Duel-of-Thoughts Reasoning

Anonymous ACL submission

Abstract

We present Duel-of-Thoughts (DoT), a general-purpose, black-box inference paradigm for test-time scaling (TTS) of large language models (LLMs), specifically tailored for best-of-N selection. DoT reframes reasoning as a binary comparison process, where the model deliberately contrasts two representative yet distinct 'thoughts' before reaching the final answer. Such pairwise comparison is adaptively triggered only when necessary, mimicking humans' more careful thinking under high decision uncertainty. Experiments on six benchmarks—including mathematics, commonsense, multi-hop logical reasoning, event planning and resource allocation—across both open-source and closed-source models demonstrate DoT's superiority against alternative strategies.

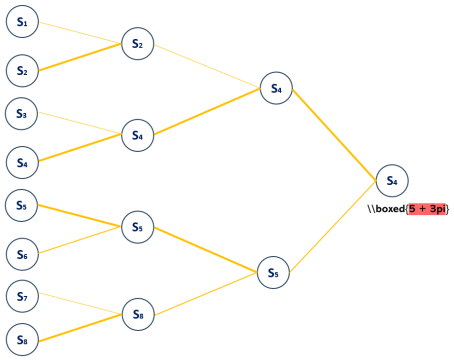


Figure 1: **Duel-of-Thoughts for Best-of-N selection**. DoT recasts best-of-N as a (series of) binary comparison task, each of which is substantially easier than the original reasoning problem and the best-of-N problem. The LLM *itself* can effectively identify reasoning errors through deliberative one-versus-one comparison, facilitating effective selection even *without* access to external reward models or internal model states.

1 Introduction

Large Language Models (LLMs) such as GPT-5 (OpenAI, 2025) and DeepSeek-R1 (Guo et al., 2025) have demonstrated remarkable progress across a wide range of complex reasoning tasks, including mathematical reasoning (Liu et al., 2025), computer programming (Qing et al., 2025), medical diagnosis (Singhal et al., 2025) and beyond.

A growing body of *test-time scaling* (TTS) techniques (Brown et al., 2024; Wu et al., 2025; Snell et al., 2025) has emerged to elicit more advanced reasoning capabilities from LLMs. Many works focus on more effective exploration of the solution space, such as beam search (Xie et al., 2023; Snell et al., 2025) and structured reasoning frameworks (Yao et al., 2023; Besta et al., 2024; Hao et al., 2023; Snell et al., 2025; Bi et al., 2025; Wang et al., 2025). Other works instead study the complementary problem of *how to select the best among N existing generations* (Wang et al., 2023; Fu et al., 2025; Kang et al., 2025; Zhao et al., 2025; Brown et al., 2024) – the primary focus of this work.

However, best-of-N selection remains challenging. Simpler methods such as majority voting (Wang et al., 2023) or using the model itself to select candidates (Ren et al., 2023) are proven sub-optimal for a wide range of tasks (Zhao et al., 2025; Wang et al., 2025). More powerful methods based on informed scoring signals exist, but they typically require domain-specific reward models (Lightman et al., 2023; Uesato et al., 2022; Cobbe et al., 2021; Li et al., 2023) or internal model states (Fu et al., 2025; Kang et al., 2025) for scoring, which may not always be available across tasks and models (for instance, tasks like commonsense answering may not come with off-the-shelf reward models, and internal states are typically inaccessible from proprietary models). These limitations significantly restrict the applicability of these methods, particularly in real-world, industrial deployments.

In this work, we propose *Duel-of-Thoughts (DoT)*, a general-purpose paradigm for best-of-N selection, designed to be widely applicable across

tasks and models. DoT is inspired by Thurstone's Law from cognitive science (Thurstone, 1927), which states that pairwise comparison of opinions is typically easier than absolute, individual scoring. Leveraging this insight, we strategically recast best-of-N selection as binary comparison problems (Figure 1). Such reformulation dramatically reduces task difficulty, allowing the model *itself* to effectively detect reasoning errors through deliberate, one-versus-one comparison, thereby naturally leading to reliable selection *without the aid of external reward models or internal signals*. Experiments across six tasks and both open- and closed-source models demonstrate that DoT offers strong accuracy while maintaining low computational overhead, especially when paired with a human-mimicking, adaptive thinking strategy.

To sum up, our contributions are as follows:

- (**General Best-of-N selection**) We propose *Duel-of-Thoughts (DoT)* for best-of-N selection, which reduces best-of-N choice to binary duel(s). DoT requires no external reward models or access to internal model states, while achieving strong accuracy and computational efficiency.
- (**Human-mimicking adaptive reasoning**) We introduce an adaptive extension of DoT that mirrors how humans reason more carefully under high uncertainty. This adaptive mechanism reduces DoT's computational cost by up to $\sim 90\%$ while preserving over 95% of the full accuracy.
- (**Extensive evaluation across models and tasks**) We assess DoT on 6 benchmarks spanning math, QA, multi-hop logical reasoning, and planning tasks with both closed and open models. The results demonstrate DoT is highly competitive even compared to internal states-based methods.

2 Preliminaries

Throughout this work, we use $p_{\text{LLM}}(\cdot)$ to denote the probability distribution induced by a LLM. A reasoning trajectory is denoted by S, and the corresponding final answer is denoted by $a = \mathcal{A}(S)$, where $\mathcal{A}$ is an answer extraction algorithm. We use a_{GT} to denote the ground truth (GT) answer.

2.1 Problem Formulation

Objective We consider the problem of selecting the best reasoning trajectory from a *pre-existing* candidate pool $\mathbf{S}$ of size $|\mathbf{S}| = N$:

$$\mathbf{S} = \{S_1, \ldots, S_N\}, \quad S_i \sim p_{\text{LLM}}(\cdot \mid q, t), \quad (1)$$

where each S_i is sampled from the LLM conditioned on the input question q and a task-specific prompt t. Any methods can be used to generate the pool $\mathbf{S}$, including both parallel thinking and structured reasoning methods. Here, we simply focus on the case where all S_i are generated independently.

Our objective is to select a trajectory $S^* \in \mathbf{S}$ whose answer matches the ground-truth answer:

$$S^* = \arg\max_{S \in \mathbf{S}} \mathbf{1}\big[\mathcal{A}(S) = a_{\text{GT}}\big]. \quad (2)$$

Assumptions We focus on designing an optimal selection mechanism under two constraints, both of which are realistic in real-world deployments:

- *No external reward models*. We assume that no process reward models (PRMs) (Lightman et al., 2023) or outcome reward models (ORMs) (Cobbe et al., 2021) are available for scoring candidates. This setting is common in non-mathematical reasoning tasks such as commonsense answering and resource planning.
- *No internal model states*. We further assume that internal model signals such as hidden states or token probabilities are inaccessible – a typical setting for proprietary models (e.g. GPT-series).

2.2 Best-of-N Selection

General methods A wide range of best-of-N sampling strategies have been proposed, each exhibiting different trade-offs and resource requirements. Popular methods include majority voting (Wang et al., 2023) or directly leveraging the LLM itself to score and aggregate candidates (Ren et al., 2023; Chen et al., 2024b). These methods are widely applicable across models and domains, yet their performance might be limited due to the lack of informed signals to guide selection (Zhao et al., 2025; Snell et al., 2025; Brown et al., 2024).

Auxiliary signals-based methods On the other hand, there exist methods that employ external reward models (RMs) or internal states to acquire more informed selection signals. One common practice is to score each candidate trajectory by a RM and then select the highest-scoring candidate (Snell et al., 2025; Wang et al., 2025; Li et al., 2023; Chen et al., 2024a). Another line of work derives confidence estimates from token probabilities and use them to guide aggregation (Fu et al., 2025; Kang et al., 2025). Despite strong performance, these approaches are subject to the availability of off-the-shelf RMs or internal states, which may not always be available across applications and models.

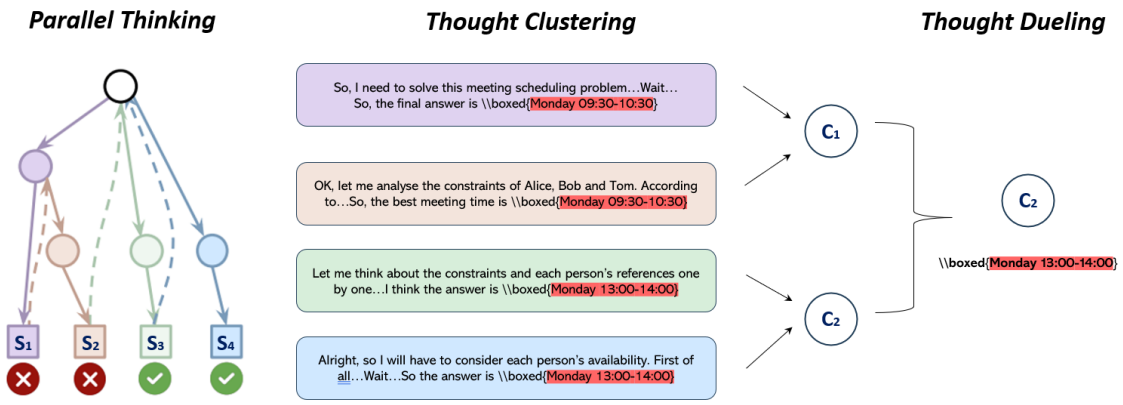


Figure 2: **Pipeline of DoT (basic version)**. DoT strategically reformulates the best-of-N selection as a binary selection problem. *Left*: parallel thinking to generate the N chains $\mathbf{S} = \{S_1, ..., S_N\}$. *Middle*: thought clustering to group chains according to their final answers $a = \mathcal{A}(S)$. *Right*: thought dueling, which prompts the LLM itself to perform one-versus-one comparison between two candidate thoughts $S' \sim \mathcal{C}_1, S'' \sim \mathcal{C}_2$ from two clusters $\mathcal{C}_1, \mathcal{C}_2$ to reach the final answer. In this example, exactly one sample from each of the two largest clusters are selected.

3 Duel-of-Thoughts

We introduce *Duel-of-Thoughts* (DoT), a powerful, fully black-box framework for best-of-N selection. DoT is inspired by a long-standing observation from classic cognitive science (Thurstone, 1927) and recent LLM-judge studies (Zheng et al., 2023):

"While absolute scoring of each opinion is difficult in isolation, comparing two opinions is easier."

This asymmetry arises because focused, *pairwise* comparison encourages deliberate contrast of reasoning steps and logical consistency, making errors more salient and easier to identify.

Building on this insight, DoT reformulates best-of-N as binary comparison tasks. Due to the reduced difficulty of such focused comparisons, each binary decision can be reliably handled by the LLM *itself* even without external reward models or internal model signals, allowing accurate selection.

In what follows, we first present a basic version of DoT that performs a single duel between two shortlisted thoughts, then discuss a full extension that incorporates all candidate thoughts.

3.1 Thoughts Clustering

In this stage, we group different generations form the existing N reasoning trajectories $\mathbf{S} = \{S_1, ...S_N\}$ into several 'thoughts', from which we select the two reference thoughts $S', S'' \in \mathbf{S}$.

Natural clusters of thoughts Different reasoning trajectories $\mathbf{S} = \{S_1, ...S_N\}$ naturally form thought clusters. For instance, one can cluster the trajectories by their embeddings (Kamthawee et al., 2024) , semantics (Farquhar et al., 2024) or even the final answer $a_i = \mathcal{A}(S_i)$. In this work, we focus on answer-based clustering[1] due to its simplicity.

Definition 1 (Thought Cluster)**.** *A Thought Cluster $\mathcal{C}$ is defined as the set of all reasoning trajectories S_i with identical final answer:*

$$\mathcal{C}(a) = \{S_i | \mathcal{A}(S_i) = a\} \tag{3}$$

where a is a specific answer in reasoning.

Given the clusters, we select two promising yet distinct *candidate thoughts* S', S'' for subsequent binary comparison. Here, we simply take S', S'' as random samples from the largest two clusters:

$$S' \sim \mathcal{C}(a'), \quad S'' \sim \mathcal{C}(a''), \tag{4}$$
$$a' = \arg\max_{a} |\mathcal{C}(a)|, \quad a'' = \arg\max_{a \neq a'} |\mathcal{C}(a)|$$

which produces two candidate thoughts S', S'' that are (i) representative of the dominant reasoning patterns and (ii) guaranteed to be diverse for capturing different reasoning logic for later comparison.

Clusterability of reasoning tasks Thought clustering applies across many real-world reasoning tasks. For instance, any mathematical or temporal reasoning tasks can be easily grouped by the final numeric answer (e.g. $\frac{3\pi}{2} + e$), and short strings such as tool-calling commands in agentic workflows (e.g. `apis.spotify.login`) are also clusterable. In Table 1, we provide more examples

[1]For open-ended generation, one can directly use the model itself to perform answer clustering, similar to the popular universal self-consistency method (Chen et al., 2024b).

of directly clusterable tasks, spanning numerical reasoning, multi-hop QA, knowledge retrieval, resource allocation, temporal reasoning and beyond.

3.2 Thoughts Dueling

In this stage, we ask the LLM *itself* to determine the best solution between the two candidate thoughts S', S'' through dedicated one-vs-one comparison. This is done by generating a new reasoning trajectory S^* based on a *duel prompt* t_{DoT} tailored for deliberative binary comparison between the two candidate thoughts S' and S'':

$$S^* \sim p_{\text{LLM}}(S^*|q, S', S'', t_{\text{DoT}}) \tag{5}$$

and the final answer is extracted as

$$a_{\text{DoT}} = \mathcal{A}(S^*) \tag{6}$$

In this process, the two candidates S', S'' only serve as *references* rather than a fixed candidate pool. The LLM can still synthesize or derive a better solution after contrasting two imperfect candidates, reducing the risk of suboptimal candidates.

Prompt template for binary comparison We design the prompt t_{DoT} to explicitly elicit the LLM to assess reasoning consistency, correctness, and logical soundness in a *side-by-side* manner:

Thought Dueling Prompt t_{DoT}

```
For the question above, there already exist
two solutions for your reference. Your task
is to carefully compare these two solutions,
analyze their potential mistakes by
contrasting them step-by-step.
Choose the best solution between these two
solutions. If neither solution is correct,
show me your own solution.

Below are the two reference solutions:

[Solution #1] ... \boxed{ANSWER 1}
[Solution #2] ... \boxed{ANSWER 2}

Compare the two solutions above carefully.
Tell me why one solution is more preferable
than the other. Show me your reasoning.
Return the final answer as \boxed{ANSWER}.
```

Figure 3: **Example of Thought Dueling Prompt t_{DoT}.**

Considering all thought clusters So far, we have discussed DoT where only the top-2 clusters are considered. This can be easily extended to top-C thought clusters by recursively applying basic DoT in a tournament-style manner; see Figure 1. This extension avoids the risk of excluding the correct thought cluster at the cost of more compute.

Algorithm 1 Duel-of-Thoughts (Basic Version)

Input: task q, task prompt t, duel prompt t_{DoT}
Output: final answer a^*
Params: candidate numbers N, threshold h
Initialization: $\mathcal{C}(a) = \varnothing$

▷ *Thought Clustering*
Sample $\{S_1, ...S_N\}$ with $S_i \sim p_{\text{LLM}}(S_i|q, t)$
for i in 1 to N **do**
  $a_i = \mathcal{A}(S_i)$
  $\mathcal{C}(a_i) \leftarrow \mathcal{C}(a_i) \cup S_i$
end for
$S' \sim \mathcal{C}(a')$ where $a' = \arg\max_a |\mathcal{C}(a)|$
$S'' \sim \mathcal{C}(a'')$ where $a'' = \arg\max_{a \neq a'} |\mathcal{C}(a)|$

▷ *Thought Dueling*
Compute $p_i = \frac{|\mathcal{C}(a_i)|}{\sum_j |\mathcal{C}(a_j)|}, \forall i$
Compute $\mathcal{H} = -\sum_i p_i \log p_i$
if $\mathcal{H} \leq h$ **then**
  $a^* = a'$
else
  $a^* = \mathcal{A}(S^*), S^* \sim p_{\text{LLM}}(S^*|q, S', S'', t_{\text{DoT}})$
end if
return a^*

4 Thought Entropy-driven Adaptive Thinking

In this section, we extend DoT with an adaptive reasoning strategy to improve its computational efficiency, which sparsely triggers thought dueling.

Human-mimicking deeper thinking Our intuition comes from the adaptive nature of human thinking: when several attempts to the same question converge to the same final answer, humans typically tend to trust that answer; however, when the attempts disagree, we often need a pause and a careful comparison. To capture this intuition, we introduce the measure of *thought entropy* to quantify convergence of thoughts:

Definition 2 (Thought Entropy)**.** *The Thought Entropy $\mathcal{H}(q)$ for a specific question q is defined as*

$$\mathcal{H} := -\sum_i p_i \log p_i \tag{7}$$

where $p_i := |\mathcal{C}(a_i)| / \sum_j |\mathcal{C}(a_j)|$ is the proportion of the thoughts belonging to the ith thought cluster.

Intuitively, a very low thought entropy $\mathcal{H}$ indicates that most reasoning trajectories lead to the same answer. This suggests that the majority answer may be a reliable estimate, thereby requiring

	GSM8K	MATH-500*	CommonsenseQA	Pronto-QA	Natural Plan	TeamAlloc
Task type	Mathematics	Mathematics	Commonsense	Logical reasoning	Temporal reasoning	Productivity
Dataset size $\lvert\mathcal{D}\rvert$	7,423	500	9,721	500	500	250
Multiple GT?	N	N	N	N	Y†	N
RM available?	Y	Y	N	N	N	N
Example answer	`1280`	$(2, \frac{\pi}{3})$	`A: Paris`	`True`	`Tue 9am-10am`	`{Tom, Sam, Ray}`
Source	(Cobbe et al., 2021)	(Hendrycks et al., 2021)	(Talmor et al., 2019)	(Saparov and He, 2023)	(Zheng et al., 2024)	(Sprague et al., 2024)

*For MATH-500, we select a 200-size subset with difficulty $\geq$ level 3 to highlight the gap between different methods.
† For task with multiple ground truths (GT), accuracy is assessed via any-match.

Table 1: **Summary of benchmarks considered.** For Natural Plan dataset, we take the Meeting scheduling and planning subset of the original dataset (Zheng et al., 2024). TeamAlloc corresponds to the team allocation subset of the MUSR dataset (Sprague et al., 2024). These datasets vary substantially in domain, scale, and input length, yet all exhibit *directly clusterable answer spaces*, highlighting the broad applicability of the proposed DoT method.

no further comparison with minority thoughts. In contrast, a high entropy $\mathcal{H}$ suggests a significant divergence across thoughts, which often happens when the question is beyond the capacity of the model or is itself ambiguous. In such regime, a careful, one-versus-one examination of the two different reasoning paths becomes especially valuable.

Based on this insight, we propose to trigger dueling only when $\mathcal{H}$ is sufficiently high:

$$\begin{cases} a^* = a', & \text{if } \mathcal{H} \leq h, \\ a^* = a_{\text{DoT}}, & \text{if } \mathcal{H} > h. \end{cases} \tag{8}$$

where $h \in \mathbb{R}$ is a user-defined threshold, $a' = \arg\max_v |\mathcal{C}(v)|$ is the answer from majority voting, and a_{DoT} is the final answer produced by the vanilla duel-of-thoughts procedure (eq. 6).

This way, thought dueling only happens when the model's own inconsistency indicates that careful reasoning is needed, dramatically saving computation cost up to $\sim$90%, as will be shown in §5.3.

Threshold selection The selection of the thought entropy threshold h reflects the trade-offs between accuracy and computation efficiency:

- *Case* $h \to 0$. Thought dueling is triggered whenever there is any disagreement, which prioritizes accuracy at the cost of computation.
- *Case* $h \to \infty$. The model trusts majority voting and rarely invokes thought dueling, maximizing efficiency but potentially sacrificing accuracy.

A detailed empirical analysis of how varying h affects this trade-off is provided in Figure 6. In practice, we simply set h to the entropy level at which 75% of the final answers agree, which we find robust across models and application domains.

5 Experiments

5.1 Setups

Baselines We compare DoT with five baselines, all of which derive a final answer from a fixed, pre-generated candidate pool $\mathbf{S} = \{S_1, \ldots, S_N\}$:

- *CoT* (Wei et al., 2022), the vanilla chain-of-thought reasoning, which corresponds to randomly selecting one candidate $S_i \sim \mathbf{S}$ from $\mathbf{S}$.
- *Majority voting* (Wang et al., 2023), which simply chooses the most frequent answer. It can be viewed as a degenerate case of DoT that always selects the largest thought cluster.
- *LLM aggregation* (Chen et al., 2024b). This method leverages the LLM *itself* to score and aggregate candidates in a generative manner.
- *Self-verification* (Miao et al., 2024; Xie et al., 2023; Madaan et al., 2023; Shinn et al., 2023), where the model is prompted to verify and potentially revise its previous generation.
- *Self-certainty* (Kang et al., 2025). We specifically compare to this state-of-the-art, *white-box* method that leverages certainty scores derived from token probabilities[2] to select candidates.

Generation setups For each question $q \in \mathcal{D}$ in a dataset $\mathcal{D}$, we generate $N = 8$ candidate trajectories $\mathbf{S} = \{S_1, ...S_8\}$ in parallel using a temperature $\tau = 0.8$ and $p = 1.0$. *These candidates are shared among all baseline methods*. An ablation study on how different methods scale with N is provided in §5.4. A maximal numbers of 6144 tokens are generated per each response (thereby $|S_i| \leq 6144$).

[2]Closed-source models such as GPT-4o-mini only provide probabilities for the top-20 logits. When applying to GPT-4o-mini, we only use these probabilities to compute self-certainty.

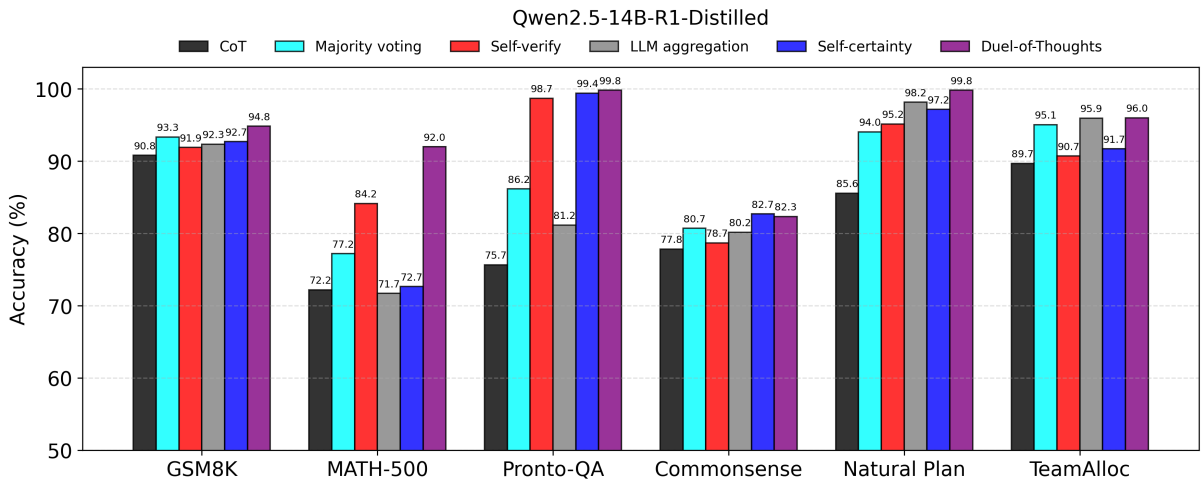


Figure 4: **Reasoning accuracy for Qwen2.5-14B-R1-Distilled**. Results are averaged over 5 independent runs.

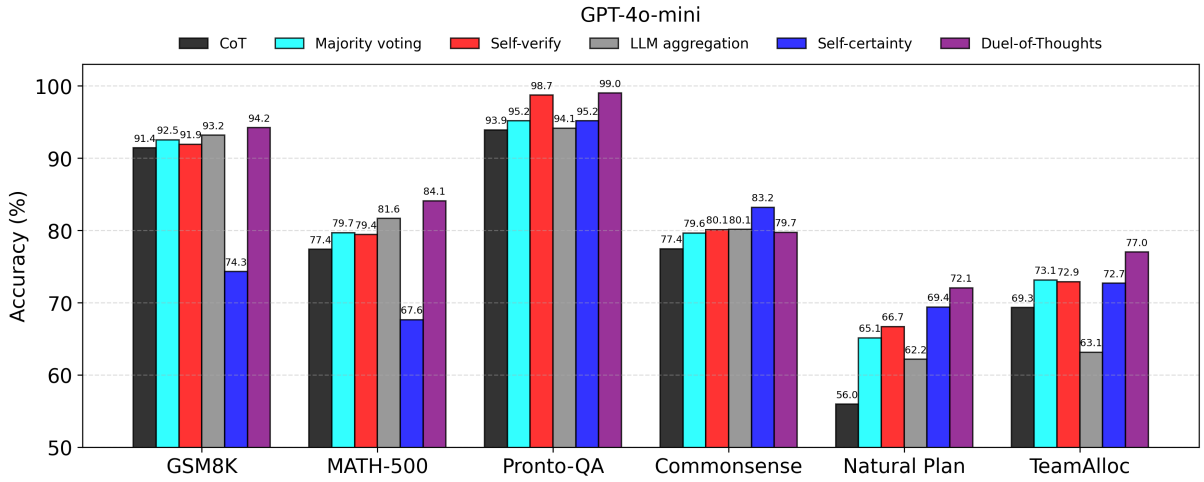


Figure 5: **Reasoning accuracy for GPT-4o-mini**. Results are averaged over 5 independent runs.

Models We evaluate our method using popular closed-source and open-source models:

- *Qwen2.5-14B-R1-Distilled*. An open-source model developed by DeepSeek (Guo et al., 2025).
- *GPT-4o-mini*. A closed-source, competent model developed by OpenAI (Hurst et al., 2024).

Results with GPT-4.1 are provided in the Appendix.

Tasks We extensively evaluate DoT across 6 different benchmarks spanning diverse domains. Table 1 provides a summary of the benchmarks, many of which do not have readily available RMs. Note that for MATH-500, a challenging subset of difficulty level ≥ 3 is chosen.

Unless specified otherwise, we mainly report the results of DoT under the basic configuration ($C = 2$), where only the top-2 clusters are considered.

5.2 Reasoning Accuracy

Overall, DoT consistently outperforms or matches the best baseline across tasks and models, being *either the champion or the runner-up method* across diverse settings, as reflected in Figures 4 and 5.

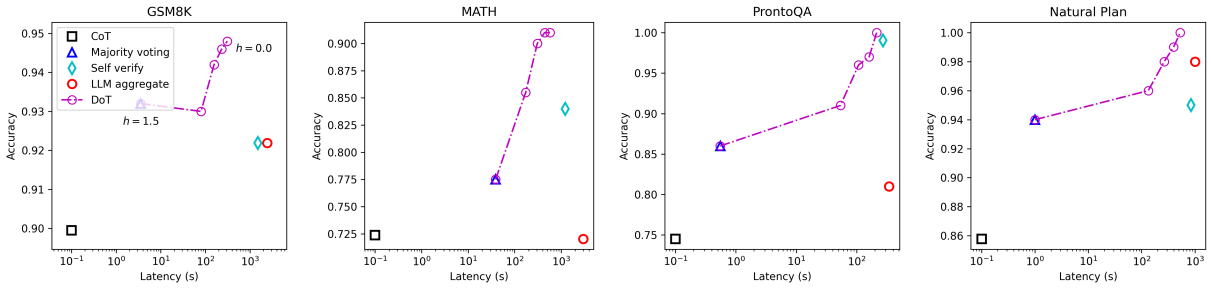
DoT vs. Majority voting DoT uniformly improves over majority voting across tasks. Table 2 analyzes this performance gap in details: while DoT and majority voting achieve identical accuracy in low-entropy regimes, DoT yields clear gains in high-entropy cases. In such settings, DoT's careful evaluation of alternative, less-frequent candidate avoids the risk of missing the correct minority.

DoT vs. Self-verification DoT also outperforms self-verification in most cases. This gap highlights that LLMs are more effective at detecting reasoning errors by *contrasting* two distinct, independently generated solutions than by revisiting a single rea-

Method	GSM8K		MATH-500		CommonsenseQA		ProntoQA		Natural Plan		TeamAlloc	
	Low $\mathcal{H}$	High $\mathcal{H}$	Low $\mathcal{H}$	High $\mathcal{H}$	Low $\mathcal{H}$	High $\mathcal{H}$	Low $\mathcal{H}$	High $\mathcal{H}$	Low $\mathcal{H}$	High $\mathcal{H}$	Low $\mathcal{H}$	High $\mathcal{H}$
Model: `Qwen2.5-14B-R1-Distilled`												
Proportion in data (%)	80.6	19.4	57.0	43.0	66.2	33.7	27.4	72.6	41.2	58.8	67.2	32.8
Majority vote Acc. (↑)	97.3	76.3	100.	47.7	92.4	55.9	100.	81.3	100.	90.0	97.0	90.9
DoT Acc. (↑)	97.3	84.5	100.	79.1	92.4	61.8	100.	100.	100.	99.2	97.0	91.2
Model: `GPT-4o-mini`												
Proportion in data (%)	93.4	6.60	76.5	23.5	90.5	9.50	96.2	3.80	31.2	68.8	70.4	29.6
Majority vote Acc. (↑)	95.1	60.6	96.7	25.5	80.7	68.4	96.9	33.3	96.8	50.7	80.0	56.7
DoT Acc. (↑)	95.1	68.4	96.7	38.3	80.7	73.7	96.9	100.	96.8	60.9	80.0	63.3

*Low $\mathcal{H}$ indicates cases where 75% of the solutions lead to the same final answer and High $\mathcal{H}$ indicates cases otherwise.

Table 2: **Thought entropy distribution** (for one typical run). The distribution of thought entropy $\mathcal{H}$ and how it can be used to save computational cost. Here, low $\mathcal{H}$ indicates cases where the thought entropy $\mathcal{H} \leq h$, where h corresponds to case where 75% of the answers agree. For most cases with low $\mathcal{H}$, taking the majority vote as the final answer is typically correct. Dedicated thought comparison is thereby only triggered in High $\mathcal{H}$ cases to reduce computation in DoT (eq.(8)). DoT improves over majority voting with more careful comparison in High $\mathcal{H}$ cases.



(a) GSM8K (b) MATH-500 (c) ProntoQA (d) Natural Plan

Figure 6: **Additional latency comparison** (the corresponding end-to-end token usage is reported in the appendix). Model: Qwen2.5-14B-R1-Distilled. We compare the additional computation time incurred by each method (excluding the time of initial candidates generation). In DoT, the trade-off between accuracy and latency is controlled by the threshold h, where thought dueling is triggered only when the thought entropy $\mathcal{H} > h$; see eq.(8).

soning trace, which allows the model to expose logical inconsistencies that are difficult to identify through self-reflection alone.

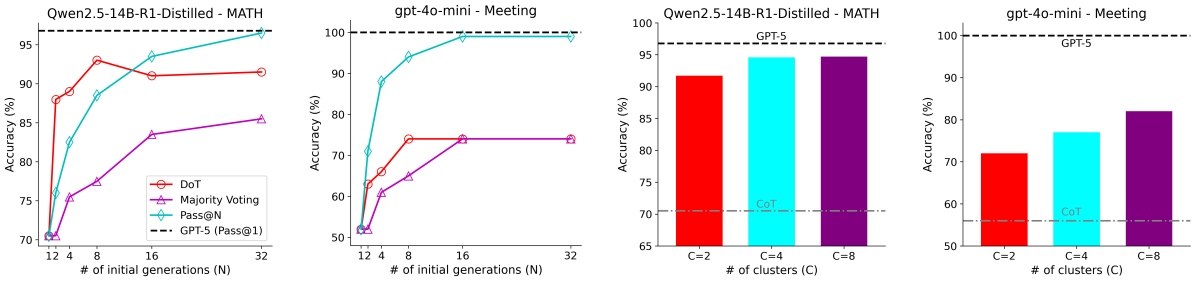
DoT vs. LLM aggregation The stronger performance of DoT over LLM aggregation suggest that aggregation is more effective when a small number of candidate solutions are presented, in line with recent studies which suggest that LLM struggle to effectively attend to and process long contexts. DoT decomposes aggregation into focused pairwise comparisons, effectively avoiding distraction and information dilution caused by long contexts.

DoT vs. Self-certainty Finally, we compare DoT with self-certainty, which leverages token probabilities in best-of-N selection. Despite requiring no token probabilities, DoT is on par with or even outperform self-certainty, particularly for GPT-4o-mini. The relatively poor performance of self-certainty on GPT-4o-mini may be due to the unavailability of full token probabilities for this model, which results in an inaccurate estimate of certainty score.

5.3 Effectiveness of Adaptive Thinking

We next demonstrate how adaptive thinking can dramatically reduce the computational cost of DoT while preserving its key advantage in accuracy.

Latency comparison Figure 6 compares the inference latency of DoT against alternative inference strategies. In DoT, the accuracy-latency trade-off is governed by the thought entropy threshold h (see eq.(8)). DoT consistently achieves a substantially better trade-off, achieving an order-of-magnitude reduction in computational cost at candidate selection/aggregation stage (see e.g. MATH-500 and Natural Plan) while attaining a comparable or even higher accuracy than alternative methods. We attribute this favorable trade-off to adaptive thinking, which triggers additional thought comparison only sparsely to avoid frequent second-round inference.



(a) Qwen, MATH-500 (b) 4o-mini, Natural Plan (c) Qwen, MATH-500 (d) 4o-mini, Natural Plan

Figure 7: **Scaling w.r.t the size of candidate pool N and the number of thought clusters C considered in DoT**. Figs (a)(b): DoT accuracy w.r.t N. Figs (c)(d): DoT accuracy w.r.t C. The accuracy of GPT-5 is shown as reference.

How does adaptive thinking actually help? Table 2 provides a more detailed analysis of how adaptive thinking reduces unnecessary inference calls. Here, we partition test instances into *low thought entropy* (Low $\mathcal{H}$), where solutions converge, and *high thought entropy* (High $\mathcal{H}$), where solutions diverge. We see that a substantial fraction of samples fall into the low-entropy regime (e.g., up to 66% on CommonsenseQA and 57% on MATH-500 for the case with Qwen-R1-Distilled). In these low entropy cases, majority voting already serves as a robust and reliable strategy, which achieves high reasoning accuracy. Thought comparison is thereby only selectively invoked for high-entropy instances, dramatically reducing unnecessary comparisons.

5.4 Scaling Studies

We examine how DoT scales with (a) the size of the candidate pool N, and (b) the number of top-C thought clusters considered in thought-dueling.

Scaling w.r.t the number of candidates N Figures 7(a,b) illustrate how DoT accuracy scales with the number of initial generations N. Across both tasks, DoT exhibits rapid performance gains as N increases, while remaining effective even for small values such as $N = 2$. Notably, DoT consistently outperforms majority voting at the same N, indicating that explicit thought comparison yields stronger benefits than simply increasing sample diversity. A sweet point is found at $N = 8$, with further increasing N only sees minor improvement in accuracy.

Scaling w.r.t the number of thought clusters C Figures 7(c,d) examine the effect of varying the number of top-C thought clusters considered in DoT. Overall, increasing C leads to monotonic improvements in accuracy, reflecting the advantage of comparing a richer set of competing solution hypotheses, albeit at the cost of more computation.

5.5 Results Summary

- Across models and benchmarks, DoT consistently achieves either the best performance or competitive runner-up accuracy compared to alternative best-of-N strategies.
- In settings where DoT performs similarly to strong baselines (e.g., LLM aggregation or self-certainty), it offers clear practical advantages, including (a) substantially lower computational cost and (b) applicability to black-box models.

6 Conclusion

Best-of-N selection is an important challenge in test-time scaling. Existing approaches often rely on external reward models or access to internal model states, which substantially limits their applicability. We remove these constraints by reformulating best-of-N selection as focused pairwise thought duels—comparisons that are substantially easier and can be handled by the model itself. To further improve computational efficiency, we introduce a human-inspired adaptive mechanism that selectively invokes deliberative comparison only under high uncertainty. Our method broadens the applicability of best-of-N methods, especially to *black-box models* and *domains without readily available reward models*. More broadly, this work suggests that when absolute scoring signals are absent, binary comparison can serve as a fundamental primitive for more reliable LLM decision making.

Beyond best-of-N selection, our findings also shed light on the hypothesized asymmetry between generation and verification. We show that, when provided with informed references, an LLM can more effectively detect reasoning errors in its own outputs. This observation suggest promising directions for comparison-driven inference paradigms and more reliable, self-reflective intelligent agents.

Limitations

While DoT is a powerful, black-box method applicable across diverse tasks and models, its effectiveness hinges on the following two key factors. *First*, similar to any other best-of-N selection methods, the final reasoning accuracy of DoT depends on the quality and diversity of the initial candidate pool $\mathbf{S} = \{S_1, \ldots, S_N\}$. If none of the candidates $S_i \in \mathbf{S}$ is correct, or if the diversity of $\mathbf{S}$ is limited (e.g., due to an overly small pool size N), DoT may fail to recover the correct answer. *Second*, DoT still relies on the LLM's ability to effectively compare two candidate thoughts S', S''. While this binary comparison task is substantially easier than the original generation task, it can still be challenging for models with limited capacity (e.g., models with <0.5B parameters). Exploring the effectiveness of DoT under such capacity constraints is an interesting direction for future work.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 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. *arXiv preprint arXiv:2407.21787*.

Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024a. Are more llm calls all you need? towards the scaling properties of compound ai systems. *Advances in Neural Information Processing Systems*, 37:45767–45790.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2024b. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*.

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

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.

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

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Krissanee Kamthawee, Can Udomcharoenchaikit, and Sarana Nutanong. 2024. Mist: mutual information maximization for short text clustering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11309–11324.

Zhewei Kang, Xuandong Zhao, and Dawn Song. 2025. Scalable best-of-n selection for large language models via self-certainty. In *Advances in neural information processing systems*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. Acemath: Advancing frontier math reasoning with post-training and reward modeling. In *Findings of the Association for*

Computational Linguistics: ACL 2025, pages 3993–4015.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. In *The Twelfth International Conference on Learning Representations*.

OpenAI. 2025. Gpt-5. https://openai.com. Large language model accessed via OpenAI API.

Y. Qing and 1 others. 2025. Livecodebench: Holistic and contamination-free evaluation of large language models for code. ArXiv preprint.

Jie Ren, Yao Zhao, Tu Vu, Peter J Liu, and Balaji Lakshminarayanan. 2023. Self-evaluation improves selective generation in large language models. *arXiv preprint arXiv:2312.09300*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, and 1 others. 2025. Toward expert-level medical question answering with large language models. *Nature Medicine*, 31(3):943–950.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. *International Conference in Learning Representation*.

Zayne Rea Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *The Twelfth International Conference on Learning Representations*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

LL Thurstone. 1927. A law of comparative judgment. *Psychological Review*, 34(4):273–286.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Xinglin Wang, Yiwei Li, Shaoxiong Feng, Peiwen Yuan, Yueqi Zhang, Jiayi Shi, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025. Every rollout counts: Optimal resource allocation for efficient test-time scaling. *Advances in Neural Information Processing Systems*.

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, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. 2025. The majority is not always right: Rl training for solution aggregation. *arXiv preprint arXiv:2509.06870*.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, and 1 others. 2024. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Appendices

A Prompts Templates

We detail the task-specific prompts t for generating the candidate reasoning trajectories $S_i \sim p_{\text{LLM}}(\cdot|q,t)$ as well as the prompt t_{final} used to generate the final trajectory S^*, from which we extract the final answer $a^* = \mathcal{A}(S^*)$.

A.1 Example prompts for rollouts

Below is the example prompt for mathematical reasoning tasks:

Generation Prompt for Math Reasoning

```
You are a helpful AI assistant that will help solving math problems. Below you are given a math problem.

Solve it step-by-step, showing your reasoning clearly. Provide your final answer as \\boxed{your answer}.
```

Below is the example prompt for the Natural Planning task:

Generation Prompt for Natural Planning

```
You are an expert at scheduling meetings. You are given a few constraints on the existing schedule of each participant, the meeting duration, and possibly some preferences on the meeting time. Note there exists a solution that works with existing schedule of every participant.

Show you reasoning step by step. Provide your final answer (scheduled date and time) as \\boxed{your answer}, where 'your answer' must be something like 'Monday, 14:00-15:05'
```

Below is the example prompt for the commonsense reasoning task:

Generation Prompt for Commonsense Reasoning

```
You are a helpful AI assistant that will answer commonsense questions. You will be given several choices.

Think carefully before answering the final choice, showing your reasoning step-by-step. You must present your final response with " boxedyour answer", where 'your answer' must be the letter of the option. If there is no correct answer, give a random answer.
```

A.2 Example prompts for BoN methods

Below is the example prompt used in the proposed Duel-of-Thoughts (DoT) method.

Prompt Template for DoT

```
For the question above, there already exist two solutions for your reference. Your task is to carefully compare these two solutions, analyze their potential mistakes by contrasting them step-by-steps.
Choose the best solution between these two solutions according to your analysis.

Below are the two reference solutions:
---
[Solution #1] ... \boxed{ANSWER 1}
[Solution #2] ... \boxed{ANSWER 2}
---
Compare the two solutions above carefully. Tell me why one solution is more preferable than the other. Show me your reasoning.
Return the final answer as \boxed{ANSWER}.
```

Below is the example prompt used in the self-verification baseline.

Prompt Template for Self-verification

```
Now there is already an answer:
---
[Solution] ... \boxed{ANSWER}
---
Your task is to check carefully whether this solution is correct. If the original answer is correct, return the final answer in \\boxed{original final answer}. If the original answer is incorrect, correct the mistakes, and return your own answer, again in the format \\boxed{your own answer}.
```

Below is the example prompt used in the LLM aggregation baseline.

Prompt Template for LLM aggregation

```
For the provided question, there are already multiple solutions:
---
[Solution #1] ... \boxed{ANSWER 1}
[Solution #2] ... \boxed{ANSWER 2}
...
[Solution #N] ... \boxed{ANSWER N}
---
Please compare the solutions above and assign a score between 0 and 10 to them. Choose the best solution among them according to the scores. Check and verify the correctness of each solutions carefully.
Conclude your final choice as \\boxed{Solution #i}, where "i" is the index of the best solution.
```

Q: The binary number 10101001110_{2} is equal to what number in base eight?

A (DoT): Okay, so I have this problem where I need to convert the binary number 10101001110_2 to base eight. **I've been given two solutions, A and B**, and I need to figure out which one is correct. Let me go through each solution step by step. **Starting with Solution #1**. The person explains that converting binary to octal involves grouping the binary digits into sets of three from the right. That makes sense because 8 is 2^3, so each octal digit corresponds to three binary digits. They took the binary number 10101001110 and added two leading zeros to make it 001 010 100 1110. Wait, hold on, that doesn't seem right. If you group from the right, the original number is 10101001110. Let me count the digits: 1 0 1 0 1 0 0 1 1 1 0. That's 11 digits. To make groups of three, starting from the right, the first group will have two digits, and the rest will have three each. So, actually, you should pad with one zero on the left to make it 12 digits, which can be divided into four groups of three. But in Solution #1, they added two zeros, making it 001 010 100 1110. Wait, that's 13 digits? No, 001 010 100 1110 is 13 digits? Let me check: 001 is three, 010 is three, 100 is three, and 1110 is four. That doesn't add up. Wait, maybe I'm miscounting. The original number is 11 digits. Adding two zeros makes it 13 digits, which isn't a multiple of three. Hmm, that seems off. Maybe they made a mistake in padding. Let me think. If the original number is 11 digits, to make it a multiple of three, we need to add two zeros to the left, making it 13 digits, but 13 isn't a multiple of three. Wait, 11 divided by 3 is 3 with a remainder of 2, so we need to add one zero to make it 12 digits. So, the correct padding would be adding one zero to the left, making it 010101001110. Then, grouping into three from the right: 01 010 100 1110. Wait, no, that's still not right. Let me try again. The original number is 10101001110. Let's write it out: 1 0 1 0 1 0 0 1 1 1 0 Starting from the right, the first group is the last three digits: 110. Then moving left, the next three: 001. Then the next three: 101. Then the remaining two digits: 01. So, to make it groups of three, we need to pad the left with one zero, making it 010101001110. Then grouping as 010 101 001 110. Each group is three digits. Now converting each group: 010_2 is 2_8, 101_2 is 5_8, 001_2 is 1_8, and 110_2 is 6_8. So combining them, it's 2516_8. Wait, that's different from what Solution #1 says. They got 1246_8, but according to my calculation, it's 2516_8. So maybe Solution #1 made a mistake in padding or grouping. Looking back at Solution #1, they wrote the binary as 001 010 100 1110. That seems incorrect because 1110 is four digits, which shouldn't happen. Each group should be three digits. So they probably messed up the padding. Instead of adding two zeros, they should have added one zero to make it 12 digits, which can be grouped into four sets of three. So their grouping was wrong, leading to an incorrect octal number. **Now, looking at Solution #2**. They converted the binary to decimal first, which is 1358_{10}, and then converted that to octal. Let's verify their steps. The binary number is 10101001110. Let's convert that to decimal: Starting from the right, the positions are 0 to 10. So: Position 10: $1 \times 2^{10} = 1024$ Position 9: $0 \times 2^9 = 0$ Position 8: $1 \times 2^8 = 256$ Position 7: $0 \times 2^7 = 0$ Position 6: $1 \times 2^6 = 64$ Position 5: $0 \times 2^5 = 0$ Position 4: $0 \times 2^4 = 0$ Position 3: $1 \times 2^3 = 8$ Position 2: $1 \times 2^2 = 4$ Position 1: $1 \times 2^1 = 2$ Position 0: $0 \times 2^0 = 0$ Adding them up: 1024 + 256 + 64 + 8 + 4 + 2 = 1358. That's correct. Then converting 1358 to octal: 1358 ÷ 8 = 169 with remainder 6 169 ÷ 8 = 21 with remainder 1 21 ÷ 8 = 2 with remainder 5 2 ÷ 8 = 0 with remainder 2 Reading the remainders from last to first: 2 5 1 6, so 2516_8. That's the same as my correct grouping method. So Solution #2 is correct. **But wait, in Solution #1, they got 1246_8, which is different. So why is there a discrepancy? Because Solution #1 grouped the binary incorrectly**. They added two zeros instead of one, leading to an incorrect grouping, which resulted in the wrong octal number. Therefore, Solution #2 is correct, and Solution #1 made a mistake in padding and grouping the binary digits. **So the correct octal number is 2516_8.**

GT: = 2516_8 | **Solution #1 = majority voting solution** | **Solution #2 = alternative solution**

Figure 8: **Case study: DoT vs. majority voting**. Model: Qwen-2.5-14B-R1-Distilled. In this example, majority voting selects an incorrect reasoning chain (Solution #1). By explicitly comparing the majority-voted solution with an alternative candidate (Solution #2), DoT identifies the reasoning error and correctly selects Solution #2.

Q: You need to schedule a meeting for Ronald, Teresa and Carol for one hour between the work hours of 9:00 to 17:00 on Monday. Here are the existing schedules for everyone during the day: Ronald is busy on Monday during 9:00 to 10:00, 11:00 to 12:00; Teresa is busy on Monday during 10:30 to 11:00, 14:00 to 14:30; Carol has blocked their calendar on Monday during 9:00 to 12:30, 14:00 to 15:30, 16:00 to 17:00; Find a time that works for everyone's schedule and constraints.

A (DoT): Let's evaluate both solutions against the schedules and constraints of Ronald, Teresa, and Carol. ### Schedules Overview: - **Ronald**: Busy from 9:00 to 10:00 and 11:00 to 12:00 - **Teresa**: Busy from 10:30 to 11:00 and 14:00 to 14:30 - **Carol**: Busy from 9:00 to 12:30, 14:00 to 15:30, and 16:00 to 17:00

Solution A: **Monday, 12:30-13:30** - **Ronald**: Available (free after 12:00) - **Teresa**: Available (free after 11:00) - **Carol**: Available (free after 12:30) **Validation**: - All three participants are available during this time. So, Solution A works.

Solution B: **Monday, 15:30-16:30** - **Ronald**: Available (free after 12:00) - **Teresa**: Available (free after 14:30) - **Carol**: Busy (blocked until 15:30) **Validation**: - Carol is busy until 15:30. This means that she is not available until 15:30, but since the meeting ends at 16:30, it overlaps with her busy time.

Conclusion Solution A is valid as all participants are available during that time, while Solution B is invalid due to Carol's conflicting schedule. Thus, the better solution is: \[\boxed{**Monday, 12:30-13:30**} \]

GT: Monday, 12:30-13:30

A (LLM aggregate): To evaluate the provided solutions, we need to analyze the availability of each participant against the proposed meeting times: - **Ronald**: Busy from 9:00 to 10:00 and 11:00 to 12:00 - **Teresa**: Busy from 10:30 to 11:00 and 14:00 to 14:30 - **Carol**: Busy from 9:00 to 12:30, 14:00 to 15:30, and 16:00 to 17:00

Now let's evaluate each solution:
1. **Solution #1: Monday, 15:30-16:30** - **Ronald:** Free - **Teresa:** Free - **Carol:** Free - **Score:** 10
2. **Solution #2: Monday, 12:30-13:30** - **Ronald:** Free - **Teresa:** Free - **Carol:** Busy - **Score:** 0 (not all are available)
3. **Solution #3: Monday, 12:30-13:30** - **Ronald:** Free - **Teresa:** Free - **Carol:** Busy - **Score:** 0 (not all are available)
4. **Solution #4: Monday, 12:30-13:30** - **Ronald:** Free - **Teresa:** Free - **Carol:** Busy - **Score:** 0 (not all are available)
5. **Solution #5: Monday, 12:00-13:00** - **Ronald:** Busy - **Teresa:** Free - **Carol:** Busy - **Score:** 0 (not all are available)
6. **Solution #6: Monday, 12:30-13:30** - **Ronald:** Free - **Teresa:** Free - **Carol:** Busy - **Score:** 0 (not all are available)
7. **Solution #7: Monday, 12:30-13:30** - **Ronald:** Free - **Teresa:** Free - **Carol:** Busy - **Score:** 0 (not all are available)
8. **Solution #8: Monday, 12:30-13:30** - **Ronald:** Free - **Teresa:** Free - **Carol:** Busy - **Score:** 0 (not all are available)

Summary of Scores: - Solution #1: 10 - Solutions #2, #3, #4, #5, #6, #7, #8: 0 each The best solution is \(\boxed{Solution \#1}\) i.e. \boxed{**Monday, 15:30-16:30**}

GT: Monday, 12:30-13:30

Figure 9: **Case study: DoT vs. LLM-based aggregation**. Model: GPT-4o-mini. In this example, directly using an LLM to aggregate candidate solutions overlooks fine-grained time constraints and select the wrong solution (Solution #1), likely due to information dilution from an overly long context. In contrast, DoT operates on a reduced and focused comparison context, enabling it to correctly identify the reasoning error and select the correct solution.

B Case studies

In Figure 8 and Figure 9, we provide case studies to compare DoT and two alternative best-of-N strategies: majority voting and LLM aggregation.

C Usage of LLMs

We used LLMs solely as a writing assistant for grammar checking. None of the research ideation, experimental design, or analysis relied on LLMs.

	MATH-500	CSQA	Natural Plan
CoT	2002	699	3446
DoT	+831	+75	+1049
Majority voting	+0	+0	+0
Self-verify	+1435	+366	+1742
LLM aggregation	+1791	+548	+1554

(a) **Qwen-R1-distill**

	MATH-500	CSQA	Natural Plan
CoT	751	221	971
DoT	+139	+45	+380
Majority voting	+0	+0	+0
Self-verify	+581	+202	+719
LLM aggregation	+596	+482	+678

(b) **GPT-4o-mini**

Table 3: Average additional token usage of different methods (denoted as '+') compared to initial rollouts ('CoT').

	$C = 2$	$C = 4$	$C = 8$	CoT
MATH-500	+831	+1632	+2472	2002×8
Natural Plan	+1049	+2237	+3120	3446×8

(a) **Qwen-R1-Distill**

	$C = 2$	$C = 4$	$C = 8$	CoT
MATH-500	+139	+212	+788	751×8
Natural Plan	+380	+512	+743	921×8

(b) **GPT-4o-mini**

Table 4: Additional average token usage of DoT under thought cluster number C (with CoT shown as reference).

	$t = 0.3$	$t = 0.5$	$t = 0.8$
CoT	80.9	79.4	72.1
DoT	**90.2**	**93.1**	**92.0**
Majority voting	86.8	88.5	77.2

(a) **Qwen-R1-Distill, MATH-500**

	$t = 0.3$	$t = 0.5$	$t = 0.8$
CoT	59.5	55.6	56.2
DoT	65.0	**71.4**	**72.1**
Majority voting	**65.2**	66.2	65.2

(b) **GPT-4o-mini, Natural Plan**

Table 5: Temperature sensitivity under different decoding temperatures t. Digits shown correspond to accuracy.

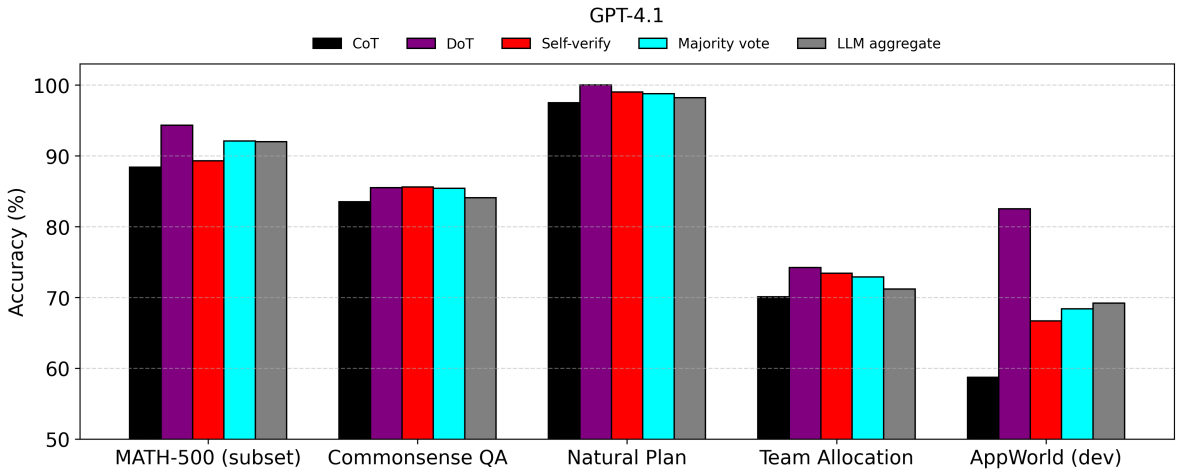


Figure 10: **Reasoning accuracy for GPT-4.1**. Results are averaged over 5 independent runs.

D Further empirical results

Full token usage report We report the actual token usage in Table 3. Overall, DoT significantly reduces additional token usage compared to alternative inference-time scaling strategies such as self-verification or LLM aggregation, introducing only small-to-moderate additional token usages.

In Table 4, we further report the token usages of DoT under different number of thought clusters C.

Sensitivity w.r.t temperature t We investigate how DoT is sensitive to different decoding temperature t. The results in Table 5 demonstrate that DoT remains highly robust across various temperatures.

Further evaluation on GPT-4.1 We further evaluate DoT on the more recent proprietary model GPT-4.1; see Figure 10. The results demonstrate that DoT remains highly effective for this model, with notable improvement on the challenging AppWorld task. where alternative strategies struggle.