiming Xiong, and Shafiq Joty. 2025. Evaluating judges as evaluators: The JETTS benchmark of LLM-as-judges as test-time scaling evaluators. In *Forty-second International Conference on Machine Learning*.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. 2025. Scaling test-time compute for llm agents. *Preprint*, arXiv:2506.12928.

A Proof of Equations

Lemma 1. *Let $m, \phi, \bar{\phi}$ follow the definition in §2.1. If $\lim_{m\to\infty} \bar{\phi} = \phi \circ \mathsf{f}$, we can derive that $\lim_{m\to\infty} \delta_m = 0$.*

Proof. Considering that $\bar{\phi} = \mathsf{avg}(\phi_i) = \mathsf{avg}(\phi \circ \mathsf{f} + \delta_i)$, we can derive that

$$\mathsf{avg}(\phi \circ \mathsf{f} + \delta_i) = \phi \circ \mathsf{f}(m \to \infty)$$

Therefore, $\mathsf{avg}(\delta_i) = 0(m \to \infty)$. Assume, for contradiction, that $\lim_{m\to\infty} \delta_m \neq 0$. Then, there exists some $\epsilon > 0$ such that for large enough m, $\delta_m \geq \epsilon$. For large m, the average of the first m terms is

$$\frac{\delta_1 + \delta_2 + \cdots + \delta_m}{m}$$

Since the average tends to 0, for sufficiently large m, we must have

$$\frac{\delta_1 + \delta_2 + \cdots + \delta_m}{m} < \epsilon$$

However, if there are infinitely many $\delta_m \geq \epsilon$, this contradicts the fact that the average tends to 0. Thus, $\lim_{m\to\infty} \delta_m = 0$. □

Considering Lemma 1, in the following proof, we substitute $m \to \infty$ with $\delta_m \to 0$.

A.1 Proof of Equation 4

Theorem 1. *Let $D, L, m, \phi, \bar{\phi}$ follow the definition in §2.1. We can derive that:*

$$\lim_{\delta_i \to 0} \mathbb{E}_D \left[L(\bar{\phi}, y)\right] = \frac{1}{m} \sum_{i=1}^{m} L\left(\phi \circ \mathsf{f}, y\right)$$

Proof.

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_D \left[L\left(\phi_i, y\right)\right] \tag{7}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_D \left[L\left(\phi \circ \mathsf{f} + \delta_i, y\right)\right] \tag{8}$$

$$= \mathbb{E}_D \left[L\left(\phi \circ \mathsf{f}, y\right)\right] (\delta_i \to 0) \tag{9}$$

Considering that:

$$\bar{\phi} = \frac{1}{m} \sum_{i=1}^{m} \phi_i \tag{10}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \phi \circ \mathsf{f}(\delta_i \to 0) \tag{11}$$

$$= \phi \circ \mathsf{f} \tag{12}$$

We can derive that:

$$\mathbb{E}_D \left[\frac{1}{m} \sum_{i=1}^{m} L\left(\phi_i, \bar{\phi}\right)\right] \tag{13}$$

$$= \mathbb{E}_D \left[\frac{1}{m} \sum_{i=1}^{m} L\left(\phi \circ \mathsf{f} + \delta_i, \bar{\phi}\right)\right] \tag{14}$$

$$= \mathbb{E}_D \left[\frac{1}{m} \sum_{i=1}^{m} L\left(\phi \circ \mathsf{f}, \phi \circ \mathsf{f}\right)\right] (\delta_i \to 0) \tag{15}$$

$$= 0 \tag{16}$$

Based on Equation 2, we can derive that:

$$\mathbb{E}_D \left[L(\bar{\phi}, y)\right] \tag{17}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_D \left[L\left(\phi_i, y\right)\right] \tag{18}$$

$$- \mathbb{E}_D \left[\frac{1}{m} \sum_{i=1}^{m} L\left(\phi_i, \bar{\phi}\right)\right] \tag{19}$$

$$= \mathbb{E}_D \left[L\left(\phi \circ \mathsf{f}, y\right)\right] (\delta_i \to 0) \tag{20}$$

□

A.2 Proof of Equation 5

Theorem 2. *Let $D, L, m, \phi, \bar{\phi}$ follow the definition in §2.2. we can derive that:*

$$\lim_{\delta_i \to 0} \mathbb{E}_D \left[L(\bar{\phi}, y)\right] \tag{21}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_D \left[L(\phi \circ \mathsf{f}_i, y)\right] \tag{22}$$

$$- \mathbb{E}_D [\frac{1}{m} \sum_{i=1}^{m} L(\phi \circ \mathsf{f}_i, \bar{\phi})] \tag{23}$$

Proof.

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_D \left[L\left(\phi_i, y\right)\right] \tag{24}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_D \left[L\left(\phi \circ \mathsf{f}_i + \delta_i, y\right)\right] \tag{25}$$

$$= \frac{1}{m} \sum_{i=1}^{m} L\left(\phi \circ \mathsf{f}_i, y\right) (\delta_i \to 0) \tag{26}$$

$$\lim_{\delta_i \to 0} \mathbb{E}_D \left[\frac{1}{m} \sum_{i=1}^{m} L\left(\phi_i, \bar{\phi}\right)\right] \tag{27}$$

$$= \mathbb{E}_D \left[\frac{1}{m} \sum_{i=1}^{m} L\left(\phi \circ f_i, \bar{\phi}\right)\right] \tag{28}$$

Based on Equation 2, we can derive that:

$$\mathbb{E}_D\left[L(\bar{\phi}, y)\right] \tag{29}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_D\left[L\left(\phi_i, y\right)\right] \tag{30}$$

$$- \mathbb{E}_D\left[\frac{1}{m}\sum_{i=1}^{m}L\left(\phi_i, \bar{\phi}\right)\right] \tag{31}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_D\left[L(\phi \circ \mathtt{f}_i, y)\right] \tag{32}$$

$$- \mathbb{E}_D[\frac{1}{m}\sum_{i=1}^{m}L(\phi \circ \mathtt{f}_i, \bar{\phi})](\delta_i \to 0) \tag{33}$$

□

B Additional Information

B.1 Prompts of FORMAT-ADAPTER

The prompts of FORMAT-ADAPTER are shown in Table 3. The prompt for the instruction rewriting is provided in the code since this prompt is too long. The prompts of the answer generation of each task follow Dubey et al. (2024), which can be found in `https://huggingface.co/datasets/meta-llama/Llama-3.1-8B-Instruct-evals`.

B.2 Baselines of Main Experiments

Single is to generate one answer using one format with Chain-of-Thought (Wei et al., 2022). The prompts we used follow Dubey et al. (2024).

Self-Consistency is similar to Single, while we generate multiple answers for each question. The generation number is the same as the format number of FORMAT-ADAPTER for each task, and we set the temperature as 0.5 and top_p as 0.9. The prompts are the same as with Single.

Tree-of-Thought is to generate the reasoning process step by step, where it evaluates the results of each step, which are used as the input for the next step. The parameters and prompts we used follow the defaults outlined in the paper.

DTV asks models to generate Isabelle formulations (Nipkow et al., 2002) to answer the questions, which can be executed automatically to ensure the logical correctness of the consistent answers. The parameters and prompts we used follow the defaults outlined in the paper.

Self-Certainty is a reward-free confidence metric for best-of-N sampling in LLMs, which directly uses the model's own token probability distributions during generation. At each decoding step, it measures how far the predicted distribution deviates from a uniform distribution, where a more peaked distribution indicates higher certainty. These token-level scores are averaged to obtain a sentence-level self-certainty score for each sampled answer. The model then ranks candidate responses by this score and, using a Borda-style weighted voting scheme, achieves scalable and efficient selection that also generalizes to open-ended tasks.

Shortest MV is a parallel test-time scaling method that re-ranks answers by combining vote counts with chain-of-thought length. Given multiple sampled solutions, the model first clusters them by their final answers. For each answer category, it counts the number of supporting solutions and computes their average length. Shortest MV then assigns a score and selects the answer with the highest score. This scoring favors clusters that are both popular and concise, leveraging the empirical finding that correct solutions are usually shorter than incorrect ones.

Forest-of-Thought is a test-time compute scaling framework that runs multiple reasoning trees in parallel and aggregates them via collective decision-making (Bi et al., 2025). FoT employs sparse activation to selectively activate only the most relevant trees (or key nodes/paths) for inference, improving both efficiency and accuracy. It also incorporates a dynamic self-correction strategy to revise intermediate mistakes in real time and reduce error propagation. Finally, it adopts consensus-guided decision-making (and optional early-stopping strategies) to select the final answer under a given compute budget.

B.3 Reasoning Formats of FORMAT-ADAPTER

In this section, we list the reasoning formats generated by different LLMs on various datasets, as shown in Table 4. We rename some reasoning categories in the experiments of §4.4 to ensure that similar categories can be compared together. Different formats in Table 4 may be more suitable for different types of questions. For instance, for numerical representation, "12" is appropriate for numerical computations, whereas "twelve" is more

The prompt of Format Generation

You are requested to generate possible answer formats that can be changed for the given task, where I want to generate different answers in different formats of the given task.
For each task, you MUST generate the possible answer formats quoted with ** of the task, the number of answer formats of each task MUST > 3.
Here are several examples:

—

Task: Code Generation.
In this task, you are given a question, and then you should generate the Python code to answer the question.
Input: Today is the last day of the first quarter of 2008. What is the date one year ago from today?
Output:
```python
from datetime import datetime, timedelta
today = datetime(2008, 3, 31)
one_year_ago = today - timedelta(days=365)
```
The possible answer formats that can be changed are:
1. Natural Language: The natural languages of questions can be changed, like change as **Chinese, French, German, Spanish**.
2. Code Language: The code languages of answers can be changed, like change to **Java, C++, R, JavaScript**.

—

Based on the above examples, generate the possible answer formats to be changed for the following task.

Task: {task_name}
{task_definition}
Output: {answer}

The prompt of Answer Scoring

Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider correctness and helpfulness. You will be given a assistant's answer. Identify and correct any mistakes. Be as objective as possible. After providing your explanation, you must rate the response on a scale of 1 to 10 by strictly following this format: "[rating]", for example: "Rating: [5]".

[Question]
{question}

[The Start of Assistant's Answer]
{answer}
[The End of Assistant's Answer]

Table 3: The prompts of FORMAT-ADAPTER.

suitable for non-numerical queries, such as "how many 'e' are in 12?"

From Table 4, we can observe that: *(i)* Compared to small-scale LLMs, large-scale LLMs are capable of generating a wider variety of reasoning formats, leading to a more significant performance improvement, as demonstrated in Table 6; *(ii)* Compared with simple datasets (e.g., GSM8K), a greater number of reasoning formats are generated on more complex datasets (e.g., MATH, GPQA), as more solving approaches are available for complex questions, thus resulting in more diverse reasoning formats.

Although the number of synthesized formats varies across different tasks and models, the comparability of the results is reliable. That is because some variation in the number of formats does exist between different tasks; however, these quantitative differences can be considered variations in the intermediate process, reflecting the model's tendency to synthesize different formats depending on the task. Furthermore, since the experimental setup is consistent across different tasks (e.g., instructions, hyperparameters), we consider the comparability of our results to be reliable.

Model	GSM8K-Hard	MATH500	ARC-C-Hard	GPQA
Llama-3.1-8B	**natural language (6)**, code language (2), mathematical notation (2), text format (2), answer style (2), response format (1)	**natural language (6)**, step-by-step format (3), text format (2), explanation level (1), mathematical notation (2)	**natural language (8)**, answer format (2), code language (6), explanation level (4), answer style (2), output format (3)	natural language (5), answer format (5), **explanation level (6)**, code language (5), answer style (4), explanation format (5), step-by-step format (4), explanation style (5), mathematical notation (4)
Llama-3.1-70B	mathematical notation (4), natural language (4), problem format (4), answer format (4), **reasoning style (3)**, unit of measurement (3)	mathematical notation (3), problem format (5), **solution approach (3)**, answer format (3), unit system (3), problem complexity (3)	natural language (4), answer format (1), question type (1), answer choice format (4), **context format (3)**, answer justification (1)	natural language (4), answer format (9), explanation format (1), **question type (4)**, candidate answer format (7), explanation style (4), answer choice format (7), mathematical notation (6)
gpt-5-nano	natural language (4), mathematical expression (4), **explanation style (4)**, number representation (5)	**natural language (6)**, explanation format (2), notation style (2), answer presentation (2), units in solution (2), solution format (3), mathematical representation (3), concluding sentence format (3)	**natural language (5)**, numerical representation (3), answer structure (2), answer explanation (4), response format (3), question format (4), contextual explanation (2), answer representation (8)	natural language (4), numerical representation (3), answer presentation (2), **explanation detail (2)**, answer format (3)

Table 4: The reasoning categories generated by FORMAT-ADAPTER on different models and datasets. The number after each category is the format number corresponding to the category. The category with the best performance under each setting is marked in **bold**.

C Additional Discussion

C.1 Efficiency of FORMAT-ADAPTER

In this section, we discuss the efficiency of FORMAT-ADAPTER. For all experiments conducted on gpt-5-nano, the total cost is approximately $600, averaging around $0.30 per question. We focus on two main aspects: the efficiency of format generation and the efficiency during inference.

C.1.1 Efficiency during Format Generation

Let the number of generated formats be M, and $t_{\mathcal{M}}$ represents the average time that LLM $\mathcal{M}$ takes to process a single data point. Considering that format generation requires both generation and rewriting, the efficiency of format generation is $2Mt_{\mathcal{M}} = O(Mt_{\mathcal{M}})$.

Based on the discussion, we can adjust M to control the efficiency of format generation. Furthermore, in practical applications, since format generation is performed offline, the cost of this step can be ignored during online inference.

C.1.2 Efficiency during Reasoning

Let the number of formats selected for each query during inference be denoted as m, the total number of user queries be N, and t_s represents the time to select a single format. Since inference involves format selection, answer generation, and answer scoring, the total inference efficiency is given by $NMt_s + 2mNt_{\mathcal{M}}$. Given that $t_s \ll t_{\mathcal{M}}$ in practice, the overall inference efficiency simplifies to $O(mNt_{\mathcal{M}})$.

Method	EM	#Token
Self-Consistency	51.4	13889.9
Tree-of-Thought	52.8	24611.4
DTV	55.4	17816.4
Self-Certainty	59.0	20321.7
Shortest MV	58.2	18259.2
Forest-of-Thought	59.2	26267.0
FORMAT-ADAPTER	60.4	19297.0

Table 5: The average output tokens per question on MATH using Llama-3.1-8B.

It can be observed that the inference efficiency of FORMAT-ADAPTER is comparable to that of Self-Consistency, while FORMAT-ADAPTER offers a significant performance improvement. Considering that prior research indicates a positive correlation between model performance and inference time (Snell et al., 2024; Zhong et al., 2024), it is important to balance efficiency and performance based on the specific application scenario. For example, when computational resources are limited, the number of reasoning formats used can be reduced to enhance inference efficiency.

Question:
Paul and Jesse each choose a number at random from the first six prime numbers. What is the probability that the sum of the numbers is even?

Answer with Original Format:
The probability of Paul choosing an odd prime is 5/6, and the probability of Jesse choosing an odd prime is also 5/6. Therefore, the probability of both choosing odd primes is:
5 / 6 × 5 / 6 = 25 / 36
The total probability of an even sum is \boxed{25 / 36}.

Answer with Set Notation:
All primes: P = {2, 3, 5, 7, 11, 13}
Even primes: E = {2}
Odd primes: O = {3, 5, 7, 11, 13}
Sum is even if both choose even primes or both choose odd primes.
P(E, E) = 1 / 36
P(O, O) = 25 / 36
P(even) = 1 / 36 + 25 / 36 = 13 / 18
So the answer is: \boxed{13 / 18}

Figure 8: An example sampled from MATH answered using different reasoning formats. The correct part is marked in red, and the incorrect part is marked in green.

C.1.3 Average Output Tokens of Different Method

To compare the efficiency of FORMAT-ADAPTER with other baselines in practical applications, we measure the average number of tokens output per question, as shown in Table 5. Although FORMAT-ADAPTER is less efficient than Self-Consistency, our method is closer to that of Tree-of-Thought. Considering the performance improvements of FORMAT-ADAPTER over both Self-Consistency and Tree-of-Thought, a balance between efficiency and performance must be considered in practical applications.

C.2 Case Study

To better understand how FORMAT-ADAPTER improves reasoning performance, we present a case study, as shown in Figure 8. From the figure, it can be observed that when using the original reasoning format, the model overlooks that 2 is also an odd number, leading to an incorrect answer. However, when reasoning with the set notation, the model successfully accounts for all odd numbers, resulting in the correct answer. Therefore, utilizing different reasoning formats helps the model approach questions from multiple perspectives and different questions require different reasoning formats. As such, it is essential to integrate various reasoning formats to obtain the correct solution.

D Additional Experiments

D.1 FORMAT-ADAPTER Compared with Multi-Format Baseline

In this section, we compare FORMAT-ADAPTER with baselines using multiple formats, including CLSP (Qin et al., 2023), MultiPoT (Luo et al., 2024), and FlexTaF (Zhang et al., 2024).

Dataset	Method	8B		70B	
		Vote	Oracle	Vote	Oracle
MATH	CLSP	53.0	66.2	66.9	74.9
	MultiPoT	57.4	66.5	72.2	79.1
	Ours	**60.1**	**88.6**	**77.2**	**88.6**
WikiTQ	FlexTaF	38.0	70.3	41.5	79.0
	Ours	**55.2**	**79.7**	**63.1**	**84.0**

Table 6: EM of FORMAT-ADAPTER (Ours) and baselines using multiple reasoning formats on Llama-3.1. The best results of each setting are marked in **bold**.

The performance comparison between FORMAT-ADAPTER and baselines employing multiple reasoning formats is presented in Table 6. Following the setup of MultiPoT, we select 263 problems from MATH500 that can be resolved using code-based solutions. About FlexTaF, we conduct experiments on the WikiTQ dataset (Pasupat and Liang, 2015), which is the mainstream benchmark of the table QA task. As observed from the table, FORMAT-ADAPTER achieves an average improvement of 4.7% over the best performance of the baselines under each setting, demonstrating the effectiveness of our method.

D.2 Performance Using All Generated Formats

To validate the necessity of the reasoning format selection of FORMAT-ADAPTER, we compare its performance with that of using all formats without selection. The experimental results, as shown in Table 7, indicate that FORMAT-ADAPTER consistently outperforms that directly using all reasoning formats across all settings, which demonstrates the importance of selecting appropriate reasoning formats.

D.3 Answer Variability of FORMAT-ADAPTER

To demonstrate that FORMAT-ADAPTER can generate different answers using various reasoning formats, we calculate the number of distinct answers generated by our method and Self-Consistency. The results are shown in Table 8, where we observe that, on average, the number of answers generated by our method is significantly higher than that of Self-Consistency. This proves that employing different reasoning formats can guide the model to generate diverse answers.

Model	Method	GSM8K-Hard	MATH500	ARC-C-Hard	GPQA
Llama-3.1-8B	All	53.9	54.0	42.2	33.9
	FORMAT-ADAPTER	**54.7**	**56.8**	**57.4**	**33.9**
Llama-3.1-70B	All	73.8	70.2	69.9	47.5
	FORMAT-ADAPTER	**76.2**	**70.4**	**71.5**	**51.0**

Table 7: The performance with all formats or the formats selected by FORMAT-ADAPTER. All denotes using all generated formats. The best performance under each setting is marked in **bold**.

Model	Method	GSM8K-Hard	MATH500	ARC-C-Hard	GPQA
Llama-3.1-8B	Self-Consistency	2.0	2.3	1.0	2.3
	FORMAT-ADAPTER	**12.4**	**5.9**	**3.6**	**3.5**
Llama-3.1-70B	Self-Consistency	1.3	1.9	1.0	1.8
	FORMAT-ADAPTER	**9.5**	**7.8**	**2.5**	**3.2**

Table 8: The average number of distinct answers generated for each question.

D.4 Robustness of FORMAT-ADAPTER

To validate the robustness of FORMAT-ADAPTER, we run it five times with different random seeds, as shown in Table 9. From the table, it can be observed that the performance of our method does not fluctuate significantly. As discussed in § 2, generating multiple answers enhances robustness and reduces performance variability.

D.5 Evaluation of the Score Quality

In this section, we evaluate the quality of the scoring of our method. We use the evaluation metrics $\frac{\sum_{d \in D} \texttt{Valid}(d)}{|D|} \times 100$. $D = \{d\}$ represents all the data, and $\texttt{Valid}(d)$ indicates that if d is correct, it is represented by the corresponding score; otherwise, it is represented by $1-$ the corresponding score. The statistical results are shown in Table 10. It can be observed that the evaluation result is not high, which is consistent with the conclusions of previous studies, highlighting the bottleneck of our method. To address this issue, a possible approach is to select the most appropriate reasoning format for each task using the training data first, and then use the selected reasoning format during inference without relying on LLMs for scoring.

D.6 Repeated Sampling with Different Formats

In this section, we evaluate the performance of FORMAT-ADAPTER after repeated sampling, comparing the use of the single format and FORMAT-ADAPTER. The experimental results are shown in Figure 9. As can be observed from the figure, when the sampling scale is > 1, the performance using a single format is lower than that of FORMAT-ADAPTER, which demonstrates the effectiveness of our method.

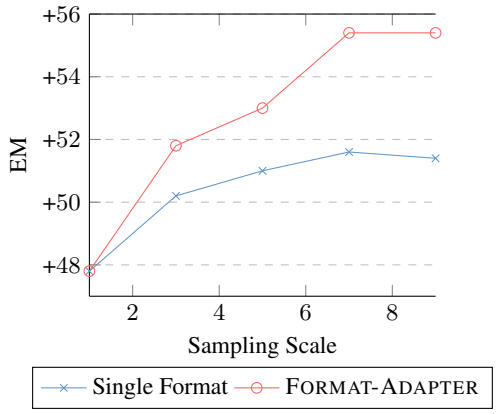

Figure 9: The performance of repeat sampling using single format and FORMAT-ADAPTER on MATH with Llama-3.1-8B.

D.7 Comparison with Code-Based Reasoning

To ensure that the performance improvement of our method, FORMAT-ADAPTER, stems from the diversity of answers rather than the use of a code executor, we compare our approach with the Self-Consistency method, which generates Python code. The experimental results are presented in Table 11. It is evident that even when code is utilized, FORMAT-ADAPTER still outperforms Self-Consistency, demonstrating that the performance enhancement of FORMAT-ADAPTER is indeed primarily attributable to the diversity of the generated answers.

Dataset	8B		70B	
	Vote	Oracle	Vote	Oracle
GSM8K-Hard	54.3 ± 0.4	89.6 ± 0.2	76.1 ± 0.3	94.4 ± 0.6
MATH500	56.7 ± 0.2	74.9 ± 0.2	70.2 ± 0.4	85.0 ± 0.5
ARC-C-Hard	57.2 ± 0.2	91.2 ± 0.4	71.5 ± 0.2	88.2 ± 0.6
GPQA	33.2 ± 0.8	93.3 ± 0.4	51.1 ± 0.7	96.2 ± 0.5
WikiTQ	55.0 ± 0.4	79.3 ± 0.5	63.0 ± 0.1	83.5 ± 0.6

Table 9: The average performance of FORMAT-ADAPTER with five running using Llama-3.1.

Dataset	8B	70B
GSM8K-Hard	47.7	66.2
MATH500	46.4	56.0
ARC-C-Hard	52.3	60.2
GPQA	45.7	48.1

Table 10: The average score quality of FORMAT-ADAPTER with Llama-3.1 on different datasets.

Method	MATH	GPQA
SC with Python	51.8	31.2
FORMAT-ADAPTER	**56.8**	**33.9**

Table 11: The performance of FORMAT-ADAPTER compared with self-consistency using python (SC with Python) using Llama-3.1-8B. The best performance under each setting is marked in **bold**.

D.8 Scoring with Different Methods

Considering that the score quality of LLM-as-a-Judge could be limited in §3.3, in this section, we discuss how to generate the score when the computational resources are sufficient. We first feed the candidate answers to the LLM in a pairwise manner to obtain the model preferences, and then use the win rate as the score for each answer. The experimental results are shown in Table 12, from which we can observe that the above scoring method achieves better performance across all settings. Considering that this approach incurs a higher computational cost, in practice, users should choose an appropriate scoring strategy based on available compute resources and performance requirements.

Method	GSM8K-Hard	MATH500	ARC-C-Hard	GPQA
-	53.9	54.2	52.7	30.1
LLM-as-a-Judge	54.7	60.4	57.4	34.3
Pairwise Preference	57.4	61.2	58.6	35.1

Table 12: The performance of FORMAT-ADAPTER with different scoring methods on Llama-3.1-8B. "-" denotes the performance without scoring.