Tsinghua Science and Technology
Doi: 10.26599/TST.2026.9010003

Research/Review Article

Aggressive Speculative Decoding for Efficient Chain-of-Thought Reasoning

Huanran Zheng, Xiaoling Wang(✉), Jingwei Zhang, Ya Zhou

Abstract Large language models (LLMs) have demonstrated excellent performance in complex tasks through Chain-of-Thought (CoT) reasoning. However, LLMs usually generate long thinking content during reasoning, leading to high inference latency. Speculative decoding (SD) is a general acceleration method but has two limitations that weaken its acceleration effect in CoT reasoning scenarios. Therefore, in this study, we proposed the aggressive speculative decoding (ASD) method for efficient CoT reasoning, alleviating the two shortcomings. First, SD adopts a strict acceptance strategy to ensure the generated results are consistent with the target model. However, multiple feasible reasoning trajectories may lead to correct answers. Therefore, ASD adopts an aggressive acceptance strategy, which accepts more draft tokens based on the deviation between the probabilities. Second, SD only generates a small fixed number of draft tokens in each drafting stage. In contrast, ASD adopts an aggressive drafting strategy, which adaptively generates a suitable number of draft tokens by estimating the expected transformation value of the inference latency through confidence. Extensive experiments on different reasoning datasets and LLMs demonstrate the effectiveness of our method. In particular, ASD achieved a significantly better acceleration effect than SD on all datasets while ensuring accuracy.

Keywords Large language model; Chain-of-thought; Speculative decoding; Efficient inference

1 Introduction

Chain-of-Thought (CoT) reasoning enables large language models (LLMs) to achieve excellent performance on various complicated reasoning tasks [1–5]. Through appropriate CoT prompts [6, 7] or reinforcement learning [8–11], LLMs can generate a long thinking content before answering the question, which is used to break down and analyze the question, thereby significantly improving the accuracy. Furthermore, OpenAI's o1 series models [12] found that the reasoning capabilities of LLMs improved as the length of the thinking content increased. However, existing LLMs are typically based on the autoregressive transformer architecture, which needs to generate content step by step. This results in extremely high inference latency in the CoT reasoning process, affecting the user experience.

Speculative decoding (SD) [13, 14] is a general method for accelerating the inference process of LLMs. As shown in Fig. 1, the main idea of SD is to first use a smaller and more efficient model as the draft model to generate multiple draft tokens quickly and then use the target LLM to verify in parallel whether these tokens are accepted. Therefore, SD makes full use of the computing power of parallel computing devices such as GPUs, significantly reducing the inference latency of LLM on general tasks. However, SD has some limitations in the CoT reasoning scenario, resulting in a weak acceleration effect.

Although significant efforts [15–17] have been made to improve the performance of SD, these works still consider general generation tasks. There is a lack of research on decoding algorithms specifically designed to accelerate the CoT reasoning process. Therefore, in this study, we propose an aggressive speculative decoding (ASD) method to address the shortcomings of the SD in the CoT reasoning scenario, thereby effectively accelerating the speed of LLMs to complete complex reasoning tasks and filling the gap in this research field. Specifically, through careful analysis, we summarize the two main limitations of the SD method in the CoT reasoning scenario and make corresponding improvements.

First, during the verification phase, SD only accepts draft tokens that are the same as the results generated by the target model. However, multiple feasible reasoning trajectories can produce correct results for the same question. Therefore, SD's acceptance strategy is unreasonable in the CoT reasoning scenario. To solve this problem, ASD employed an aggressive acceptance strategy. Specifically, CoT-decoding [18] uncovered that LLMs can generate inherent reasoning trajectories by considering alternative top-k tokens rather than solely relying on the top-1 greedy decoding. Therefore, even if the draft tokens are not the tokens with the highest

• Huanran Zheng and Xiaoling Wang are with the East China Normal University, Shanghai 200062, China (e-mail: hrzheng@stu.ecnu.edu.cn, xlwang@cs.ecnu.edu.cn)
• Jingwei Zhang and Ya Zhou are with the Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China (e-mail: gtzjw@hotmail.com, ccyzhou@guet.edu.cn)
• Corresponding author: Xiaoling Wang.
Manuscript received: 2025-09-25; revised: 2025-11-11; accepted: 2025-12-22

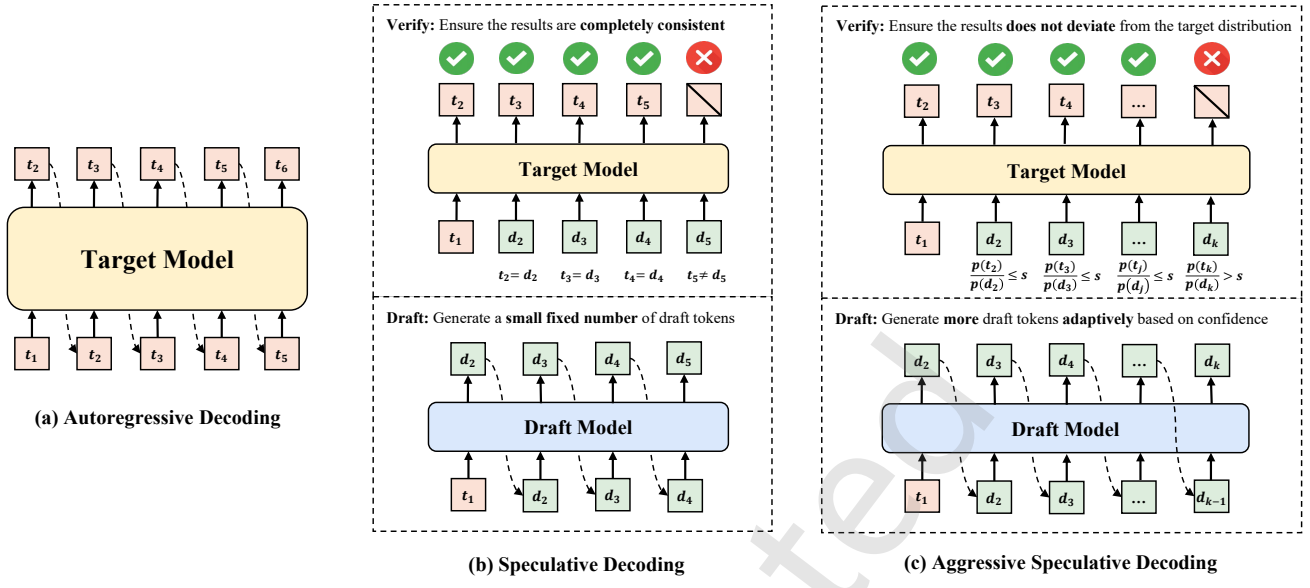


Fig. 1 Comparison between autoregressive decoding (a), speculative decoding (b), and our aggressive speculative decoding (c). $p(\cdot)$ is the probability distribution predicted by the target model, s is the hyperparameter threshold. We proposed three different acceptance strategies. Here, we take the strategy based on the probability ratio as an example.

prediction probability of the target LLM, they may still be the content of other inherent reasoning trajectories. Based on this, ASD will only reject draft tokens that severely deviate from the probability distribution predicted by target LLMs, which can significantly improve the average acceptance rate of draft tokens while ensuring the correctness of the reasoning trajectory.

Second, SD only generates a fixed conservative number of draft tokens in each iteration to reduce the computation time waste due to the rejection. However, this strategy does not consider the differences between questions. When faced with complex questions, it can ensure a high acceptance rate. But, when faced with simple problems, SD can only generate a fixed small number of tokens per iteration, leading to a speed bottleneck. Therefore, ASD adopts an aggressive drafting strategy. Specifically, ASD estimates the expected transformation value of the inference latency caused by each draft token through confidence. If the current draft token causes the expected inference latency to increase, the draft generation process will automatically stop. Through this method, ASD can adaptively generate as many draft tokens as possible according to the difficulty of different questions, enhancing the acceleration effect.

We conducted extensive experiments on various reasoning datasets based on Llama3.1 series models [19] and the DeepSeek-R1-Distill-Qwen series models [9]. The experimental results verify the effectiveness of our method. ASD can alleviate the shortcomings of SD in CoT reasoning scenarios and significantly reduce the inference latency of LLMs. Moreover, ASD can easily trade off efficiency and accuracy according to different requirements. For instance, on the GSM8K [20] dataset, under high-accuracy settings, ASD can achieve a **3.6x** speedup without sacrificing performance. Under high-efficiency settings, ASD can even achieve a **4.0x** speedup with only a 0.3% drop in accuracy. Furthermore, ASD is a plug-and-play method that can be easily applied to different LLMs without training.

To sum up, our contributions are as follows:

(1) To the best of our knowledge, there is still a lack of research on efficient decoding algorithms specifically for Chain-of-Thought (CoT) reasoning scenarios, and this study fills the gap in this field.

(2) We carefully analyzed the limitations of the speculative decoding (SD) method in the CoT reasoning scenario, made targeted improvements, and proposed the aggressive speculative decoding (ASD) method.

(3) Comprehensive experiments were conducted using four reasoning datasets. The experiments confirm that our ASD method can effectively alleviate the two main limitations of the SD method and significantly improve the inference speed of LLMs without any training.

2 Related work

2.1 Chain-of-Thought reasoning

Existing work on improving the reasoning capabilities of LLMs mainly focuses on studying how to enable LLMs to

perform Chain-of-Thought (CoT) reasoning better [21–24]. In the early stage, [6] proposed the CoT prompting technology, which provides LLMs with several demonstrations of reasoning processes through in-context learning [25, 26], guiding the model to think step by step when dealing with complex problems, significantly improving the model's performance on various reasoning tasks.

Benefiting from the remarkable performance of CoT prompting, much subsequent work follows this idea and improves it. For instance, Program-of-Thoughts [7] enhances the accuracy of numerical reasoning tasks by generating program codes to express reasoning steps and executing these codes through an external interpreter to complete the calculation. Self-Ask [27] lets the model explicitly ask itself follow-up questions before answering the initial question and leverages a search engine to answer follow-up questions, effectively improving the accuracy of the answers. Least-to-most prompting [28] breaks down a complex problem into a series of simpler subproblems and then solves them in sequence. Tree-of-Thoughts [29] allows LLMs to make deliberate decisions by considering multiple reasoning paths and self-evaluating choices to decide the following action. These methods effectively improve the reasoning ability of LLMs through prompt engineering but often involve manually intensive prompt engineering.

Different from the above methods, the OpenAI o1 model [12] internalizes the form of CoT reasoning into the model generation process through large-scale reinforcement learning, enabling the model to think deeply before answering each question. The OpenAI o1 model has achieved outstanding performance on various reasoning benchmarks, attracting widespread attention. For example, [30] explored the data distillation from existing o1-like models and showed the effectiveness of data distillation. [8] leveraged Monte Carlo Tree Search (MCTS) to generate code data with reasoning processes to train O1-Coder. Recently, DeepSeek-R1 [9] has achieved great success. It uses the group relative policy optimization (GRPO) algorithm [31] for training, which saves the training cost of reinforcement learning and achieves performance competitive with OpenAI o1.

Although CoT reasoning has achieved excellent performance, it significantly increases the inference latency of the model. Since existing LLMs use autoregressive decoding, the longer the length of the generated thinking context, the greater the inference latency. However, efficient CoT reasoning has been less explored and is still in the early stages of research. Recently, [32] proposed a cascade system of different sizes LLMs to save the cost of using LLMs in reasoning tasks but did not consider the issue of inference speed. Compressed Chain-of-Thought [33] (CCoT) elicits reasoning with a short sequence of continuous embeddings, allowing for much greater throughput. However, CCoT requires retraining the LLMs, affecting the model's reasoning ability. Unlike the above approaches, this study specifically explores the decoding algorithm for efficient CoT reasoning and proposes the aggressive speculative decoding (ASD) method, which can significantly improve the inference speed while ensuring the generation quality.

2.2 Speculative decoding

Speculative decoding (SD) [13, 14] is a general method that can accelerate the inference process of LLMs without compromising the generation quality or modifying the model itself. In each iteration, SD first uses a small model as the draft model to generate multiple draft tokens quickly and then uses target LLMs for parallel verification to determine whether the draft tokens can be accepted. In this way, SD enables LLMs to generate multiple tokens in one forward pass, breaking the speed bottleneck of autoregressive decoding.

The performance of SD is mainly determined by the efficiency of the draft model and the average acceptance rate of draft tokens. Therefore, researchers have done a lot of work on these two aspects to further improve the acceleration effect of SD. For example, Medusa [34] adds additional decoding headers to LLM, enabling it to generate multiple future tokens from different decoding headers as draft tokens in parallel, significantly improving the generation speed of draft tokens. EAGLE [35] generates draft tokens by performing autoregressive decoding based on the feature level of LLMs, which can utilize the knowledge of LLMs to improve the quality of generated draft tokens. DistillSpec [16] employs knowledge distillation to better align the draft model with the target LLM, thereby increasing the acceptance rate of draft tokens. SpecInfer [36] constructs a candidate tree-based speculative inference and verification system, which significantly increases the average acceptance length in each iteration by constructing a draft tokens tree.

Although these methods can enhance the performance of SD, they are all oriented towards general generation tasks. In the CoT reasoning scenario, these methods still have some limitations. For instance, the draft models of Medusa and EAGLE can quickly generate draft tokens. However, their reasoning ability has a significant gap with the target LLMs, resulting in high rejection rates when performing CoT reasoning. The number of nodes in the candidate draft tree constructed by SpecInfer increases exponentially as the tree

depth increases, which limits the length of draft sequences during the drafting process, leading to a bottleneck in the acceleration effect. Recently, RSD [37] has improved SD to adapt to reasoning scenarios. However, RSD requires an additional process reward model to enable better evaluation of the generated drafts, which can reduce FLOPs but does not improve the speed. In contrast, our method is plug-and-play and can be applied to various LLMs without any training. Furthermore, it can address the shortcomings of SD, thereby significantly improving the CoT reasoning speed of LLMs.

3 Preliminary

The speculative decoding (SD) framework usually contains two models: one is the target model that wants to be accelerated, and the other is a draft model that is much faster than the target model. Based on these two models, SD divides each iteration into two stages: drafting and verification (as shown in Fig. 1). In the drafting stage, SD uses the draft model to quickly generate the subsequent m draft tokens $(d_{i+1}, d_{i+2}, ..., d_{i+m})$ based on the previous content $t_{\leqslant i}$. In the verification phase, SD inputs $t_{\leqslant i}$ and draft tokens together into the target model for a forward pass, and then determines whether to accept $(d_{i+1}, d_{i+2}, ..., d_{i+m})$ based on the predicted probability distribution. There are two acceptance strategies: speculative decoding and speculative sampling, which are equivalent to the greedy search and standard sampling of the target model. In this section, we mainly introduce the speculative decoding acceptance strategy. If you are interested in speculative sampling strategy, please refer to the original paper [13, 14]. The SD acceptance strategy performs a greedy search on the probability distribution predicted by the target model to obtain the prediction results of the target model $(t_{i+1}, t_{i+2}, \ldots, t_{i+m}, t_{i+m+1})$. Then, it will determine whether to accept the draft token from left to right:

$$\begin{cases} \text{Accept } d_j, \text{ if } d_j = t_j \\ \text{Reject } d_j, \text{ if } d_j \neq t_j \end{cases} \tag{1}$$

where $i+1 \leqslant j \leqslant i+m$. Furthermore, when the first rejection occurs, SD will accept the t_j to correct the currently rejected draft token d_j and discard all subsequent draft tokens. Due to the causal mask, the generation process of all accepted tokens is exactly the same as autoregressive decoding. Therefore, SD can guarantee that the generated results are consistent with the results when the target model performs the greedy search. In addition, if d_{i+m} is same as t_{i+m}, t_{i+m+1} will also be accepted. Therefore, SD generates at least one token in each iteration and at most $m+1$ tokens. Finally, the accepted tokens are concatenated with the previous content as the input for the next iteration.

4 Method

This section introduces our proposed ASD method in detail. First, we carefully analyze the limitations of the SD method in the CoT reasoning scenario and give the motivation for our approach in Section 4.1. Then, in Section 4.2, we propose corresponding solutions to the SD's problems as the two main modules of our ASD method. Finally, we give the overall process of the ASD algorithm in Section 4.3.

4.1 Motivation

SD method can achieve excellent performance in general tasks, which can significantly accelerate the target LLMs without affecting the generation results. However, it still has some limitations in the CoT reasoning scenario.

First, SD's acceptance strategy is too strict. As introduced in Section 3, this acceptance strategy can ensure that the results generated by SD are completely consistent with the results generated by autoregressive target LLMs. However, this strict acceptance strategy is unnecessary in the CoT reasoning scenario. For example, as shown in Fig. 2, for the same problem, multiple feasible reasoning trajectories can get the correct answer. In particular, the two thinking contents sampled by Llama-3.1-8B-Instruct directly calculate 9 through "16-3-4", while Llama-3.1-70B-Instruct first calculates 7 through "3+4" and then calculates 9 through "16-7". In addition, CoT-decoding [18] uncovered that LLMs are capable of generating correct reasoning trajectories by considering alternative top-k tokens. Therefore, rejecting draft tokens because of the different reasoning strategies or styles is too harsh and will bring unnecessary computational overhead.

Second, SD only generates a small fixed number of draft tokens in each iteration. The average acceptance rate is the key factor affecting the SD acceleration effect. Therefore, SD generates only a small fixed number of draft tokens in each iteration to ensure the acceptance rate, which does not consider the difficulty differences between samples. In previous work [34, 35, 38], this number is usually fixed at around 5. When faced with complex samples, it can ensure a high acceptance rate. However, when faced with simple problems, SD can only generate a fixed small number of tokens per iteration, leading to a speed bottleneck. Moreover, due to the development of knowledge distillation technology [9, 30, 39], the reasoning ability of existing smaller models has been significantly enhanced. For most reasoning tasks,

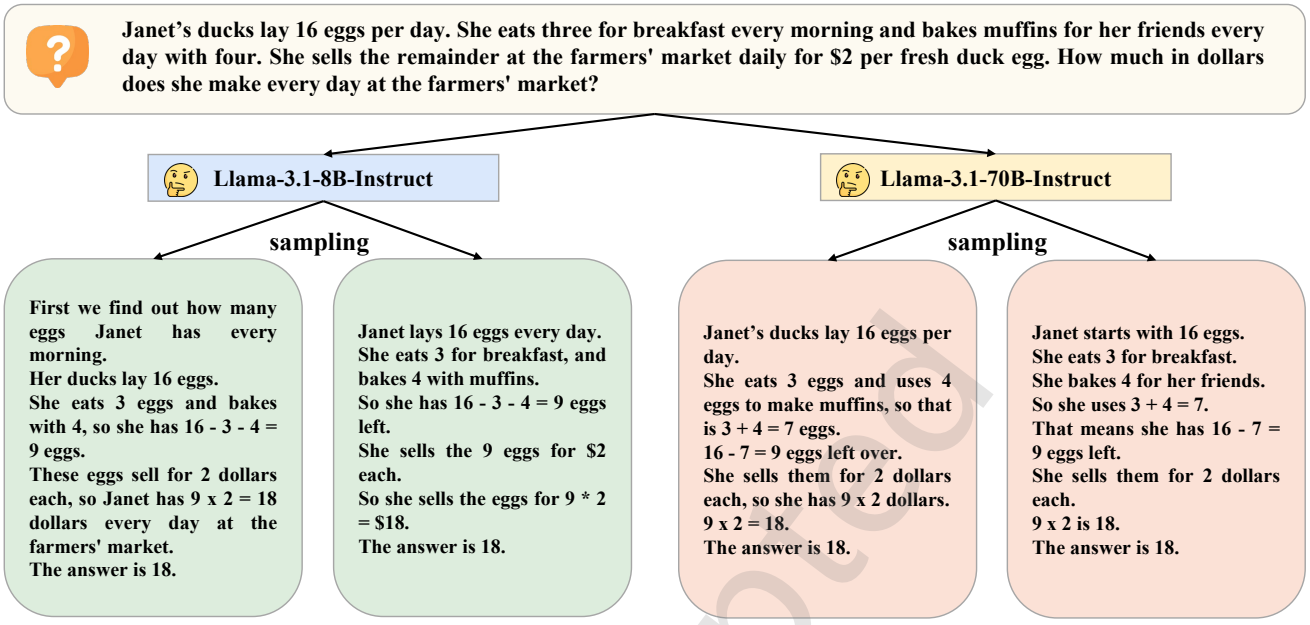


Fig. 2 This figure shows that for the same problem, LLMs can generate multiple feasible reasoning trajectories. For brevity, we do not show the CoT demonstration content.

the gap between their performance and that of large models has been narrowed. For example, on the GSM8K dataset [20], the accuracy of Llama-3.1-70B-Instruct is 93.4%, and the accuracy of Llama-3.1-8B-Instruct is 84.5%. These results show that if Llama-3.1-8B-Instruct is used as a draft model, in most cases, Llama-3.1-70B-Instruct can accept the entire draft thinking content, and only one forward pass is needed to generate the correct answer.

The above two limitations lead to the poor acceleration effect of the SD method in CoT reasoning scenarios. Therefore, to better improve the inference speed of existing LLMs, it is essential to design a special efficient decoding algorithm for CoT reasoning scenarios.

4.2 Aggressive speculative decoding

Because of the shortcomings of the SD method, we propose the aggressive speculative decoding (ASD) method specifically for the CoT reasoning scenario. Specifically, ASD makes two improvements to the SD method.

4.2.1 Aggressive acceptance strategy

As shown in Fig. 2, for the same problem, LLMs can generate multiple correct reasoning trajectories by sampling from the probability distribution. Therefore, we propose aggressive acceptance strategies. During the validation phase, ASD will accept the draft token if it does not deviate too much from the probability distribution predicted by the target LLM. In particular, we propose three feasible strategies to determine whether the draft token deviates from the target probability distribution:

- Based on the absolute probability. Define the current draft token to be judged as d_j, the current probability distribution predicted by the target LLM as p, and the token with the largest probability value as t_j. If $p(d_j)$ is greater than the set threshold s_1, it means that LLM has high confidence in predicting d_j, so accept it. In addition, if $d_j = t_j$, which satisfies the acceptance condition of SD, we also accept d_j regardless of the value of $p(d_j)$.
- Based on the probability difference. The acceptance strategy based on absolute probability cannot handle situations where there are multiple high-probability tokens in the target probability distribution. Even if $p(d_j)$ is less than s_1, and d_j is different from t_j, d_j should be accepted when $p(d_j)$ is very close to $p(t_j)$. Therefore, this strategy uses the difference between $p(t_j)$ and $p(d_j)$ to determine whether to accept d_j. If $p(t_j) - p(d_j)$ is less than the set threshold s_2, it means that the confidence of target LLM for predicting t_j and d_j is similar, then d_j is accepted.
- Based on the probability ratio. If $p(t_j)/p(d_j)$ is less than the set threshold s_3, accept d_j. Compared with the strategy based on probability difference, this strategy is more sensitive to the situation when $p(t_j)$ is small. Because the difference cannot be greater than $p(t_j)$, but

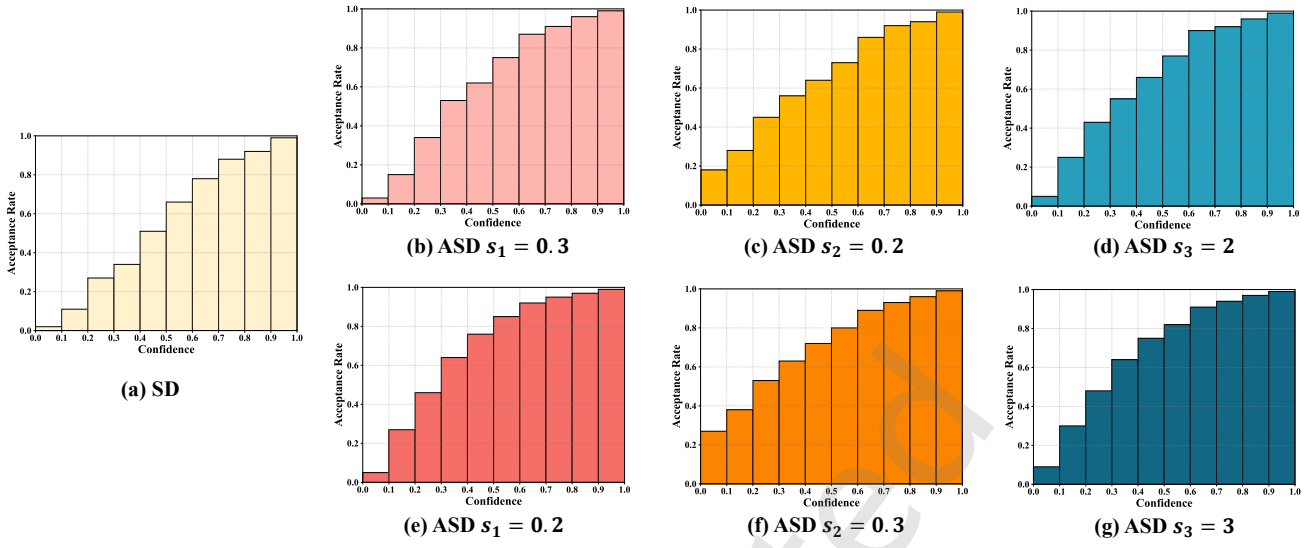


Fig. 3 Statistical histograms of confidence (probability) and acceptance rate of draft tokens under different acceptance strategies and various threshold settings on the GSM8k dataset.

the ratio can be very large.

Through the above aggressive acceptance strategies, our ASD method can accept more thinking content generated by the draft model, significantly improving the acceleration effect. In addition, since they can ensure that the reasoning process does not deviate from the target probability distribution, the impact on the accuracy of the final answer is negligible. Further mathematical proofs and analyses are presented in Appendix A.

4.2.2 *Aggressive drafting strategy*

Our ASD method adopts an aggressive drafting strategy that can adaptively generate as many draft tokens as possible according to the difficulty of questions, making full use of the reasoning ability of the draft model. Following the approach proposed by [40], we first explored the correlation between the confidence (probability) and the acceptance rate of draft tokens under different acceptance strategies. Specifically, we conducted experiments on the GSM8K dataset, requiring the draft model to generate a sequence of length 30 each time and counted each draft token's confidence score and acceptance rate. Note that we do not include draft tokens after the first rejection to ensure independence in the statistical process. We divided the confidence scores into 10 interval bins and plotted the corresponding histograms. Fig. 3 shows that confidence and acceptance rate have a strong positive correlation under different acceptance strategies and threshold settings. Then, unlike in [40] which directly uses confidence, we regard the statistical acceptance rate of the interval bin where the confidence of draft token d_j is located as d_j's independent acceptance probability a_j. Furthermore, considering the entire draft sequence, d_j being accepted means that it and all the draft tokens before it are accepted, so its joint probability of being accepted is:

$$o_j = \prod_{i=1}^{j} a_i. \tag{2}$$

We can then estimate the expected inference latency reduction brought by d_j based on o_j. Note that we only make estimates based on ideal conditions due to the complexity of system-level factors (such as KV cache, I/O, etc.). Specifically, assuming that the time required for a forward pass of the draft model and the target model is s_d and s_t, the expected reduction in inference latency of d_j can be evaluated by the following formula:

$$s_t - [(1 - o_j) * (s_t + s_d) + o_j * s_d] = o_j * s_t - s_d \tag{3}$$

Therefore, if $o_j < s_d/s_t$, continuing to generate the draft token is more likely to increase the inference latency, and ASD will stop drafting. Based on this principle, ASD can adaptively determine the number of draft tokens generated in each iteration and enhance the acceleration effect.

4.3 Overall process

Algorithm 1 generalizes the overall process of ASD, taking the aggressive acceptance strategy based on the probability ratio as an example. Given the target model $p(.|.)$, the draft model $q(.|.)$, and the set threshold s_3, we first need to calculate the statistical acceptance rate of different confidence intervals and the time required for one forward pass of the two models as described in Section 4.2.2. After the initial prompt and

Algorithm 1 Aggressive Speculative Decoding

Given: Target model $p(.|.)$, Draft model $q(.|.)$, Threshold s_3
Input: Initial prompt sequence $x_1, x_2, ..., x_i$, Maximum sequence length T
while i ¡ T **do**
  for $j = 1$ to $(T - i)$ **do**
    $d_j \leftarrow \arg\max q(d|x_1, x_2, ..., x_i, d_1, ..., d_{j-1})$
    Calculate o_j based on the Eq. (2)
    if $o_j < s_d/s_t$ **then**
      m ← j, Break
    end if
  end for
  $(t_1, ..., t_{m+1}) \leftarrow \arg\max p(t|x_1, ..., x_i, d_1, ..., d_m)$
  for $j = 1$ to m **do**
    if $p(t_j)/p(d_j) > s_3$ **then**
      $x_{i+j} \leftarrow d_j$
    else
      $x_{i+j} \leftarrow t_j$, $n \leftarrow j$, Break
    end if
  end for
  if $(d_1, ..., d_m)$ all accept **then**
    $x_{i+m+1} \leftarrow t_{m+1}$, $i \leftarrow i + m + 1$
  else
    $i \leftarrow i + n$
  end if
end while

maximum length are entered, ASD starts to iterate and quickly generate subsequent content.

In the iteration process, ASD first adopts the aggressive drafting strategy and continuously generates draft tokens until $o_j < s_d/s_t$. Then, ASD predicts $(t_1, ..., t_m, t_{m+1})$ in parallel and uses the aggressive acceptance strategy to verify whether the draft tokens are accepted. Note that if all draft tokens are accepted, ASD will further accept t_{m+1}. Finally, the accepted tokens are concatenated with the previous content as the input for the next round of iteration.

Through the aggressive acceptance strategy and aggressive drafting strategy, our ASD method can effectively alleviate the limitations of the SD method in the CoT reasoning scenario and significantly improve the acceleration effect. In addition, ASD inherits the SD's advantages and can be applied to various existing LLMs in a plug-and-play manner without any training.

5 Experiment

In this section, we verify the effectiveness of our proposed ASD method on different datasets and LLMs. We first introduce our experimental setup in Section 5.1. Then, we report the main results in Section 5.2. We also perform ablation studies in Section 5.3 to demonstrate the contributions of individual components of ASD. Finally, we conduct a more in-depth analysis of our ASD approach in Section 5.4.

5.1 Experimental setup

5.1.1 Models and datasets

We selected two series of LLMs, LLaMA-3.1-Instruct series models [19] and DeepSeek-R1-Distill-Qwen series models [9], for experiments. Specifically, for the LLaMA-3.1-Instruct, we use the 8B model as the draft model and the 70B model as the target model. For the DeepSeek-R1-Distill-Qwen, we use the 1.5B model as the draft model and the 32B model as the target model. In addition, to comprehensively evaluate our method, we selected the following different reasoning task datasets:

(1) GSM8K [20], a high-quality dataset of grade school math word problems to support the task of question answering on basic mathematical problems that require multi-step reasoning. Its test set contains 1319 samples. We randomly select 319 as the validation set for hyper-parameter selection in our experiment and the remaining 1000 as the test set for evaluating model performance.

(2) MATH [41], a more difficult mathematical reasoning task dataset that contains challenging competition mathematics problems. We adopt MATH-500[①] as our test set.

(3) DATE, a symbolic reasoning task dataset in BIG-Bench Hard [42], which requires understanding and inferring dates. We do not provide options but let the model infer the answer directly.

(4) ARC-C [43], a commonsense reasoning dataset containing multiple-choice science questions. We randomly select 500 samples from it as our test set.

5.1.2 Metrics and baselines

We use accuracy (ACC) to evaluate the quality of generated results. We use three metrics to measure the acceleration effect of different methods:

(1) Walltime speedup: The actual speedup ratio relative to autoregressive decoding, which may be affected by different operating environments.

① https://huggingface.co/datasets/HuggingFaceH4/MATH-500

Table 1 Evaluation metrics of different methods on various reasoning task datasets and different LLMs. This table measures the comparative performance of the baselines and our method in terms of ACC (%). Walltime speedup, average acceptance rate α, and average acceptance length τ are used to measure the acceleration effect. The highest scores are in **bold** in the table.

		GSM8K				MATH				DATE				ARC-C			
Model	Method	ACC	Speedup	α	τ	ACC	Speedup	α	τ	ACC	Speedup	α	τ	ACC	Speedup	α	τ
LLaMA-3.1-Instruct	AD (Target Model)	93.0	1.0x	-	-	58.8	1.0x	-	-	88.8	1.0x	-	-	90.7	1.0x	-	-
	AD (Draft Model)	79.6	6.0x	-	-	43.0	5.7x	-	-	67.2	6.0x	-	-	81.0	6.1x	-	-
	SD (10 draft tokens)	93.0	2.5x	0.79	7.9	58.8	2.4x	0.73	7.3	88.8	2.7x	0.87	8.7	**90.7**	2.0x	0.63	6.3
	SD (20 draft tokens)	93.0	2.4x	0.60	12.0	58.8	2.3x	0.50	10.0	88.8	2.5x	0.68	13.6	**90.7**	1.8x	0.43	8.6
	MEDUSA	93.0	2.7x	0.70	3.5	58.8	2.6x	0.68	3.4	88.8	3.2x	0.82	4.1	**90.7**	2.6x	0.70	3.5
	ASD ($s_1 = 0.3$)	**93.4**	3.5x	0.90	15.5	56.8	3.0x	0.72	14.4	**89.2**	3.5x	0.98	25.6	**90.7**	3.0x	0.78	9.8
	ASD ($s_1 = 0.2$)	92.2	**3.8x**	0.93	22.1	57.2	3.2x	0.73	20.1	88.4	3.7x	0.98	32.0	89.7	**3.4x**	**0.80**	**14.0**
	ASD ($s_2 = 0.2$)	92.9	3.6x	0.94	16.7	56.4	3.0x	0.72	15.4	86.4	3.5x	0.98	26.6	89.3	2.9x	0.74	9.9
	ASD ($s_2 = 0.3$)	92.5	3.8x	0.94	20.7	55.2	3.3x	**0.75**	19.9	84.8	3.6x	**0.99**	31.1	86.7	3.1x	0.76	12.4
	ASD ($s_3 = 2$)	**93.4**	3.6x	0.94	18.1	**59.0**	3.2x	**0.75**	16.5	89.0	3.6x	0.98	28.4	**90.7**	3.0x	0.77	10.8
	ASD ($s_3 = 3$)	92.7	**4.0x**	**0.98**	**22.6**	57.2	**3.3x**	**0.75**	**20.5**	87.6	**3.8x**	**0.99**	**32.6**	88.3	**3.4x**	**0.80**	13.9
DeepSeek-R1-Distill-Qwen	AD (Target Model)	90.4	1.0x	-	-	75.8	1.0x	-	-	83.2	1.0x	-	-	91.0	1.0x	-	-
	AD (Draft Model)	65.8	3.9x	-	-	45.4	3.9x	-	-	36.0	4.0x	-	-	50.3	4.0x	-	-
	SD (10 draft tokens)	90.4	1.9x	0.70	7.0	**75.8**	1.9x	0.71	7.1	83.2	1.9x	0.72	7.2	**91.0**	1.3x	0.48	4.8
	SD (20 draft tokens)	90.4	1.7x	0.50	10.0	**75.8**	1.6x	0.52	10.4	83.2	1.8x	0.60	12.0	**91.0**	0.9x	0.29	5.8
	MEDUSA	90.4	2.3x	0.66	3.3	**75.8**	2.2x	0.64	3.2	83.2	2.3x	0.66	3.3	**91.0**	1.8x	0.58	2.9
	ASD ($s_1 = 0.3$)	90.1	2.5x	0.74	14.2	74.8	2.6x	0.85	14.8	**84.4**	2.4x	0.83	18.7	90.3	1.7x	0.56	5.0
	ASD ($s_1 = 0.2$)	89.7	**2.8x**	**0.88**	16.4	73.6	**2.9x**	0.87	20.8	83.2	**2.6x**	**0.87**	**23.1**	88.3	**1.9x**	**0.62**	6.2
	ASD ($s_2 = 0.2$)	90.5	2.5x	0.78	13.8	73.8	2.5x	0.85	15.5	82.4	2.3x	0.85	18.8	87.3	1.7x	0.60	5.2
	ASD ($s_2 = 0.3$)	89.4	2.7x	0.80	16.7	72.2	**2.9x**	0.88	19.4	81.6	2.4x	0.85	21.2	86.3	1.8x	0.60	6.1
	ASD ($s_3 = 2$)	**90.8**	2.5x	0.75	14.7	**75.8**	2.6x	0.87	17.1	83.2	2.4x	**0.87**	20.3	**91.0**	1.6x	0.51	5.6
	ASD ($s_3 = 3$)	90.0	2.6x	0.74	**20.3**	74.6	**2.9x**	**0.91**	**21.5**	82.0	2.5x	**0.87**	23.0	90.3	1.8x	0.58	**6.3**

(2) Average acceptance rate α: The average acceptance rate of draft tokens generated in each iteration. A low acceptance rate means most generated draft tokens are rejected, resulting in a poor acceleration effect.

(3) Average acceptance length τ: The average number of new tokens generated per iteration. The larger α and τ, the better the acceleration effect. Furthermore, both α and τ are not affected by hardware configuration and can provide a more objective evaluation.

To verify the effectiveness of our method, we selected various baselines for comparison: (1) Autoregressive Decoding (AD). We adopt a greedy search for decoding. (2) Vanilla Speculative Decoding [13]. We adopt two settings, generating 10 and 20 draft tokens per iteration, respectively. (3) MEDUSA [34], which adds additional decoding heads to LLM and generates draft tokens in a non-autoregressive way.

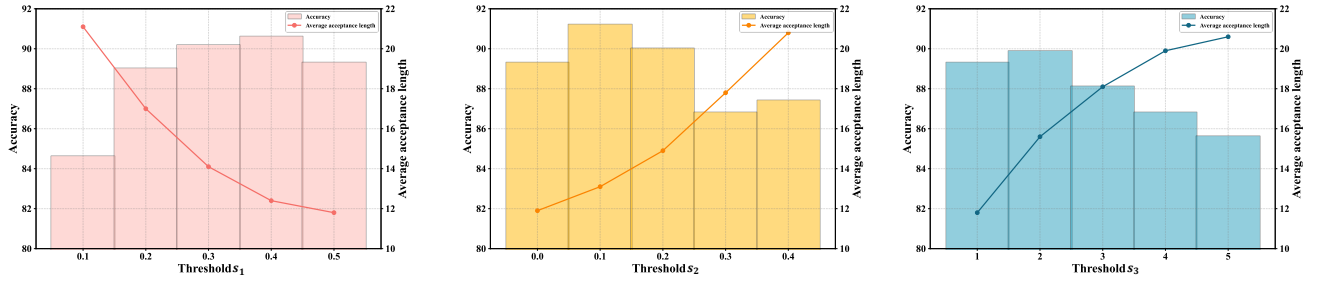
5.1.3 Implementation details

We conduct experiments on NVIDIA RTX A6000 48GB GPUs. We use the CoT prompting technology to guide the LLMs in performing CoT reasoning. Specifically, for GSM8K and MATH, we adopt the prompt in the chain-of-thought hub [44]. For DATE and ARC-C, we manually write CoT demonstrations. In the ASD verification phase, we use two different settings: high accuracy and high efficiency. The high-accuracy setting aims to speed up the target LLM while ensuring accuracy is as unaffected as possible. In contrast, the high-efficiency setting pursues better acceleration effects and can accept a slight decrease in accuracy. In the high-accuracy setting, the thresholds of the three different aggressive acceptance strategies are $s_1 = 0.3$, $s_2 = 0.2$, and $s_3 = 2$. For the high-efficiency setting, the thresholds are: $s_1 = 0.2$, $s_2 = 0.3$ and $s_3 = 3$. For the ASD draft generation stage, we calculated the average time required for one forward pass of the target model (LLaMA-3.1-Instruct 70B: 256ms, DeepSeek-R1-Distill-Qwen-32B: 118ms) and the draft model (LLaMA-3.1-Instruct 8B: 39ms, DeepSeek-R1-Distill-Qwen-1.5B: 25ms) respectively. Then, we count the acceptance rates in different confidence intervals based on the GSM8K validation set for various strategies and thresholds (as shown in Fig. 3). Our experiments focus on the scenario with a batch size of 1, representing the use case where the LLMs are hosted locally for personal use.

5.2 Main results

The main experimental results are shown in Table 1. As can be seen, our proposed ASD method can significantly improve the inference speed while ensuring high accuracy, performing better than baselines.

First, the aggressive acceptance strategy causes almost no accuracy loss in the high-accuracy setting. This experimental result verifies that there are multiple feasible reasoning trajectories in the CoT reasoning stage, so it is not necessary to use the strict acceptance strategy of SD to generate correct results, which verifies the rationality of our method.



(a) Based on absolute probability (b) Based on the probability difference (c) Based on probability ratio

Fig. 4 Performance across various aggressive acceptance strategies with different thresholds. We conduct experiments on the GSM8K validation set, and the results reveal the trade-off between accuracy and efficiency under different threshold settings.

Second, combining the aggressive acceptance strategy and aggressive drafting strategy can significantly improve the average acceptance length and acceptance rate. Specifically, SD can achieve a high acceptance rate when generating 10 draft tokens in each iteration, but its average acceptance length is low. On the contrary, when SD generates 20 draft tokens per iteration, its acceptance length is improved, but the average acceptance rate is reduced. In contrast, our ASD method can adaptively generate an appropriate number of draft tokens in each iteration through the aggressive drafting strategy and accept more tokens through the aggressive acceptance strategy. More importantly, even if we estimate the acceptance rate based only on the GSM8K dataset, the aggressive drafting strategy still performs well on other datasets, which shows that the mapping between confidence scores and acceptance rates is generalizable across different datasets. In addition, although MEDUSA can generate draft tokens faster, its draft sequence length is limited and the generation quality is poor, resulting in a low average acceptance length. Therefore, ASD can still achieve better acceleration than MEDUSA.

Third, ASD is generalizable and can perform well on different models and datasets. As we can see, when based on DeepSeek-R1-Distill-Qwen series models, the inference latency after using SD on the ARC-C dataset is similar to or even higher than that of AD. This is because the reasoning ability of DeepSeek-R1-Distill-Qwen-1.5B on the ARC-C dataset is much weaker than that of DeepSeek-R1-Distill-Qwen-32B, resulting in a very low acceptance rate of SD. However, our ASD method can still achieve adequate acceleration in this case.

Finally, ASD can flexibly adapt to various scenarios by setting different thresholds. When pursuing high accuracy, we can use the high-accuracy setting. Experimental results show that when $s_3 = 2$, ASD can achieve significant acceleration without affecting accuracy. When speed is a priority, we can adapt the high-efficiency setting, which can further improve the acceleration effect with less than 2% accuracy degradation.

5.3 Ablation study

In this section, we comprehensively verify the effectiveness of components in ASD, including the aggressive acceptance strategy and the aggressive drafting strategy. We conduct experiments on the GSM8K validation set based on the DeepSeek-R1-Distill-Qwen series models.

5.3.1 Effectiveness of aggressive acceptance strategy

We propose three aggressive acceptance strategies to address the shortcomings of the SD method. To verify the effectiveness of the proposed aggressive acceptance strategies, we fix the number of draft tokens generated in each iteration to 30 and evaluate the accuracy and average acceptance length under different threshold settings. Since the number of draft tokens generated each time is fixed, the average acceptance length can effectively measure the model efficiency. The higher the average acceptance length, the faster the inference speed.

The experimental results are shown in the Fig. 4. The three aggressive acceptance strategies are all the better than the SD acceptance strategy in CoT scenarios. They can flexibly balance generation quality and inference speed by setting different thresholds. Specifically, the strategy based on the probability difference and the strategy based on the probability ratio degenerates into the SD acceptance strategy when their thresholds $s_2 = 0.0$ and $s_3 = 1$. Then, when their thresholds gradually increase, the average acceptance length of each iteration increases, i.e., the inference speed is significantly improved. In addition, it is exciting that when $s_2 = 0.1$ and $s_3 = 2$, they can even achieve higher accuracy than the SD acceptance strategy. We believe this may be because the thinking content generated by the smaller model may be more suitable for some simple problems, and our aggressive acceptance strategies can accept the results generated by the draft model as much as possible when facing simple problems. Similarly,

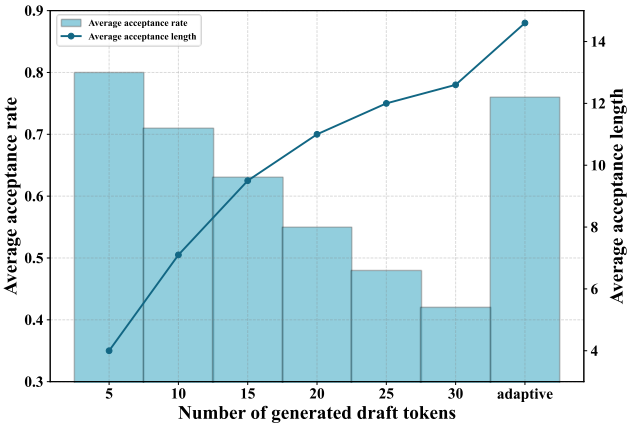


Fig. 5 The impact of the number of generated draft tokens in each iteration on the average acceptance rate and average acceptance length. 'adaptive' means using our proposed aggressive drafting strategy.

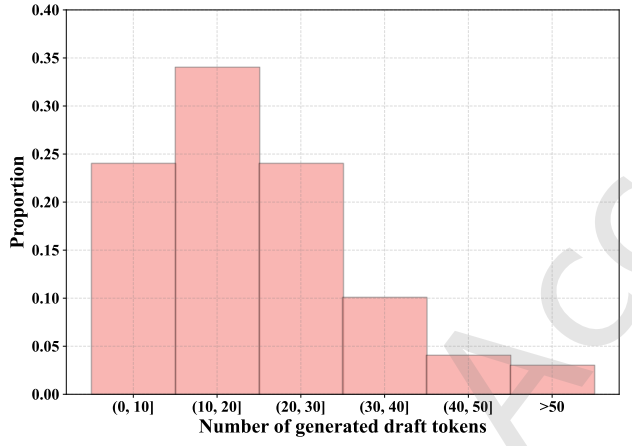


Fig. 6 Distribution of the number of generated draft tokens in each iteration when using the aggressive drafting strategy.

the acceptance strategy based on absolute probability can also significantly improve the average acceptance length while ensuring accuracy by setting an appropriate threshold s_1.

These experiment results verify the effectiveness of our proposed aggressive acceptance strategies. According to the experimental results, we adopted two different threshold settings, high accuracy and high efficiency, as mentioned in the implementation details, to adapt well to various application scenarios.

5.3.2 *Effectiveness of aggressive drafting strategy*

To verify the effectiveness of the proposed aggressive drafting strategy, we adopt the acceptance strategy based on probability ratio with $s_3 = 2$ and compare the average acceptance rate and average acceptance length under different drafting strategies.

As shown in Fig. 5, when using the fixed number, the average acceptance length gradually increases as the number of generated draft tokens in each iteration increases. However, the average acceptance rate continues to decrease. Because this strategy cannot distinguish the difficulty differences between different samples, directly increasing the number of draft tokens generated in each iteration can not improve the acceleration effect of the SD method.

Unlike the SD method, our method can adaptively determine the number of draft tokens that should be generated in each iteration. Fig. 6 shows the distribution of the generated draft tokens number in each iteration when using our aggressive drafting strategy. Our aggressive drafting strategy can generate draft sequences of various lengths, and sometimes even more than 50 draft tokens can be generated in one iteration. Furthermore, even if a large number of draft tokens are generated, our strategy can still maintain a high average acceptance rate. These experimental results show that estimating the expected reduction in inference latency caused by each draft token through the statistical acceptance rate with different confidence intervals is reasonable. Therefore, our aggressive drafting strategy can generate an appropriate number of tokens in each drafting phase, significantly improving the average acceptance length while ensuring a high average acceptance rate.

Table 2 Performance of different methods in the internal CoT reasoning scenario based on the DeepSeek-R1-Distill-Qwen series models. The highest scores are in **bold** in the table.

	MATH				AIME 2024			
Method	ACC	Speedup	α	τ	ACC	Speedup	α	τ
AD (Target Model)	85.4	1.0x	-	-	60.0	1x	-	-
AD (Draft Model)	74.4	3.7x	-	-	23.3	3.7x	-	-
SD (10 draft tokens)	**85.4**	2.3x	0.83	8.3	60.0	1.4x	0.47	4.7
ASD ($s_1 = 0.3$)	84.8	**2.9x**	**0.91**	22.3	**66.7**	3.2x	**0.57**	11.7
ASD ($s_2 = 0.2$)	83.6	2.8x	0.86	21.2	53.3	3.1x	0.56	11.9
ASD ($s_3 = 2$)	**85.4**	**2.9x**	0.90	**23.9**	56.7	**3.3x**	**0.57**	**12.0**

5.4 Analysis

5.4.1 *Analysis of confidence and acceptance rate*

The aggressive drafting strategy we proposed requires statistics on the acceptance rate of draft tokens at different confidence levels. Therefore, we conducted experiments on the validation set of GMS8K, divided the confidence scores into 10 intervals, and calculated the average acceptance rate of each interval. The experimental results are shown in Fig. 3. It can be seen that the confidence of the draft token has a clear positive relationship with its acceptance rate. The higher the confidence, the higher the acceptance rate. In addition, our aggressive acceptance strategies can achieve a higher acceptance rate in the same confidence interval compared with the SD acceptance strategy. Furthermore, the higher the

Table 3 Case study of our method ($s_2 = 2$) and SD method (10 draft tokens) based on DeepSeek-R1-Distill-Qwen series models. For brevity, we omit the CoT prompt demonstrations and line breaks. We employ // to divide the content generated by each iteration.

Question	A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Vanilla SD	The robe takes 2 bolts of blue fiber. It // takes half that much white fiber, so 2 / // 2 = 1 bolt of // white fiber. Total bolts // are 2 + 1 = 3. The // answer is 3.
ASD	The robe takes 2 bolts of blue fiber. Half that much white fiber is 2 / // 2 = 1 bolt. So total is 2 + 1 = 3 bolts. The answer is 3.
Question	Altitudes $\overline{AD}$ and $\overline{BE}$ of $\triangle ABC$ intersect at H. If $\angle BAC = 54^\circ$ and $\angle ABC = 52^\circ$, then what is $\angle AHB$?
Vanilla SD	The // point // $(// 0,3$ is on the positive $ // y$-axis. // So // , $ // r = // n // sqrt{0^2 + 3^2} // = 3$, // and $ n // theta = n frac{π}{2}$. Thus //, the polar coordinates are $(3, n frac{n // pi}{2}$. The answer is (3 // , $\pi/2$).
ASD	The // point // is $(0, 3)$, so $ // x=0$, $y = 3$. $r = \sqrt{x^2 + y^2} = \sqrt{0 + 9} = 3$. θ is the angle made with the positive x-axis. Since // the point is on the positive y-axis, $\theta = \frac{\pi}{2}$. So the polar // coordinates are $(3, \frac{\pi}{2})$. The answer is $(3, \pi/2)$
Question	Today's meeting is rescheduled to 11 am tomorrow, 10/16/1924. What is the date one week from today in MM/DD/YYYY?
Vanilla SD	Tomorrow // is 10/16/192//4, // so today is 10/15/1//924. One week from today is 1//0/22/1924. The // answer is 10/22/19//24.
ASD	Today's meeting is rescheduled to 11 am tomorrow, 10/16/1924. So today // is 10/15/1924. One week from today is 10/22/1924. The answer is 10/22/1924.
Question	A goat gets energy from the grass it eats. Where does the grass get its energy? A. soil B. sunlight C. water D. air
Vanilla SD	1. The goat // eats grass and // gets // energy from // it. 2. Grass // is a plant, and // plants produce // their // own food // through photosynthesis. 3. In // photosynthesis, plants use sunlight, // water, and carbon dioxide to produce // glucose, which is their // energy // source // . 4. Therefore, the grass gets its energy from // sunlight. The answer is B.
ASD	1. The goat // eats grass and // gets // energy from // it. 2. The grass is a plant, and plants make //their own food through photosynthesis. 3. Photosynthesis is // the process by which plants convert sunlight // into energy. 4. Therefore, the grass gets its energy from sunlight. The answer is B.

tolerable value of the discrepancy between the draft token and the target distribution is set by the threshold, the higher the acceptance rate. Therefore, our ASD method can generate and accept more draft tokens in each round of iteration.

5.4.2 *Analysis of internal CoT reasoning scenario*

In the above experiment, we used CoT prompting technology to guide LLMs in performing reasoning. This approach requires manually writing explicit CoT demonstrations. There is also a mainstream method that uses reinforcement learning or supervised fine-tuning to enable LLMs to spontaneously perform internal CoT reasoning, such as the o1 model [12] and the DeepSeek-R1 series models [9]. Therefore, we explore the performance of our ASD method in the internal CoT reasoning scenario. Specifically, we conduct experiments on the MATH and AIME 2024[①] datasets based on the DeepSeek-R1-Distill-Qwen series models. The AIME 2024 dataset contains challenging questions from the 2024 American invitational mathematics examination that require deep thinking from LLMs to answer.

As shown in Table 2, our ASD method still performs excellently in the internal CoT reasoning scenario. Moreover, the ASD approach achieves a higher speedup in the internal CoT reasoning scenario than when using CoT prompting. Internal CoT reasoning produces very long thinking content that may contain repeated or similar text, thus increasing the acceleration effect of ASD. In particular, when using CoT prompting, the DeepSeek-R1-Distill-Qwen 32B model generates an average of 265 tokens per question on the MATH dataset. In this case, the ASD method achieves a speedup of 2.5x-2.6x. When using internal CoT reasoning, the DeepSeek-R1-Distill-Qwen 32B model generates an average of 1097 tokens per question, and ASD can achieve a speedup of 2.8x-2.9x. Furthermore, we found that the average token number of results generated by ASD on the AIME 2024 dataset is smaller than that of AD. We believe this is because ASD adopts the aggressive acceptance strategy, which reduces the number of times LLMs repeat self-reflection. Therefore, ASD achieved a very high speedup on this dataset. These experimental results further validate the potential of our approach, which can be effectively applied to LLMs that spontaneously perform deep thinking.

① https://huggingface.co/datasets/HuggingFaceH4/aime_2024

Table 4 Performance of different methods in the non-CoT tasks based on the LLaMA-3.1-Instruct series models. The highest scores are in **bold**.

	HumanEval				CNN/DM			
Method	Pass@1	Speedup	α	τ	Rouge-2	Speedup	α	τ
SD (10 draft tokens)	**78.2**	2.3x	0.71	7.1	14.9	1.2x	0.43	4.3
ASD ($s_1 = 0.3$)	76.4	3.1x	0.76	16.0	14.6	**2.4x**	**0.66**	7.3
ASD ($s_2 = 0.2$)	76.8	3.0x	0.76	16.2	**15.1**	**2.4x**	0.65	**7.5**
ASD ($s_3 = 2$)	78.0	**3.3x**	**0.78**	**17.2**	14.9	2.3x	0.65	7.3

5.4.3 *Generalization on non-CoT tasks*

To further explore the performance of our method on non-COT tasks, we conducted experiments on the HumanEval [45] (code generation) and CNN/DM [46] (text summarization) datasets using the LLaMA-3.1-Instruct series models. As shown in Table 4, even on non-COT tasks, our proposed

ASD method still achieves a significant speedup over SD. These experimental results further demonstrate the potential of our approach. Even in scenarios where explicit reasoning is not required, LLMs can still produce diverse and correct responses. Our method effectively leverages this property to substantially improve both the average acceptance rate and the average acceptance length. In future work, we plan to further evaluate the effectiveness and generalization of our method across more datasets and tasks.

5.4.4 Case study

We display a case study of SD and our ASD approach in Table 3. As we can see, for the same question, the number of iterations required for SD is significantly more than that for ASD. Specifically, for complex questions, SD can only accept a few tokens in each iteration, resulting in many draft tokens being rejected. For simple problems, SD can only generate at most 11 tokens per iteration, which leads to a speed bottleneck. In contrast, ASD adaptively generates as many draft tokens as possible in each iterations through the aggressive draft strategy and adopts the aggressive acceptance strategy to improve the average acceptance rate. Therefore, ASD can generate a dozen or even dozens of tokens in one iteration and maintain a high acceptance rate, significantly improving the acceleration effect.

6 Conclusion

In this study, we analyze the limitations of SD in the CoT reasoning scenarios and make corresponding improvements to propose the ASD approach. First, we design aggressive acceptance strategies that can accept more draft tokens while ensuring generation quality. Second, we propose an aggressive drafting strategy that adaptively decides the appropriate number of draft tokens to generate in each iteration by estimating the expected reduction in inference latency. Experimental results indicate that ASD can achieve a significantly better acceleration effect than SD in the CoT reasoning scenario while ensuring accuracy.

Although our method achieved excellent results, it still warrants further exploration and improvement. First, ASD currently requires manual threshold setting. In future work, we will explore adaptive threshold optimization mechanisms to mitigate the impact of different models and datasets, thereby enhancing the adaptability and usability of this method across various applications. Second, our proposed aggressive drafting strategy relies on confidence mappings statistically derived from the GSM8K dataset. In the future, we plan to employ more diverse datasets or explore alternative approaches to further improve the generalization and robustness of the aggressive drafting strategy.

Acknowledgements

This work was supported by NSFC grant (No. 62136002 and 62477014), Ministry of Education Research Joint Fund Project (8091B042239), and Fundamental Research Funds for the Central Universities.

A Mathematical Analysis for Aggressive Acceptance Strategies

In this section, we provide the mathematical justification and convergence analysis for the three aggressive acceptance strategies proposed in Section 4.2.1. Specifically, we analyze the Kullback–Leibler (KL) divergence between the distribution of tokens accepted by ASD and the target model's probability distribution, demonstrating that appropriate threshold settings can guarantee a bounded deviation from the target distribution.

Let $p(x)$ denote the target model's probability distribution, and let $\pi_{\text{ASD}}(x)$ represent the actual distribution of tokens accepted by ASD. The KL divergence between the ASD output distribution and the target distribution is defined as:

$$D_{\text{KL}}(\pi_{\text{ASD}}|p) = \sum_x \pi_{\text{ASD}}(x) \log \frac{\pi_{\text{ASD}}(x)}{p(x)}. \tag{4}$$

A smaller KL divergence indicates that the ASD output distribution is closer to the target distribution, ensuring that the reasoning trajectory does not deviate from what the target model would generate.

A.1 Absolute Probability-Based Acceptance Strategy

Define the acceptance region $\mathcal{A}_1 = \{x : p(x) \geqslant s_1\}$. For $\mathcal{A}_1$ to be non-empty, s_1 must satisfy $s_1 \leqslant p_{\max}$, where $p_{\max} = \max_x p(x)$ denotes the probability of the most likely token under the target distribution. Then, the ASD output distribution can be written as:

$$\pi_{\text{ASD}}(x) = \begin{cases} \frac{p(x)}{\sum_{x' \in \mathcal{A}_1} p(x')} & \text{if } x \in \mathcal{A}_1 \\ 0 & \text{otherwise} \end{cases}$$

The KL divergence between π_{ASD} and p can be decomposed as

$$D_{\text{KL}}(\pi_{\text{ASD}}\|p) = \sum_{x \in \mathcal{A}_1} \pi_{\text{ASD}}(x) \log \frac{\pi_{\text{ASD}}(x)}{p(x)}$$

For $x \in \mathcal{A}_1$, we have $p(x) \geqslant s_1$, which implies

$$\log \frac{\pi_{\text{ASD}}(x)}{p(x)} \leqslant \log \frac{1}{s_1} = -\log(s_1)$$

Using the fact that $\sum_{x\in\mathcal{A}_1}\pi_{\text{ASD}}(x)=1$ and $s_1\leqslant p_{max}$, we obtain

$$D_{\text{KL}}(\pi_{\text{ASD}}\|p)\leqslant-\log(s_1)\leqslant-\log(p_{max})$$

Therefore, as s_1 increases, the ASD output distribution will converges toward the target distribution.

A.2 Probability Difference-Based Acceptance Strategy

Define the acceptance region as $\mathcal{A}_2=\{x:p(x)\geqslant p_{\max}-s_2\}$, where $0\leqslant s_2\leqslant p_{\max}$.

Following the derivation steps in A.1, we obtain:

$$\log\frac{\pi_{\text{ASD}}(x)}{p(x)}\leqslant\log\frac{1}{p_{\max}-s_2}=-\log(p_{\max}-s_2)$$

Furthermore, since $\sum_{x\in\mathcal{A}_2}\pi_{\text{ASD}}(x)=1$, it follows that:

$$\begin{aligned}D_{\text{KL}}(\pi_{\text{ASD}}\|p)&=\sum_{x\in\mathcal{A}_2}\pi_{\text{ASD}}(x)\log\frac{\pi_{\text{ASD}}(x)}{p(x)}\\&\leqslant-\log(p_{\max}-s_2)\end{aligned}$$

Therefore, when p_{max} is large, the ASD output distribution converges to the target distribution as s_2 decreases.

A.3 Probability Ratio-Based Acceptance Strategy

Similarly, define the acceptance region as $\mathcal{A}_3=\{x:p(x)\geqslant p_{\max}/s_3\}$, where $1\leqslant s_3$. Then, we can obtain:

$$\log\frac{\pi_{\text{ASD}}(x)}{p(x)}\leqslant\log\frac{s_3}{p_{\max}}=\log(s_3)-\log(p_{\max})$$

Finally,

$$\begin{aligned}D_{\text{KL}}(\pi_{\text{ASD}}\|p)&=\sum_{x\in\mathcal{A}_3}\pi_{\text{ASD}}(x)\log\frac{\pi_{\text{ASD}}(x)}{p(x)}\\&\leqslant\log(s_3)-\log(p_{\max})\end{aligned}$$

Therefore, the smaller s_3 is, the closer the result generated by our method is to the target distribution.

The mathematical proof above provides further theoretical guidance for setting the thresholds in our three proposed aggressive acceptance strategies. Moreover, it demonstrates that an appropriately chosen threshold ensures that the reasoning trajectory obtained by our method remains consistent with that of the target model.

Reference

[1] T. B. Brown, B. Mann, N. Ryder, and et al., Language models are few-shot learners, *ArXiv*, vol. abs/2005.14165, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:218971783

[2] J. Chen, F. Wang, S. Pang, M. Chen, M. Xi, T. Zhao, and J. Yin, A privacy policy text compliance reasoning framework with large language models for healthcare services, *Tsinghua Science and Technology*, vol. 30, no. 4, pp. 1831–1845, 2025. [Online]. Available: https://www.sciopen.com/article/10.26599/TST.2024.9010089

[3] B. Wang, Y. Liu, H. Tian, R. Hua, K. Chang, J. Xia, X. Dai, Z. Gao, S. Liu, R. Wang, X. Zhou, and W. Wei, Llm4deu: Fine tuning large language model for medical diagnosis in outpatient and emergency department visits of neurosurgery, *Tsinghua Science and Technology*, vol. 30, no. 6, pp. 2487–2504, 2025. [Online]. Available: https://www.sciopen.com/article/10.26599/TST.2024.9010125

[4] OpenAI, Gpt-4 technical report, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257532815

[5] DeepSeek-AI, Deepseek-v3 technical report, *ArXiv*, vol. abs/2412.19437, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:275118643

[6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, F. Xia, Q. Le, and D. Zhou, Chain of thought prompting elicits reasoning in large language models, *ArXiv*, vol. abs/2201.11903, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246411621

[7] W. Chen, X. Ma, X. Wang, and W. W. Cohen, Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, *Trans. Mach. Learn. Res.*, vol. 2023, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253801709

[8] Y. Zhang, S. Wu, Y. Yang, J. Shu, J. Xiao, C. Kong, and J. Sang, o1-coder: an o1 replication for coding, *ArXiv*, vol. abs/2412.00154, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:274436851

[9] DeepSeek-AI, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, *ArXiv*, vol. abs/2501.12948, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:275789950

[10] W. Liu, X. Xu, J. Wu, and J. Jiang, Federated meta reinforcement learning for personalized tasks, *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 911–926, 2024. [Online]. Available: https://www.sciopen.com/article/10.26599/TST.2023.9010066

[11] L. Dong, N. Li, G. Gong, and X. Lin, Offline reinforcement learning with constrained hybrid action implicit representation towards wargaming decision-making, *Tsinghua Science and Technology*, vol. 29, no. 5, pp. 1422–1440, 2024. [Online]. Available: https://www.sciopen.com/article/10.26599/TST.2023.9010100

[12] OpenAI, Learning to reason with llms, 2024. [Online]. Available: https://openai.com/index/learning-to-reason-with-llms/

[13] Y. Leviathan, M. Kalman, and Y. Matias, Fast inference from transformers via speculative decoding, in *International*

Conference on Machine Learning, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254096365

[14] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. M. Jumper, Accelerating large language model decoding with speculative sampling, *ArXiv*, vol. abs/2302.01318, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:256503945

[15] C. Du, J. Jiang, Y. Xu, J. Wu, S. Yu, Y. Li, S. Li, K. Xu, L. Nie, Z. Tu, and Y. You, Glide with a cape: A low-hassle method to accelerate speculative decoding, *ArXiv*, vol. abs/2402.02082, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267412316

[16] Y. Zhou, K. Lyu, A. S. Rawat, A. K. Menon, A. Rostamizadeh, S. Kumar, J.-F. Kagy, and R. Agarwal, Distillspec: Improving speculative decoding via knowledge distillation, *ArXiv*, vol. abs/2310.08461, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263909387

[17] M. Elhoushi, A. Shrivastava, D. Liskovich, B. Hosmer, B. Wasti, L. Lai, A. Mahmoud, B. Acun, S. Agarwal, A. Roman, A. Aly, B. Chen, and C.-J. Wu, Layerskip: Enabling early exit inference and self-speculative decoding, *ArXiv*, vol. abs/2404.16710, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269362647

[18] X. Wang and D. Zhou, Chain-of-thought reasoning without prompting, *ArXiv*, vol. abs/2402.10200, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267681847

[19] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, Llama: Open and efficient foundation language models, *ArXiv*, vol. abs/2302.13971, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257219404

[20] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, Training verifiers to solve math word problems, *ArXiv*, vol. abs/2110.14168, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:239998651

[21] Q. Li, H. Sun, F. Xiao, Y. Wang, X. Gao, and B. Bhanu, Ps-cot-adapter: adapting plan-and-solve chain-of-thought for scienceqa, *Science China Information Sciences*, vol. 68, no. 1, p. 119101, 2025.

[22] H. Xu, Y. Li, L. Ma, C. Li, Y. Dong, X. Yuan, and H. Liu, Autonomous embodied navigation task generation from natural language dialogues, *Science China Information Sciences*, vol. 68, no. 5, pp. 1–14, 2025.

[23] H. Wu, Y. Zhang, Z. Han, Y. Hou, L. Wang, S. Liu, Q. Gong, and Y. Ge, Cot-driven framework for short text classification: Enhancing and transferring capabilities from large to smaller model, *Knowledge-Based Systems*, vol. 311, p. 113057, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705125001042

[24] R. Li, Y. Wang, Z. Wen, M. Cui, and Q. Miao, Different paths to the same destination: Diversifying llms generation for multi-hop open-domain question answering, *Knowledge-Based Systems*, vol. 309, p. 112789, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705124014230

[25] M.-K. Ghali, A. Farrag, D. Won, and Y. Jin, Enhancing knowledge retrieval with in-context learning and semantic search through generative ai, *Knowledge-Based Systems*, vol. 311, p. 113047, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705125000942

[26] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, Rethinking the role of demonstrations: What makes in-context learning work? *ArXiv*, vol. abs/2202.12837, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247155069

[27] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis, Measuring and narrowing the compositionality gap in language models, *ArXiv*, vol. abs/2210.03350, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252762102

[28] D. Zhou, N. Scharli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. H. Chi, Least-to-most prompting enables complex reasoning in large language models, *ArXiv*, vol. abs/2205.10625, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248986239

[29] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, Tree of thoughts: Deliberate problem solving with large language models, *ArXiv*, vol. abs/2305.10601, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258762525

[30] Z. Huang, H. Zou, X. Li, Y. Liu, Y. Zheng, E. Chern, S. Xia, Y. Qin, W. Yuan, and P. Liu, O1 replication journey - part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *ArXiv*, vol. abs/2411.16489, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:274234336

[31] Z. Shao, P. Wang, Q. Zhu, R. Xu, J.-M. Song, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, Deepseekmath: Pushing the limits of mathematical reasoning in open language models, *ArXiv*, vol. abs/2402.03300, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267412607

[32] M. Yue, J. Zhao, M. Zhang, L. Du, and Z. Yao, Large language model cascades with mixture of thoughts representations for cost-efficient reasoning, *ArXiv*, vol. abs/2310.03094, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263671564

[33] J. Cheng and B. V. Durme, Compressed chain of thought: Efficient reasoning through dense representations, *ArXiv*, vol. abs/2412.13171, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:274789675

[34] T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. huai Chen, and T. Dao, Medusa: Simple llm inference

acceleration framework with multiple decoding heads, *ArXiv*, vol. abs/2401.10774, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267061277

[35] Y. Li, F. Wei, C. Zhang, and H. Zhang, Eagle: Speculative sampling requires rethinking feature uncertainty, *ArXiv*, vol. abs/2401.15077, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267301131

[36] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, Z. Zhang, R. Y. Y. Wong, A. Zhu, L. Yang, X. Shi, C. Shi, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia, Specinfer: Accelerating large language model serving with tree-based speculative inference and verification, *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:267094734

[37] B. Liao, Y. Xu, H. Dong, J. Li, C. Monz, S. Savarese, D. Sahoo, and C. Xiong, Reward-guided speculative decoding for efficient llm reasoning, *ArXiv*, vol. abs/2501.19324, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:276079559

[38] Y. Li, F. Wei, C. Zhang, and H. Zhang, Eagle-2: Faster inference of language models with dynamic draft trees, in *Conference on Empirical Methods in Natural Language Processing*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:270702281

[39] E. Shang, H. Liu, J. Zhang, R. Zhao, and J. Du, Ensemble knowledge distillation for federated semi-supervised image classification, *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 112–123, 2025. [Online]. Available: https://www.sciopen.com/article/10.26599/TST.2023.9010156

[40] H. Zheng and X. Wang, Faster speculative decoding via effective draft decoder with pruned candidate tree, in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 9856–9868. [Online]. Available: https://aclanthology.org/2025.acl-long.486/

[41] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, Let's verify step by step, *arXiv preprint arXiv:2305.20050*, 2023.

[42] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso *et al.*, Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, *arXiv preprint arXiv:2206.04615*, 2022.

[43] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, Think you have solved question answering? try arc, the ai2 reasoning challenge, *arXiv:1803.05457v1*, 2018.

[44] Y. Fu, L. Ou, M. Chen, Y. Wan, H. Peng, and T. Khot, Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance, *arXiv preprint arXiv:2305.17306*, 2023.

[45] M. Chen, J. Tworek, H. Jun, and et al., Evaluating large language models trained on code, 2021.

[46] R. Nallapati, B. Zhou, C. Dos Santos, Ç. Gulçehre, and B. Xiang, Abstractive text summarization using sequence-to-sequence rnns and beyond, in *Proceedings of the 20th SIGNLL conference on computational natural language learning*, 2016, pp. 280–290.

Author biography

Huanran Zheng received the BEng degree from Nanjing Normal University in 2021. He is currently a PhD candidate at the School of Computer Science and Technology, East China Normal University, China. His research interests include large language models and efficient inference algorithms.

Xiaoling Wang received the PhD degree from Southeast University in 2003. She is currently a professor at the School of Computer Science and Technology, East China Normal University, China. Her research interests include artificial intelligence, data management technology, and data analysis.

Jingwei Zhang received the PhD degree from East China Normal University in 2012. He is currently a professor at the School of Computer and Information Security, Guilin University of Electronic Technology. His main research interests include big data management and analysis, and smart applications.

Ya Zhou received the M.S degree from Fudan University in 1996. She is currently a professor at the School of Computer and Information Security, Guilin University of Electronic Technology. Her main research interests include distributed computing and optimization, and big data analysis.