InfoGuide: Information-Theoretic Modulation for Classifier-Free Guidance in Large Language Models

Anonymous ACL submission

Abstract

Classifier-Free Guidance (CFG) has emerged as a powerful test-time intervention for steering text-to-image generative models. However, applying static CFG to autoregressive Large Language Models (LLMs) for complex Chain-of-Thought (CoT) reasoning often severely degrades performance. In this paper, we reveal that uniform token-level guidance disrupts the delicate logical dependencies of CoT by distorting structural syntax, inducing severe entropy spikes that lead to trajectory divergence, and amplifying out-of-distribution tail-token hallucinations. To overcome these fundamental pathologies, we propose **InfoGuide**, an Information-Theoretic CFG framework designed specifically for reasoning LLMs. InfoGuide dynamically modulates guidance intensity at each decoding step by leveraging Jensen-Shannon Divergence to measure distributional shifts and Information Gain to quantify predictive uncertainty. Furthermore, we introduce robust structural safeguards: Unconditional Logit Clipping, Sequence Length Decay, and Confident Token Bypassing—to stabilize long reasoning chains and optimize inference efficiency. Extensive experiments across the Qwen model family on rigorous mathematical benchmarks (e.g., GSM8K, AMC, AIME) demonstrate that InfoGuide not only prevents the reasoning degradation caused by vanilla CFG but also significantly enhances complex problem-solving accuracy, outperforming standard decoding baselines.

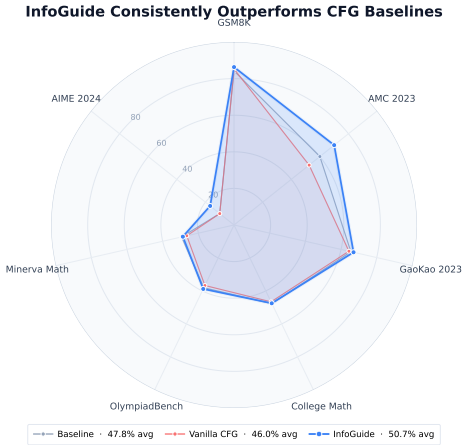

Figure 1: **Performance comparison on mathematical reasoning benchmarks.** Our proposed **InfoGuide** consistently outperforms both the standard Baseline and Vanilla CFG across various datasets (e.g., GSM8K, AMC 2023, AIME 2024), achieving the highest average accuracy (50.7%). Notably, applying Vanilla CFG uniformly degrades the average performance compared to the baseline (46.0% vs. 47.8%), highlighting the necessity and effectiveness of our dynamic guidance approach.

1 Introduction

Classifier-Free Guidance (CFG) has fundamentally transformed the landscape of generative artificial intelligence, most notably within text-to-image diffusion models (Ho and Salimans, 2022). By extrapolating between conditional and unconditional model predictions, CFG amplifies the influence of the conditioning signal without the need for an auxiliary classifier. While traditionally applied as a static penalty, recent advancements have explored dynamic CFG scheduling and adaptive guidance to further optimize generation fidelity, alignment, and safety (Ho and Salimans, 2022; Roy et al., 2025), supported by a deeper theoretical understanding of its classifier-centric roots (Zhao and Schwing, 2026).

Building upon its profound success in continuous diffusion spaces, CFG has been adapted for discrete text generation in autoregressive (AR) large language models (LLMs) (Sanchez et al., 2023). In the AR setting, CFG operates at the token level during decoding by interpolating the logits generated from a conditional prompt and an unconditional null prompt. This mechanism shares fundamental principles with Contrastive Decoding, which has demonstrated significant out-of-the-box improve-

ments for LLM reasoning by contrasting expert and amateur distributions (O'Brien and Lewis, 2023; Phan et al., 2024). These interventions effectively steer the language model to adhere more strictly to user instructions without requiring computationally expensive parameter updates.

Despite its efficacy in general text generation, applying vanilla CFG to complex reasoning tasks, such as CoT mathematical problem solving—reveals critical limitations. Extended reasoning trajectories demand precise, logical, and coherent step-by-step deductions. Recent literature highlights that reasoning models are prone to localized uncertainty, overthinking, and redundancy, necessitating careful test-time interventions (Wang et al., 2026; Yang et al., 2026). Vanilla CFG, however, applies a static and uniform penalty across all generated tokens. This rigid application disrupts the delicate structural dependencies inherent in CoT reasoning. By uniformly pushing the output away from the unconditional prior, static CFG forces the model to deviate unnaturally from its standard learned distribution. This injects noise into the reasoning trajectory, exacerbates thinking-answer disagreements (Kim et al.), and causes logical progression to become highly unstable. As observed in recent confidence-driven decoding and test-time ensemble studies, uniform token-level interventions often lead to semantic drift and mode collapse (Tang et al., 2026; Zeng et al., 2026), which can even degrade overall reasoning performance compared to unguided baselines (as illustrated in Figure 1).

To overcome these pathologies and harness the benefits of guidance for complex problem-solving, we propose a novel framework to stabilize CFG for reasoning LLM sampling, which we term **InfoGuide** (Information-Theoretic Classifier-Free Guidance). Instead of applying a rigid, uniform guidance scale, InfoGuide dynamically modulates the guidance intensity at each decoding step. By evaluating the real-time predictive landscape of the model, InfoGuide ensures that guidance is applied only when it is logically beneficial, thereby preserving the structural integrity of the reasoning trajectory while strictly aligning the output with the core problem constraints.

The core novelty of InfoGuide lies in its rigorous grounding in information theory, coupled with robust structural safeguards. We utilize Jensen-Shannon Divergence (JSD) to measure the distributional shift induced by the prompt, and Information Gain to quantify the reduction in predictive uncertainty. By synthesizing these metrics, InfoGuide applies strong guidance only when the prompt provides critical, uncertainty-resolving information, while attenuating the guidance for syntactic or highly certain tokens. Furthermore, to explicitly stabilize long CoT generation, we introduce Unconditional Logit Clipping to prevent unbounded penalties on tail-end tokens, and Sequence Length Decay to maintain reasoning flexibility over extended trajectories.

In summary, our contributions are as follows:

- **Analysis of CFG Limitations:** We reveal why static CFG fails in autoregressive reasoning, specifically highlighting trajectory noise and logit distortion.

- **InfoGuide Framework:** We propose a dynamic guidance method that uses information theory to calibrate token-level penalties and introduces novel mechanisms to stabilize long CoT sequences.

- **Empirical Performance:** Experiments on Qwen models demonstrate significant accuracy improvements on mathematical benchmarks like GSM8K, AMC, and AIME (see Figure 1).

2 Related Work

CFG for image generation builds upon the foundational concept of Classifier Guidance (Dhariwal and Nichol, 2021), which steers the sampling process of an unconditional diffusion model $P_\theta(x)$ towards a text condition c using an auxiliary classifier $P_\phi(c|x)$. By applying a guidance strength γ, the generation is modeled as $P(x|c) \propto P_\theta(x) \cdot P_\phi(c|x)^\gamma$, an approach that successfully trades sample diversity for higher fidelity and has been widely adapted for domain adaptation (Gal et al., 2022), robust image manipulation (Kim et al., 2022), and open-domain image editing (Crowson et al., 2022). CFG subsequently advanced this paradigm by eliminating the need for a separately trained, noise-robust classifier. Instead, a single model is trained with conditioning dropout to support both conditional and unconditional generation, allowing the sampling probability to be approximated via Bayes' rule as $P_\theta(x|c) \propto P_\theta(x|c)^\gamma / P_\theta(x)^{\gamma-1}$. This implicit classifier approach has become a cornerstone of modern high-resolution latent diffusion models

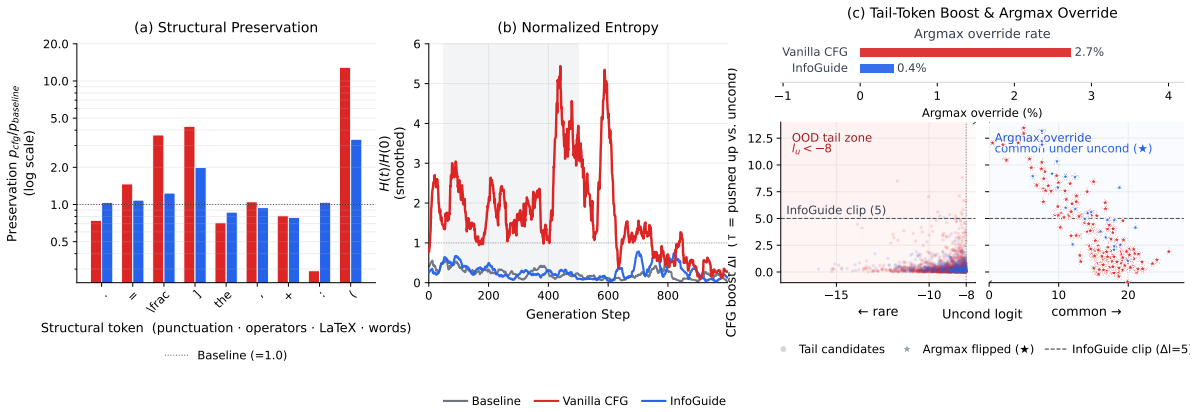

Figure 2: Analysis of Vanilla CFG limitations and InfoGuide improvements. (a) Structural Preservation: Vanilla CFG severely distorts the probability of structural tokens, while InfoGuide closely matches the baseline. (b) Normalized Entropy: Vanilla CFG causes massive entropy spikes leading to CoT trajectory divergence, whereas InfoGuide maintains stability. (c) Tail-Token Boost & Argmax Override: Vanilla CFG is highly vulnerable to rare token distortion (OOD tail zone), resulting in a high argmax override rate. InfoGuide restricts overrides primarily to common tokens, significantly reducing the flip rate and stabilizing generation.

(Rombach et al., 2022). Furthermore, the probabilistic formulation of CFG naturally translates into vector arithmetic in the latent space, enabling advanced control mechanisms such as negative prompting and compositional visual generation (Du et al., 2020).

CFG for LLMs has emerged as a inference-time technique for controlling natural language generation. Its application to text was notably pioneered by Sanchez et al. (Sanchez et al., 2023), who demonstrated that CFG can effectively steer standard autoregressive language models to "stay on topic" and improve alignment by interpolating between conditional and unconditional prompt logits. The broader utility of such test-time interventions in autoregressive architectures has been further emphasized by recent findings demonstrating that minimal inference-time adjustments can significantly enhance LLM reasoning capabilities (Yang et al., 2025b). Concurrently, the development of diffusion models for discrete text data has provided a new frontier for controllable generation, beginning with foundational adaptations like Diffusion-LM (Li et al., 2022) and DiffuSeq (Gong et al., 2022). As these architectures matured, CFG was naturally integrated to guide the iterative denoising process in language diffusion models (Lovelace et al., 2024). Recently, this paradigm has been substantially scaled up through Masked Diffusion Models (MDMs) and Large Language Diffusion Models, exemplified by state-of-the-art frameworks such as LLaDA (Nie et al., 2025, 2026) and Dream-7B (Ye et al., 2025). To address the limitations of relying on static unconditional inputs within these advanced frameworks, Li et al. (Li et al., 2026) introduced Adaptive Classifier-Free Guidance (A-CFG), a novel approach that dynamically tailors the unconditional input by temporarily re-masking low-confidence tokens, thereby focusing the guidance precisely on areas of model uncertainty during the reverse generation process.

3 Methodology

3.1 Preliminaries

CFG (Gal et al., 2022) enhances the alignment of generative models by extrapolating between conditional and unconditional distributions, bypassing the need for an auxiliary classifier. While originally designed for continuous image diffusion, it has been naturally extended to discrete text diffusion frameworks.

In text diffusion models, let x_t denote the noisy or masked sequence at timestep t, and c be the conditioning signal. The network, parameterized by θ, predicts the categorical distribution of the clean sequence, denoted as $p_\theta(x_0|x_t, c)$. To apply CFG, the model is also evaluated with a null condition $\emptyset$ to obtain the unconditional distribution $p_\theta(x_0|x_t, \emptyset)$.

The guided distribution $\tilde{p}_\theta$ is formulated by extrapolating these two distributions in the log-space, which is equivalent to the following weighted product:

$$\tilde{p}_\theta(x_0|x_t, c) \propto p_\theta(x_0|x_t, c)^\gamma \cdot p_\theta(x_0|x_t, \emptyset)^{1-\gamma}$$

where $\gamma \geq 1$ is the guidance scale. This formulation actively shifts the probability mass away from the unconditional prior, penalizing tokens that are highly probable in a generic context but irrelevant to the specific condition c.

CFG for Autoregressive LLMs. Building upon this paradigm, CFG has recently been adapted for autoregressive (AR) Large Language Models (Sanchez et al., 2023). In the AR setting, text generation is formulated as sequential next-token prediction. Let $x_{<t}$ denote the prefix of previously generated tokens, and c represent the user prompt acting as the condition. The model predicts the conditional next-token distribution $p_\theta(x_t|x_{<t}, c)$.

By replacing the user prompt c with a null condition $\emptyset$ (e.g., an empty string), we obtain the unconditional prior $p_\theta(x_t|x_{<t}, \emptyset)$. The guided next-token distribution $\tilde{p}_\theta$ is computed using an identical log-space extrapolation:

$$\tilde{p}_\theta(x_t|x_{<t}, c) \propto p_\theta(x_t|x_{<t}, c)^\gamma \cdot p_\theta(x_t|x_{<t}, \emptyset)^{1-\gamma}$$

This token-level guidance amplifies the influence of the user prompt at each generation step, effectively steering the AR decoding process without requiring any modification to the model weights.

3.2 Limitations of CFG in AR Models

While Vanilla CFG effectively steers generation towards the user prompt, its static and uniform application across all tokens introduces severe pathologies in autoregressive LLMs, particularly during long CoT reasoning. We identify three critical limitations, as illustrated in Figure 2:

- **Structural Distortion:** Vanilla CFG drastically alters the frequency and probability distribution of common structural tokens (e.g., punctuation, mathematical operators, and LaTeX formatting). As shown in Figure 2(a), the preservation ratio $p_{cfg}/p_{baseline}$ for Vanilla CFG deviates significantly from the baseline of 1.0. In contrast, InfoGuide maintains a token distribution that closely mirrors the baseline, preserving the essential structural integrity required for coherent reasoning.

- **Entropy Spikes and Trajectory Divergence:** Applying a static penalty uniformly increases the predictive entropy of the language model over time. Figure 2(b) illustrates that Vanilla CFG causes massive spikes in normalized entropy during next-token prediction. This divergence forces the model out of its standard learned distribution, frequently resulting in catastrophic generation failures such as repetitive loops (mode collapse) and CoT collapse. InfoGuide dynamically modulates the guidance, keeping the entropy distribution stable and closely aligned with the baseline.

- **Vulnerability to Tail-Token Distortion:** Vanilla CFG is highly susceptible to extreme values in the unconditional logits. In Figure 2(c), the x-axis represents the unconditional logits, transitioning from rare (out-of-distribution) to common tokens. Vanilla CFG aggressively boosts rare tokens in the "OOD tail zone," leading to a high argmax override rate (2.7%) where the model is forced to output hallucinated or irrelevant tokens. InfoGuide mitigates this by restricting argmax flips primarily to common tokens, drastically reducing the override rate (0.4%) and ensuring a stable, on-topic generation trajectory.

These fundamental flaws necessitate a paradigm shift from static, uniform guidance to a dynamic, token-aware, and information-theoretic modulation strategy.

3.3 Information-Theoretic CFG

To address the limitations of static guidance, we propose Information-Theoretic CFG. This framework dynamically calibrates token-level guidance based on real-time predictive uncertainty and distributional discrepancies, while employing structural safeguards to prevent logit distortion.

Information-Theoretic Modulation. The optimal intensity of guidance is highly context-dependent and fluctuates throughout the AR generation trajectory. We introduce a dynamic modulation mechanism grounded in information theory, leveraging JSD and relative entropy dynamics to rigorously quantify the token-level guidance requirement.

Distributional Discrepancy via JSD. To measure the distributional shift induced by the conditioning signal, we employ JSD. Defining the mixture distribution $M = \frac{1}{2}(P_c + P_\emptyset)$, the divergence is formulated as:

$$D_{JS} = \frac{1}{2}D_{KL}(P_c \parallel M) + \frac{1}{2}D_{KL}(P_\emptyset \parallel M)$$

Normalizing this by $\ln(2)$ yields a bounded scalar $\bar{D}_{JS} \in [0, 1]$. A high $\bar{D}_{JS}$ signifies that the conditioning context strongly dictates the current token prediction, necessitating amplified guidance.

Conversely, a low $\bar{D}_{JS}$ indicates syntactic tokens, where guidance should be attenuated to preserve fluency.

Entropy-Aware Information Gain Weighting. To account for relative predictive confidence, we incorporate an information gain factor derived from the differential Shannon entropy, $\Delta H = H(P_\emptyset) - H(P_c)$. A positive ΔH implies the conditioning signal resolves predictive uncertainty. We project this into a monotonic modulation factor Φ_{IG}:

$$\Phi_{IG} = \begin{cases} 1 + \omega_{ig} \cdot \Delta H & \text{if } \Delta H > 0 \\ \frac{1}{1+\omega_{ig} \cdot |\Delta H|} & \text{otherwise} \end{cases}$$

where ω_{ig} is a scaling hyperparameter. The composite dynamic multiplier for the CFG scale at step t is $\mu_t = (\bar{D}_{JS})^\alpha \cdot \Phi_{IG}$. This ensures guidance is maximized only when the prompt induces both a significant distributional shift and a reduction in uncertainty.

3.4 Stabilizing Reasoning Trajectory

While information-theoretic modulation optimizes guidance intensity, applying CFG over extended reasoning chains introduces distinct structural instabilities, such as logit distortion and semantic drift. To ensure robust and coherent generation, we introduce three complementary safeguards to stabilize the reasoning trajectory and improve inference efficiency.

Unconditional Logit Clipping A common pathology in standard CFG arises when the unconditional model assigns extremely low logits to obscure or highly context-specific tokens. When subtracted during log-space extrapolation, these unbounded negative logits artificially inflate the guided probability of hallucinated tokens. To mitigate this, we introduce Unconditional Logit Clipping. Let $l_\emptyset$ denote the unconditional logits. We dynamically lower-bound these logits relative to their maximum value:

$$\tilde{l}_\emptyset = \max(l_\emptyset, \max(l_\emptyset) - \tau_{clip})$$

where τ_{clip} is a predefined clipping range. This bounds the maximum penalty CFG can apply, preventing catastrophic hallucination on tail-end tokens.

Sequence Length Decay Applying a constant CFG scale over extended sequences, such as long CoT reasoning, often leads to semantic drift and mode collapse. To preserve diversity in later generation stages, we implement a sequence length decay mechanism. After an initial warmup period of T_{warmup} steps, the base CFG scale γ_t decays exponentially:

$$\gamma_t = 1 + (\gamma_{base} - 1) \cdot \lambda^{\max(0,\, t - T_{warmup})}$$

where $\lambda \in (0, 1)$ is the decay factor and γ_{base} is the initial scale. This ensures strong initial alignment with the prompt while gradually relaxing constraints to maintain reasoning flexibility.

Confident Token Bypassing To optimize computational efficiency and protect highly certain predictions from unnecessary distortion, we introduce a confidence-based bypass mechanism. At each generation step, we evaluate the maximum probability of the conditional distribution, $p_{max} = \max(P_c)$. If p_{max} exceeds a strict confidence threshold τ_{conf}, or if the decayed CFG scale approaches 1.0, we assume the model is highly certain of the next token (e.g., standard grammar or obvious continuations). In such cases, we entirely skip the unconditional forward pass and CFG modulation, directly sampling from P_c. This significantly reduces inference overhead while preserving the integrity of deterministic token sequences.

4 Experiments

4.1 Implementations

To rigorously evaluate the efficacy of our proposed dynamic CFG approach, we conduct comprehensive experiments across a diverse suite of mathematical reasoning benchmarks using state-of-the-art autoregressive language models.

Models. We select a range of models from the Qwen family to assess the generalizability of our method across different parameter scales and architectural specializations. Specifically, we utilize the general-purpose instruction-tuned models, Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct, to establish a baseline for standard reasoning tasks. To evaluate performance on domain-specific, rigorous reasoning, we employ the mathematically optimized Qwen2.5-1.5B-Math-Instruct and Qwen2.5-7B-Math-Instruct (Yang et al., 2024). Furthermore, we include the recently introduced reasoning-focused Qwen3-0.6B-thinking model (Yang et al., 2025a) to investigate the impact of our dynamic CFG on models natively trained for extended CoT generation.

Dataset	Qwen2.5 1.5B Inst.			Qwen2.5 7B Inst.			Qwen2.5 Math 1.5B			Qwen2.5 Math 7B			Qwen3-0.6B-think		
	Base	CFG	InfoGuide	Base	CFG	InfoGuide	Base	CFG	InfoGuide	Base	CFG	InfoGuide	Base	CFG	InfoGuide
GSM8K	74.1	47.1	**74.2**	92.2	90.8	**92.6**	84.2	85.3	**86.3**	95.2	95.1	**95.8**	75.9	75.5	**77.7**
Minerva Math	**18.0**	10.7	17.3	**35.7**	33.5	34.9	**29.4**	26.5	28.7	**37.1**	29.8	36.8	**21.7**	18.8	19.9
GaoKao 2023	47.0	34.5	**48.1**	63.4	63.0	**64.4**	65.5	64.4	**67.0**	65.8	**66.0**	65.2	54.5	50.2	**55.3**
OlympiadBench	**21.0**	17.3	20.0	**41.3**	39.2	39.6	38.1	36.7	**38.8**	41.6	40.6	**42.2**	28.4	26.7	**29.0**
College Math	**38.2**	31.6	37.5	**47.2**	46.5	**47.2**	**47.7**	46.6	47.5	46.7	**46.8**	46.7	35.3	32.3	**35.7**
AIME 2024	6.7	0.0	**6.7**	13.3	16.7	**16.7**	10.0	10.0	**16.7**	16.7	13.3	**16.7**	3.33	3.33	**10.0**
AMC 2023	25.0	20.0	**32.5**	52.5	50.0	**60.0**	60.0	52.5	**70.0**	62.5	52.5	**65.0**	**37.5**	30.0	35.0
Average	32.9	23.0	**33.8**	49.4	48.5	**50.8**	47.8	46.0	**50.7**	52.2	49.2	**52.6**	36.7	33.8	**37.5**

Table 1: Performance comparison across different Qwen models and guidance configurations. The best results for each model setting are highlighted in bold.

Datasets. Our evaluation spans a diverse spectrum of mathematical reasoning benchmarks, ranging from foundational problems to advanced STEM tasks. We utilize GSM8K (Cobbe et al., 2021) for grade-school math and the Gaokao from (Zhong et al., 2024) for high-school standardized exams. For highly complex, multi-step logical reasoning, we employ competition-level datasets including AMC23, AIME24, and Olympiad (He et al., 2024). Finally, advanced higher-education quantitative proficiency is assessed using CollegeMath (Tang et al., 2024) and MinervaMath (Lewkowycz et al., 2022).

4.2 Main Results

Table 1 encapsulates the empirical benchmarking of our dynamic CFG framework. The results demonstrate that our adaptive approach systematically outperforms both standard inference (Base) and static CFG baselines. By securing the highest aggregate accuracy across all five distinct model architectures, our method validates its robust efficacy and architectural agnosticism in complex reasoning tasks.

Vanilla CFG Degrades Reasoning. Applying a static vanilla CFG consistently harms mathematical reasoning. Across all models, it causes a substantial drop in average accuracy (e.g., from 32.9% to 23.0% on Qwen2.5-1.5B-Instruct). This suggests that uniformly amplifying contrast disrupts the precise, token-by-token logical dependencies essential for rigorous CoT.

Superiority of Dynamic CFG. Our dynamic approach effectively mitigates these negative effects while extracting the benefits of guidance. It yields the highest average scores across all model scales, with pronounced gains on complex datasets. For instance, on AMC 2023, our method improves Qwen2.5-Math-1.5B from 60.0% to 70.0%, steering the model toward accurate problem-solving trajectories without breaking logical consistency.

Effectiveness on Reasoning Models. Our method also benefits models natively optimized for extended reasoning. On the CoT-focused Qwen3-0.6B-think, vanilla CFG drops average performance to 33.8%, whereas our dynamic approach elevates it to 37.5% (surpassing the 36.7% baseline). This confirms that dynamic CFG is complementary to intrinsic reasoning training paradigms.

4.3 Ablation Study

To systematically evaluate the contribution of each component in our proposed framework, we conduct a step-by-step ablation study using the Qwen2.5-Math-1.5B model. We assess performance on three representative datasets: GSM8K, AIME 2024, and AMC 2023. The results are summarized in Table 2.

Impact of Information-Theoretic Modulation. Compared to the static baseline (CFG scale 2.0), which degrades AMC 2023 performance from 60.0% to 52.5%, introducing dynamic modulation based on Jensen-Shannon Divergence (JSD) and Information Gain significantly recovers and enhances reasoning capabilities. Using JSD alone boosts AMC 2023 to 65.0%, while Information Gain alone yields a peak AIME score of 20.0%. Combining both metrics (JSD + Info Gain) provides a balanced and robust foundation, outperforming static CFG across all three benchmarks.

Impact of Trajectory Stabilization. Building upon the dynamic modulation, the addition of our structural safeguards further elevates performance. Integrating Unconditional Logit Clipping prevents tail-end hallucination, yielding a substantial leap on AMC 2023 to 70.0% and GSM8K to 86.3%. Subsequent addition of Sequence Length Decay (CFG Decay) helps maintain reasoning flexibility in longer trajectories, stabilizing AIME 2024 per-

Configuration	Components					Datasets		
	JSD	Info Gain	Uncond Clip	CFG Decay	Skip Conf	GSM8K	AIME24	AMC23
Base Inference						84.2	10.0	60.0
Static CFG (2.0)						85.3	10.0	52.5
+ JSD	✓					85.1	13.3	65.0
+ Info Gain		✓				85.4	**20.0**	60.0
+ JSD & Info Gain	✓	✓				85.4	16.7	62.5
+ Uncond Clip	✓	✓	✓			86.3	13.3	**70.0**
+ CFG Decay	✓	✓	✓	✓		85.9	16.7	67.5
+ Skip Conf	✓	✓	✓	✓	✓	**86.3**	16.7	**70.0**

Table 2: Ablation study of the proposed dynamic CFG components on Qwen2.5-Math-1.5B. 'Uncond Clip' denotes Unconditional Logit Clipping, and 'Skip Conf' denotes Confident Token Bypassing.

formance at 16.7%. Finally, enabling Confident Token Bypassing (Skip Conf) achieves our peak comprehensive performance (86.3% on GSM8K, 16.7% on AIME, 70.0% on AMC) while simultaneously optimizing inference efficiency by skipping unnecessary unconditional forward passes.

Table 3: Cross-model performance on GSM8K and AIME24. All models are Instruct versions. "V-CFG" denotes Vanilla CFG. Best results for each model are in **bold**.

Model	Method	GSM8K	AIME24
Gemma3-2B	Direct	**58.5**	3.3
	V-CFG	52.3	0.0
	InfoGuide	57.2	**6.6**
Gemma3-7B	Direct	78.2	13.3
	V-CFG	76.2	10.0
	InfoGuide	**78.6**	**16.7**
LLaMA3-1B	Direct	54.2	3.3
	V-CFG	53.5	3.3
	InfoGuide	**54.5**	**10.0**
LLaMA3-7B	Direct	81.0	13.3
	V-CFG	80.2	13.3
	InfoGuide	**83.5**	**16.7**

4.4 Cross-Model Generalization

To verify that the effectiveness of InfoGuide is not limited to a specific model architecture, we extended our evaluation to the Gemma-3 and LLaMA-3 model families across different scales (1B to 7B parameters). As shown in Table 3, Vanilla CFG consistently degrades or stagnates reasoning performance compared to standard direct decoding, suffering catastrophic failures on complex tasks (e.g., dropping to 0% on AIME24 for Gemma3-2B-Inst). In contrast, InfoGuide demonstrates strong cross-model generalizability. It not only prevents the performance degradation induced by Vanilla CFG but also provides consistent improvements over the base models. This enhancement is particularly pronounced on the highly challenging AIME24 benchmark, where InfoGuide doubles the accuracy for Gemma3-2B-Inst (from 3.3% to 6.6%) and triples it for LLaMA3-1B-Inst (from 3.3% to 10.0%), confirming that our dynamic information-theoretic intervention robustly unlocks reasoning capabilities across diverse LLM architectures.

4.5 Efficiency

A major drawback of standard CFG is the computational overhead of requiring two forward passes per token, which effectively halves generation speed. Figure 3 illustrates the per-sample inference latency across the GSM8K and AIME24 benchmarks.

While Vanilla CFG drastically increases latency—dropping throughput from 61.87 to 31.85 tokens/s on AIME24. **InfoGuide** significantly mitigates this bottleneck. Thanks to our confident token bypassing mechanism, InfoGuide skips unnecessary unconditional passes. This restores the generation throughput to 51.49 tokens/s on AIME24 and reduces the average latency from 27.41s (Vanilla CFG) down to 15.20s. Similar efficiency gains are observed on GSM8K (50.37 tokens/s vs. Vanilla's 31.74 tokens/s). Ultimately, InfoGuide delivers superior reasoning performance with an inference speed nearly comparable to standard direct decoding. The speed is measured with Qwen2.5-Math-1.5B-Instruct model and H20 GPUs.

4.6 Visualization

To intuitively illustrate how InfoGuide improves reliability, Figure 4 visualizes the generation process on a math word problem. We highlight token-level

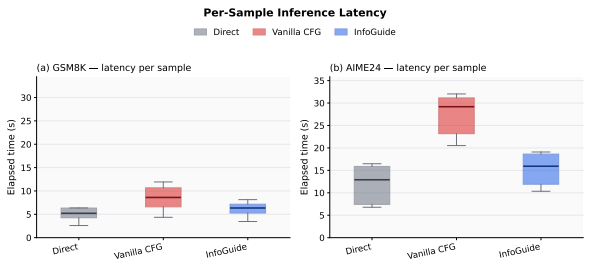

Figure 3: Per-sample inference latency on (a) GSM8K and (b) AIME24. InfoGuide significantly reduces the severe computational overhead of Vanilla CFG, achieving inference speeds nearly on par with standard Direct decoding.

Figure 4: Token generation visualization. Colors indicate uncertainty (yellow/red) and red borders mark "argmax flips". Vanilla CFG (top) forces disruptive flips that cause a logical error, while InfoGuide (bottom) maintains stable, low-uncertainty reasoning to reach the correct answer.

uncertainty (yellow to red) and mark "argmax flips" (red borders) where guidance overrides the base model's original top-1 prediction.

Under Vanilla CFG (Figure 4, top), static penalties force the model into a logical error: applying the 150% increase to the repair cost $(50,000)$ instead of the house price. This mistake coincides with a dense cluster of high-uncertainty tokens and forced argmax flips, demonstrating how unconstrained CFG disrupts intrinsic reasoning logic.

In contrast, InfoGuide (Figure 4, bottom) dynamically modulates guidance based on confidence. By avoiding harmful interventions during critical logical deductions, it correctly applies the multiplier to the initial value $(80,000)$. The generation exhibits significantly lower uncertainty and no detrimental flips at key mathematical nodes, successfully arriving at the correct answer $(70,000)$.

5 Conclusion

In this paper, we systematically investigated the failure modes of standard Classifier-Free Guidance when applied to autoregressive Large Language Models for complex CoT reasoning. We revealed that static, uniform guidance disrupts reasoning trajectories by distorting structural syntax, inducing catastrophic entropy spikes, and amplifying tail-token hallucinations. To address these fundamental pathologies, we introduced **InfoGuide**, an information-theoretic CFG framework. By dynamically modulating the guidance scale at the token level using Jensen-Shannon Divergence and Information Gain, InfoGuide applies guidance precisely where it resolves predictive uncertainty. Furthermore, our proposed structural safeguards—Unconditional Logit Clipping, Sequence Length Decay, and Confident Token Bypassing—effectively stabilize long reasoning chains. Extensive empirical evaluations across the Qwen model family and rigorous mathematical benchmarks demonstrated that InfoGuide not only prevents the degradation caused by vanilla CFG but significantly elevates the reasoning capabilities of LLMs. Ultimately, InfoGuide presents a robust, training-free intervention that unlocks the true potential of guidance for complex problem-solving.

6 Limitations

While InfoGuide significantly improves reasoning performance, we note the following limitations for future research:

- **Computational Overhead:** Like all CFG methods, InfoGuide requires an unconditional forward pass. Although our *Confident Token Bypassing* reduces this burden, inference latency remains higher than standard decoding. Future work could integrate speculative decoding to further improve efficiency.

- **Hyperparameter Sensitivity:** InfoGuide introduces structural parameters (e.g., τ_{clip}, λ). While our default configuration is robust across mathematical tasks, extending this framework to vastly different domains (e.g., code generation or open-ended writing) may require domain-specific tuning.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*, 9.

Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. 2022. Vqgan-clip: Open domain image generation and editing with natural language guidance. In *European conference on computer vision*, pages 88–105. Springer.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Yilun Du, Shuang Li, and Igor Mordatch. 2020. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33:6637–6647.

Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. 2022. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. 2022. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2426–2435.

Taehyeon Kim, Youngsoo Jang, Hyunsoo Lee, Yu Jin Kim, and Moontae Lee. Contrastive thinking decoding can improve answers for reasoning models.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Pengxiang Li, Shilin Yan, Jiayin Cai, Renrui Zhang, Ruichuan An, Ziyu Guo, and Xiaowei Gao. 2026. Adaptive classifier-free guidance via dynamic low-confidence masking. *Advances in Neural Information Processing Systems*, 38:47362–47381.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343.

Justin Lovelace, Varsha Kishore, Yiwei Chen, and Kilian Weinberger. 2024. Diffusion guided language modeling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14936–14952.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025. Scaling up masked diffusion models on text. In *International Conference on Learning Representations*, volume 2025, pages 82974–82997.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2026. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646.

Sean O'Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*.

Phuc Phan, Hieu Tran, and Long Phan. 2024. Distillation contrastive decoding: Improving llms reasoning with contrastive decoding and distillation. *arXiv preprint arXiv:2402.14874*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Soham Roy, Abhishek Mishra, Aakash Sen Sharma, Shirish Karande, and Murari Mandal. 2025. Guardians of generation: Dynamic inference-time copyright shielding with adaptive guidance for ai image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2551–2561.

Guillaume Sanchez, Honglu Fan, Alexander Spangher, Elad Levi, Pawan Sasanka Ammanamanchi, and Stella Biderman. 2023. Stay on topic with classifier-free guidance. *arXiv preprint arXiv:2306.17806*.

Lexiang Tang, Weihao Gao, Bingchen Zhao, Lu Ma, Bang Yang, Yuexian Zou, and 1 others. 2026. Thinking by subtraction: Confidence-driven contrastive decoding for llm reasoning. *arXiv preprint arXiv:2602.18232*.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*.

Qianyue Wang, Jinwu Hu, Yufeng Wang, Huanxiang Lin, Bolin Chen, Zhiquan Wen, Yaofo Chen, and Mingkui Tan. 2026. Beyond model scaling: Test-time intervention for efficient deep reasoning. *arXiv preprint arXiv:2601.11252*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Chenxu Yang, Qingyi Si, Mz Dai, Dingyu Yao, Mingyu Zheng, Minghui Chen, Zheng Lin, and Weiping Wang. 2026. Test-time prompt intervention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34223–34231.

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

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

Zhichen Zeng, Qi Yu, Xiao Lin, Ruizhong Qiu, Xuying Ning, Tianxin Wei, Yuchen Yan, Jingrui He, and Hanghang Tong. 2026. Harnessing consistency for robust test-time llm ensemble. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 3528–3545.

Xiaoming Zhao and Alex Schwing. 2026. Studying classifier (-free) guidance from a classifier-centric perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13271–13279.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the association for computational linguistics: NAACL 2024*, pages 2299–2314.

A More Ablation on Hyperparameters.

In this section, we present a detailed ablation study on the key hyperparameters of the InfoGuide framework, specifically the base CFG scale (γ_{base}) and the confidence threshold (τ_{conf}). We conduct these experiments on the Qwen2.5-Math-7B-Instruct model, evaluating on the AMC 2023 and AIME 2024 benchmarks.

Base CFG Scale (γ_{base}): Figure 5 illustrates the impact of varying the base CFG scale from 1.0 to 2.0. We observe that setting $\gamma_{base} = 1.0$ yields the optimal performance across both benchmarks. As the scale increases beyond 1.0, performance initially drops, indicating that excessive baseline guidance disrupts the delicate reasoning trajectory, even with dynamic modulation. This highlights the efficacy of our information-theoretic components (JSD and Information Gain), which are sufficient to guide the model effectively without requiring a high baseline penalty.

Confidence Threshold (τ_{conf}): Figure 6 demonstrates the sensitivity of the model to the confidence threshold used for bypassing the unconditional forward pass. The results show that performance is remarkably stable across a wide range of thresholds from 0.85 to 1.00. This robustness indicates that the model's internal confidence is a highly reliable indicator for safe bypassing. We select $\tau_{conf} = 0.85$ as the default setting, as it maximizes inference efficiency by skipping unnecessary computations without compromising reasoning accuracy.

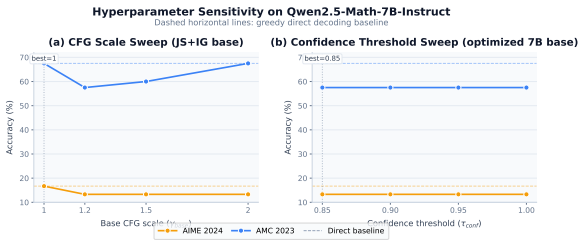

Figure 5: Impact of the base CFG scale (γ_{base}) on mathematical reasoning performance. The optimal performance is achieved at $\gamma_{base} = 1.0$, demonstrating that our dynamic modulation provides sufficient guidance without needing a high baseline penalty.

B Visualizations on AIME 2024

To further understand the internal generation dynamics of our proposed method, we visualize the token-level generation trajectories of different methods on the AIME 2024 dataset. In the following heatmaps, the background color represents the model's uncertainty (calculated as $1-p_{max}$), where darker colors indicate higher uncertainty. The red outlines indicate "argmax flips," where the finally sampled token differs from the model's original top-1 prediction.

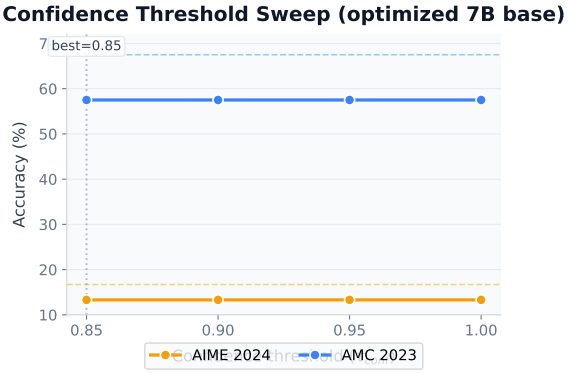

Figure 6: Sensitivity analysis of the confidence threshold (τ_{conf}). Performance remains highly stable across the 0.85 to 1.00 range, allowing us to safely set $\tau_{conf} = 0.85$ to maximize inference efficiency.

Case Study 1: Geometric and Algebraic Reasoning (Rescue #18). Figure 7 presents a complex geometry problem requiring the intersection of a line segment and a specific family of curves. Both the Baseline and Vanilla CFG models fail to maintain a coherent reasoning trajectory. Specifically, Vanilla CFG suffers from severe error accumulation, evidenced by a high density of argmax flips (red outlines) and elevated uncertainty (darker backgrounds) during the algebraic manipulation phase. In contrast, InfoGuide maintains a highly confident and stable generation process. By dynamically modulating the guidance, InfoGuide successfully navigates the complex algebraic substitutions and correctly identifies the unique intersection point, yielding the correct answer.

Case Study 2: System of Logarithmic Equations (Rescue #24). Figure 8 illustrates a problem involving a system of logarithmic equations. The Baseline and Vanilla CFG methods attempt to solve the problem by immediately converting the expressions into exponential forms. This suboptimal strategy leads to overly complicated algebraic fractions, causing the models to become uncertain and ultimately produce incorrect results. InfoGuide, however, exhibits a much more strategic reasoning path. It confidently introduces linear substitutions

($a = \log_2 x$, $b = \log_2 y$, $c = \log_2 z$), transforming the complex non-linear system into a straightforward linear system. The visualization shows that InfoGuide's trajectory is remarkably clean, with minimal uncertainty and very few argmax flips, demonstrating how effective information-guided modulation can steer the model toward optimal problem-solving strategies.

AIME24 Rescue #18

Question

Let $O=(0,0)$, $A=\left(\frac{1}{2},0\right)$, and $B=\left(0,\frac{\sqrt{3}}{2}\right)$ be points in the coordinate plane. Let $\mathcal{F}$ be the family of segments $\overline{PQ}$ of unit length lying in the first quadrant with P on the x-axis and Q on the y-axis. There is a unique point C on $\overline{AB}$, distinct from A and B, that does not belong to any segment from $\mathcal{F}$ other than $\overline{AB}$. Then $OC^2=\frac{p}{q}$, where p and q are relatively prime positive integers. Find $p+q$.

Ground truth: 23

Method	Correct	Pred	# tokens	Argmax flips
Baseline	N	1	1129	0 (0.0%)
Vanilla CFG	N	5	1139	36 (3.2%)
InfoGuide	Y	23	1242	5 (0.4%)

Baseline (wrong)

Token heatmap: background = uncertainty (1-p_max); red outline = argmax flip

To solve the problem, we need to find the unique point C on the line segment $\overline{AB}$ that does not belong to any segment from the family $\mathcal{F}$ other than $\overline{AB}$. The family $\mathcal{F}$ consists of segments $\overline{PQ}$ of unit length with P on the x-axis and Q on the y-axis. The coordinates of P and Q can be written as $P = (a, 0)$ and $Q = (0, b)$ where $a^2 + b^2 = 1$. The equation of the line segment $\overline{AB}$ can be determined using the coordinates of A and B. The coordinates of A are $\left(\frac{1}{2}, 0\right)$ and the coordinates of B are $\left(0, \frac{\sqrt{3}}{2}\right)$. The slope of the line segment $\overline{AB}$ is: $$\text{slope} = \frac{\frac{\sqrt{3}}{2} - 0}{0 - \frac{1}{2}} = -\sqrt{3}$$ Thus, the equation of the line passing through A and B is: $$y = -\sqrt{3}x + \frac{\sqrt{3}}{2}$$ We need to find the point C on this line such that C does not belong to any segment from $\mathcal{F}$ other than $\overline{AB}$. This means that C must be the point where the line segment $\overline{AB}$ intersects the line $x^2 + y^2 = 1$ (the unit circle) at a point other than A and B. Substitute $y = -\sqrt{3}x + \frac{\sqrt{3}}{2}$ into $x^2 + y^2 = 1$: $$x^2 + \left(-\sqrt{3}x + \frac{\sqrt{3}}{2}\right)^2 = 1$$

$$\text{Expanding and simplifying: } x^2 + 3x^2 - 3x + \frac{3}{4} = 1$$

$$4x^2 - 3x + \frac{3}{4} = 1$$

$$16x^2 - 12x + 3 = 4$$

$$16x^2 - 12x - 1 = 0$$

Using the quadratic formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ where $a = 16$, $b = -12$, and $c = -1$: $$x = \frac{12 \pm \sqrt{144 + 64}}{32} = \frac{12 \pm \sqrt{208}}{32} = \frac{12 \pm 4\sqrt{13}}{32} = \frac{3 \pm \sqrt{13}}{8}$$ Since x must be between 0 and $\frac{1}{2}$, we take the smaller root: $$x = \frac{3 - \sqrt{13}}{8}$$ Substitute $x = \frac{3 - \sqrt{13}}{8}$ back into the equation of the line to find y: $$y = -\sqrt{3}\left(\frac{3 - \sqrt{13}}{8}\right) + \frac{\sqrt{3}}{2} = \frac{-3\sqrt{3} + \sqrt{39}}{8} + \frac{4\sqrt{3}}{8} = \frac{\sqrt{39} + \sqrt{3}}{8}$$ Thus, the coordinates of C are $\left(\frac{3 - \sqrt{13}}{8}, \frac{\sqrt{39} + \sqrt{3}}{8}\right)$. The distance OC is: $$OC = \sqrt{\left(\frac{3 - \sqrt{13}}{8}\right)^2 + \left(\frac{\sqrt{39} + \sqrt{3}}{8}\right)^2} = \sqrt{\frac{(3 - \sqrt{13})^2 + (\sqrt{39} + \sqrt{3})^2}{64}} = \sqrt{\frac{9 - 6\sqrt{13} + 13 + 39 + 6\sqrt{13} + 3}{64}} = \sqrt{\frac{64}{64}} = 1$$

Therefore, $OC^2 = 1$, and the answer is: $$\boxed{1}$$ <|im_end|>

Vanilla CFG (wrong)

Token heatmap: background = uncertainty (1-p_max); red outline = argmax flip

Given points $O = (0,0)$, $A = \left(\frac{1}{2}, 0\right)$, and $B = \left(0, \frac{\sqrt{3}}{2}\right)$, we need to find a unique point C on $\overline{AB}$ that does not belong to any segment from $\mathcal{F}$ other than $\overline{AB}$. The family $\mathcal{F}$ consists of segments $\overline{PQ}$ of unit length lying in the first quadrant with P on the x-axis and Q on the y-axis. Therefore, any segment $\overline{PQ}$ in $\mathcal{F}$ satisfies $x + y = 1$ for some $x, y > 0$. The equation of line $\overline{AB}$ can be found using the two-point form of the line equation: $$y - 0 = \frac{\frac{\sqrt{3}}{2} - 0}{0 - \frac{1}{2}}\left(x - \frac{1}{2}\right)$$ $$\Rightarrow y = -\sqrt{3}x + \frac{\sqrt{3}}{2}.$$ We need to find the point C on $\overline{AB}$ that does not satisfy $x + y = 1$. Let $C = (x_0, y_0)$ be such a point. Then $y_0 = -\sqrt{3}x_0 + \frac{\sqrt{3}}{2}$ and $x_0 + y_0 = k$ for some $k \neq 1$. Substituting y_0 into the equation $x_0 + y_0 = k$, we get: $$x_0 + (-\sqrt{3}x_0 + \frac{\sqrt{3}}{2}) = k \Rightarrow (1 - \sqrt{3})x_0 + \frac{\sqrt{3}}{2} = k \Rightarrow x_0 = \frac{k - \frac{\sqrt{3}}{2}}{1 - \sqrt{3}}.$$ Rationalizing the denominator, we have: $$x_0 = \frac{(k - \frac{\sqrt{3}}{2})(1 + \sqrt{3})}{(1 - \sqrt{3})(1 + \sqrt{3})} = \frac{(k + k\sqrt{3}) - \frac{\sqrt{3}}{2}}{2} = \frac{(k - \frac{\sqrt{3}}{2}) - 2k\sqrt{3} + \sqrt{3}}{4}.$$ Since C does not belong to $\mathcal{F}$ other than $\overline{AB}$, C must be such that $x_0 + y_0 \neq 1$. The only value of k that satisfies this condition is $k = \frac{1}{2}$. Substituting $k = \frac{1}{2}$ into the equation for x_0, we get: $$x_0 = \frac{\frac{1}{2} - \frac{\sqrt{3}}{2}}{1 - \sqrt{3}} = \frac{\frac{1 - \sqrt{3}}{2}}{1 - \sqrt{3}} = \frac{1 - \sqrt{3}}{2(1 - \sqrt{3})} = \frac{1}{2}.$$ Then, substituting $x_0 = \frac{1}{2}$ into the equation of line $\overline{AB}$, we get: $$y_0 = -\sqrt{3} \cdot \frac{1}{2} + \frac{\sqrt{3}}{2} = 0.$$ This is not possible since $y_0 > 0$. Instead, we should use the correct value of k that satisfies the condition. The correct value is $k = \frac{1}{2}$, and we need to find the correct x_0 and y_0 such that $x_0 + y_0 = \frac{1}{2}$. Solving for x_0 and y_0, we get: $$x_0 = \frac{1}{2}, \quad y_0 = 0.$$ Then, $C^2 = x_0^2 + y_0^2 = \left(\frac{1}{2}\right)^2 + 0^2 = \frac{1}{4}$. Therefore, $p = 1$ and $q = 4$, and $p + q = 5$. The final answer is: $$\boxed{5}.$$ <|im_end|>

InfoGuide (correct)

Token heatmap: background = uncertainty (1-p_max); red outline = argmax flip

To solve the problem, we first need to determine the coordinates of point C on segment $\overline{AB}$ that does not belong to any segment from $\mathcal{F}$ other than $\overline{AB}$. The coordinates of points A and B are given as $A = \left(\frac{1}{2}, 0\right)$ and $B = \left(0, \frac{\sqrt{3}}{2}\right)$. The equation of line segment $\overline{AB}$ can be derived using the two-point form of the equation of a line: $$y - 0 = \frac{\frac{\sqrt{3}}{2} - 0}{0 - \frac{1}{2}}\left(x - \frac{1}{2}\right) \Rightarrow y = -\sqrt{3}\left(x - \frac{1}{2}\right)$$ $$\Rightarrow y = -\sqrt{3}x + \frac{\sqrt{3}}{2}.$$ Next, we need to find the point C on $\overline{AB}$ such that C does not lie on any segment from $\mathcal{F}$ other than $\overline{AB}$. A segment from $\mathcal{F}$ has endpoints $P = (a, 0)$ and $Q = (0, b)$ with $a^2 + b^2 = 1$ and $a, b > 0$. The equation of such a segment is: $$\frac{x}{a} + \frac{y}{b} = 1.$$ We need to find the intersection of this line with $\overline{AB}$. Setting $y = -\sqrt{3}x + \frac{\sqrt{3}}{2}$ in the equation of the segment, we get: $$\frac{x}{a} + \frac{-\sqrt{3}x + \frac{\sqrt{3}}{2}}{b} = 1 \Rightarrow \frac{x}{a} - \frac{\sqrt{3}x}{b} + \frac{\sqrt{3}}{2b} = 1 \Rightarrow x\left(\frac{1}{a} - \frac{\sqrt{3}}{b}\right) = 1 - \frac{\sqrt{3}}{2b}.$$ Solving for x, we get: $$x = \frac{1 - \frac{\sqrt{3}}{2b}}{\frac{1}{a} - \frac{\sqrt{3}}{b}} = \frac{2b - \sqrt{3}a}{2b - \sqrt{3}a}.$$ Since $a^2 + b^2 = 1$, we can express b in terms of a as $b = \sqrt{1 - a^2}$. Substituting $b = \sqrt{1 - a^2}$ into the equation for x, we get: $$x = \frac{2\sqrt{1 - a^2} - \sqrt{3}a}{2\sqrt{1 - a^2} - \sqrt{3}a}.$$ To find the specific value of x that corresponds to the point C that does not belong to any segment from $\mathcal{F}$ other than $\overline{AB}$, we need to consider the case when $a = \frac{1}{2}$ and $b = \frac{\sqrt{3}}{2}$. Substituting these values into the equation for x, we get: $$x = \frac{2 \cdot \frac{\sqrt{3}}{2} - \sqrt{3} \cdot \frac{1}{2}}{2 \cdot \frac{\sqrt{3}}{2} - \sqrt{3} \cdot \frac{1}{2}} = \frac{\frac{\sqrt{3}}{2} - \frac{\sqrt{3}}{2}}{\frac{\sqrt{3}}{2} - \frac{\sqrt{3}}{2}} = \frac{\frac{\sqrt{3}}{2}}{\frac{\sqrt{3}}{2}} = 1.$$ This value of x corresponds to the point B, which is not the point we are looking for. Instead, we need to consider the case when $a = \frac{1}{2}$ and $b = \frac{\sqrt{3}}{2}$ and find the point C that does not lie on any segment from $\mathcal{F}$ other than $\overline{AB}$. This point is the intersection of $\overline{AB}$ with the line $x = \frac{1}{2}$, which is the point $C = \left(\frac{1}{2}, \frac{\sqrt{3}}{4}\right)$. Finally, we calculate $$OC^2 = \left(\frac{1}{2}\right)^2 + \left(\frac{\sqrt{3}}{4}\right)^2 = \frac{1}{4} + \frac{3}{16} = \frac{4}{16} + \frac{3}{16} = \frac{7}{16}.$$ Thus, $p = 7$ and $q = 16$, and $p + q = 23$. The answer is: $$\boxed{23}.$$ <|im_end|>

AIME24 Rescue #24

Question

Let \$x,y\$ and \$z\$ be positive real numbers that satisfy the following system of equations:
\\[\\log_2\\left({x \\over yz}\\right) = {1 \\over 2}\\]\\[\\log_2\\left({y \\over xz}\\right) = {1 \\over 3}\\]\\[\\log_2\\left({z \\over xy}\\right) = {1 \\over 4}\\]
Then the value of \$\\left|\\log_2(x^4y^3z^2)\\right|\$ is \$\\tfrac{m}{n}\$ where \$m\$ and \$n\$ are relatively prime positive integers. Find \$m+n\$.

Ground truth: 33

Method | Correct | Pred | # tokens | Argmax flips
Baseline | N | 15 | 1249 | 0 (0.0%)
Vanilla CFG | N | 14 | 779 | 13 (1.7%)
InfoGuide | Y | 33 | 1580 | 4 (0.3%)

Baseline (wrong)

Token heatmap: background = uncertainty (1-p_max); red outline = argmax flip

To solve the given system of equations , we start by rewriting each equation in exponential form . The first equation is : \\ [\\ log _2\\ left (\\ frac {x }{ yz }\\ right) = \\ frac {1}{ 2} \\ ] This implies : \\ [\\ frac {x }{ yz } = 2^{ 1/2} = \\ sqrt {2} \\ ] So , we have : \\ [x = y z \\ sqrt {2} \\ ] The second equation is : \\ [\\ log _2\\ left (\\ frac {y }{ xz }\\ right) = \\ frac {1}{ 3} \\ ] This implies : \\ [\\ frac {y }{ xz } = 2^{ 1/3} \\ ] So , we have : \\ [y = x z \\ cdot 2^{ 1/3} \\ ] The third equation is : \\ [\\ log _2\\ left (\\ frac {z }{ xy }\\ right) = \\ frac {1}{ 4} \\ ] This implies : \\ [\\ frac {z }{ xy } = 2^{ 1/4} \\ ] So , we have : \\ [z = xy \\ cdot 2^{ 1/4} \\ ] Now we have three equations : 1. \\(x = y z \\ sqrt {2}\\ ) 2. \\(y = x z \\ cdot 2^{ 1/3}\\ ) 3. \\(z = xy \\ cdot 2^{ 1/4}\\ ) We can substitute the expression for \\(z\\) from the third equation into the first equation . From the third equation , we have : \\ [z = xy \\ cdot 2^{ 1/4} \\ ] Sub stit uting this into the first equation , we get : \\ [x = y \\ cdot (xy \\ cdot 2^{ 1/4}) \\ sqrt {2} = xy ^2 \\ cdot 2^{ 1/4} \\ cdot \\ sqrt {2} = xy ^2 \\ cdot 2^{ 3/4} \\ ] So , we have : \\ [x = xy ^2 \\ cdot 2^{ 3/4} \\ ] Div iding both sides by \\(x\\) (since \\(x \\ ne q 0\\ )), we get : \\ [1 = y ^2 \\ cdot 2^{ 3/4} \\ ] Solving for \\(y^2\\ ), we get : \\ [y^2 = 2^{- 3/4} \\ ] Taking the square root of both sides , we get : \\ [y = 2^{- 3/8} \\ ] Now , substit uting \\(y = 2^{- 3/8}\\ ) into the third equation , we get : \\ [z = x \\ cdot 2^{- 3/8} \\ cdot 2^{ 1/4} = x \\ cdot 2^{- 3/8 + 2/8} = x \\ cdot 2^{- 1/8} \\ ] So , we have : \\ [z = x \\ cdot 2^{- 1/8} \\ ] Sub stit uting \\(z = x \\ cdot 2^{- 1/8}\\ ) into the first equation , we get : \\ [x = y \\ cdot (x \\ cdot 2^{- 1/8}) \\ sqrt {2} = 2^{- 3/8} \\ cdot x \\ cdot 2^{- 1/8} \\ cdot 2^{ 1/2} = 2^{- 3/8 - 1/8 + 4/8} \\ cdot x = 2^{ 0} \\ cdot x = x

\\ ] This is consistent , so our expressions for \\(y\\) and \\(z\\) are correct . Now , we need to find the value of \\ (\\ left | \\ log _2(x ^4 y ^3 z ^2) \\ right |\\ ). First , we express \\(x\\ ), \\(y\\ ), and \\(z\\) in terms of each other . From \\(y = 2^{- 3/8}\\ ) and \\(z = x \\ cdot 2^{- 1/8}\\ ), we substitute into the expression for \\(x\\ ): \\ [x = y z \\ sqrt {2} = 2^{- 3/8} \\ cdot x \\ cdot 2^{- 1/8} \\ cdot 2^{ 1/2} = x \\ cdot 2^{- 3/8 - 1/8 + 4/8} = x \\ cdot 2^{ 0} = x

\\ ] This is consistent , so we can use the values directly . Now , we calculate \\(x^4 y ^3 z ^2\\ ): \\ [x^4 y ^3 z ^2 = x ^4 \\ left (2^{- 3/8}\\ right)^ 3 \\ left (x \\ cdot 2^{- 1/8}\\ right)^ 2 = x ^4 \\ cdot 2^{- 9/8} \\ cdot x ^2 \\ cdot 2^{- 2/8} = x ^6 \\ cdot 2^{- 11/8} \\ ] Since \\(x = 2^{ 3/8}\\ ), we substitute \\(x\\) into the expression : \\ [x^6 = (2^{ 3/8})^6 = 2^{ 18/8} = 2^{ 9/4} \\ ] So , we have : \\ [x^4 y ^3 z ^2 = 2^{ 9/4} \\ cdot 2^{- 11/8} = 2^{ 18/8 - 11/8} = 2^{ 7/8} \\ ] Therefore , we have : \\ [\\ left | \\ log _2(x ^4 y ^3 z ^2) \\ right | = \\ left | \\ log _2(2^{ 7/8}) \\ right | = \\ left | \\ frac {7}{ 8} \\ right | = \\ frac {7}{ 8} \\ ] The value of \\(m+n \\) is : \\ [7 + 8 = 15

\\ ] So , the answer is : \\ [\\ boxed {15} \\ ]<|im_end|>

Vanilla CFG (wrong)

Token heatmap: background = uncertainty (1-p_max); red outline = argmax flip

Given the system of logarith mic equations : \\ [\\ log _2\\ left ((x \\ over y z}\\ right) = \\ frac {1}{ 2}, \\ quad \\ log _2\\ left ({ y \\ over x z}\\ right) = \\ frac {1}{ 3}, \\ quad \\ log _2\\ left ({ z \\ over xy }\\ right) = \\ frac {1}{ 4}, \\ ] we convert these into exponential form : \\ [\\ frac {x }{ yz } = 2^{ 1/2} = \\ sqrt {2}, \\ quad \\ frac {y }{ xz } = 2^{ 1/3}, \\ quad \\ frac {z }{ xy } = 2^{ 1/4}. \\ ] Rew riting these equations , we get : \\ [x = y z \\ cdot \\ sqrt {2}, \\ quad y = x z \\ cdot 2^{ 1/3}, \\ quad z = xy \\ cdot 2^{ 1/4}. \\ ] We multiply all three equations together : \\ [xyz = (yz \\ cdot \\ sqrt {2})(xz \\ cdot 2^{ 1/3})(xy \\ cdot 2^{ 1/4}). \\ ] S impl ifying the right -hand side , we have : \\ [xyz = x ^2 y ^2 z ^2 \\ cdot 2^{ 1/2 + 1/3 + 1/4}. \\ ] Since \\(x, y , z \\) are positive real numbers , we can divide both sides by \\(xyz \\ ): \\ [1 = xyz \\ cdot 2^{ 1/2 + 1/3 + 1/4}. \\ ] We need to find the value of \\(1/2 + 1/3 + 1/4\\ ): \\ [\\ frac {1}{ 2} + \\ frac {1}{ 3} + \\ frac {1}{ 4} = \\ frac {6}{ 12} + \\ frac {4}{ 12} + \\ frac {3}{ 12} = \\ frac {13}{ 12}. \\ ] Thus , we have : \\ [1 = xyz \\ cdot 2^{ 13/12} \\ impl ies xyz = 2^{- 13/12}. \\ ] We need to find the value of \\ (\\ left |\\ log _2(x ^4 y ^3 z ^2)\\ right |\\ ): \\ [\\ log _2(x ^4 y ^3 z ^2) = \\ log _2((xyz)^ 4 \\ cdot y ^2) = \\ log _2((2^{- 13/12})^4 \\ cdot y ^2) = \\ log _2(2 ^{- 13/3} \\ cdot y ^2). \\ ] Since \\(y = \\ frac {x z \\ cdot 2^{ 1/3}}{x} = \\ frac {2^{ 1/3}}{x x} = 2^{ 1/3}\\ ), we substitute \\(y = 2^{ 1/3}\\ ): \\ [\\ log _2(2^{- 13/3} \\ cdot (2^{ 1/3}) ^2) = \\ log _2(2^{- 13/3} \\ cdot 2^{ 2/3 }) = \\ log _2(2^{- 13/3 + 2/3}) = \\ log _2(2^{- 11/3}) = - \\ frac {11}{ 3}. \\ ] Thus , the value of \\ (\\ left |\\ log _2(x ^4 y ^3 z ^2)\\ right |\\ ) is \\ (\\ frac {11}{ 3}\\ ), and \\(m+n = 11+3 = 14\\ ). Therefore , the answer is : \\ [\\ boxed {14}. \\ ]<|im_end|>

InfoGuide (correct)

Token heatmap: background = uncertainty (1-p_max); red outline = argmax flip

Given the system of equations : \\ [\\ log _2\\ left (\\ frac {x }{ yz }\\ right) = \\ frac {1}{ 2}, \\ quad \\ log _2\\ left (\\ frac {y }{ xz }\\ right) = \\ frac {1}{ 3}, \\ quad \\ log _2\\ left (\\ frac {z }{ xy }\\ right) = \\ frac {1}{ 4}, \\ ] we can rewrite these equations in terms of logarith ms of \\(x\\ ), \\(y\\ ), and \\(z\\ ): \\ [\\ log _2 x - \\ log _2 (yz) = \\ frac {1}{ 2} \\ impl ies \\ log _2 x - \\ log _2 y - \\ log _2 z = \\ frac {1}{ 2}, \\ ] \\ [\\ log _2 y - \\ log _2 (xz) = \\ frac {1}{ 3} \\ impl ies \\ log _2 y - \\ log _2 x - \\ log _2 z = \\ frac {1}{ 3}, \\ ] \\ [\\ log _2 z - \\ log _2 (xy) = \\ frac {1}{ 4} \\ impl ies \\ log _2 z - \\ log _2 x - \\ log _2 y = \\ frac {1}{ 4}. \\ ] Let 's denote \\ (\\ log _2 x = a \\ ), \\ (\\ log _2 y = b \\ ), and \\ (\\ log _2 z = c \\ ). The system of equations becomes : \\ [a - b - c = \\ frac {1}{ 2}, \\ quad b - a - c = \\ frac {1}{ 3}, \\ quad c - a - b = \\ frac {1}{ 4}. \\ ] Adding all three equations , we get : \\ [(a - b - c) + (b - a - c) + (c - a - b) = \\ frac {1}{ 2} + \\ frac {1}{ 3} + \\ frac {1}{ 4}. \\ ] This simpl ifies to : \\ [-a - b - c = \\ frac {6}{ 12} + \\ frac {4}{ 12} + \\ frac {3}{ 12} = \\ frac {13}{ 12}. \\ ] Thus , \\ [a + b + c = - \\ frac {13}{ 12}. \\ ] Now we can solve for \\(a\\ ), \\(b\\ ), and \\(c\\) individually . From the first equation : \\ [a - b - c = \\ frac {1}{ 2} \\ impl ies a = b + c + \\ frac {1}{ 2}. \\ ] Sub stit uting \\(a = b + c + \\ frac {1}{ 2}\\ ) into \\(a + b + c = - \\ frac {13}{ 12}\\ ), we get : \\ [b + c + \\ frac {1}{ 2} + b + c = - \\ frac {13}{ 12} \\ impl ies 2b + 2c + \\ frac {1}{ 2} = - \\ frac {13}{ 12} \\ impl ies 2b + 2c = - \\ frac {13}{ 12 } - \\ frac {6}{ 12} = - \\ frac {19}{ 12} \\ impl ies b + c = - \\ frac {19}{ 24}. \\ ] Then , \\ [a = b + c + \\ frac {1}{ 2} = - \\ frac {19}{ 24} + \\ frac {12}{ 24} = - \\ frac {7}{ 24}. \\ ] Similarly , from the second equation : \\ [b - a - c = \\ frac {1}{ 3} \\ impl ies b = a + c + \\ frac {1}{ 3}. \\ ] Sub stit uting \\(b = a + c + \\ frac {1}{ 3}\\ ) into \\(a + b + c = - \\ frac {13}{ 12}\\ ), we get : \\ [a + a + c + \\ frac {1}{ 3} + c = - \\ frac {13}{ 12} \\ impl ies 2a + 2c + \\ frac {1}{ 3} = - \\ frac {13}{ 12} \\ impl ies 2a + 2c = - \\ frac {13}{ 12} - \\ frac {4}{ 12} = - \\ frac {17}{ 12} \\ impl ies a + c = - \\ frac {17}{ 24}. \\ ] Then , \\ [b = a + c + \\ frac {1}{ 3} = - \\ frac {17}{ 24} + \\ frac {8}{ 24} = - \\ frac {9}{ 24} = - \\ frac {3}{ 8}. \\ ] Finally , from the third equation : \\ [c - a - b = \\ frac {1}{ 4} \\ impl ies c = a + b + \\ frac {1}{ 4}. \\ ] Sub stit uting \\(c = a + b + \\ frac {1}{ 4}\\ ) into \\(a + b + c = - \\ frac {13}{ 12}\\ ), we get : \\ [a + b + a + b + \\ frac {1}{ 4} = - \\ frac {13}{ 12} \\ impl ies 2a + 2b + \\ frac {1}{ 4} = - \\ frac {13}{ 12} \\ impl ies 2a + 2b = - \\ frac {13}{ 12} - \\ frac {3}{ 12} = - \\ frac {16}{ 12} = - \\ frac {4}{ 3} \\ impl ies a + b = - \\ frac {2}{ 3}. \\ ] Then , \\ [c = a + b + \\ frac {1}{ 4} = - \\ frac {2}{ 3} + \\ frac {3}{ 12} = - \\ frac {8}{ 12} + \\ frac {3}{ 12} = - \\ frac {5}{ 12}. \\ ] Thus , we have : \\ [a = - \\ frac {7}{ 24}, \\ quad b = - \\ frac {3}{ 8} = - \\ frac {9}{ 24}, \\ quad c = - \\ frac {5}{ 12} = - \\ frac {10}{ 24}. \\ ] We need to find the value of \\ (\\ left | \\ log _2 (x^4 y ^3 z ^2) \\ right |\\ ): \\ [\\ log _2 (x^4 y ^3 z ^2) = 4a + 3b + 2c = 4 \\ left (- \\ frac {7}{ 24} \\ right) + 3 \\ left (- \\ frac {9}{ 24} \\ right) + 2 \\ left (- \\ frac {10}{ 24} \\ right) = - \\ frac {28}{ 24} - \\ frac {27}{ 24} - \\ frac {20}{ 24} = - \\ frac {75}{ 24} = - \\ frac {25}{ 8}. \\ ] Thus , \\ [\\ left | \\ log _2 (x^4 y ^3 z ^2) \\ right | = \\ frac {25}{ 8}. \\ ] The value of \\(m+n \\) is : \\ [25 + 8 = 33. \\ ] The answer is : \\ [\\ boxed {33}. \\ ]<|im_end|>

C Source Code

The following is the cleaned and finalized implementation of the **InfoGuide** framework. The code has been formatted to fit within the column margins.

```python
"""
InfoGuide implementation.
Loaded at inference time via environment variable
EVAL_INFOGUIDE_GENERATOR_PROGRAM
"""

import time
import math
import torch
import torch.nn.functional as F

from model_utils import (
    _apply_top_k_top_p_filtering,
    _get_eos_token_ids,
)

class InfoGuideGenerator:
    def __init__(
        self, model, tokenizer, uncond_prompt,
        cfg_scale=2.0, temperature=0.0, top_p=1.0,
        top_k=0, stop_id_sequences=None,
        add_special_tokens=True, apc_alpha=0.05,
        cfg_decay=0.997, warmup_steps=50,
        confidence_threshold=0.85, js_power=1.0,
        uncond_clip_range=5.0, neg_delta_weight=0.5,
        info_gain_weight=0.5,
        token_significance_boost=1.2,
        tsw_strength=0.8,
        temperature_compensation=0.3, **kwargs,
    ):
        self.model = model
        self.tokenizer = tokenizer
        self.uncond_prompt = uncond_prompt
        self.cfg_scale = cfg_scale
        self.temperature = temperature
        self.top_p = top_p
        self.top_k = top_k
        self.stop_id_sequences = stop_id_sequences or []
        self.add_special_tokens = add_special_tokens
        self.eos_token_ids = _get_eos_token_ids(tokenizer)

        self.apc_alpha = apc_alpha
        self.cfg_decay = cfg_decay
        self.warmup_steps = warmup_steps
        self.confidence_threshold = confidence_threshold
        self.js_power = js_power
        self.uncond_clip_range = uncond_clip_range
        self.neg_delta_weight = neg_delta_weight
        self.info_gain_weight = info_gain_weight
        self.token_significance_boost =
      token_significance_boost
        self.tsw_strength = tsw_strength
        self.temp_comp = temperature_compensation

    def _sample_token(self, logits):
        if self.temperature and self.temperature > 0:
            logits = logits / self.temperature
            logits = _apply_top_k_top_p_filtering(
                logits, top_k=self.top_k, top_p=self.top_p
            )
            probs = F.softmax(logits, dim=-1)
            return torch.multinomial(probs, num_samples=1)
        return torch.argmax(logits, dim=-1, keepdim=True)

    def _forward_prompt(self, text):
        inputs = self.tokenizer(
            text, return_tensors="pt",
            add_special_tokens=self.add_special_tokens
        )
        input_ids = inputs.input_ids.to(self.model.device)
        attn_mask =
      inputs.attention_mask.to(self.model.device)
        outputs = self.model(
            input_ids=input_ids,
            attention_mask=attn_mask,
            use_cache=True,
        )
        return (
            outputs.past_key_values,
            attn_mask,
            outputs.logits[:, -1, :].float()
        )

    def _append_token(self, token_ids, past_kv, attn_mask):
        next_attn = torch.ones_like(
            token_ids, dtype=attn_mask.dtype
        )
        attn_mask = torch.cat([attn_mask, next_attn], dim=-1)
        outputs = self.model(
            input_ids=token_ids,
            attention_mask=attn_mask,
            past_key_values=past_kv,
            use_cache=True,
        )
        return (
            outputs.past_key_values,
            attn_mask,
            outputs.logits[:, -1, :].float()
        )

    def _is_finished(self, generated_ids, token_id):
        if token_id in self.eos_token_ids:
            return True
        gen_text = self.tokenizer.decode(
            generated_ids, skip_special_tokens=True
        )
        return any(
            sw in gen_text for sw in self.stop_id_sequences
        )

    @torch.no_grad()
    def generate_one(
        self, prompt, max_new_tokens=2048,
        sample_id=None, sample_metadata=None
    ):
        start_time = time.time()

        kv_cond, cond_attn, logits_cond = (
            self._forward_prompt(prompt)
        )
        kv_uncond, uncond_attn, logits_uncond = (
            self._forward_prompt(self.uncond_prompt)
        )

        generated_ids = []
        uncond_pending_tokens = []
        finished = False
        token_metrics_list = []

        while len(generated_ids) < max_new_tokens and not
      finished:
            step = len(generated_ids)

            if step < self.warmup_steps:
                base_scale = self.cfg_scale
            else:
                decay_factor = self.cfg_decay ** (
                    step - self.warmup_steps
                )
                base_scale = 1.0 + (self.cfg_scale - 1.0) *
      decay_factor

            p_cond = F.softmax(logits_cond, dim=-1)
            p_max = torch.max(p_cond, dim=-1)[0].item()

            if p_max > self.confidence_threshold or
      (base_scale - 1.0) < 0.05:
                need_uncond = False
                logits_cfg = logits_cond
                masked_count = 0
                current_scale = 1.0
            else:
                need_uncond = True

                if uncond_pending_tokens:
                    pending_tensor = torch.tensor(
                        [uncond_pending_tokens],
                        device=self.model.device
                    )
                    kv_uncond, uncond_attn, logits_uncond = (
                        self._append_token(
                            pending_tensor, kv_uncond,
      uncond_attn
                        )
                    )
```

```
                uncond_pending_tokens = []

            p_uncond = F.softmax(logits_uncond, dim=-1)

            m = 0.5 * (p_cond + p_uncond)
            js_div = 0.5 * torch.sum(
                p_cond * torch.log(p_cond / m + 1e-10) +
                p_uncond * torch.log(p_uncond / m +
1e-10),
                dim=-1
            ).item()
            normalized_js = js_div / 0.693147

            entropy_cond = -torch.sum(
                p_cond * torch.log(p_cond + 1e-10), dim=-1
            ).item()
            entropy_uncond = -torch.sum(
                p_uncond * torch.log(p_uncond + 1e-10),
dim=-1
            ).item()
            info_gain = entropy_uncond - entropy_cond

            if info_gain > 0:
                ig_mod = 1.0 + self.info_gain_weight *
info_gain
            else:
                ig_mod = 1.0 / (
                    1.0 + self.info_gain_weight *
abs(info_gain)
                )

            multiplier = math.pow(
                normalized_js, self.js_power
            ) * ig_mod

            if normalized_js > 0.8:
                multiplier *=
self.token_significance_boost

            current_scale = min(
                5.0, 1.0 + (base_scale - 1.0) * multiplier
            )

            max_uncond_logits = torch.max(
                logits_uncond, dim=-1, keepdim=True
            )[0]
            min_uncond = max_uncond_logits -
self.uncond_clip_range
            logits_uncond_clipped = torch.maximum(
                logits_uncond, min_uncond
            )

            delta = logits_cond - logits_uncond_clipped
            if self.neg_delta_weight != 1.0:
                delta = torch.where(
                    delta > 0, delta,
                    delta * self.neg_delta_weight
                )

            if self.tsw_strength > 0.0:
                delta = delta * (1.0 - self.tsw_strength
* p_uncond)

            logits_cfg = logits_cond + (current_scale -
1.0) * delta

            if self.temp_comp > 0.0:
                comp_factor = 1.0 + (current_scale - 1.0)
* self.temp_comp
                logits_cfg = logits_cfg / comp_factor

            if self.apc_alpha > 0.0:
                max_logits = torch.max(
                    logits_cond, dim=-1, keepdim=True
                )[0]
                dynamic_apc = min(
                    1.0, self.apc_alpha * max(1.0,
current_scale)
                )
                threshold = max_logits +
math.log(dynamic_apc)
                mask = logits_cond < threshold
                logits_cfg = logits_cfg.masked_fill(
                    mask, -float('inf')
                )
                masked_count = mask.sum().item()
            else:
                masked_count = 0

        token_ids = self._sample_token(logits_cfg)
        token_id = int(token_ids[0, 0].item())
        generated_ids.append(token_id)

        token_metrics_list.append({
            "step": step,
            "p_max": p_max,
            "cfg_scale": current_scale,
            "masked_count": masked_count,
            "need_uncond": need_uncond
        })

        if self._is_finished(generated_ids, token_id):
            finished = True
            break

        kv_cond, cond_attn, logits_cond = (
            self._append_token(token_ids, kv_cond,
cond_attn)
        )

        if need_uncond:
            kv_uncond, uncond_attn, logits_uncond = (
                self._append_token(token_ids, kv_uncond,
uncond_attn)
            )
        else:
            uncond_pending_tokens.append(token_id)

    gen_text = self.tokenizer.decode(
        generated_ids, skip_special_tokens=True
    )
    for stop_word in self.stop_id_sequences:
        if stop_word in gen_text:
            gen_text = gen_text.split(stop_word)[0]

    elapsed = time.time() - start_time
    tps = len(generated_ids) / elapsed if elapsed > 0
else 0.0

    telemetry = {
        "sample_id": sample_id,
        "text": gen_text,
        "system_metrics": {
            "num_tokens": len(generated_ids),
            "tokens_per_second": tps,
        },
    }
    return gen_text, telemetry
```