Learning When to Sample: Confidence-Aware Selective Sampling for Efficient Chain-of-Thought Reasoning

Anonymous ACL submission

Abstract

Large language models (LLMs) can achieve strong reasoning performance through chain-of-thought (CoT) reasoning, yet they often generate unnecessarily long reasoning paths that incur high inference cost. Self-consistency-based approaches push accuracy higher still, but they require sampling and aggregating multiple reasoning trajectories, leading to substantial computational overhead. In this paper, we introduce a confidence-aware selective sampling framework that, at inference time, analyzes a single reasoning trajectory to adaptively determine whether to rely on that trajectory alone or trigger multi-path sampling. The framework uses trajectory-level numeric and sentence-level linguistic features extracted from reasoning states to guide selective multi-path reasoning. We train it on MedQA and evaluate it in-domain on MedQA and under calibration-only transfer on MathQA, MedMCQA, and MMLU, without further fine-tuning. Experimental results show that the proposed framework maintains comparable performance to full and efficient multi-path reasoning baselines, with accuracy changes of -0.41 ± 0.58 and -0.31 ± 0.58 percentage points, respectively, while reducing token usage by $71.7 \pm 5.0\%$ and $36.6 \pm 9.1\%$. These findings demonstrate that reasoning trajectories contain rich signals for uncertainty estimation, enabling a simple, transferable mechanism to balance accuracy and efficiency in LLM reasoning.

1 Introduction

Large language models (LLMs) have recently demonstrated strong reasoning capabilities, enabling effective performance on a wide range of tasks, including mathematical problem-solving, commonsense reasoning, and scientific question answering (Wei et al., 2022b; Kojima et al., 2022; Sprague et al., 2025). This emerging reasoning ability represents an important step toward general-purpose problem-solving systems, as LLMs can decompose complex problems and produce coherent intermediate solutions (Wei et al., 2022a; Yao et al., 2023a). Despite these promising observations, LLM reasoning remains unstable. A suboptimal local decoding decision can disrupt subsequent reasoning steps and propagate through the sequential generation trajectory (Peng et al., 2025; Song et al., 2026). This phenomenon limits the reliability of single-pass decoding, suggesting that a single generation is often insufficient for robust problem solving (Wang et al., 2023; Song et al., 2026).

To address this gap, prior work has introduced various sampling-based aggregation methods. One representative method is self-consistency (Wang et al., 2023), which samples multiple chain-of-thought (CoT) paths and then aggregates them through mechanisms such as majority voting and probabilistic consensus. It has been widely shown that self-consistency can substantially enhance robustness and accuracy compared to its single-path counterpart (Wang et al., 2023; Xue et al., 2023). However, these gains often require substantially more test-time computation, through longer outputs or repeated reasoning-path generation (Snell et al., 2025; Wu et al., 2025; Muennighoff et al., 2025). Such increased generation directly raises inference cost and energy consumption (Samsi et al., 2023; Fernandez et al., 2025), making the balance between reasoning performance and inference efficiency a fundamental challenge.

Motivated by this concern, recent work on efficient reasoning has explored adaptive test-time computation and selective sampling strategies that reduce redundant reasoning paths. Dynamic voting, for instance, reduces inference cost by terminating multi-sampled reasoning once sufficient agreement is reached across sampled chains (Xue et al., 2023). More recent adaptive self-consistency methods further incorporate reasoning-path quality or response-level reliability signals to decide when additional samples are needed (Wan et al., 2025; Kim et al.,

2026). Broader test-time scaling studies also show that the amount of inference compute should be allocated adaptively rather than uniformly across all instances (Wu et al., 2025; Zhang et al., 2025a; Alomrani et al., 2025). Despite these efficiency gains, most methods still rely on observing multiple sampled trajectories or response-level agreement before deciding to stop, which incurs nontrivial computational overhead (Xue et al., 2023; Wan et al., 2025; Kim et al., 2026). As a result, the stopping decision may vary across runs and requires generating multiple responses, rather than relying on a single reasoning trajectory (Kadavath et al., 2022; Yona et al., 2024).

In this work, we propose a confidence-aware selective sampling framework that evaluates a completed single CoT reasoning trajectory to control inference-time reasoning cost. Specifically, our framework estimates the model's confidence and analyzes linguistic patterns of the initial reasoning trajectory, then adaptively determines whether to accept the answer deemed likely correct or trigger additional sampling for one assessed as likely wrong. We evaluate the proposed framework on five popular LLMs: GPT-OSS 20B (OpenAI, 2025), Llama 3.1 8B Instruct (Llama Team, AI @ Meta, 2024), Qwen 2.5 7B (Qwen Team, 2025), and Qwen 3 14B/32B (Yang et al., 2025). The experiments are conducted across four multiple-choice benchmarks: MedQA (Jin et al., 2021), MathQA (Amini et al., 2019), MedMCQA (Pal et al., 2022), and MMLU (Hendrycks et al., 2020). The results demonstrate that the proposed framework substantially improves inference efficiency, yielding major token savings, while preserving strong reasoning accuracy. In summary, our contributions are threefold:

- We introduce a confidence-aware selective sampling framework that analyzes a single completed CoT to determine whether additional multi-path reasoning is necessary, avoiding unnecessary sampling and computational overhead.
- We define an attention-based recurrent neural network (RNN) decision model that leverages sentence-level numeric and linguistic features to capture temporal reasoning dynamics and assess the reasoning reliability.
- We demonstrate the generalizability and robustness of the proposed approach through extensive evaluation across multiple LLMs and benchmarks, and provide ablation evidence that numeric and linguistic features provide complementary signals.

2 Related Work

2.1 Reasoning in Large Language Models

LLMs have demonstrated substantial improvements in reasoning through methods that encourage structured intermediate steps. CoT prompting enables models to generate interpretable step-by-step reasoning traces and improves accuracy across complex reasoning tasks (Wei et al., 2022b; Kojima et al., 2022; Sprague et al., 2025). To enhance reliability and mitigate stochastic failures, self-consistency samples multiple reasoning paths and selects the answer that is most frequently generated (Wang et al., 2023). Other prominent techniques include question decomposition and planning-based frameworks, such as Least-to-Most prompting and Tree-of-Thoughts, which break complex queries into subtasks or explore alternative reasoning trajectories to strengthen coherence and coverage (Zhou et al., 2022; Yao et al., 2023a). Recent advances have further expanded these capabilities through test-time compute scaling, where models allocate additional inference computation to improve reasoning performance (Snell et al., 2025; Bi et al., 2025; Muennighoff et al., 2025; Zhang et al., 2025a). Self-refinement methods enable models to iteratively critique and improve their own outputs without additional training (Madaan et al., 2023), while process reward models provide fine-grained step-level supervision to guide reasoning and correct intermediate errors (Setlur et al., 2025; Zhang et al., 2025b; Khalifa et al., 2025). Additionally, tool-augmented approaches extend intrinsic reasoning capabilities by integrating external retrieval or code execution modules, facilitating grounded or verifiable computation, as exemplified by ReAct, Toolformer, and Program-of-Thoughts prompting (Yao et al., 2023b; Schick et al., 2023; Chen et al., 2023). While these methods often improve reasoning performance, they also introduce additional inference overhead through longer generations, tool calls, or multiple reasoning paths, motivating methods that preserve reasoning quality while reducing unnecessary computation (Samsi et al., 2023; Fernandez et al., 2025; Wu et al., 2025).

2.2 Uncertainty Estimation in Reasoning

Uncertainty estimation has become an essential diagnostic tool for assessing and improving the quality of reasoning in LLMs. Prior work has examined how model confidence, often derived from token probabilities, correlates with correctness and can reveal whether a model can distinguish reliable answers from unreliable ones (Kadavath et al., 2022). Other studies have investigated whether LLMs can faithfully verbalize their intrinsic uncertainty or learn to express uncertainty through self-training (Yona et al., 2024; Liu et al., 2024). Beyond single-path diagnostics, semantic-based approaches measure the variability across semantically equivalent generations to estimate epistemic uncertainty and detect hallucinations (Kuhn et al., 2023; Baan et al., 2023; Farquhar et al., 2024; Huang et al., 2025). More recent work studies uncertainty directly over reasoning traces, showing that CoT can improve response-wise uncertainty quantification and that internal reasoning states can support efficient test-time reasoning assessment (Zhang and Zhang, 2025; Ni et al., 2025). Confidence-signal enhanced reasoning assigns confidence scores to intermediate reasoning steps and uses them to weight and aggregate multiple CoT paths, improving robustness over vanilla self-consistency by explicitly leveraging uncertainty structure (Razghandi et al., 2025). Unlike prior work that primarily uses uncertainty for calibration, hallucination detection, or reweighting multiple completed reasoning paths, our approach uses uncertainty-derived features from a single trajectory to determine whether further sampling is necessary, thereby reducing the cost of multi-path reasoning.

2.3 Adaptive and Early-Exit Reasoning

To mitigate the high computational burden associated with extended reasoning traces, adaptive test-time computation aims to allocate inference resources according to question difficulty rather than applying the same computation budget to every question (Snell et al., 2025; Wu et al., 2025; Alomrani et al., 2025). In multi-path reasoning, dynamic voting mitigates inconsistency by aggregating predictions from multiple reasoning trajectories and terminating sampling once sufficient agreement is observed (Xue et al., 2023). More recent adaptive self-consistency methods go beyond count-based voting by incorporating rationale quality or response-level reliability signals to determine whether additional samples are needed (Wan et al., 2025; Kim et al., 2026). However, these strategies often require sampling multiple complete reasoning chains or observing response-level agreement before deciding to stop, which remains costly in terms of token usage and computational overhead.

Another line of work trains a decision model to learn dynamic stopping or computation-allocation policies. Confident adaptive language modeling introduces token-level early exits inside the Transformer and calibrates their confidence so that the final sequence quality remains within a user-specified tolerance of full computation (Schuster et al., 2022). Earlier adaptive computation methods similarly learn how much internal computation to use for each input, either by varying the number of recurrent update steps or by exiting at different network depths (Graves, 2016; Teerapittayanon et al., 2016; Xin et al., 2020). Recent latent-reasoning approaches train models to decide when to stop internal reasoning, often using reinforcement learning or latent-state objectives (Hao et al., 2024; Ning et al., 2025). Related compute-allocation work formulates test-time computation as a constrained optimization problem and trains lightweight policies to allocate additional compute only to difficult instances (Zhai et al., 2026). However, these stopping or compute-allocation policies are often tied to architecture-specific internal representations, latent-state training objectives, or model-specific calibration, which may limit transfer across LLM backbones, prompt formats, and benchmark domains.

3 Method

3.1 Sentence-Level Per-Choice Prediction from Logits

To monitor the reasoning dynamics during CoT generation, we derive sentence-level answer-choice probabilities from token-level logits. Given a question q with K answer choices $\{A_i\}_{i=1}^{K}$, an LLM generates a CoT reasoning trace r token by token. We segment the generated reasoning trace into sentences $\{s_n\}_{n=1}^{N}$ and estimate the model's belief over answer choices after each sentence.

Let $r_{1:n}$ denote the reasoning prefix up to the n-th sentence, i.e., $r_{1:n} = s_1 \oplus \cdots \oplus s_n$. To estimate the likelihood of each option A_i given this prefix,

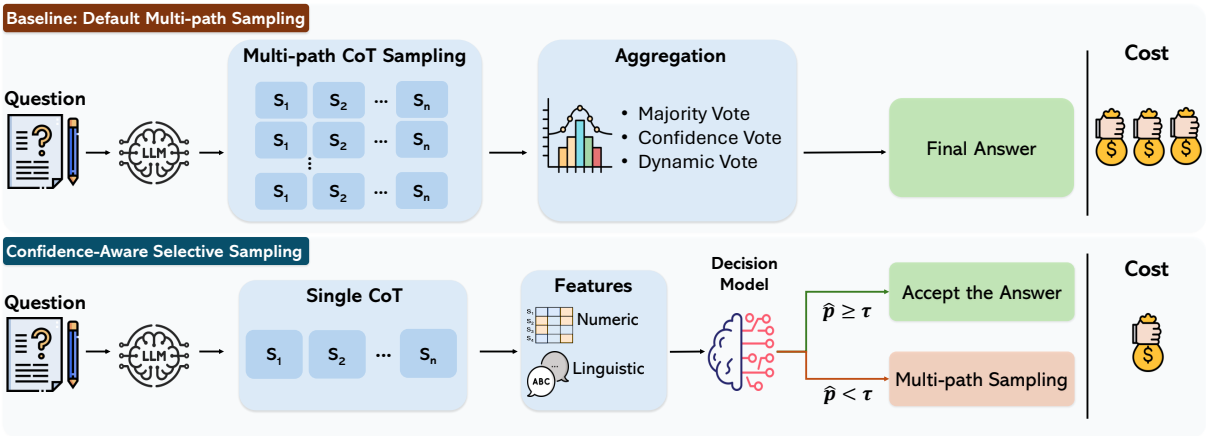

Figure 1: **Overview of the framework.** For each question, an LLM generates a reasoning trajectory, from which numeric and linguistic features are extracted. Our decision model then analyzes this sequence of features and estimates the probability $\hat{p}$ that the initial answer will be correct or wrong. A confidence threshold τ determines whether additional multi-path reasoning is necessary. Unlike the default multi-path sampling method, our confidence-aware selective sampling reduces inference cost.

we compute the conditional log-probability score:

$$\alpha_{n,i} = \frac{1}{|A_i|} \sum_{j=1}^{|A_i|} \log p(a_{ij} \mid q, r_{1:n}, a_{i,<j}), \quad (1)$$

where a_{ij} is the j-th token of answer option A_i. We use length normalization to reduce bias toward shorter answer options. We then normalize these scores across all options:

$$p_n(A_i) = \frac{\exp(\alpha_{n,i})}{\sum_{k=1}^{K} \exp(\alpha_{n,k})}. \quad (2)$$

This yields a probability vector $\mathbf{p}_n = [p_n(A_1), p_n(A_2), \ldots, p_n(A_K)]$ for each s_n, representing the model's evolving beliefs over answer choices. These per-sentence trajectories are stored for later analysis and used to construct the feature trajectory for the selective-sampling stage.

3.2 Decision Model

Given a question, the LLM produces a completed CoT trajectory and a final answer. A GRU-based decision model estimates the probability that the final answer is correct based on the completed trajectory. The decision model outputs a scalar probability $\hat{p} \in [0, 1]$ and applies a routing threshold τ:

$$\begin{cases} \text{likely-correct}, & \text{if } \hat{p} \geq \tau, \\ \text{likely-wrong}, & \text{otherwise.} \end{cases} \quad (3)$$

Predictions in the likely-correct group accept the final answer, while predictions in the likely-wrong group are routed to the multi-path aggregation module, instantiated as dynamic voting in our experiments.

3.2.1 Feature Extraction

We extract a sentence-level trajectory from the completed CoT. The CoT text is segmented into N sentences $\{s_n\}_{n=1}^{N}$ using punctuation and newline boundaries. For each sentence s_n, we retain a scalar confidence signal $p_n \in (0, 1)$ derived from the answer-choice distribution, an uncertainty signal H_n computed from token-level generation entropy, and the prefix length L_n, defined as the cumulative number of word tokens up to sentence n. The resulting trajectory is

$$\mathcal{Z} = \{(p_n, H_n, L_n, s_n)\}_{n=1}^{N}. \quad (4)$$

We then construct $\mathbf{X} \in \mathbb{R}^{N \times D}$ by concatenating trajectory-based numeric features with sentence-level linguistic features, yielding $D = 32$ features per sentence. Additional preprocessing and implementation details are provided in Appendix A.

Numeric trajectory features. Table 1 summarizes the numeric features computed from (p_n, H_n, L_n). These features capture local confidence and uncertainty, prefix-level information, temporal changes, and short-term stability along the completed reasoning trajectory.

Linguistic features. In addition to numeric signals, we extract lightweight linguistic features from the sentence text s_n and its relation to the prompt (question and answer options). These features

Table 1: Numeric trajectory features computed at sentence position s_n.

Idx	Feature	Description
1	p_n	Scalar confidence signal derived from the answer-choice distribution after sentence s_n.
2	H_n	Mean next-token generation entropy over the tokens in sentence s_n.
3	$p_n/\log(1+L_n)$	Confidence normalized by the cumulative prefix length.
4	Δp_n	$p_n - p_{n-1}$
5	ΔH_n	$H_n - H_{n-1}$.
6	$\sigma_n^{(k)}$	Standard deviation over the last up to k confidence values.
7	$r_n^{(k)}$	Range over the last up to k confidence values.
8	L_n	Cumulative number of word tokens up to and including sentence s_n.
9	$\mathrm{EMA}(p)_n$	Exponential moving average of the confidence signal up to sentence n.
10	$\Delta\mathrm{EMA}(p)_n$	$\mathrm{EMA}(p)_n - \mathrm{EMA}(p)_{n-1}$.
11	$z(p_n)$	Trajectory-level z-score of the confidence signal.
12	$z(\mathrm{EMA}(p)_n)$	Trajectory-level z-score of the EMA confidence signal.

capture text statistics, reasoning style, and topical alignment, without using any text embeddings. Table 2 summarizes the linguistic features.

3.2.2 Model Architecture

We instantiate the trainable component as a GRU-based decision model that operates on the complete feature trajectory $\mathbf{X} \in \mathbb{R}^{N\times D}$ extracted from a completed CoT. The model consists of the following blocks.

Feature gating block. We employ a trajectory-conditioned feature gating block to adaptively reweight the input feature sequence $\mathbf{X}$. We first compute a trajectory-level summary by mean pooling over valid sentence positions:

$$\mathbf{s} = \frac{\sum_{n=1}^{N} m_n \mathbf{X}_n}{\sum_{n=1}^{N} m_n},$$

where $m_n \in \{0,1\}$ indicates whether sentence position n is valid. The summary vector is passed through a lightweight two-layer multilayer perceptron (MLP) with ReLU and sigmoid activations to produce channel-wise gating weights $\mathbf{g} \in (0,1)^D$. The gated feature sequence is then computed as

$$\tilde{\mathbf{X}}_n = \mathbf{X}_n \odot \mathbf{g}.$$

The same gating vector is applied to all sentence positions within a trajectory, allowing the decision model to emphasize or suppress feature dimensions according to the global characteristics of the completed reasoning trajectory.

Table 2: Sentence-level linguistic features extracted from each sentence s_n.

Category	Features
Sentence statistics	token count; character count; average token length of s_n
Lexical composition	stop word ratio; content word ratio in s_n
Punctuation	comma, period, question, and exclamation counts; punctuation density in s_n
Character patterns	digit ratio; uppercase ratio in s_n
Prompt overlap	overlap counts between s_n and the question/options; normalized overlap ratios
Reasoning markers	hedge word count; certainty word count; logical connector count in s_n
Position	normalized position of s_n within the CoT

GRU encoder block. The gated sequence $\tilde{\mathbf{X}}$ is encoded by a single-layer unidirectional gated recurrent unit (GRU) with hidden size $d_h = 64$. The GRU produces contextual representations

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N] \in \mathbb{R}^{N\times d_h},$$

which capture temporal dependencies in the feature dynamics across the reasoning trajectory. For variable-length trajectories, padded sentence positions are excluded from recurrent encoding using packed sequence representations.

Multi-head self-attention block. The sequence of contextual representations $\mathbf{U}$ is further processed by a lightweight 4-head self-attention block. The block follows a pre-normalization transformer-style structure consisting of LayerNorm, multi-head self-attention with residual connections, and a position-wise feed-forward sublayer. This component allows each sentence representation to incorporate information from other reasoning steps, enhancing the modeling of long-range dependencies across the trajectory.

Position-wise projection head. Each contextual representation is mapped to a scalar logit by a position-wise MLP. The head consists of LayerNorm followed by a two-layer MLP with hidden dimension 32 and ReLU activation, producing per-sentence logits $\{\ell_n\}_{n=1}^{N}$ and probabilities

$$q_n = \mathrm{sigmoid}(\ell_n).$$

The final-position probability $\hat{p} = q_N$ is used as the trajectory-level score for detecting whether the final answer is likely to be correct.

3.2.3 Threshold Calibration

We determine dataset-specific confidence thresholds using validation-set profiling. Specifically, we sweep the confidence threshold $\tau \in [0, 1]$ and evaluate both prediction accuracy and token reduction on the validation set. For each dataset, we first identify the maximum validation accuracy achieved across all candidate thresholds. We then select the threshold that yields the largest token reduction while keeping the validation accuracy within 0.5 percentage points of this maximum. This procedure prioritizes efficiency gains while preventing material degradation in predictive performance.

4 Experiments

Datasets We evaluate our method on four multiple-choice question answering datasets that cover both medical and general domains.

- **MedQA** (Jin et al., 2021): A dataset derived from medical licensing exam questions, containing expert-authored problems across multiple clinical topics. It assesses medical reasoning and factual recall in clinical question-answering settings.
- **MedMCQA** (Pal et al., 2022): A challenging corpus of medical questions collected from Indian medical entrance exams. It contains over 200,000 questions and tests domain comprehension and knowledge integration.
- **MathQA** (Amini et al., 2019): A dataset of arithmetic and commonsense mathematical problems requiring multi-sentence symbolic reasoning. We use the multiple-choice version standardized by prior reasoning benchmarks.
- **MMLU** (Hendrycks et al., 2020): A broad evaluation benchmark spanning 57 subjects, ranging from science, technology, engineering, and mathematics (STEM) to humanities, designed to assess general knowledge and reasoning in large language models.

Models: We evaluate our framework on a diverse set of open-source LLMs spanning multiple architectures and parameter scales. The experiments are conducted using GPT-OSS 20B (OpenAI, 2025), Meta Llama 3.1 8B Instruct (Llama Team, AI @ Meta, 2024), Qwen 2.5 7B (Qwen Team, 2025), and Qwen 3 14B/32B (Yang et al., 2025). All of the models support autoregressive CoT generation with probability outputs, enabling a unified routing pipeline across different reasoning engines. We report evaluation results for GPT-OSS 20B in the main experiments and defer results for the other models to Appendix B and Appendix C.

Baselines: We compare our approach against representative multi-path reasoning baselines and confidence-aware variants:

- **Self-Consistency (SC):** Aggregates multiple sampled reasoning paths and selects the most frequent answer to enhance reasoning accuracy (Wang et al., 2023).
- **Confidence Enhanced Reasoning (CER):** Aggregates multiple reasoning paths by weighting them according to confidence estimated from reasoning steps (Razghandi et al., 2025).
- **Dynamic Voting (DV):** Adaptively samples reasoning paths and stops once a voting consensus is reached, reducing the computation relative to self-consistency (Xue et al., 2023).

Experiment Setup: We train and evaluate all decision models using the same data splits on a single NVIDIA H100 GPU. For each LLM, the decision model is trained only on the MedQA training split. For each target dataset, we use 500 validation examples solely for threshold calibration and 1,000 held-out examples for testing. No decision-model parameters are updated outside MedQA. We sample 10 CoTs for multi-path experiments with a temperature of 1.0. Full data preprocessing and training details are provided in Appendix A.

5 Results

5.1 Confidence Threshold Calibration

Figure 2 illustrates the accuracy (blue) versus token reduction (orange) tradeoffs across confidence threshold choices. The selected optimal thresholds correspond to the maximum token reduction under the $\leq 0.5\%$ accuracy-drop constraint. Based on this profiling, we selected: $\tau_{\text{MedQA}} = 0.65$, $\tau_{\text{MathQA}} = 0.20$, $\tau_{\text{MedMCQA}} = 0.60$, and $\tau_{\text{MMLU}} = 0.80$. The calibration results for other LLMs are shown in Appendix B Figures 4–7.

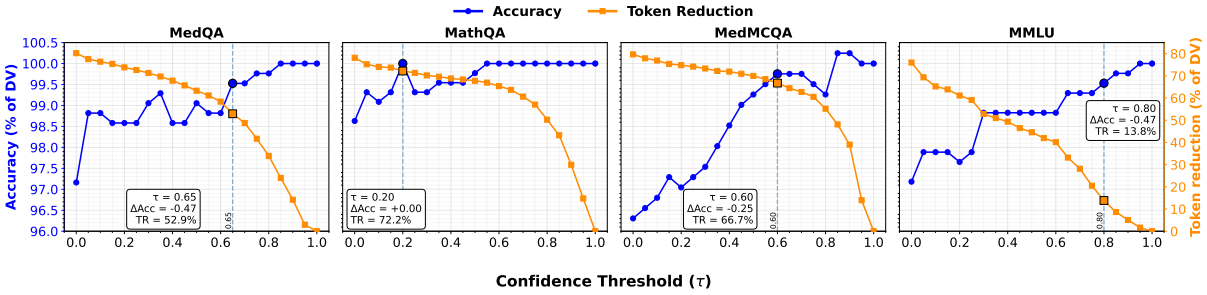

Figure 2: **Confidence Threshold Calibration.** Accuracy (blue, % of DV) and token reduction (orange, % vs. DV) versus confidence threshold τ on MedQA, MathQA, MedMCQA, and MMLU. $\tau = 1.0$ is the DV baseline (100% accuracy, 0% reduction).

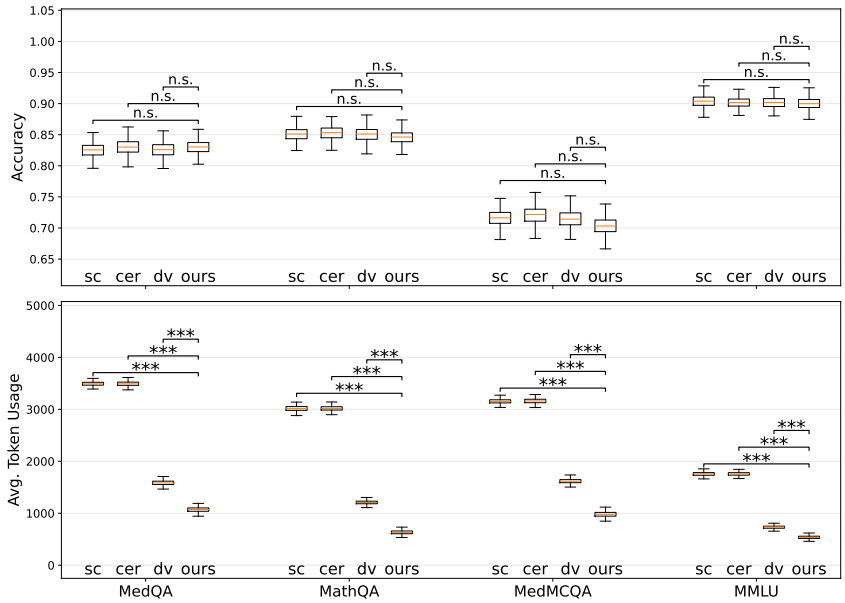

Figure 3: **Distribution of accuracy and token usage across datasets.** Boxplots show the performance of SC, CER, DV, and Ours on MedQA, MathQA, MedMCQA, and MMLU using GPT-OSS 20B. The top panel reports accuracy, while the bottom panel reports average token usage. Statistical significance is evaluated using paired bootstrap with 2,000 resamples; $n.s.$ indicates not significant and * indicates $p < 0.05$; ** indicates $p < 0.01$; *** indicates $p < 0.001$. Higher accuracy is better, whereas lower token usage indicates greater efficiency.

5.2 Accuracy and Efficiency

Figure 3 presents a comparison of accuracy and token usage between our proposed model and the baselines for GPT-OSS 20B. Across all four datasets, the differences in accuracy between our method and the multi-path baselines were not statistically significant ($n.s.$) under bootstrap testing. In MedQA, MathQA, MedMCQA and MMLU, our performance remains within a narrow margin of SC, CER, and DV, indicating that selective multi-path reasoning preserves the final prediction accuracy.

In contrast, the token usage reductions of our framework are substantial and statistically significant ($p < 0.05$) when comparing to SC, CER, and DV on every dataset. Relative to SC and CER, our framework reduces token consumption by approximately 69–79% across tasks. Even compared to DV, token consumption is reduced by 27–48%. The tradeoff results for remaining LLMs are shown in Appendix C, Figures 8–11, which show the same finding: our solution significantly reduces token usage while accuracy remains close to multi-path baselines in most settings.

For each LLM, we train the decision model on MedQA CoT traces and evaluate it on MedQA, as well as under calibration-only transfer to MathQA, MedMCQA, and MMLU. For all datasets, only the routing threshold is calibrated on the corresponding validation set, while the decision-model parameters are kept fixed. Since different LLMs exhibit

Table 3: **Ablation of Feature Gating (FG) block and Multi-Head Self-Attention (MHSA) block**. The variant without FG and MHSA serves as the reference. Best results are in bold. ⇑: higher is better; ⇓: lower is better.

Dataset	Variant	Accuracy⇑	Token Usage⇓
MedQA	FG ✗, MHSA ✗	0.822	1230
	FG ✓, MHSA ✗	0.825(↑0.003)	1185(↓3.66%)
	FG ✗, MHSA ✓	0.828(↑0.006)	1206(↓1.95%)
	FG ✓, MHSA ✓	**0.830(↑0.008)**	**1026(↓16.59%)**
MathQA	FG ✗, MHSA ✗	0.847	704
	FG ✓, MHSA ✗	**0.848(↑0.001)**	689(↓2.13%)
	FG ✗, MHSA ✓	0.847(↑0.000)	692(↓1.70%)
	FG ✓, MHSA ✓	0.847(↑0.000)	**629(↓10.65%)**
MedMCQA	FG ✗, MHSA ✗	**0.704**	1164
	FG ✓, MHSA ✗	0.701(↓0.003)	1100(↓5.50%)
	FG ✗, MHSA ✓	0.701(↓0.003)	1133(↓2.66%)
	FG ✓, MHSA ✓	**0.704(↑0.000)**	**983(↓15.55%)**
MMLU	FG ✗, MHSA ✗	**0.900**	691
	FG ✓, MHSA ✗	**0.900(↑0.000)**	576(↓16.64%)
	FG ✗, MHSA ✓	0.898(↓0.002)	570(↓17.51%)
	FG ✓, MHSA ✓	**0.900(↑0.000)**	**536(↓22.43%)**
Average Performance Across All Datasets			
	FG ✗, MHSA ✗	0.818	947
	FG ✓, MHSA ✗	0.819(↑0.001)	888(↓6.23%)
	FG ✗, MHSA ✓	0.819(↑0.001)	900(↓4.96%)
	FG ✓, MHSA ✓	**0.820(↑0.002)**	**794(↓16.16%)**

distinct reasoning dynamics, we train a separate decision model for each LLM while keeping the feature extraction pipeline and model architecture fixed.

As shown for GPT-OSS 20B in Figure 3, our method achieves accuracy comparable to multi-path reasoning while substantially reducing token usage. Similar trends are observed for other models, as shown in Appendix C, Figure 8–11. These suggest that the decision model can learn a trajectory-based decision rule that transfers across domains.

We further observe that larger models, such as GPT-OSS 20B, tend to produce more separable confidence and uncertainty trajectories between correct and incorrect reasoning paths. This clearer separation allows the decision model to accept initial reasoning more frequently, leading to larger efficiency gains. In contrast, smaller models, such as Llama 3.1 8B Instruct, produce noisier trajectory signals, which limits token savings but still preserves the overall accuracy–efficiency advantage of the proposed approach.

5.3 Ablation Study

We conduct two ablation studies to analyze the contributions of key architectural design choices and input feature types.

We first evaluate the contribution of the feature gating (FG) block and the multi-head self-attention (MHSA) block (Table 3). Using either module alone yields only limited or inconsistent improvements over the variant with both blocks disabled. In contrast, enabling both modules consistently achieves the best accuracy–efficiency trade-off.

Table 4: **Feature Ablation study.** We compare training with numeric (reference), linguistic, and both feature types combined. ⇑: higher is better; ⇓: lower is better.

Dataset	Variant	Accuracy⇑	Token Usage⇓
MedQA	Numeric only	0.825	1148
	Linguistic only	0.825(↑0.000)	1222(↑6.45%)
	Numeric + Linguistic	**0.830(↑0.005)**	**1070(↓6.79%)**
MathQA	Numeric only	0.845	736
	Linguistic only	0.847(↑0.002)	681(↓7.47%)
	Numeric + Linguistic	**0.847(↑0.002)**	**630(↓14.40%)**
MedMCQA	Numeric only	0.704	1069
	Linguistic only	0.702(↓0.002)	1144(↑7.02%)
	Numeric + Linguistic	**0.704(↑0.000)**	**983(↓8.04%)**
MMLU	Numeric only	0.898	583
	Linguistic only	0.899(↑0.001)	595(↑2.06%)
	Numeric + Linguistic	**0.900(↑0.002)**	**536(↓8.06%)**
Average Performance Across All Datasets			
	Numeric only	0.818	884
	Linguistic only	0.818(↑0.000)	911(↑3.05%)
	Numeric + Linguistic	**0.820(↑0.002)**	**805(↓8.94%)**

Similarly, to determine the optimal feature set, we compare numeric-only, linguistic-only, and combined features (Table 4). Combining the two yields the best performance, improving both accuracy and token efficiency.

6 Discussion and Conclusion

In this paper, we proposed a confidence-aware selective sampling framework that improves the accuracy–efficiency trade-off of LLM reasoning. Rather than applying costly multi-path reasoning to every query, our framework analyzes the sentence-level dynamics of a completed reasoning trajectory and predicts whether additional reasoning is needed. Empirically, we find that sentence-level features, including probability trends, entropy dynamics, and convergence patterns, provide effective signals of reasoning reliability. We further show that these signals transfer across datasets and domains: a decision model trained on MedQA traces transfers to MathQA, MedMCQA, and MMLU with only threshold calibration. Finally, trajectories from larger LLMs exhibit clearer separability between reliable and unreliable reasoning paths, suggesting that improving trajectory quality and uncertainty separability remains an important direction for smaller LLMs.

7 Limitations

Our work has several limitations that can serve as a basis for further investigation. First, the framework is evaluated primarily on multiple-choice scientific and medical question answering tasks, where reasoning trajectories exhibit relatively structured patterns. Its effectiveness for open-ended generation, long-form reasoning, or dialogue settings can be explored in future research. Second, the method analyzes completed reasoning trajectories and therefore cannot be directly used for online early-exit decisions during generation. Adapting the framework to operate causally in intermediate sentences would require additional modeling and calibration in future work. Third, monitoring numeric and linguistic features depends on access to internal signals and is mainly validated on open-source LLMs. Future work could improve the proposed method by focusing solely on the reasoning trajectory to evaluate proprietary LLMs.

References

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, Peng Cheng, Yunzhou Wang, Pengyi Liao, Hanrui Huang, Bin Wang, Jianye Hao, and Mark Coates. 2025. Reasoning on a budget: A survey of adaptive and controllable test-time compute in LLMs. *arXiv preprint arXiv:2507.02076*.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 2357–2367.

Joris Baan, Nico Daheim, Evgenia Ilia, Dennis Ulmer, Haau-Sing Li, Raquel Fernández, Barbara Plank, Rico Sennrich, Chrysoula Zerva, and Wilker Aziz. 2023. Uncertainty in natural language generation: From theory to applications. *arXiv preprint arXiv:2307.15703*.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2025. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 4253–4267. PMLR.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

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

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. 2025. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32556–32569.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):42:1–42:55.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *Preprint*, arXiv:2504.16828.

Junseok Kim, Nakyeong Yang, Kyungmin Min, and Kyomin Jung. 2026. Reliability-aware adaptive self-consistency for efficient sampling in llm reasoning. *arXiv preprint arXiv:2601.02970*.

Takeshi Kojima, Shixiang Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations (ICLR)*. ArXiv:2302.09664.

Shudong Liu, Zhaocong Li, Xuebo Liu, Runzhe Zhan, Derek F. Wong, Lidia S. Chao, and Min Zhang. 2024. Can LLMs learn uncertainty on their own? expressing uncertainty effectively in a self-training manner. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21635–21645. Association for Computational Linguistics.

Llama Team, AI @ Meta. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20275–20321, Suzhou, China. Association for Computational Linguistics.

Jingwei Ni, Ekaterina Fadeeva, Tianyi Wu, Mubashara Akhtar, Jiaheng Zhang, Elliott Ash, Markus Leippold, Timothy Baldwin, See-Kiong Ng, Artem Shelmanov, and Mrinmaya Sachan. 2025. ReProbe: Efficient test-time scaling of multi-step reasoning by probing internal states of large language models. *Preprint*, arXiv:2511.06209.

Alex Ning, Yen-Ling Kuo, and Gabe Gomes. 2025. Learning when to stop: Adaptive latent reasoning via reinforcement learning. *arXiv preprint arXiv:2511.21581*.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Jingyu Peng, Maolin Wang, Xiangyu Zhao, Kai Zhang, Wanyu Wang, Pengyue Jia, Qidong Liu, Ruocheng Guo, and Qi Liu. 2025. Stepwise reasoning error disruption attack of llms. *Preprint*, arXiv:2412.11934.

Qwen Team. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Ali Razghandi, Seyed Mohammad Hadi Hosseini, and Mahdieh Soleymani Baghshah. 2025. Cer: Confidence enhanced reasoning in llms. *Preprint*, arXiv:2502.14634.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From words to watts: Benchmarking the energy costs of large language model inference. *arXiv preprint arXiv:2310.03003*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*. ArXiv:2207.07061.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*.

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

Peiyang Song, Pengrui Han, and Noah Goodman. 2026. Large language model reasoning failures. *arXiv preprint arXiv:2602.06176*.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *Preprint*, arXiv:2409.12183.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *International Conference on Pattern Recognition (ICPR)*, pages 2464–2469.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of*

the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 3613–3635, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35.

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

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251.

Mingfeng Xue, Dayiheng Liu, Wenqiang Lei, Xingzhang Ren, Baosong Yang, Jun Xie, Yidan Zhang, Dezhong Peng, and Jiancheng Lv. 2023. Dynamic voting for efficient reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3085–3104.

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

S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Gal Yona, Roee Aharoni, and Mor Geva. 2024. Can large language models faithfully express their intrinsic uncertainty in words? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7752–7764. Association for Computational Linguistics.

Zhiyuan Zhai, Bingcong Li, Bingnan Xiao, Ming Li, and Xin Wang. 2026. Adaptive test-time compute allocation for reasoning llms via constrained policy optimization. *arXiv preprint arXiv:2604.14853*.

Boxuan Zhang and Ruqi Zhang. 2025. CoT-UQ: Improving response-wise uncertainty quantification in LLMs with chain-of-thought. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26114–26133, Vienna, Austria. Association for Computational Linguistics.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. 2025a. What, how, where, and how well? a survey on test-time scaling in large language models. *arXiv preprint arXiv:2503.24235*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516.

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

A Implementation Details

CoT generation and answer extraction. For each question, we construct a chat-style prompt containing the question and answer options, and instruct the LLM to produce concise step-by-step reasoning followed by a final answer line in the format `Answer: X`, where `X` is one of the provided option letters. The CoT portion is separated from the final answer line using this marker. If the final answer marker is absent, we fall back to forced-choice scoring over the candidate answer letters using the completed CoT prefix.

Sentence-level probability and entropy extraction. For each sentence prefix $r_{1:n}$, we append an answer prompt and compute length-normalized conditional log-probabilities for all candidate answer options. The resulting scores are normalized with a softmax to obtain the answer-choice distribution. Token-level generation entropy is computed

from the next-token distribution during CoT generation and then averaged over the tokens aligned to each sentence span. Sentence spans are obtained using punctuation-based segmentation, and token-to-sentence alignment is performed using tokenizer offset mappings.

Trajectory preprocessing. For each trajectory, we keep only sentence positions where both the confidence signal and entropy signal are available and finite. The prefix length L_n is computed as the cumulative number of regex-based word tokens up to sentence n. Rolling statistics use a window of up to $k = 4$ recent sentence-level confidence values. The exponential moving average uses smoothing coefficient $\alpha = 0.8$. Trajectory-level z-scores are computed within each completed CoT trajectory.

Linguistic feature extraction. The linguistic feature set is rule-based and does not use text embeddings. Stop words are removed when computing content-word overlap with the question and answer options. Hedge, certainty, and logical-marker counts are computed using fixed lexicons. Hedge words include terms such as *maybe*, *might*, *could*, *possibly*, *perhaps*, *appears*, *seems*, *suggests*, *likely*, and *unlikely*. Certainty words include *therefore*, *thus*, *hence*, *certainly*, *clearly*, *definitely*, *must*, *always*, and *never*. Logical markers include *because*, *so*, *therefore*, *thus*, *hence*, *however*, *but*, *although*, *while*, and *whereas*.

Training objective. The decision model is trained as a binary classifier. The label is $y = 1$ if the initial CoT answer is correct and $y = 0$ otherwise. We use binary cross-entropy loss with masking over valid sentence positions for variable-length trajectories. To account for class imbalance, the positive class weight is set to the ratio between the number of negative and positive training examples.

Optimization and model selection. We train the decision model with AdamW using a learning rate of 2×10^{-3}, weight decay of 10^{-4}, batch size 64, and 20 training epochs. Gradients are clipped to a maximum norm of 1.0. We use random seed 1337 for Python, NumPy, and PyTorch. The checkpoint with the lowest validation loss is selected for evaluation.

Sequence batching and padding. CoT trajectories have variable numbers of sentences. During batching, trajectories are padded to the maximum sequence length within each batch, and padded sentence positions are masked during training and evaluation. No fixed sentence-level maximum length is imposed beyond the generation and preprocessing limits.

Reproducibility details. We use the tokenizer chat template when available and otherwise fall back to a plain role-based prompt. For multi-path baselines, we sample up to 10 reasoning trajectories with temperature 1.0 using the same prompt and answer-extraction procedure. For each trajectory, we extracted CoT, answer-choice probabilities, sentence-level entropy values, and number of generated tokens.

B Confidence Threshold Calibration for Additional LLMs

The calibrated confidence threshold for each LLM on the validation set is shown in Figures 4–7. The key observation is that the accuracy–efficiency trade-off is model dependent, but most calibration curves contain a usable region where token reduction increases while accuracy remains close to the dynamic-voting reference. This supports lightweight validation-set threshold calibration: for each dataset, we select the threshold that maximizes token reduction while keeping validation accuracy within 0.5 percentage points of the best validation accuracy.

C Accuracy–Efficiency Trade-off Results for Additional LLMs

For each LLM, the decision model is trained on MedQA traces and evaluated on MedQA, MathQA, MedMCQA, and MMLU. MedQA serves as the in-domain setting, while the remaining datasets assess calibration-only transfer. Figures 8–11 show that the main trend is consistent across backbones: selective sampling substantially reduces generated token usage while keeping accuracy close to multi-path reasoning baselines. The amount of token reduction varies by model and dataset, reflecting differences in the separability of confidence and entropy trajectories; noisier trajectories lead to more conservative routing, while clearer trajectories allow more single-path outputs to be accepted.

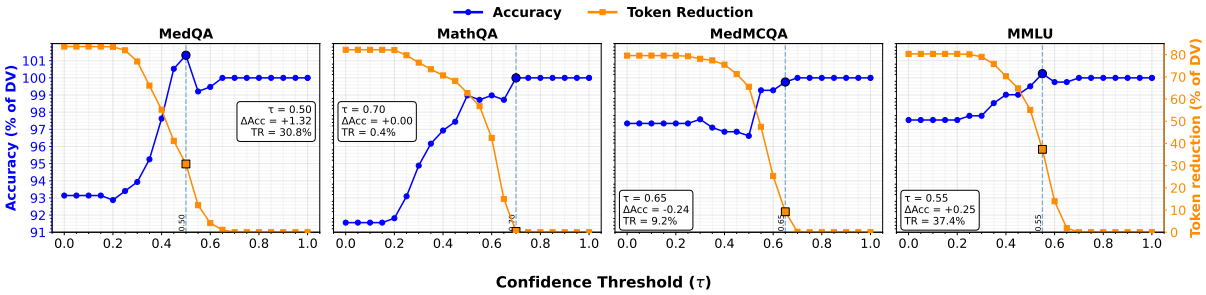

Figure 4: **Confidence Threshold Calibration for Llama 3.1 8B Instruct.** Accuracy (blue, % of DV) and token reduction (orange, % vs. DV) versus confidence threshold τ on MedQA, MathQA, MedMCQA, and MMLU. $\tau = 1.0$ is the DV baseline (100% accuracy, 0% reduction).

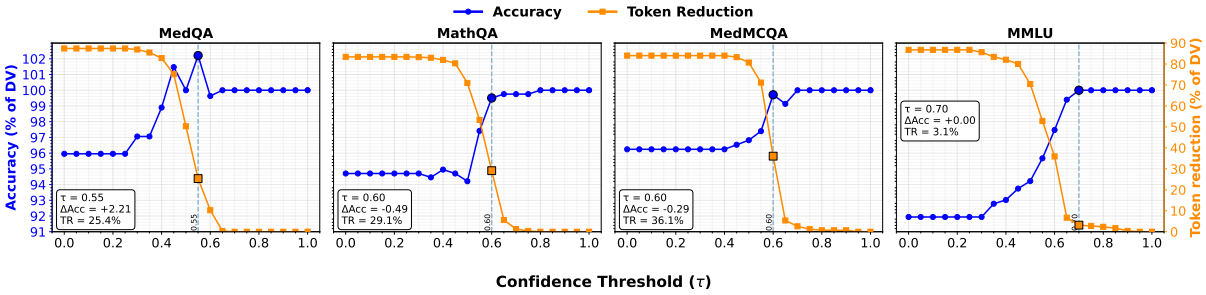

Figure 5: **Confidence Threshold Calibration for Qwen 2.5 7B.** Accuracy (blue, % of DV) and token reduction (orange, % vs. DV) versus confidence threshold τ on MedQA, MathQA, MedMCQA, and MMLU. $\tau = 1.0$ is the DV baseline (100% accuracy, 0% reduction).

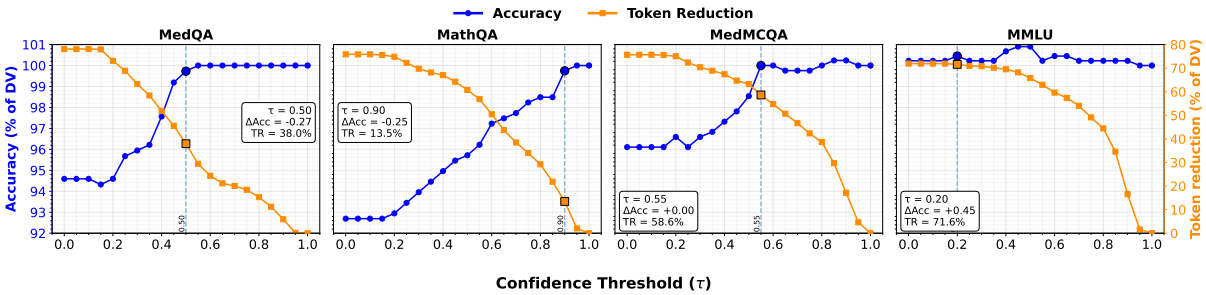

Figure 6: **Confidence Threshold Calibration for Qwen 3 14B.** Accuracy (blue, % of DV) and token reduction (orange, % vs. DV) versus confidence threshold τ on MedQA, MathQA, MedMCQA, and MMLU. $\tau = 1.0$ is the DV baseline (100% accuracy, 0% reduction).

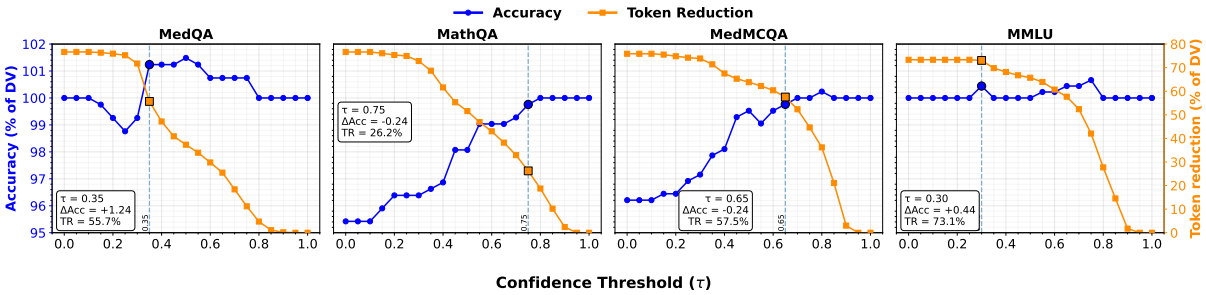

Figure 7: **Confidence Threshold Calibration for Qwen 3 32B.** Accuracy (blue, % of DV) and token reduction (orange, % vs. DV) versus confidence threshold τ on MedQA, MathQA, MedMCQA, and MMLU. $\tau = 1.0$ is the DV baseline (100% accuracy, 0% reduction).

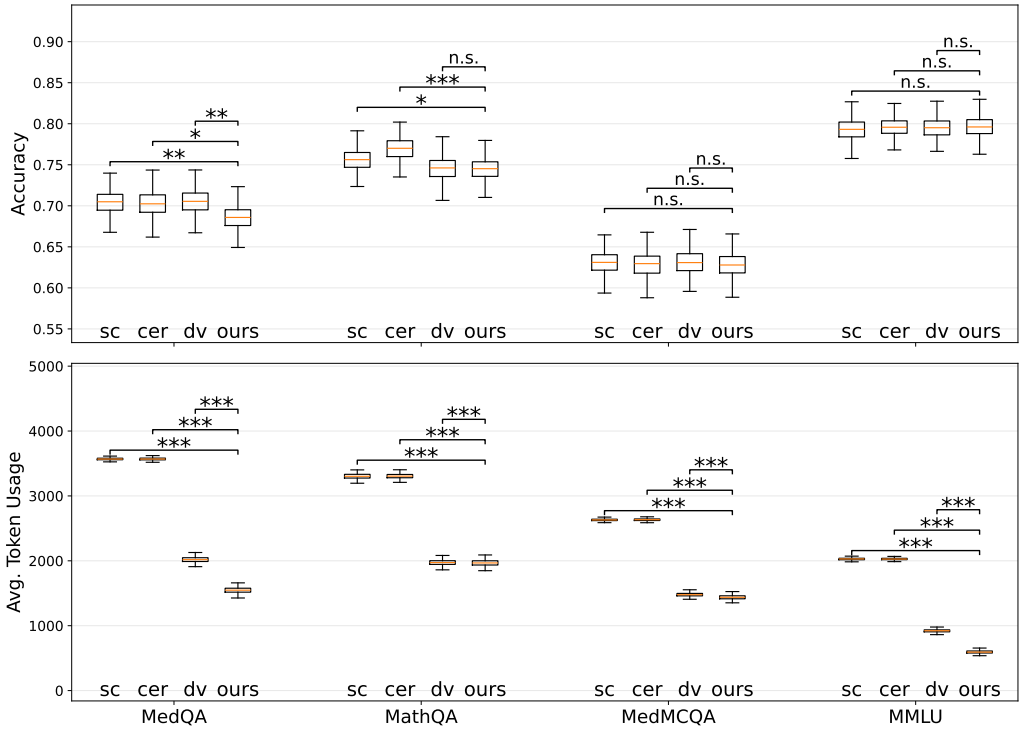

Figure 8: **Distribution of accuracy and token usage across datasets.** Boxplots show the performance of SC, CER, DV, and Ours on MedQA, MathQA, MedMCQA, and MMLU using Llama 3.1 8B Instruct. The top panel reports accuracy, while the bottom panel reports average token usage. Statistical significance is evaluated using paired bootstrap with 2,000 resamples; $n.s.$ indicates not significant and * indicates $p < 0.05$; ** indicates $p < 0.01$; *** indicates $p < 0.001$. Higher accuracy is better, whereas lower token usage indicates greater efficiency.

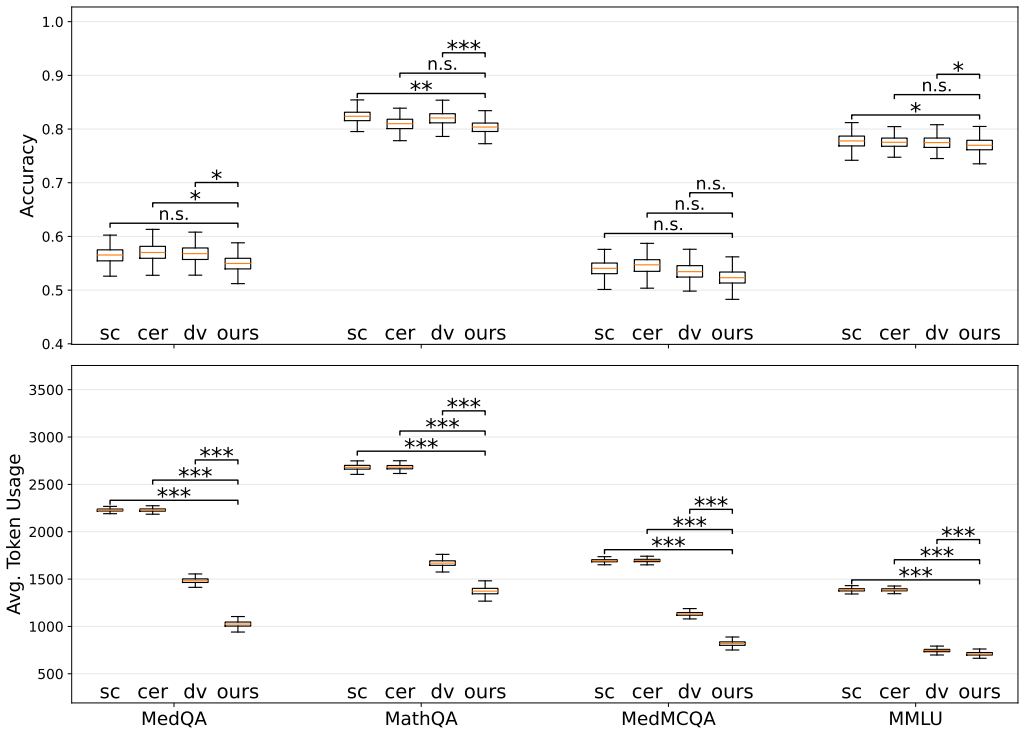

Figure 9: **Distribution of accuracy and token usage across datasets.** Boxplots show the performance of SC, CER, DV, and Ours on MedQA, MathQA, MedMCQA, and MMLU using Qwen 2.5 7B. The top panel reports accuracy, while the bottom panel reports average token usage. Statistical significance is evaluated using paired bootstrap with 2,000 resamples; $n.s.$ indicates not significant and * indicates $p < 0.05$; ** indicates $p < 0.01$; *** indicates $p < 0.001$. Higher accuracy is better, whereas lower token usage indicates greater efficiency.

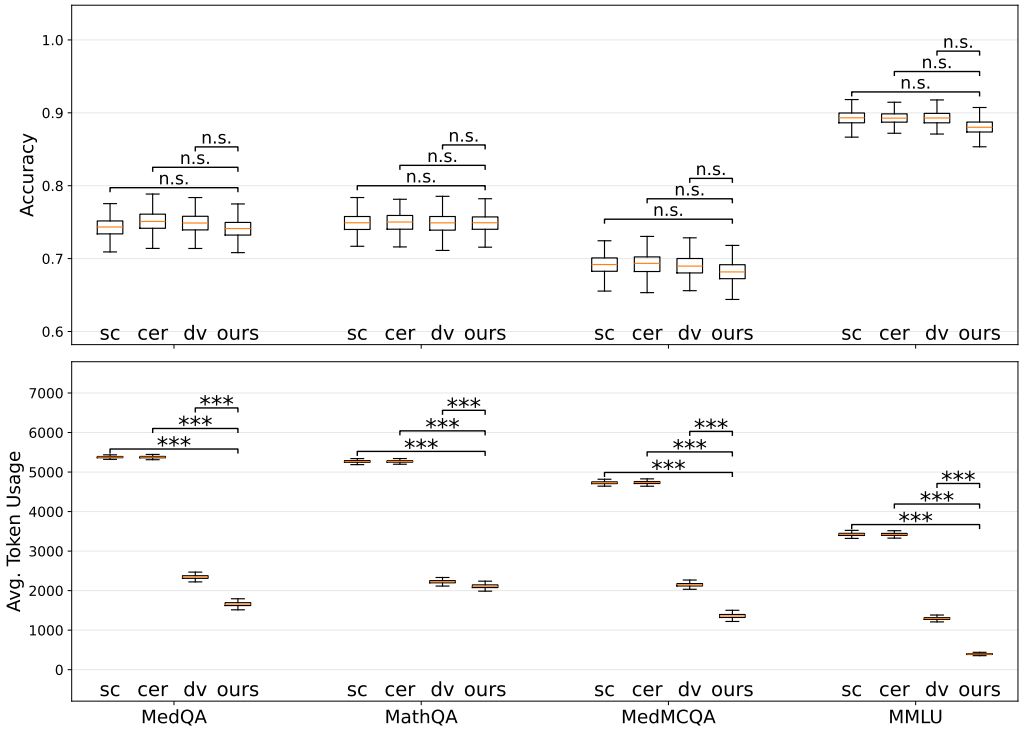

Figure 10: **Distribution of accuracy and token usage across datasets.** Boxplots show the performance of SC, CER, DV, and Ours on MedQA, MathQA, MedMCQA, and MMLU using Qwen 3 14B. The top panel reports accuracy, while the bottom panel reports average token usage. Statistical significance is evaluated using paired bootstrap with 2,000 resamples; $n.s.$ indicates not significant and * indicates $p < 0.05$; ** indicates $p < 0.01$; *** indicates $p < 0.001$. Higher accuracy is better, whereas lower token usage indicates greater efficiency.

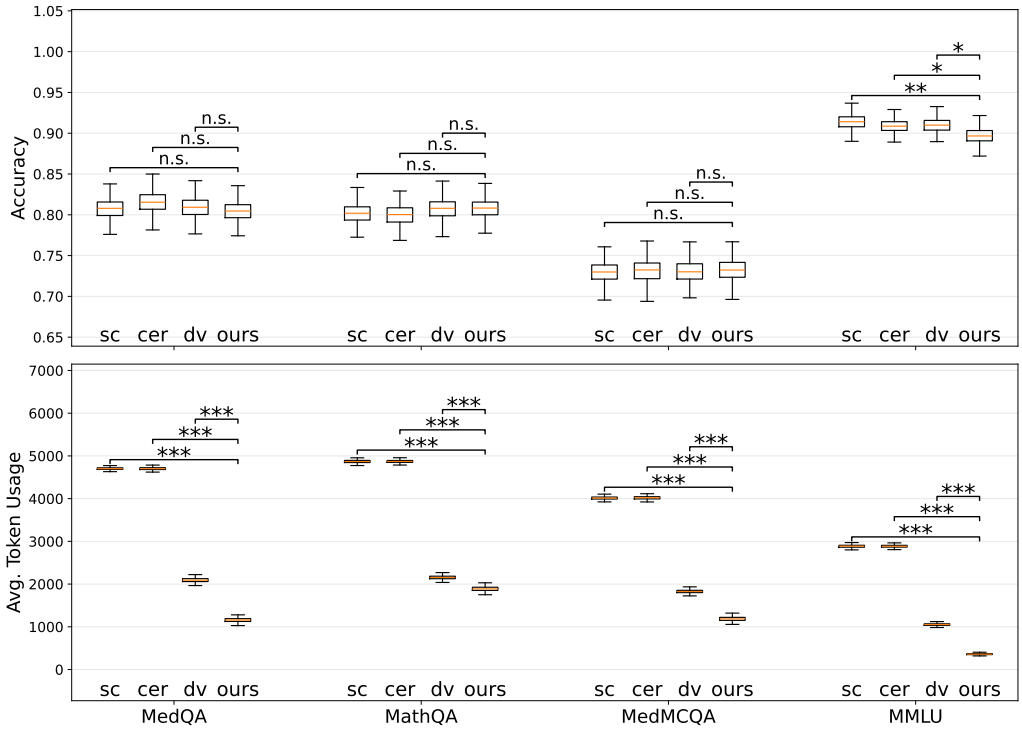

Figure 11: **Distribution of accuracy and token usage across datasets.** Boxplots show the performance of SC, CER, DV, and Ours on MedQA, MathQA, MedMCQA, and MMLU using Qwen 3 32B. The top panel reports accuracy, while the bottom panel reports average token usage. Statistical significance is evaluated using paired bootstrap with 2,000 resamples; $n.s.$ indicates not significant and * indicates $p < 0.05$; ** indicates $p < 0.01$; *** indicates $p < 0.001$. Higher accuracy is better, whereas lower token usage indicates greater efficiency.