Entropy-based Exploration Conduction for Multi-step Reasoning

Anonymous ACL submission

Abstract

In large language model (LLM) reasoning, multi-step processes have proven effective for solving complex tasks. However, the depth of exploration can significantly affect the reasoning performance. Existing methods to automatically decide the depth often bring high cost and lack flexibility, and thus undermine the model's reasoning accuracy. To address these issues, we propose **Entro**py-based Exploration Depth Con**duction** (**Entro-duction**), a novel method that dynamically adjusts the exploration depth during multi-step reasoning by monitoring LLM's output entropy and variance entropy. We employ these two metrics to capture the model's current uncertainty and the fluctuation of uncertainty across consecutive reasoning steps. Based on the observed changes, the LLM selects whether to deepen, expand or stop exploration according to the probability. In this way, we balance the reasoning accuracy and exploration effectiveness. Experimental results across four benchmark datasets demonstrate the efficacy of Entro-duction. We further conduct experiments and analysis on the components of Entro-duction to discuss their contributions to reasoning performance.

1 Introduction

Large language models (LLMs) have demonstrated remarkable reasoning capabilities across various domains (Brown et al., 2020; Touvron et al., 2023; Patterson et al., 2022; Fu et al., 2022; Wei et al., 2022). Meanwhile, facing complex tasks, they encounter challenges in generating accurate and effective multi-step reasoning. Specifically, LLMs may exhibit over-reasoning or under-reasoning, which both indicate the depth of exploration for a given problem does not match expectations (Ahn et al., 2024; Mirzadeh et al., 2024; Huang and Chang, 2022; Fu et al., 2023). This mismatch could lead to several issues: (1) inaccurate, insufficient, or redundant reasoning outcomes; (2) unnecessary computation costs (Yeo et al., 2025; Yang et al., 2024; Lightman et al., 2023). These issues exist because i) LLMs lack evaluation and regulatory mechanisms for their reasoning process; ii) there are significant variations in the reasoning process required for different tasks, and LLMs are unable to accurately judge and adjust the depth of exploration for a task solely based on their training knowledge.

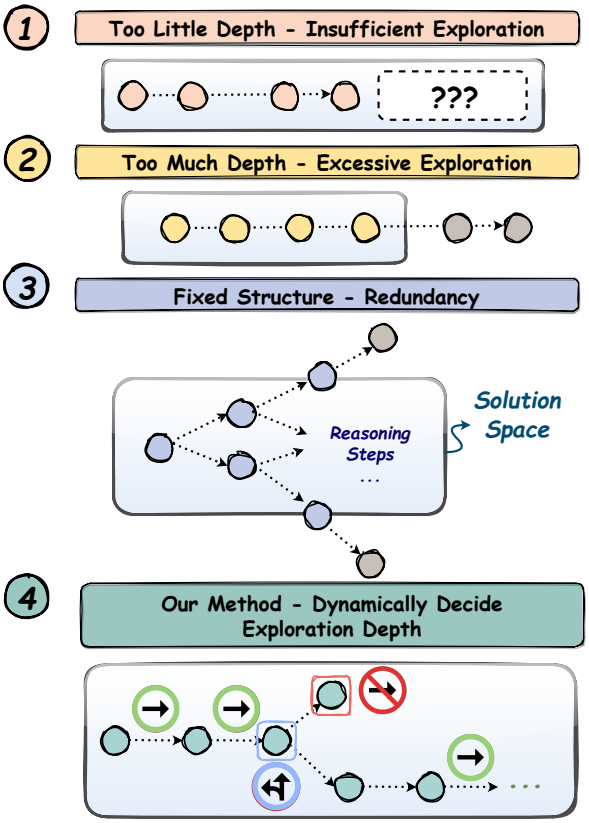

Figure 1: Reasoning depth mismatching solution space.

Existing methods for optimizing the exploration depth of multi-step reasoning in LLMs fall into two types: outcome-based optimization and process-based optimization. Outcome-based optimization aligns the LLM's reasoning exploration with human expectation after it has generated complete

reasoning and reached conclusions (Jin et al., 2024; Liu et al., 2024a; Ton et al., 2024; Yu et al., 2024). These approaches rely on post-training of LLMs, which is costly and does not provide an immediate improvement on the current task. Further, due to the diversity of reasoning tasks, the enhancement gain is task-specific. In process-based optimization, the LLMs supervise and evaluate their own reasoning process through self-critique or by labeling reasoning steps to enhance outputs (Ma et al., 2025; Yang et al., 2024; Pan et al., 2023). The advantages of process-based optimization are the immediacy of optimization and its low cost. However, it requires the LLMs to have high reasoning capabilities and a substantial knowledge base. Moreover, since the process is usually opaque, it is difficult for humans to effectively supervise and interpret these optimization processes. And the model's own illusions, biases, and errors may be reinforced during this process (Stechly et al., 2023, 2024b; Liang et al., 2024b; Song et al., 2025).

Our Target. Given the existing challenges, we want to develop a method that guides LLMs to automatically, concisely, and transparently determine the appropriate depth of exploration based on task information and the model's reasoning state. We aim to enable the model to look ahead during reasoning, plan dynamically, and decide whether further exploration is necessary to achieve optimal reasoning outcomes. Our target involves enhancing multi-step reasoning performance and reducing unnecessary exploration.

Our Method. To tackle these challenges, we propose **Entro**py-based Exploration Depth Con**duction** (**Entro-duction**), a novel method to help LLM assess the adequacy of exploration during multi-step reasoning processes, thus enhancing reasoning outcomes. Inspired by Entropy Uncertainty Measurement, we employ entropy changes in the LLM's reasoning process to evaluate the model's uncertainty of reasoning, which reflects the LLM's reasoning confidence, and accordingly switch exploration strategy. Specifically, we use entropy and variance entropy as rewards to update the LLM's probability distribution for its next exploratory action, whether to deepen, stop, or expand exploration. This distribution then drives a behavior selection mechanism that enhances reasoning when exploration is insufficient and avoids redundant reasoning when it is sufficient.

In summary, our contributions involve:

1. We propose Entro-duction to help LLMs dynamically evaluate the adequacy of exploration based on LLMs' reasoning uncertainty to enhance reasoning performance and avoid unnecessary exploration.

2. We further design an entropy-based exploration behavior selection mechanism, which refers to LLMs' expectancy and confidence in reasoning.

3. We conduct a series of experiments to demonstrate the effectiveness of Entro-duction on various reasoning tasks. Our results and analysis show that the Entro-duction outperforms baseline methods.

2 Related Work

Reasoning Steps and Structures. When responding to queries, LLMs typically provide direct outputs. For complex questions, direct outputs often fail to deliver correct answers because they may overlook deeper logical connections or contextual information (Xia et al., 2024; Minaee et al., 2024). Multi-step reasoning involves instructing LLMs to decompose and progressively address problems, breaking down complex tasks into smaller, manageable units to significantly enhance reasoning capabilities (Chu et al., 2023). The simplest structure of multi-step reasoning is the Chain of Thought (CoT) (Wei et al., 2022; Wang and Zhou, 2024), which links reasoning steps by connecting distinct thoughts into a linear, coherent sequence. (Li et al., 2024; Jin and Lu, 2024; Sprague et al., 2024).

To enable more comprehensive exploratory reasoning, some studies based on CoT have developed structured reasoning methods, such as self-consistent CoT (CoT-SC), Complex CoT, and Tree-of-Thought (ToT) (Wang et al., 2022; Zhang et al., 2024b; Yao et al., 2024; Liu et al., 2024b; Mo and Xin, 2024; Zhang et al., 2024a). These methods are called reasoning structures. They guide LLMs to do multi-directional exploration of problem solution spaces for superior reasoning solutions by capturing more complex and varied logical relationships (Xia et al., 2024; Stechly et al., 2024a; Liang et al., 2024a). However, the breadth and depth of these reasoning structures highly depend on predefined settings and vary greatly across different tasks, limiting their generalizability and flexibility.

Optimization of Reasoning Depth. The depth of reasoning structures refers to the number of lay-

ers or steps in the reasoning process, namely, the number of reasoning steps an LLM must undertake before reaching a final answer (Plaat et al., 2024; Gomez, 2023). For any given task, the optimal number of reasoning layers often correlates with the task's complexity and the level of detail required (Zhang et al., 2024b). Current methods in determining these layers or optimize reasoning structures automatically. These methods include using reinforcement learning algorithms to optimize the number of reasoning steps or dynamically adjusting the depth of exploration during the reasoning process(Jin et al., 2024; Liu et al., 2024a; Hoffmann et al., 2022).

The main issues of these methods include: (1) the automated algorithms may lack precision due to the lack of precision; (2) making dynamic adjustments without fully understanding task characteristics could harm the reasoning process; (3) for highly complex or novel tasks, preset reasoning structures may be inappropriate, and could limit the model's applicability and flexibility. These issues together ultimately affect the LLM's reasoning reliability and efficiency, and lead to inaccurate reasoning outcomes or redundant exploration.

3 Methodology

3.1 Problem Formulation

Given a reasoning task Q and an LLM $\mathcal{R}$ as the reasoner, the multi-step reasoning process is to generate a reasoning structure $\mathcal{S}$. Structure $\mathcal{S}$ comprises directed links connecting sentence-level reasoning nodes. A reasoning chain is a unidirectional sequence that begins at an initial reasoning node and concludes at a terminal node:

$$\mathcal{C}_i = \mathcal{T}_{i1} \rightarrow \mathcal{T}_{i2} \rightarrow \cdots \rightarrow \mathcal{T}_{i,j} \rightarrow \cdots, i = 1, ..., m, \tag{1}$$

where $\mathcal{T}_{i,j}$ is the j-th node in the i-th chain,m is the total number of chains in $\mathcal{S}$. The chain length $|\mathcal{C}_i|$ is the total number of nodes within $\mathcal{C}_i$. We define the depth $\mathcal{S}_d$ of $\mathcal{S}$ as the maximum length of any reasoning chain within the structure:

$$\mathcal{S}_d = \max\{|\mathcal{C}_i|\}. \tag{2}$$

Since $\mathcal{S}$ could contain several valid reasoning chains, the reasoning conclusion L is made through a voting mechanism V:

$$L = V(\mathcal{S}, \mathcal{R}). \tag{3}$$

In this paper, our goal is to generate a structure $\mathcal{S}$ such that it achieves optimal reasoning accuracy, denoted as $\text{Acc}(L)$, with an optimal depth $\mathcal{S}_d$.

3.2 Reasoning State Evaluation

The essence of the multi-step reasoning process is to explore the solution space of task $\mathcal{Q}$. Our exploration goal is to cover as many potential reasoning paths as possible to ensure the accuracy and completeness of the solution. However, exhaustive exploration is inefficient and often impractical. As the problem's complexity grows, the solution space expands exponentially, driving the computational and time costs to untenable levels. Consequently, we must balance the breadth and depth of exploration. Achieving this balance calls for a method that can look forward from each current reasoning state, predicting and adjusting subsequent exploration steps, so that we do not miss crucial paths or waste resources on unnecessary ones.

We employ uncertainty and stability to describe the reasoning state. Uncertainty measures the divergence of current thought processes, as shown in Figure 2. In a reasoning step, a high uncertainty indicates the presence of multiple possible directions or conclusions, this means a wide scope of exploration is necessary. Conversely, low uncertainty, where there are few or even a single possible outcome, indicates a more focused path. Specifically, we employ entropy as a metric for uncertainty to quantify the number of potential paths that need exploration at any given moment and to gauge the confidence level in the conclusions.

Definition 1: Entropy. Consider a reasoning step represented by a sentence, denoted as $\mathcal{T}_{i,j}$, which consists of a sequence of n tokens:

$$\mathcal{T}_{i,j} = \{t_{ij1}, t_{ij2}, \ldots, t_{ijn}\}. \tag{4}$$

Each token t_{ijk} matches a logit l_{ijk}, which is the model's raw output before the softmax function. The collection of logits for the entire sentence is:

$$l_{ij} = \{l_{ij1}, l_{ij2}, \ldots, l_{ijn}\}. \tag{5}$$

We calculate the probability p_{ijk} of each token t_{ijk} by applying the softmax function to its corresponding logit l_{ijk}:

$$p_{ijk} = \frac{\exp(l_{ijk})}{\sum_{r=1}^{n} \exp(l_{ijr})}. \tag{6}$$

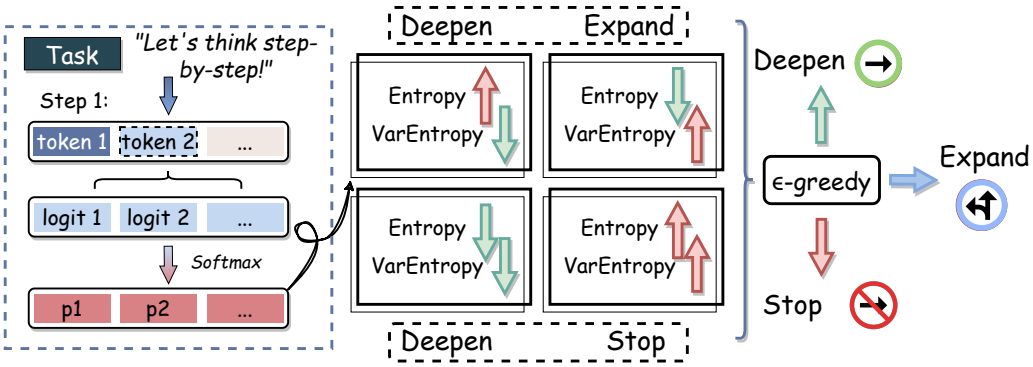

Figure 2: Framework of Entro-duction. We obtain two metrics, entropy and variance entropy, by calculating the probabilities of the logits at each reasoning step. Subsequently, we employ the $epsilon$-greedy method to select the appropriate exploration behavior based on changes in both metrics.

Then the entropy of the sentence $\mathcal{T}_{i,j}$ is:

$$H(\mathcal{T}_{i,j}) = -\sum_{k=1}^{n} p_{ijk} \log_2(p_{ijk}). \tag{7}$$

This measures the uncertainty or information content encoded in the probability distribution $\{p_{ij1}, p_{ij2}, \ldots, p_{ijn}\}$.

To compare the entropies across reasoning steps of varying lengths, we define the normalized entropy as:

$$\widetilde{H}(\mathcal{T}_{i,j}) = \frac{H(\mathcal{T}_{i,j})}{\log_2(n)}. \tag{8}$$

Here, $\log_2(n)$ is the maximum possible entropy when all n tokens have uniform probability. Hence, the normalized entropy is bounded between 0 and 1 for consistent comparisons.

Similarly, we employ variance entropy to capture how much uncertainty fluctuates across consecutive reasoning steps. It indicates the consistency or divergence of the thought process.

Definition 2: Variance Entropy. For reasoning step $\mathcal{T}_{i,j}$ of length n, let:

$$\overline{H}(\mathcal{T}_{i,j}) = \frac{1}{n} \sum_{k=1}^{n} H(t_{ijk}), \tag{9}$$

be the average token-level entropy in $\mathcal{T}_{i,j}$. We define the variance entropy as:

$$\sigma_H^2(\mathcal{T}_{i,j}) = \frac{1}{n} \sum_{k=1}^{n} \left[H(t_{ijk}) - \overline{H}(\mathcal{T}_{i,j})\right]^2. \tag{10}$$

For comparisons, we define the normalized variance entropy:

$$\widetilde{\sigma_H^2}(\mathcal{T}_{i,j}) = \frac{\sigma_H^2(\mathcal{T}_{i,j})}{\log_2(n)}. \tag{11}$$

In this way, we have a normalized concise metric for tracking fluctuations in uncertainty within each reasoning step.

3.3 Exploration Behaviors

With two metrics for reasoning state defined above, we further consider how to use changes in these metrics to determine exploration behavior strategies. The possible scenarios are listed below:

1. ***Entropy*** ↓**,** ***Variance Entropy*** ↓**:** The reasoning step becomes more certain, and the overall thought process more coherent. This indicates that information is becoming more focused, and the reasoning process is stable and effective. The LLM should continue to explore in this direction.

2. ***Entropy*** ↑ **,** ***Variance Entropy*** ↓**:** The reasoning step introduces more uncertainty, but the fluctuations between different steps are decreasing. This suggests that while a broader range of possibilities has emerged, the overall direction has not become dispersed. The LLM should continue to explore in this direction.

3. ***Entropy*** ↓ **,** ***Variance Entropy*** ↑**:** The uncertainty of reasoning is decreasing, but the fluctuation between reasoning steps is increasing. This indicates potential divergences in local steps, and we should consider increasing exploration in different directions to cover possible solutions.

4. ***Entropy*** ↑ **,** ***Variance Entropy*** ↑**:** The reasoning process becomes simultaneously more complex and more unstable. This indicates that current exploration might have strategic deviations, but another possibility is that the

model is exploring a new or more challenging direction. We need to consider avoiding ineffective exploration while maintaining the potential for the model to tackle challenging problems.

Accordingly, we design a mechanism that adjusts the probability of exploration behaviors based on entropy and variance entropy changes. We define the exploration behaviors as follows:

Deepen: This behavior extends the current reasoning chain $\mathcal{C}_i$ by adding a new node $\mathcal{T}_{i,j+1}$:

$$\mathcal{C}_i \to \mathcal{C}_i \cup \{\mathcal{T}_{i,j+1}\}. \tag{12}$$

Expand: This behavior divides the current reasoning chain at $\mathcal{T}_{i,j}$, creates two separate chains $\mathcal{C}_i$ and $\mathcal{C'}_i$. Each chain extending from the split point generates a new node:

$$\mathcal{C}_i \to (\mathcal{C}_i \cup \{\mathcal{T}_{i,j+1}\}), \mathcal{C'}_i \to \left(\mathcal{C}_i \cup \{\mathcal{T'}_{i,j+1}\}\right). \tag{13}$$

Stop: This behavior terminates the extension of the current chain $\mathcal{C}_i$ at the current node $\mathcal{T}_{i,j}$:

$$\mathcal{C}_i \to \mathcal{C}_i \setminus \{\mathcal{T}_{i,j+1}\}. \tag{14}$$

3.4 Behavior Selection Mechanism

At the j-th reasoning step, we define:

$$\Delta H_j = H(\mathcal{T}_{j+1}) - H(\mathcal{T}_j), \tag{15}$$

$$\Delta\sigma^2_{H,j} = \sigma^2_H(\mathcal{T}_{j+1}) - \sigma^2_H(\mathcal{T}_j), \tag{16}$$

which denotes the changes in entropy and variance entropy, respectively. We define the state:

$$\mathbf{s}_j = \left(\Delta H_j, \Delta\sigma^2_{H,j}\right), \tag{17}$$

and the set of possible actions as

$$\mathcal{A} = \{\text{Deepen}, \text{Expand}, \text{Stop}\}. \tag{18}$$

We introduce a mapping $\Phi : \mathbb{R}^2 \to \mathcal{A}$ that assigns to each state $(\Delta H_j,\ \Delta\sigma^2_{H,j})$ a "best" action a^*_j:

$$\Phi(\Delta H, \Delta\sigma^2_H) = \begin{cases} \text{Deepen}, & \text{if } (\Delta H < 0, \Delta\sigma^2_H < 0) \\ & \text{or } (\Delta H > 0, \Delta\sigma^2_H < 0), \\ \text{Expand}, & \text{if } \Delta H < 0, \Delta\sigma^2_H > 0 \\ \text{Stop}, & \text{if } \Delta H > 0, \Delta\sigma^2_H > 0 \end{cases} \tag{19}$$

Algorithm 1 Entro-duction

Input: Reasoning task $\mathcal{Q}$; LLM reasoner $\mathcal{R}$; max steps J; exploration rate ϵ.
Output: Reasoning structure $\mathcal{S}$ and conclusion L.
Initialize $\mathcal{S}$; set $j \leftarrow 1$; initialize chains $\{\mathcal{C}_i\}$.
while $j \leq J$ **do**
  for each active chain $\mathcal{C}_i$ **do**
    Compute $H(T_j)$ and $\sigma^2_H(\mathcal{T}_j)$ according to Eqs. 7, 10.
    Compute ΔH_j and $\Delta\sigma^2_{H,j}$ according to Eqs. 15, 16).
    Determine $a^*_j \leftarrow \Phi(\Delta H_j, \Delta\sigma^2_{H,j})$ according to Eq. 19.
    Sample action a_j with probability $\pi_j(a \mid s_j)$ according to Eq. 20).
    Execute a_j:
      i) **Deepen**: append $\mathcal{T}_{j+1}$ to $\mathcal{C}_i$.
      ii) **Expand**: branch $\mathcal{C}_i$ into two chains with $\mathcal{T}_{j+1}$ and $\mathcal{T}'_{j+1}$.
      iii) **Stop**: finalize $\mathcal{C}_i$ (no further expansion).
  end for
  if all chains stopped or $j = J$ **then**
    break
  end if
  $j \leftarrow j + 1$
end while
$L \leftarrow V(\mathcal{S}, \mathcal{R})$ (final conclusion via consensus).
return $\mathcal{S},\ L$

Then, at each step j, we sample the actual action a_j according to an ϵ-greedy rule:

$$\pi_j\big(a \mid \mathbf{s}_j\big) = \begin{cases} 1-\epsilon, & a = a^*_j, \\ \dfrac{\epsilon}{|\mathcal{A}| - 1}, & a \neq a^*_j. \end{cases} \tag{20}$$

Given the current state $\mathbf{s}_j = (\Delta H_j, \Delta\sigma^2_{H,j})$, we first compute $a^*_j = \Phi(\mathbf{s}_j)$, then draw a_j from $\pi_j(\cdot \mid \mathbf{s}_j)$. If $a_j =$ ***Stop***, the reasoning ends; otherwise, the system transitions to the next state $\mathbf{s}_{j+1}$, where the new entropy measures yield an updated state.

4 Experiments

In this section, we first compare the reasoning performance and reasoning steps used between baseline methods and Entro-duction. Subsequently, we present ablation studies to analyze the contributions of each part of our strategies. Following this, we

examine how different parameter settings impact Entro-duction's overall robustness.

4.1 Experiment Settings

Datasets. Entro-duction is a general approach applicable to various LLMs and reasoning tasks. Here, we test across four reasoning tasks with benchmark datasets, including two mathematical tasks (GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021)) and two commonsense question-answering tasks (StrategyQA (Geva et al., 2021), CommonsenseQA (Talmor et al., 2018)). Here, GSM8K challenges language models with multi-step math reasoning tasks, assessing their complex reasoning capabilities, while SVAMP focuses on simpler, one-step math reasoning tasks. StrategyQA tests strategic reasoning skills for deriving implicit strategies and using deductive reasoning to answer questions. CommonsenseQA (CSQA) tests the ability to handle commonsense reasoning with everyday knowledge. In evaluating performance on these datasets, we primarily focus on reasoning accuracy (%) as the key metric.

Baselines. We compare Entro-duction with two strong baseline types: (1) Reasoning structures, including Chain of Thought (CoT), Chain of Thought with Self-Consistency (CoT-SC), Tree of Thought (ToT) and Complex CoT:

- CoT: Guides the model to solve problems step-by-step and generates a coherent reasoning chain that leads to a conclusion.
- CoT-SC: Generates multiple reasoning chains and uses a majority vote to determine the final output. We sample answer 8 (CoT-SC@maj8) and 64 (CoT-SC@maj64) times to employ majority vote for selection.
- ToT: Expands the reasoning process into a tree-like structure where multiple branches represent different reasoning pathways.
- Complex CoT: Engages with complex samples and selects the best solution from various intricate reasoning paths for tackling multi-faceted and challenging problems.

And (2) Reasoning depth optimization methods, including Self-talk (Shwartz et al., 2020; Molfese et al., 2024) and Distillation-Reinforcement-Reasoning (DRR) (Yang et al., 2024):

- Self-talk: Enhances reasoning by eliciting LLMs to generate exploratory questions, uncovering implicit background knowledge and selecting the best answer.
- DRR: Distills LLM reasoning processes into synthetic data by training a lightweight model to provide feedback.

For detailed settings of baselines, please refer to Appendix A.

Implementation Details. We conduct the experiments utilizing the `Llama-3.1-8B-Instruct`[1]. The temperature for all models is set to the default value of 0.7, with a maximum token limit of 128. All tasks are performed on an NVIDIA 4090 GPU.

4.2 Overall Performance

The overall performance is reported in Table 1. For the baselines, we compare reasoning accuracies across four datasets, and for reasoning structures, we additionally measure the number of steps required. Since each structure's steps and branches are predefined, we adopt configurations that can perform well and that further increasing the step count often does not bring significant gains. Specifically, for math tasks, CoT is set to 8 steps, CoT-SC adopts 3 parallel chains of 8 steps each, and ToT generates three branches per step for five layers. For commonsense tasks, CoT is set to 5 steps, CoT-SC adopts 3 parallel chains of 5 steps each, and ToT generates three branches per step for five layers.

Compared to reasoning structures, our Entro-duction approach achieves both higher accuracy and fewer reasoning steps. For instance, on GSM8K, CoT reaches 0.75 accuracy, CoT-SC@maj64 0.80, and Complex CoT 0.81, while Entro-duction attains 0.85. A similar advantage appears on SVAMP (up to 0.92), and on the commonsense tasks StrategyQA and CSQA, Entro-duction scores 0.70 and 0.79 respectively, surpassing most baselines. Moreover, tree-structured methods ToT require hundreds of steps (over 100 in several cases), while Entro-duction needs much fewer steps, only more than CoT. Even compared to fixed-step approaches such as CoT and Complex CoT, Entro-duction delivers higher accuracy in a similar or slightly increased number of steps.

Compared to reasoning depth optimization methods, Entro-duction consistently attains higher per-

[1] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

Method	Math				Commonsense			
	GSM8K		SVAMP		StrategyQA		CSQA	
	Accuracy	# Steps	Accuracy	# Steps	Accuracy	# Steps	Accuracy	# Steps
CoT	75.2	8.0	83.4	8.0	57.7	5.0	75.6	5.0
CoT-SC@maj8	78.1	24.0	87.5	24.0	68.3	15.0	78.2	15.0
CoT-SC@maj64	80.2	24.0	89.6	24.0	67.1	15.0	78.7	15.0
ToT	72.6	121.0	83.3	121.0	65.8	121.0	73.5	121.0
Complex CoT	81.4	8.0	86.2	8.0	65.7	5.0	73.9	5.0
Self-talk	79.1	/	83.7	/	61.5	/	70.0	/
DRR	83.0	/	90.2	/	67.7	/	82.1	/
Entro-duction	85.4	9.5	92.0	11.20	70.3	9.6	79.6	7.1

Table 1: Performance comparison across different reasoning methods with accuracy and number of steps.

formance. Self-talk achieves accuracies of 0.79, 0.61, and 0.70 on GSM8K, StrategyQA, and CSQA, all below Entro-duction 's 0.85, 0.70, and 0.79. DRR demonstrates decent performance on SVAMP (0.90) and CSQA (0.82), but still trails Entro-duction on GSM8K (0.83 vs. 0.85) and StrategyQA (0.67 vs. 0.70). Moreover, these methods often rely on additional training or separate models, while Entro-duction balances accuracy and a relatively low reasoning overhead without training.

4.3 Ablation Study

4.3.1 Impact of Jointly Using Entropy and Variance Entropy

To validate the necessity of jointly using entropy and variance entropy, we conduct experiments across four datasets to validate the necessity of jointly using entropy and variance entropy. We set four different scenarios: *Base* (neither used), *Entropy* (only entropy used), *Variance* (only variance entropy used), and *Both* (both used).

As shown in Figure 3, when using only entropy, the model tends to stop reasoning prematurely in scenarios with many potential outcomes but fewer overall fluctuations (Scenario 2). Using only variance entropy can capture changes in fluctuations between reasoning steps. It slightly outperforms using entropy alone, but still proves inadequate for handling various uncertainty scenarios Scenario 3), with accuracies mostly close to or below *Base*.

4.3.2 Impact of Expansion in Reasoning

Compared with ***Deepen*** and ***Stop*** that directly affect reasoning depth, ***Expand*** is a key behavior in Entro-duction to branch out the current reasoning path to cover more potential solutions. We further validate the necessity of the behavior ***Expand*** by

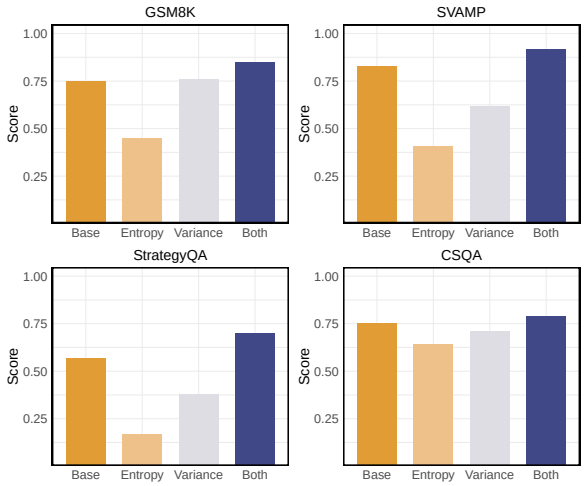

Figure 3: Comparison of adjusting with entropy and/or variance entropy.

comparing three settings across datasets: *Base* (no behavior selection), *w/o* (only using ***Deepen*** and ***Stop***), and *w/* (enabling ***Expand***).

As shown in Figure 4, the accuracies of *w/o* are generally lower than those of *w/*, particularly in tasks requiring multiple reasoning paths or branching thought processes, such as SVAMP and StrategyQA. The result indicates that using only ***Deepen*** and ***Stop*** limits the exploration of potential directions, while expanding the exploration contributes to improving the completeness of the reasoning.

4.3.3 Impact of Soft Stop

In some complex tasks, we notice that the initial reasoning process could see an increase in both entropy and variance entropy. However, the model may still expect valid exploration in the continued reasoning. In this case, if we adopt a "hard stop", which means immediate shutdown, it could terminate exploration in advance of arriving at the

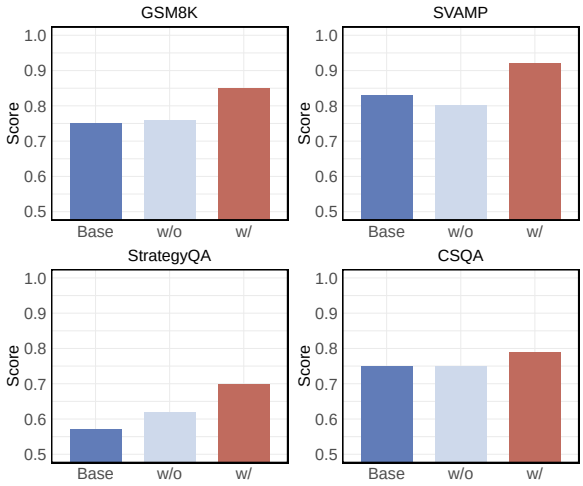

Figure 4: Impact of the behavior ***Expand***.

correct conclusion. Instead, we introduce a "soft stop" mechanism to balance the need for thorough exploration and the risk of redundant reasoning. The model continues for several additional steps before stopping. In our experiments, we implement four settings: *Base* (no stopping strategy), *Stop@1* (hard stop with immediate termination), *Stop@2* (soft stop with one more reasoning step before stopping) and *Stop@3* (soft stop with two more reasoning steps before stopping).

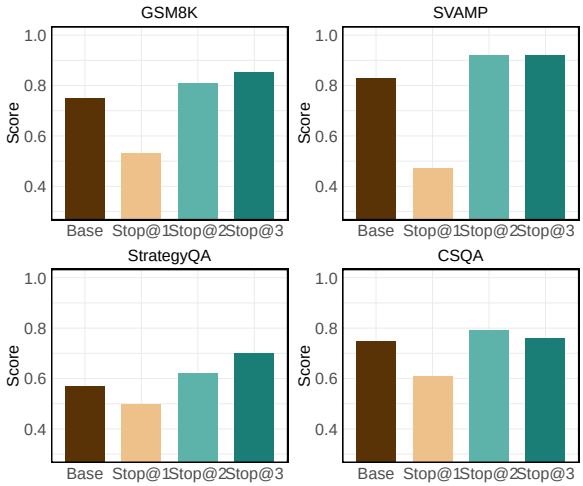

Figure 5: Comparison of stopping strategies.

As shown in Figure 5, we can see that the hard stop strategy has the lowest outcomes across all four datasets, lower than not employing any stopping strategy. The soft stop strategy consistently has the best results. Moreover, on datasets SVAMP and CSQA, Stop@2 performs as well or better than Stop@3. This result suggests that a soft stop extending two to three steps is sufficient to complete effective exploration without the need for extensive further reasoning.

4.4 Robustness Study

We further discuss the impact of ϵ-greedy strategy by testing four ϵ values (0.05, 0.1, 0.25, 0.5).

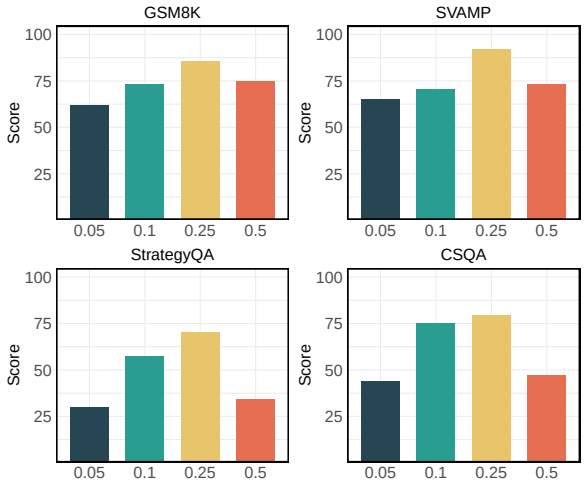

Figure 6: Comparison of choice of ϵ.

As shown in Figure 6, $\epsilon = 0.25$ consistently achieves the highest accuracy across all four datasets. This result demonstrates that this value enhances the model's performance by effectively exploring the solution space. Specifically, when ϵ is set too low ($\epsilon = 0.05$), the model's performance is poor. This is likely due to insufficient exploration that relies heavily on the known strategy, thus unable to explore potential solutions hidden in the space. Meanwhile, when $\epsilon = 0.5$, it outperforms $\epsilon = 0.1$ in mathematical tasks and underperforms in commonsense tasks. This result indicates that tasks requiring reasoning with stringent logical structure and relatively more steps need broader exploration to identify the correct solutions. For commonsense tasks, which require more precise adopting of knowledge for quick decision-making. In this type of task, over-exploration may lead the model away from the question background and thus miss the intuitive commonsense answers.

5 Conclusion

In this study, we introduce Entro-duction, a novel approach that dynamically adjusts the exploration depth during LLM multi-step reasoning by monitoring the entropy and variance entropy. Entro-duction leverages the change of both metrics to select exploration behavior to enhance reasoning performance and avoid redundant reasoning steps. Our experiments across multiple reasoning datasets demonstrate the effectiveness of the Entro-duction and its components.

Limitations

We develop a framework with dynamic depth adjustment strategies for LLMs. If not precisely calibrated, it might lead to suboptimal reasoning performance.. This may limit the Entro-duction method's ability to adaptively balance exploration and exploitation in real-time. Besides, the experiments are conducted on four benchmark datasets on one Llama model, which may not provide a comprehensive view of the Entro-duction's generalization capability across LLMs with varying sizes and pre-training processes. Moreover, the Entro-duction is mainly evaluated on specific tasks and these tasks cannot fully reflect the complexities of real-world scenarios where reasoning tasks can be variable and with more complex and external solution spaces. We left these potential explorations as future work.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*.

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

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Kye Gomez. 2023. Tree of thoughts. https://github.com/kyegomez/tree-of-thoughts.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Ziqi Jin and Wei Lu. 2024. Self-harmonized chain of thought. *arXiv preprint arXiv:2409.04057*.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*.

Xun Liang, Shichao Song, Zifan Zheng, Hanyu Wang, Qingchen Yu, Xunkai Li, Rong-Hua Li, Feiyu Xiong, and Zhiyu Li. 2024a. Internal consistency and self-feedback in large language models: A survey. *arXiv preprint arXiv:2407.14507*.

Yuxin Liang, Zhuoyang Song, Hao Wang, and Jiaxing Zhang. 2024b. Learning to trust your feelings: Leveraging self-awareness in llms for hallucination mitigation. *arXiv preprint arXiv:2401.15449*.

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

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

Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Xingyu Wang, Jiaxing Wang, Hailong Yang, and Jing Li. 2024b. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. *arXiv preprint arXiv:2409.17539*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. *Preprint*, arXiv:2502.09601.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Shentong Mo and Miao Xin. 2024. Tree of uncertain thoughts reasoning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12742–12746. IEEE.

Francesco Maria Molfese, Simone Conia, Riccardo Orlando, and Roberto Navigli. 2024. Zebra: Zero-shot example-based retrieval augmentation for commonsense question answering. *arXiv preprint arXiv:2410.05077*.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *Preprint*, arXiv:2308.03188.

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

David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean. 2022. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. 2024. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. *arXiv preprint arXiv:2004.05483*.

Xiaoshuai Song, Yanan Wu, Weixun Wang, Jiaheng Liu, Wenbo Su, and Bo Zheng. 2025. Progco: Program helps self-correction of large language models. *arXiv preprint arXiv:2501.01264*.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.

Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. 2023. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. *Preprint*, arXiv:2310.12397.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024a. Chain of thoughtlessness: An analysis of cot in planning. *arXiv preprint arXiv:2405.04776*.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024b. On the self-verification limitations of large language models on reasoning and planning tasks. *arXiv preprint arXiv:2402.08115*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Jean-Francois Ton, Muhammad Faaiz Taufiq, and Yang Liu. 2024. Understanding chain-of-thought in llms through information theory. *Preprint*, arXiv:2411.11984.

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

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

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yu Xia, Rui Wang, Xu Liu, Mingyan Li, Tong Yu, Xiang Chen, Julian McAuley, and Shuai Li. 2024. Beyond chain-of-thought: A survey of chain-of-x paradigms for llms. *arXiv preprint arXiv:2404.15676*.

Diji Yang, Linda Zeng, Kezhen Chen, and Yi Zhang. 2024. Reinforcing thinking through reasoning-enhanced reward models. *arXiv preprint arXiv:2501.01457*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

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

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

Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. 2024a. Ratt: Athought structure for coherent and correct llmreasoning. *arXiv preprint arXiv:2406.02746*.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2024b. On the diagram of thought. *arXiv preprint arXiv:2409.10038*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Experimental Settings

A.1 Baselines

In CoT, we prompt the model with the sentense "`Let's think step-by-step.`". For CoT-SC, we use a majority vote to identify the most probable correct solution, with the same few-shot examples as the standard CoT method. For Complex CoT we follow the setting in (Zhou et al., 2022). The setting of Self-talk and DRR method follows the setting in (Yang et al., 2024).

A.2 Answer-Cleaning

We showcase our answer-cleaning process with GSM8K as an example. In the context of the GSM8K dataset, the answer-cleansing process is crucial for ensuring the accuracy and usability of predictions from large language models. Initially, the raw prediction, referred to as `pred`, often contains numerical answers formatted with commas or mixed with textual content. To standardize these predictions, we first remove any commas to normalize the numbers. Subsequently, we use regular expressions to extract all numerical values from this cleaned string. Given the nature of GSM8K tasks, where a specific numerical answer is typically required, our algorithm strategically selects either the first or last number based on predefined logic tailored to the dataset's requirements. This selection process is designed to pick the most relevant number based on its position in the model's output. The final step produces a `cleansed_pred`, which is the processed and formatted numerical answer ready for evaluation against the dataset's ground truth.

Algorithm 2 Answer Cleansing for GSM8K Dataset

```
Input: pred ▷ Raw prediction from the model
Output: cleansed_pred ▷ Cleansed numerical prediction
Remove commas from pred
Extract all numbers from pred using regex
Select the first or last number based on context
return cleansed_pred
```