BEYOND UNIFORM SOFT THINKING: ADAPTIVE GATING AND CALIBRATION OF SOFT TOKENS FOR LATENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Decoding-time soft thinking extends discrete chain-of-thought reasoning by feeding back a probability-weighted mixture of token embeddings rather than a single-token embedding. However, applying such mixtures at every step does not distinguish between steps with a clear leading token and those with several plausible continuations. Recent selective methods decide when to activate soft mixing, but construct mixtures with a fixed candidate budget or over the full vocabulary. Selecting the steps does not determine how widely to mix, since their next-token distributions can still differ in candidate count and probability concentration. We therefore propose *Adaptive Gating and Calibration of Soft Tokens* (AGC-ST), a training-free decoding method that determines both when and how widely to mix from the local next-token distribution. Its gate combines entropy and the top-1/top-2 probability margin to select steps where probability mass is broadly distributed and the two most likely tokens have similar probabilities. For each gated step, the Rényi-0.5 effective number, which reflects both candidate count and probability concentration, sets a support cap. A cumulative-mass prefix then selects the final mixing support within this cap. We evaluate AGC-ST on five reasoning models from the Qwen and Llama families, spanning 7B–32B parameters, across five benchmarks covering mathematical, coding, and scientific reasoning. With one shared hyperparameter configuration, AGC-ST achieves the highest accuracy among the evaluated methods in all 25 model–benchmark settings.

1 INTRODUCTION

Reasoning is a central capability of large language models (LLMs) and underlies their performance on mathematics, coding, and other complex tasks (Jaech et al., 2024; Guo et al., 2025; Yang et al., 2025; Grattafiori et al., 2024; Comanici et al., 2025). The dominant interface for eliciting such reasoning is chain-of-thought (CoT), which expresses intermediate reasoning steps as natural-language tokens (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023; Sprague et al., 2025; Pfau et al., 2024). This discrete format is readable and fits autoregressive decoding, but each step selects a single token from the next-token distribution and uses its embedding as the next input (Holtzman et al., 2020; Yao et al., 2023), which can bottleneck reasoning when multiple next-token alternatives remain plausible.

Latent reasoning seeks to alleviate this bottleneck by shifting the reasoning state from discrete tokens to continuous representations (Hao et al., 2024; Geiping et al., 2025; Xu et al., 2025; Zhu et al., 2025; Shen et al., 2025; Li & Liang, 2021; Deng et al., 2025; Gozeten et al., 2025). In contrast to trained latent-state methods and recurrent-depth approaches (Hao et al., 2024; Shen et al., 2025; Zheng & Lee, 2026; Deng et al., 2026; Geiping et al., 2025; Zhu et al., 2025; Chen et al., 2025), *decoding-time soft thinking* (Zhang et al., 2025) applies this shift at the input-embedding level without training. Rather than feeding forward a single selected token's embedding, it feeds forward a *soft token*, a mixture of token embeddings weighted by next-token probabilities, so that the next reasoning step is conditioned on several next-token alternatives rather than on one.

However, repeatedly applying soft mixing does not necessarily preserve the influence of multiple alternatives over a rollout. Autoregressive feedback can amplify small differences until one mixture component becomes dominant (Backour, 2026), while vanilla Soft Thinking has been observed to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

rely predominantly on the highest-probability token when predicting subsequent steps (Wu et al., 2025). Token-level analyses further highlight heterogeneity across reasoning steps, suggesting that reasoning-related signals are concentrated in a small subset of high-uncertainty or transition-like steps (Wang et al., 2025; Qian et al., 2025; Hwang et al., 2026), while many steps are execution-like continuations such as digits, operators, or punctuation. These observations motivate selective application of soft mixing, taking into account the differences across reasoning steps.

Several recent methods adopt selective soft-mixing strategies: SwiReasoning (Shi et al., 2026) switches between latent and explicit reasoning at the block level, SeLaR (Fu & Luo, 2026) gates step selection on Shannon entropy, and ThinkRouter (Xu et al., 2026) routes according to max-token confidence. These methods primarily decide *when* to apply soft mixing, but construct each mixture with a fixed candidate budget or over the full vocabulary. The mixing support determines both which candidates contribute and how much weight each receives, so step selection alone leaves open *how widely* to mix.

We therefore frame soft-token decoding around two complementary, distribution-conditioned decisions. At each reasoning step, we ask:

Two Questions for Soft-Token Methods

Gating: Should soft mixing be applied at this reasoning step?
Mixing: If so, how many candidate tokens should be mixed?

Fig. 1 grounds these two questions in next-token distributions observed along discrete CoT trajectories. Most reasoning steps have concentrated distributions with a clear leading token, making discrete decoding a natural default. High-entropy, low-margin steps are fewer, but leave several continuations plausible. This contrast motivates step-level *gating*. Within this high-entropy, low-margin region, the number of retained candidates varies even within individual model–benchmark runs, and steps with the same number of candidates still differ in how probability mass is distributed. Thus, deciding where to mix does not determine how widely to mix. These observations motivate adaptively *calibrating* the mixing breadth from each selected step's filtered distribution.

Guided by this analysis, we propose AGC-ST (**A**daptive **G**ating and **C**alibration of **S**oft **T**okens), a training-free decoding method illustrated in Fig. 2. For gating, AGC-ST combines two statistics of the filtered next-token distribution: entropy measures how widely probability mass is spread, while the top-1/top-2 margin indicates how closely matched the two most likely tokens are in probability. AGC-ST applies soft mixing only at steps with high entropy and a small margin; other reasoning steps retain discrete sampling. For calibration, we use the Rényi-0.5 effective number to translate the filtered distribution into an effective candidate count: it equals the number of candidates when their probabilities are uniform and decreases as probability mass becomes more concentrated. Rounding up this count sets the maximum support size, and within this cap a cumulative-mass prefix keeps only the leading candidates needed to reach a target mass.

Our contributions are: **(1)** We formulate soft-token decoding around two distribution-conditioned decisions: whether to apply soft mixing and how widely to mix. Analysis of discrete CoT traces motivates this distinction: high-entropy, low-margin steps are sparse, yet vary in retained candidate counts and probability concentration (Sec. 3.1). **(2)** We design AGC-ST to combine a joint entropy-margin gate with a Rényi-0.5 effective-number cap and a cumulative-mass prefix that selects the final mixing support within the cap (Sec. 3.2). **(3)** We evaluate AGC-ST on five reasoning models from the Qwen and Llama families (7B–32B) across five benchmarks covering mathematical, coding, and scientific reasoning. With one shared hyperparameter configuration, AGC-ST achieves the highest accuracy among the evaluated methods in all 25 model–benchmark settings (Sec. 4).

2 Preliminaries: Discrete Decoding and Soft Thinking

Discrete CoT Reasoning. Let $\mathcal{V}$ be the vocabulary of the LLM π_θ, and let $E : \mathcal{V} \to \mathbb{R}^d$ denote its token embedding map. Given a question Q, CoT reasoning generates a reasoning trace R followed by an answer sequence A. At each step, the model defines a next-token distribution $p_t(v) = \pi_\theta(v \mid X_{<t})$ over tokens $v \in \mathcal{V}$, where $X_{<t}$ contains the question and previously generated tokens.

In practical decoding, truncation rules such as top-k or top-p restrict p_t to a candidate set $C_t \subseteq \mathcal{V}$. For nonnegative weights w with $\sum_{u \in B} w(u) > 0$, let $\mathcal{N}_B(w)(v) = w(v) / \sum_{u \in B} w(u)$ for $v \in B$

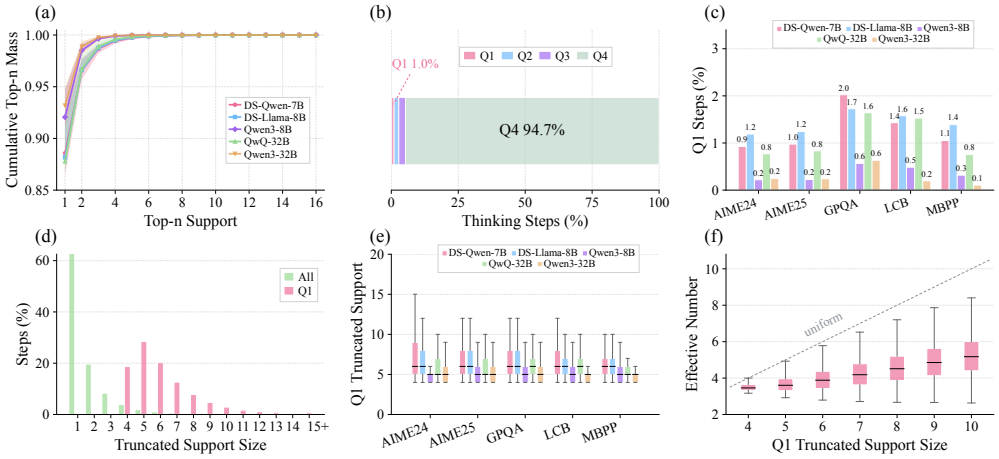

Figure 1: **Distributional motivation for gating and mixing.** Statistics use 25 discrete CoT model–benchmark runs. **Gating (a–c):** Most reasoning steps have concentrated distributions, whereas high-entropy, low-margin (Q1) steps are less frequent but spread probability mass across multiple plausible candidates. **Mixing breadth (d–f):** Q1 truncated support sizes vary across steps, including within individual runs (d,e). Even at a fixed truncated support size, the order-2 effective number varies across Q1 steps (f), reflecting differences in probability concentration. Shading in (a) shows the range across benchmarks; the dashed line in (f) marks the effective number for uniform probabilities. Region and effective-number definitions are given in Sec. 3.1; statistical details are in App. D.6.

denote normalization over B. The filtered next-token distribution is

$$\bar{p}_t(v) = \mathcal{N}_{C_t}(p_t)(v), \quad v \in C_t. \tag{1}$$

We refer to the cardinality $|C_t|$ of the filtered distribution's support as the *truncated support size*. The decoder then chooses one token by greedy decoding, $x_t = \arg\max_{v \in C_t} \bar{p}_t(v)$, or by sampling, $x_t \sim \bar{p}_t$. This token is appended to R as r_t during reasoning or to A as a_t during answer generation, and its embedding $E(x_t)$ is fed into the next step.

Soft Thinking Reasoning. Soft thinking (Zhang et al., 2025) replaces the single-token input of a reasoning step with a probability-weighted mixture of token embeddings: given the filtered distribution $\bar{p}_t$, it selects a support $\mathcal{S}_t \subseteq C_t$, normalizes its weights as $\tilde{p}_t(v) = \mathcal{N}_{\mathcal{S}_t}(\bar{p}_t)(v)$ for $v \in \mathcal{S}_t$, and forms

$$\tilde{e}_t = \sum\nolimits_{v \in \mathcal{S}_t} \tilde{p}_t(v) E(v). \tag{2}$$

The next reasoning step receives $\tilde{e}_t$ instead of a single-token embedding, while answer generation remains discrete. The nonnegative, normalized weights place $\tilde{e}_t$ in the convex hull of the selected token embeddings (Zhang et al., 2025; Zheng et al., 2025).

3 AGC-ST

In this section, we first examine the distributional evidence for gating and mixing breadth, then present the AGC-ST decoding procedure.

3.1 Distributional Motivation for Gating and Mixing

We analyze the truncated next-token distributions of discrete CoT traces from five reasoning models on five benchmarks, using one sampled trace per question (App. D.6). At each reasoning step t, we compute two statistics of the filtered distribution $\bar{p}_t$ over C_t: the Shannon entropy $H_t = -\sum_{v \in C_t} \bar{p}_t(v) \log \bar{p}_t(v)$ and the top-1/top-2 probability margin $m_t = \bar{p}_t(v_{t,1}) - \bar{p}_t(v_{t,2})$, where $v_{t,1}$ and $v_{t,2}$ are the two highest-probability tokens in C_t. With entropy threshold $\alpha = 1.25$ and margin threshold $\beta = 0.10$, we assign each step to one of four regions: Q1 has $H_t \geq \alpha, m_t \leq \beta$, Q2 has $H_t \geq \alpha, m_t > \beta$, Q3 has $H_t < \alpha, m_t \leq \beta$, and Q4 has $H_t < \alpha, m_t > \beta$.

Gating. Fig. 1(a,b) shows that probability mass generally concentrates on a small high-probability prefix and that most reasoning steps lie in Q4. When mass is concentrated on the top token, a soft

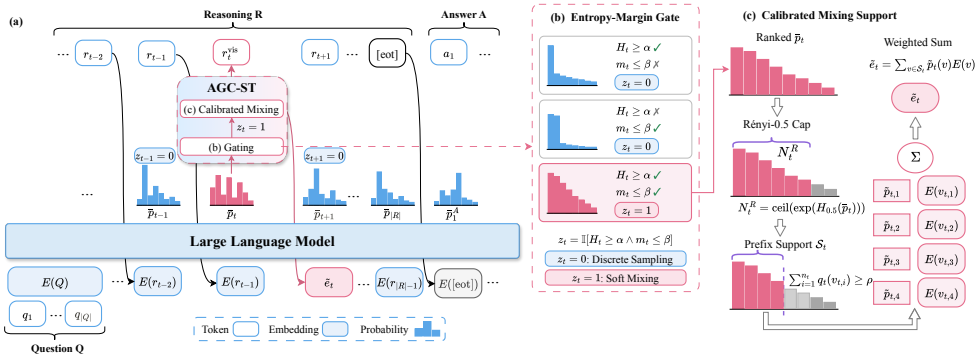

Figure 2: **Overview of AGC-ST. (a)** At each reasoning step, AGC-ST applies the gate in (b) and, when $z_t = 1$, the calibrated mixing in (c). At a gated step, the visible token r_t^{vis} is recorded, while the soft embedding $\tilde{e}_t$ is fed into the next step. Non-gated reasoning steps use discrete sampling, and answer generation remains discrete. **(b)** Three illustrative distributions show that soft mixing requires both high entropy and a small top-1/top-2 probability margin; satisfying either condition alone does not activate the gate. **(c)** Candidates ranked by $\bar{p}_t$ are restricted by a Rényi-0.5 effective-number cap. After renormalization within the cap, the smallest prefix reaching cumulative mass ρ defines $\mathcal{S}_t$. Its token embeddings are mixed using probabilities renormalized over $\mathcal{S}_t$.

mixture assigns little weight to the remaining candidates, making discrete decoding a natural default. In contrast, Q1 steps combine high entropy with a small probability gap between the two most likely tokens, leaving multiple continuations plausible. These steps account for a small fraction of reasoning steps but persist across all 25 model–benchmark runs (Fig. 1(c)). Tab. 4 provides complementary evidence: Q1 support is enriched in reasoning-transition and uncertainty markers, while Q4 support is enriched in digits, operators, and punctuation. This distributional and semantic contrast makes Q1 a natural target for step-level gating.

Mixing Breadth. Gating Q1 steps determines where soft mixing is applied, but leaves open how much support the mixture should include. Fig. 1(d,e) shows that truncated support sizes vary among Q1 steps, including within individual model–benchmark runs. Even at a fixed truncated support size, Fig. 1(f) shows variation in the order-2 effective number $1/\sum_v \bar{p}_t(v)^2$, with smaller values indicating greater probability concentration. A breakdown at 6 candidates shows that this variation is also present within every run (Fig. 7(b)). Changing the mixing support affects both which token embeddings enter the mixture and their normalized weights (App. B.3). These observations motivate adapting mixing breadth to each step's filtered distribution.

Accordingly, AGC-ST first gates soft mixing to Q1 steps and then calibrates the mixing support from each selected step's filtered distribution.

3.2 Adaptive Gating and Calibrated Mixing

As illustrated in Fig. 2, AGC-ST uses an entropy-margin gate to decide whether to mix and a Rényi-calibrated support rule to determine how widely to mix. A hybrid update then feeds either the resulting soft embedding or a single-token embedding into the next step.

Entropy-Margin Gating. At step t, AGC-ST computes the entropy H_t and the top-1/top-2 probability margin m_t of $\bar{p}_t$. The gate activates soft mixing only in the high-entropy, low-margin region,

$$z_t = \mathbb{I}\left[H_t \geq \alpha \ \wedge \ m_t \leq \beta\right], \tag{3}$$

where α and β are the entropy and margin thresholds, respectively. The entropy condition selects steps with broadly distributed probability mass. As Fig. 2(b) illustrates, high entropy can coexist with a substantial probability gap between the two most likely tokens, so the margin condition additionally requires their probabilities to be close.

Rényi-Calibrated Mixing Support. Let $(v_{t,1}, \ldots, v_{t,|C_t|})$ be the tokens in C_t sorted by descending $\bar{p}_t$. Sec. 3.1 shows that high-entropy, low-margin steps vary in both retained candidate counts and probability concentration. To account for this variation, we determine the mixing breadth from the local distribution rather than using a fixed top-k mixing budget. The Rényi effective number

$\exp(H_\gamma(\bar{p}_t))$ provides such a count: it equals M when M candidates are equally likely and decreases as probability concentrates on fewer of them. Here, H_γ denotes Rényi entropy of order γ. AGC-ST takes the ceiling of the order-0.5 effective number to obtain a distribution-conditioned support cap,

$$N_t^{\mathrm{R}} = \lceil \exp(H_{0.5}(\bar{p}_t)) \rceil = \left\lceil \left(\sum_{v \in C_t} \sqrt{\bar{p}_t(v)} \right)^2 \right\rceil . \tag{4}$$

Because Rényi effective numbers do not increase with the order, the order-0.5 effective number is no smaller than its order-1 counterpart $\exp(H_t)$. We select $\gamma = 0.5$ empirically among the tested orders in Sec. 4.3; App. C provides the derivation and support-range bounds.

Let $C_t^{\mathrm{R}} = \{v_{t,1}, \ldots, v_{t,N_t^{\mathrm{R}}}\}$ denote the capped candidate set. We renormalize within this set as $q_t(v) = \mathcal{N}_{C_t^{\mathrm{R}}}(\bar{p}_t)(v)$ for $v \in C_t^{\mathrm{R}}$, and choose the smallest prefix whose capped mass reaches ρ:

$$n_t = \min \left\{ n \in \{1, \ldots, N_t^{\mathrm{R}}\} : \sum_{i=1}^{n} q_t(v_{t,i}) \geq \rho \right\} . \tag{5}$$

The final support is $\mathcal{S}_t = \{v_{t,1}, \ldots, v_{t,n_t}\}$, with $\rho = 0.90$ in our main setting. In Fig. 2(c), the cap sets the maximum candidate count, while the mass prefix retains only the leading candidates needed to reach ρ under q_t. The resulting support satisfies $1 \leq n_t \leq N_t^{\mathrm{R}} \leq |C_t|$ (App. C.2).

Hybrid Soft-Discrete Update. For gated steps, AGC-ST forms $\tilde{e}_t$ from the selected support using Eq. 2. Non-gated steps retain discrete sampling. The next-step input representation is

$$h_t = \begin{cases} \tilde{e}_t, & z_t = 1, \\ E(r_t), \quad r_t \sim \bar{p}_t, & z_t = 0, \end{cases} \tag{6}$$

where h_t is the input representation at the next position, from which p_{t+1} is computed. As shown in Fig. 2(a), a gated step records $r_t^{\mathrm{vis}} = \arg\max_{v \in \mathcal{S}_t} \tilde{p}_t(v)$ for a readable trace, while a non-gated step records the sampled token r_t. Algorithm 1 in App. A gives the full decoding procedure.

4 Experiments

4.1 Experimental Setup

Benchmarks and Metric. We evaluate on five benchmarks in three domains: mathematics (AIME2024 and AIME2025; Zhang & Math-AI, 2024; 2025), science (GPQA-Diamond (GPQA-D); Rein et al., 2024), and coding (MBPP and LiveCodeBenchV5 (LCB); Austin et al., 2021; Jain et al., 2025). We report Avg@k (Chen et al., 2021), the mean correctness over k sampled responses per problem, with $k = 16$ on AIME2024/2025 and $k = 4$ on the other benchmarks.

Models and Baselines. We evaluate five reasoning LLMs: DeepSeek-R1-Distill-Qwen-7B (DS-Qwen-7B), DeepSeek-R1-Distill-Llama-8B (DS-Llama-8B), Qwen3-8B, QwQ-32B, and Qwen3-32B, covering both smaller distilled models and larger RL-trained models. We compare AGC-ST with seven inference-time baselines in three groups: discrete CoT decoding with greedy decoding (CoT Greedy) or sampling (CoT); full-step soft thinking, namely Soft Thinking (Zhang et al., 2025), Multiplex Thinking Inference (Multiplex-I) (Tang et al., 2026), and Soft-Thinking Gumbel (Soft-Thinking G) (Zheng et al., 2025); and selective soft thinking, namely SeLaR (Fu & Luo, 2026) and SwiReasoning (SwiR) (Shi et al., 2026). All main experiments use one shared setting, $(\alpha, \beta, \gamma, \rho) = (1.25, 0.10, 0.5, 0.90)$, selected as described in Sec. 4.3. Details on benchmarks, models, baselines, and evaluation are given in App. D.

4.2 Main Results

Tab. 1 reports results across five reasoning models from the Qwen and Llama families, spanning 7B–32B parameters. AGC-ST achieves the highest benchmark-averaged accuracy for all five models and the best accuracy in all 25 model–benchmark settings. Vanilla Soft Thinking falls below CoT Greedy in five-benchmark average accuracy for every model. Compared with vanilla Soft Thinking, AGC-ST improves this average by 3.23–12.15 percentage points, with the largest gains on the two

Table 1: Main results across five reasoning models (7B–32B) and five benchmarks. Mean is the unweighted average accuracy across the five benchmarks. Bold marks the highest accuracy in each column within a model.

Model	Method	AIME2024	AIME2025	GPQA-D	MBPP	LCB	**Mean**
		Avg@16	Avg@16	Avg@4	Avg@4	Avg@4	
DS-Qwen-7B	CoT Greedy	50.00	38.89	47.14	79.25	30.11	49.08
	CoT	54.79	40.63	48.74	86.28	38.35	53.76
	Soft Thinking	53.96	23.33	46.21	83.27	31.90	47.73
	Multiplex-I	48.96	35.83	47.10	82.78	33.15	49.56
	Soft-Thinking G	54.58	40.42	49.37	85.99	38.26	53.72
	SeLaR	50.63	37.29	50.13	83.07	26.61	49.55
	SwiR	53.75	39.79	48.74	86.67	38.17	53.42
	AGC-ST	**56.88**	**42.50**	**50.63**	**87.45**	**39.61**	**55.41**
DS-Llama-8B	CoT Greedy	31.11	22.22	37.21	75.36	27.72	38.72
	CoT	41.88	31.04	43.81	85.21	38.53	48.09
	Soft Thinking	22.08	22.92	38.01	77.72	27.51	37.65
	Multiplex-I	36.25	30.42	42.05	80.06	32.35	44.22
	Soft-Thinking G	42.50	31.25	44.44	84.53	38.62	48.27
	SeLaR	35.21	28.54	44.95	82.30	23.75	42.95
	SwiR	43.96	29.79	43.18	84.14	39.07	48.03
	AGC-ST	**44.17**	**32.29**	**46.46**	**86.38**	**39.70**	**49.80**
Qwen3-8B	CoT Greedy	68.89	65.56	57.58	89.88	53.29	67.04
	CoT	76.04	67.71	60.23	94.26	59.32	71.51
	Soft Thinking	69.58	58.75	56.06	92.02	52.69	65.82
	Multiplex-I	72.92	66.67	59.60	92.12	56.81	69.62
	Soft-Thinking G	74.58	67.29	58.96	94.55	59.77	71.03
	SeLaR	74.38	68.33	59.09	94.16	49.19	69.03
	SwiR	75.00	66.67	58.96	94.55	59.32	70.90
	AGC-ST	**77.50**	**70.00**	**61.11**	**94.84**	**59.95**	**72.68**
QwQ-32B	CoT Greedy	76.67	68.89	62.96	96.89	60.93	73.27
	CoT	77.50	69.58	63.26	97.37	62.72	74.09
	Soft Thinking	77.71	67.50	62.88	95.14	56.99	72.04
	Multiplex-I	76.46	66.04	62.37	96.89	61.83	72.72
	Soft-Thinking G	78.75	67.71	63.01	97.67	62.54	73.93
	SeLaR	76.88	65.63	60.86	94.55	47.22	69.03
	SwiR	78.33	69.17	64.65	97.37	62.90	74.48
	AGC-ST	**79.38**	**70.21**	**65.66**	**97.76**	**63.35**	**75.27**
Qwen3-32B	CoT Greedy	78.89	73.33	62.63	95.59	65.23	75.13
	CoT	81.25	71.88	65.15	96.11	65.86	76.05
	Soft Thinking	79.58	68.54	64.02	95.91	59.14	73.44
	Multiplex-I	79.17	72.92	64.90	96.01	65.68	75.73
	Soft-Thinking G	80.83	70.42	65.40	96.11	65.59	75.67
	SeLaR	80.00	72.71	64.02	94.16	56.99	73.58
	SwiR	80.00	73.13	65.53	96.11	65.41	76.04
	AGC-ST	**81.88**	**74.38**	**66.92**	**96.50**	**66.04**	**77.14**

DeepSeek-R1-distilled models. These gains also extend beyond recovering the performance lost by vanilla Soft Thinking: relative to CoT sampling, its five-benchmark mean gains are 1.09–1.71 percentage points. The strongest baseline by average accuracy varies across models: CoT leads on DS-Qwen-7B, Qwen3-8B and Qwen3-32B, Soft-Thinking G on DS-Llama-8B, and SwiR on QwQ-32B. AGC-ST exceeds these respective baseline averages by 0.79–1.65 percentage points. On the five-benchmark mean, paired 95% bootstrap intervals for the gains lie entirely above zero for all five comparisons with CoT and for 34 of the 35 model–baseline comparisons; the exception is SwiR on QwQ-32B (App. E.1).

Decoding Overhead. Fixed-512-token tests on DS-Qwen-7B and AIME2024 show no consistent additional per-token decoding overhead for AGC-ST relative to CoT (App. E.2).

4.3 Ablation and Hyperparameter Analysis

Tabs. 2 and 3 compare design choices on the AIME settings used for hyperparameter selection.

Gating. Tab. 2 removes either gate condition while keeping support calibration fixed. Entropy-only gating admits Q1∪Q2 and margin-only gating admits Q1∪Q3, whereas the joint gate selects Q1.

Table 2: Gate ablation and hyperparameter selection on AIME2024/2025, with support calibration fixed. Rate is the percentage of gated thinking steps.

Settings	DS-Qwen-7B				DS-Llama-8B				Mean
	AIME2024		AIME2025		AIME2024		AIME2025		
	Avg@16	Rate	Avg@16	Rate	Avg@16	Rate	Avg@16	Rate	
(a) Reference									
CoT	54.79	0%	40.63	0%	41.88	0%	31.04	0%	42.09
Soft Thinking	53.96	100%	23.33	100%	22.08	100%	22.92	100%	30.57
(b) Gated region									
Entropy only (α=1.25, β=1.00)	54.38	3.33%	40.42	3.60%	36.04	3.66%	28.75	3.93%	39.90
Margin only (α=0.00, β=0.10)	54.17	3.05%	39.38	3.16%	41.25	3.70%	30.21	3.87%	41.25
(c) Q1 boundary									
Tighter (α=1.50, β=0.05)	56.25	0.37%	41.25	0.40%	43.54	0.38%	31.67	0.43%	43.18
Medium (α=1.00, β=0.15)	55.00	2.74%	40.00	2.80%	42.29	3.17%	30.00	3.50%	41.82
Looser (α=0.75, β=0.20)	52.50	4.89%	40.21	5.06%	41.46	5.31%	29.58	5.73%	40.94
AGC-ST (α=1.25, β=0.10)	56.88	1.14%	42.50	1.18%	44.17	1.52%	32.29	1.70%	**43.96**

Table 3: Support ablation and hyperparameter selection on AIME2024/2025, with the joint gate fixed.

Settings	DS-Qwen-7B		DS-Llama-8B		QwQ-32B		Mean
	AIME2024	AIME2025	AIME2024	AIME2025	AIME2024	AIME2025	
	Avg@16	Avg@16	Avg@16	Avg@16	Avg@16	Avg@16	
(a) Reference							
CoT	54.79	40.63	41.88	31.04	77.50	69.58	52.57
Soft Thinking	53.96	23.33	22.08	22.92	77.71	67.50	44.58
(b) Support rule variants							
Fixed size (top-4)	55.00	38.54	42.08	31.67	76.25	68.13	51.94
Fixed size (top-6)	55.00	39.17	41.67	30.83	76.46	67.92	51.84
Full support (S_t=C_t)	55.83	36.88	41.04	29.58	76.04	68.13	51.25
Prefix only (ρ=0.90)	54.58	38.33	40.83	32.50	75.63	67.50	51.56
Rényi cap only (γ=0.5)	52.50	40.42	41.46	32.08	76.67	66.46	51.60
Rényi cap only (γ=1.5)	54.79	40.83	39.79	30.83	77.29	69.17	52.12
(c) Prefix mass ρ							
γ=0.5, ρ=0.85	53.75	37.08	45.00	30.00	76.88	69.38	52.01
γ=0.5, ρ=0.95	58.33	39.58	42.08	30.00	76.88	69.79	52.78
(d) Rényi order γ							
γ=0.33, ρ=0.90	53.13	43.75	41.25	29.58	78.96	67.08	52.29
γ=0.75, ρ=0.90	56.88	42.08	44.17	31.67	78.96	69.38	53.85
AGC-ST (γ=0.5, ρ=0.90)	56.88	42.50	44.17	32.29	79.38	70.21	**54.24**

Removing the margin or entropy condition lowers mean accuracy by 4.06 or 2.71 points, respectively. These results support the joint entropy-margin gate under the evaluated settings.

Mixing Support. Tab. 3 fixes the joint gate and changes the support used to form each mixture. In block (b), fixed top-4 and top-6 restrict mixing to the highest-probability candidates, while full support uses all candidates in C_t after the shared top-k/top-p filtering. Prefix only applies the cumulative-mass rule directly to $\bar{p}_t$, and cap only retains the Rényi-capped prefix without the subsequent mass-based selection. Each variant renormalizes over its selected support before mixing.

AGC-ST outperforms fixed top-4, fixed top-6, and full support in all six settings, with mean gains of 2.30–2.99 percentage points. At the default Rényi order, removing the cap or the mass-prefix stage lowers mean accuracy by 2.68 or 2.64 points, respectively. These results support the complete cap–prefix rule over the tested alternatives.

Hyperparameter Selection and Sensitivity. We select α, β on DS-Qwen-7B and DS-Llama-8B over AIME2024/2025, and γ, ρ on the same benchmarks with QwQ-32B added, maximizing accuracy over the respective settings. In Tab. 2(c), tightening or relaxing the joint gate lowers accuracy relative to the selected thresholds $(\alpha, \beta) = (1.25, 0.10)$. The most restrictive gate is therefore not the best-performing one among those tested. Among the masses tested in Tab. 3(c), $\rho = 0.90$ achieves the highest average accuracy across the six AIME settings. The cap-then-prefix design motivates considering orders below one (App. C.1). At $\rho = 0.90$, $\gamma = 0.5$ achieves higher mean accuracy across the six AIME settings than the other orders tested in Tab. 3(d), and we fix this configuration for the main evaluation. GPQA-D, MBPP, LCB, and both Qwen3 models were not used for hyperparameter

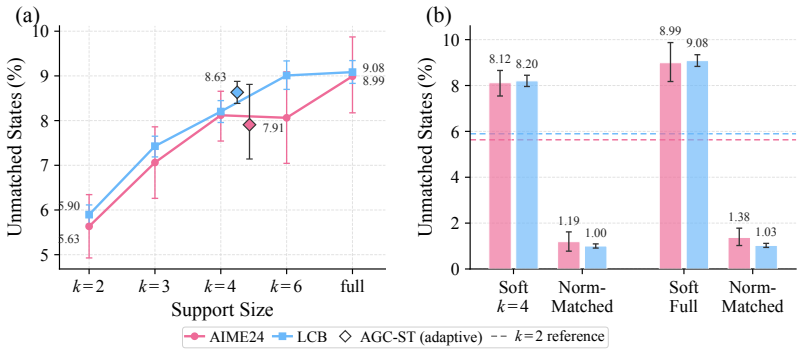

Figure 3: Same-state support-size interventions on DS-Qwen-7B: 3,781 gated states from 30 AIME2024 problems and 51,118 from 279 LCB problems. Error bars are 95% problem-level bootstrap intervals. **(a)** Unmatched rates: no single-token input from C_t reproduces the mixture's next-step argmax. Diamonds mark AGC-ST at its mean support size; k=6 uses only states with $|C_t| \geq 6$. **(b)** Mixtures and their norm-matched top-1 controls; dashed lines mark the k=2 rates.

selection. Across the 19 model–benchmark settings that involve them, AGC-ST achieves the highest accuracy among the evaluated methods.

4.4 Analysis of Adaptive Gating and Mixing

Does Support Size Matter Locally? Sec. 4.3 compares support rules by accuracy. We next examine whether support size changes the next-step prediction at states selected by AGC-ST. We hold the prefix and KV cache fixed, then feed in turn the mixture at each tested support size and each single-token input from C_t. The state must be held fixed, because AGC-ST sets its support from the current distribution: comparing different steps would change the state along with the width. An input is *unmatched* when its next-step argmax differs from that of every single-token input from C_t, including the top-1 token, and the *unmatched rate* is the percentage of gated states at which this occurs.

On DS-Qwen-7B, Fig. 3 summarizes three observations at AGC-ST-selected states. (1) Under the deployed support rule, the mixture is unmatched at 7.91% of gated states on AIME2024 and 8.63% on LCB. (2) On the same states, widening the support from k=2 to full raises this rate by about three percentage points on both benchmarks. (3) Rescaling the top-1 embedding to the mixture's norm does not reproduce the mixture's unmatched rate: at both k=4 and full support, the rescaled top-1 is unmatched at about 1% of gated states on both benchmarks, compared with 8–9% for the mixture. Support size, which calibration sets, can therefore alter the next-step prediction at gated states. These are one-step measurements, and the unmatched rate is not an accuracy proxy. AGC-ST does not maximize this rate, staying below full support on both benchmarks; the accuracy comparison with full support is in Sec. 4.3. Further controls and validation are in App. E.3.

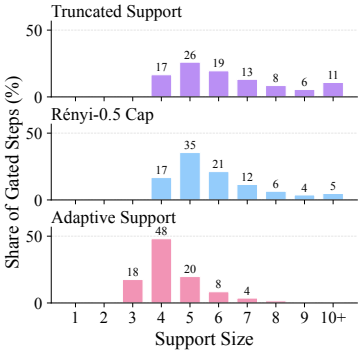

Figure 4: Support-size distributions across calibration stages on gated steps, aggregated over all models and benchmarks. Each row sums to 100%.

Adaptive Support Behavior. We next examine how AGC-ST sets the support size once a step is gated. Fig. 4 reports the support size of all gated steps at each calibration stage, collected from one sampled rollout per question across the 25 model–benchmark combinations. The support contracts from the truncated support to the Rényi-0.5 cap and then to the final support $\mathcal{S}_t$, labeled adaptive support in the figures, yet the final support still spans several widths. This matches the two-stage design: the cap bounds the candidate range, and the retained-mass prefix sets the final width from the distribution within the cap. Pairwise and model-wise views are given in Figs. 8 and 9 (App. E.4).

Qualitative Case Study. Fig. 5 shows two segments of a DS-Llama-8B trace on LCB. Each column lists the ranked candidates of one thinking step, shaded by probability; green boxes mark sampled

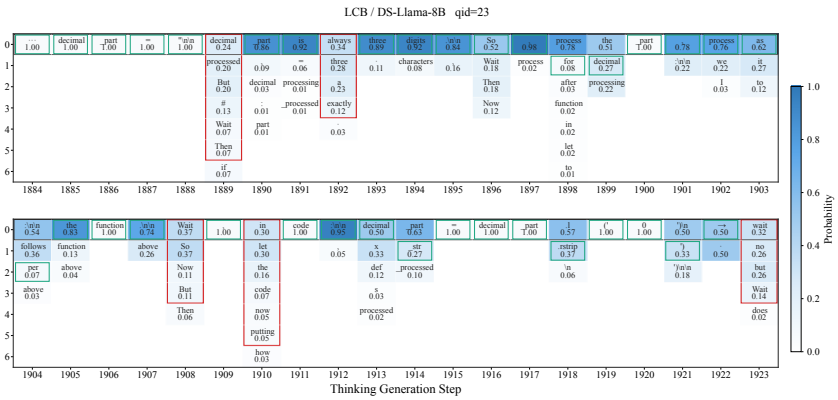

Figure 5: Segment-level heatmap for LCB with DS-Llama-8B, question id 23.

tokens, and red boxes mark gated steps, with height equal to the support size. Most steps remain discrete, including code fragments and near-deterministic tokens. Soft mixing is activated only at a few steps where several continuations remain plausible, such as steps 1908 and 1923, where “Wait”, “So”, “no”, and “but” share the probability mass. More segments and cases are in App. H.1.

5 Related Work

Latent Reasoning. Latent reasoning augments or replaces explicit CoT with internal computation, through implicit-reasoning training (Zelikman et al., 2022; Wang & Zhou, 2024; Zelikman et al., 2024; Goyal et al., 2024; Ye et al., 2025; Zheng & Lee, 2026), hidden-state feedback or distillation (Hao et al., 2024; Shen et al., 2025; Deng et al., 2026), or recurrent depth (Geiping et al., 2025; Zhu et al., 2025; Chen et al., 2025; Fu et al., 2025; Mohtashami et al., 2025). Closest to our setting are token-interface methods, which keep autoregressive decoding but feed continuous inputs (Li & Liang, 2021; Xu et al., 2025; Deng et al., 2025). Unlike these training-based approaches, AGC-ST is a training-free decoding-time method.

Soft Thinking. Soft Thinking (Zhang et al., 2025) is the closest training-free method to ours: it replaces each reasoning token with a probability-weighted mixture of token embeddings, so that the next step can condition on several candidate continuations. Such mixtures can collapse toward a dominant component (Wu et al., 2025; Backour, 2026). Later methods add stochasticity or training to soft thinking (Tang et al., 2026; Zheng et al., 2025), or restrict mixing to steps or blocks selected by a confidence signal (Shi et al., 2026; Fu & Luo, 2026; Xu et al., 2026). Their mixtures still use a fixed top-n support, a fixed number of sampled tokens, or the full vocabulary. AGC-ST instead gates jointly on entropy and the top-1/top-2 margin, selecting steps where probability is both spread out and contested, and calibrates the mixing breadth of each gated step from its local distribution. Truncation samplers (Holtzman et al., 2020; Tan et al., 2026) also adapt the candidate set to the distribution, but to sample one token rather than to form a mixture.

6 Conclusion

We presented AGC-ST, a training-free decoding method that combines entropy-margin gating with a Rényi-calibrated support rule to determine both when and how widely to apply soft mixing. With one shared configuration across five reasoning models and five benchmarks, AGC-ST achieves the highest accuracy among the evaluated methods in all 25 model–benchmark settings. AIME ablations favor the joint gate and the complete support rule over the tested alternatives in mean accuracy, while same-state interventions show that support size can alter next-step predictions at gated states.

Limitations and Future Work. First, AGC-ST makes its gating and support decisions at each step from the current distribution; coordinating these decisions over multi-token segments is a natural extension. Second, our evaluation covers five reasoning models from 7B to 32B on mathematics, science, and coding; larger models and other task types remain to be tested.

AI USE STATEMENT

We have not used generative AI tools for any task with required disclosure. Additionally, we used generative AI tools to polish the writing, including grammar, wording, and readability, and to search for and verify related literature. We have reviewed all AI-assisted work: all AI-suggested edits were reviewed and revised by the authors, and cited works were checked against the original papers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI. The open-source LLMs evaluated in this paper are the subjects of our experiments; we do not train or release a new model.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Ali Backour. The dynamics of continuous mixture collapse in language models. *arXiv preprint arXiv:2609.02049*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28241–28259, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. LLM latent reasoning as chain of superposition. *arXiv preprint arXiv:2510.15522*, 2025.

Jingcheng Deng, Zihao Wei, Liang Pang, Junhong Wu, Shicheng Xu, Zenghao Duan, and Huawei Shen. Latent-grpo: Group relative policy optimization for latent reasoning. *arXiv preprint arXiv:2604.27998*, 2026.

Renyu Fu and Guibo Luo. SeLaR: Selective latent reasoning in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

Tianyu Fu, Yichen You, Zekai Chen, Guohao Dai, Huazhong Yang, and Yu Wang. Think-at-hard: Selective latent iterations to improve reasoning language models. *arXiv preprint arXiv:2511.08577*, 2025.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024.

Halil Alperen Gozeten, M Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. *arXiv preprint arXiv:2505.23648*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Mark O Hill. Diversity and evenness: a unifying notation and its consequences. *Ecology*, 54(2): 427–432, 1973.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

Jaehui Hwang, Dongyoon Han, Sangdoo Yun, and Byeongho Heo. Oops, wait: Token-level signals as a lens into llm reasoning. *arXiv preprint arXiv:2601.17421*, 2026.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Lou Jost. Entropy and diversity. *Oikos*, 113(2):363–375, 2006.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 388–395, 2004.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. Cotformer: A chain of thought driven architecture with budget-adaptive computation cost at inference. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Dachuan Shi, Abedelkadir Asi, Keying Li, Xiangchi Yuan, Leyan Pan, Wenke Lee, and Wen Xiao. SwiReasoning: Switch-thinking in latent and explicit for pareto-superior reasoning LLMs. In *International Conference on Learning Representations*, 2026.

Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Runyan Tan, Shuang Wu, and Phillip Howard. p-less sampling: A robust hyperparameter-free approach for llm decoding. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yao Tang, Li Dong, Yaru Hao, Qingxiu Dong, Furu Wei, and Jiatao Gu. Multiplex Thinking: Reasoning via token-wise branch-and-merge. *arXiv preprint arXiv:2601.08808*, 2026.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. *Advances in Neural Information Processing Systems*, 37:66383–66409, 2024.

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

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

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. LLMs are single-threaded reasoners: Demystifying the working mechanism of Soft Thinking. *arXiv preprint arXiv:2508.03440*, 2025.

Xin Xu, Tong Yu, Xiang Chen, Haoliang Wang, Julian McAuley, and Saayan Mitra. ThinkRouter: Efficient reasoning via routing thinking between latent and discrete spaces. *arXiv preprint arXiv:2602.11683*, 2026.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

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

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. *arXiv preprint arXiv:2510.04182*, 2025.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. *Advances in Neural Information Processing Systems*, 38:169819–169851, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft Thinking: Unlocking the reasoning potential of LLMs in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025.

Zhi Zheng and Wee Sun Lee. Beyond imitation: Reinforcement learning for active latent planning. *arXiv preprint arXiv:2601.21598*, 2026.

Zhi Zheng, Yu Gu, Wei Liu, Yee Whye Teh, and Wee Sun Lee. Soft-grpo: Surpassing discrete-token llm reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*, 2025.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

APPENDIX CONTENTS

A **AGC-ST Decoding Algorithm** 15

B **Additional Motivation and Calibration Analysis** 16

B.1 Additional Distributional Motivation 16

B.2 Support-Token Enrichment 16

B.3 Why Support Size Changes Mixture Weights 17

C **Rényi Effective Number and Adaptive Range** 17

C.1 Rényi Effective-Number Background 18

C.2 Adaptive Support Range 19

D **Experimental Details** 20

D.1 Model Details 20

D.2 Benchmark Details 20

D.3 Implementation Details 21

D.4 Baseline Implementation and Hyperparameters 21

D.5 Evaluation Protocol 22

D.6 Analysis Traces 23

D.7 Licenses 23

E **Additional Experimental Results** 23

E.1 Statistical Significance of Gains over Baselines 23

E.2 Inference Latency and Decode Efficiency 24

E.3 Same-State Interventions: Controls and Validation 25

E.4 Support Calibration: Pairwise Structure and Model-Wise Decomposition 26

F **Broader Impacts** 27

G **Benchmark Prompts** 28

H **Qualitative Analysis** 30

H.1 Additional Qualitative Heatmap Cases 30

H.2 Qualitative Output Examples 33

A AGC-ST DECODING ALGORITHM

Algorithm 1 gives the full decoding procedure for AGC-ST. Generation is controlled by a maximum new-token budget rather than a separate reasoning-step budget. AGC-ST changes only the thinking phase before the end-of-thinking token; after that point, decoding follows the ordinary discrete answer path. The blue lines mark the AGC-ST-specific intervention. When the entropy-margin gate fires, the support is formed by the Rényi cap of order $\gamma = 0.5$ followed by the cumulative-mass prefix with $\rho = 0.90$, and the resulting soft embedding is fed as the next input.

Algorithm 1 AGC-ST Decoding

Require: Question Q, model π_θ, embedding map E, maximum new tokens L, thinking and answer decoding rules, entropy threshold α, margin threshold β, Rényi order $\gamma = 0.5$, prefix mass $\rho = 0.90$
Ensure: Answer A

Initialize generated tokens $O \leftarrow \emptyset$, input representations $Y \leftarrow E(Q)$, and phase THINK $\leftarrow$ `true`
for $t = 1, \ldots, L$ **do**
  Compute the next-token distribution $p_t(v)$ from the current input representations Y
  Choose the phase-specific candidate set C_t
  Set $\bar{p}_t(v) = p_t(v) / \sum_{u \in C_t} p_t(u)$ for $v \in C_t$ (Eq. 1)
  if THINK $=$ `false` **then**
    Sample $a_t \sim \bar{p}_t$ ▷ Answer decoding
    Append a_t to O and append $E(a_t)$ to Y
    if a_t is the end-of-sequence token **then**
      break
    end if
    continue
  end if
  Set $K_t = |C_t|$ and rank the candidates so that $\bar{p}_t(v_{t,1}) \geq \cdots \geq \bar{p}_t(v_{t,K_t})$
  Compute $H_t = -\sum_{v \in C_t} \bar{p}_t(v) \log \bar{p}_t(v)$ and margin $m_t = \bar{p}_t(v_{t,1}) - \bar{p}_t(v_{t,2})$
  Set $z_t = \mathbb{I}[H_t \geq \alpha \wedge m_t \leq \beta]$ (Eq. 3) ▷ Gating
  if $z_t = 0$ **then**
    Sample $r_t \sim \bar{p}_t$
    Append r_t to O and append $E(r_t)$ to Y
  else
    Compute $N_t^{\mathrm{R}} = \left\lceil \left(\sum_{i=1}^{K_t} \sqrt{\bar{p}_t(v_{t,i})}\right)^2 \right\rceil$ (Eq. 4) ▷ Support: Rényi cap
    Set $C_t^{\mathrm{R}} = \{v_{t,1}, \ldots, v_{t,N_t^{\mathrm{R}}}\}$
    Compute $q_t(v) = \bar{p}_t(v) / \sum_{u \in C_t^{\mathrm{R}}} \bar{p}_t(u)$ for $v \in C_t^{\mathrm{R}}$
    Choose $n_t = \min\{n : \sum_{i=1}^{n} q_t(v_{t,i}) \geq \rho\}$ (Eq. 5) ▷ Support: mass prefix
    Set $\mathcal{S}_t = \{v_{t,1}, \ldots, v_{t,n_t}\}$
    Compute $\tilde{p}_t(v) = \bar{p}_t(v) / \sum_{u \in \mathcal{S}_t} \bar{p}_t(u)$ for $v \in \mathcal{S}_t$
    Set $\tilde{e}_t = \sum_{v \in \mathcal{S}_t} \tilde{p}_t(v) E(v)$ (Eq. 2) ▷ Soft input
    Set the visible token $r_t^{\mathrm{vis}} = \arg\max_{v \in \mathcal{S}_t} \tilde{p}_t(v)$
    Append r_t^{vis} to O and append $\tilde{e}_t$ to Y
  end if
  if the appended token is the end-of-thinking token **then**
    THINK $\leftarrow$ `false`
  end if
end for
Extract the answer A from O after the end-of-thinking token
return A

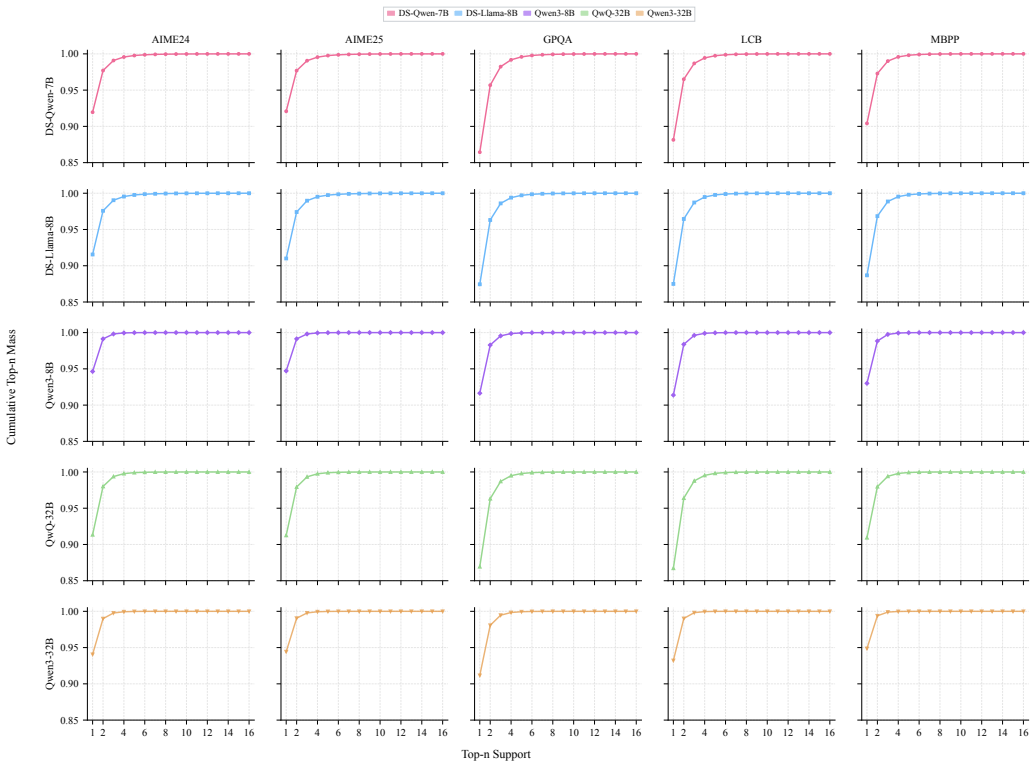

Figure 6: Model–benchmark breakdown of cumulative top-n mass in truncated next-token distributions from 25 CoT runs (five models and five benchmarks). Top-n is defined in App. D.6.

B Additional Motivation and Calibration Analysis

B.1 Additional Distributional Motivation

Fig. 1 presents both aggregate statistics and individual-run distributions. Here we provide complementary breakdowns. Fig. 6 shows that mean cumulative top-n mass rises quickly over the first few candidates in every run. Fig. 7 further examines probability concentration. Panel (a) shows that Q1 steps accumulate probability mass more slowly on average than the overall population. Panel (b) fixes the truncated support at 6 candidates and separates the 25 runs. The effective number varies across Q1 steps within every run, with interquartile ranges of 0.88–1.01, while the run medians are close to one another. At this support size, the variation is therefore not solely due to differences between runs.

B.2 Support-Token Enrichment

We conduct a support-token enrichment audit on the same 25 CoT model–benchmark runs as Fig. 1. Each thinking step is assigned to one of the regions Q1–Q4 with $\alpha = 1.25$ and $\beta = 0.10$. For each region, we accumulate token probabilities over the filtered support and compute each token's share of the region-level support mass. Tab. 4 compares Q1 and Q4 by the ratio of these support-mass shares. Tokens are decoded with their respective model tokenizers and merged by decoded text. For each direction, the table reports the ten highest enrichment ratios among tokens appearing in at least 1,000 steps in each region and accounting for at least 0.2% of support mass in the target region. The contrast indicates that Q1 support contains more reasoning-transition and uncertainty markers, whereas Q4 support is dominated by arithmetic, punctuation, and execution tokens.

The selected Q1-enriched tokens include `<sp>let`, `<sp>wait`, `<sp>consider`, and `<sp>maybe`. Q4-enriched tokens include digits and operators such as `2`, `1`, and `=`. These patterns provide descriptive evidence for interpreting the entropy-margin regions; token identity is not used by the gate.

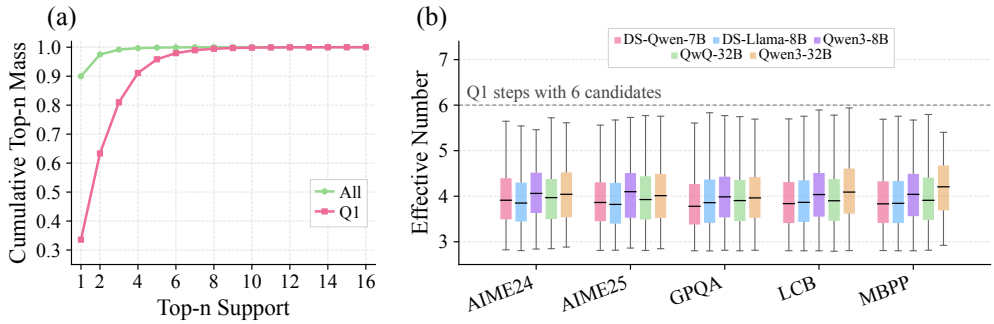

Figure 7: Probability shape behind the mixing-breadth motivation, using the same 25 CoT runs as Fig. 1. (a) Mean cumulative top-n mass for all and Q1 steps, averaging over steps across runs. (b) Order-2 effective number $1/\sum_v \bar{p}_t(v)^2$ of Q1 steps with 6 candidates in each model–benchmark run; the dashed line marks the uniform value of 6. Box-plot conventions are described in App. D.6.

Table 4: Support-token enrichment for Q1 and Q4 steps. For a token v, $f_Q(v)$ denotes the share of filtered-support probability mass assigned to v within region Q, accumulated over all 25 CoT model–benchmark runs with thresholds $\alpha = 1.25, \beta = 0.10$. The ratio column compares the target region with the reference region directly. Q1 support is enriched with lexical markers of reasoning transitions and uncertainty, while Q4 support is enriched with arithmetic, punctuation, and execution tokens.

Q1-enriched support tokens				**Q4-enriched support tokens**			
Token	$f_{Q1}(v)$	$f_{Q4}(v)$	Ratio	Token	$f_{Q4}(v)$	$f_{Q1}(v)$	Ratio
<sp>let	0.749%	0.063%	11.9×	2	2.124%	0.064%	32.9×
<sp>wait	0.529%	0.046%	11.5×	1	2.835%	0.110%	25.7×
<sp>consider	0.216%	0.020%	10.8×	=	0.782%	0.088%	8.9×
<sp>maybe	0.882%	0.096%	9.1×	<sp>-	0.566%	0.069%	8.1×
<sp>what	0.316%	0.039%	8.0×	<sp>=	0.814%	0.113%	7.2×
<sp>since	0.502%	0.065%	7.7×	)	0.736%	0.112%	6.6×
Another	0.255%	0.033%	7.7×	-	0.535%	0.114%	4.7×
<sp>when	0.637%	0.091%	7.0×	<sp>be	0.515%	0.145%	3.6×
<sp>another	0.232%	0.035%	6.6×	<sp>	3.060%	0.956%	3.2×
This	0.257%	0.044%	5.9×	<sp>→	0.327%	0.114%	2.9×

B.3 Why Support Size Changes Mixture Weights

Support size affects soft thinking not only by deciding which token embeddings enter the mixture, but also by changing their renormalized weights. Let $(v_{t,1}, \ldots, v_{t,|C_t|})$ be sorted by descending $\bar{p}_t$, and consider the prefix support $\mathcal{S}_t^{(n)} = \{v_{t,1}, \ldots, v_{t,n}\}$. The renormalized weight of the i-th retained token is

$$w_{t,i}^{(n)} = \frac{\bar{p}_t(v_{t,i})}{\sum_{j=1}^{n} \bar{p}_t(v_{t,j})}, \quad i \le n. \tag{7}$$

When the retained candidates have similar probabilities, $w_{t,i}^{(n)} \approx 1/n$. Thus, increasing the support size from $n = 4$ to $n = 6$ changes the approximate weight of each retained token from 0.25 to 0.167. Support size is therefore not merely a truncation budget: a small support amplifies the head candidates, whereas an overly large support can dilute them by admitting weaker tail candidates. This motivates calibrating the support at each step selected by the entropy-margin gate.

C Rényi Effective Number and Adaptive Range

The support rule relies on two facts: exponentiated Rényi entropy gives a real-valued effective number whose ceiling defines an integer support cap, and the final prefix support remains bounded by the

filtered candidate set and this cap. We first derive the Rényi-0.5 closed form, then record these range constraints.

C.1 Rényi Effective-Number Background

Rényi entropy provides a family of distribution-shape measures with different sensitivities to low-probability events. For a filtered distribution $\bar{p}_t$ over C_t and $\gamma \neq 1$, we define

$$H_\gamma(\bar{p}_t) = \frac{1}{1-\gamma} \log \sum_{v \in C_t} \bar{p}_t(v)^\gamma. \tag{8}$$

Following the effective-number interpretation (Hill, 1973; Jost, 2006), exponentiating this entropy gives a real-valued effective number of candidates

$$D_\gamma(\bar{p}_t) = \exp(H_\gamma(\bar{p}_t)). \tag{9}$$

This number is anchored by the uniform case. For a uniform distribution over M candidates,

$$H_\gamma = \frac{1}{1-\gamma} \log \sum_{i=1}^{M} \left(\frac{1}{M}\right)^\gamma \tag{10}$$

$$= \frac{1}{1-\gamma} \log \left(M^{1-\gamma}\right) = \log M, \tag{11}$$

and therefore $D_\gamma = M$. Thus D_γ matches the exact support size when the candidates are equally likely. For a non-uniform distribution, it gives the size of an equally weighted support with the same Rényi entropy, becoming smaller as probability mass concentrates on fewer candidates.

The same family also underlies p-less sampling (Tan et al., 2026), which truncates the next-token distribution with the threshold $L[P] = \sum_i p_i^2 = \exp(-H_2(P))$, the collision probability of P, and keeps the tokens whose probabilities are at least $L[P]$. Since $L[P] = 1/D_2(P)$, p-less compares each token's probability with the inverse of the order-2 effective number and then samples one token from the retained set. AGC-ST uses an effective number differently: the order-0.5 value bounds how many candidates enter a mixture, and each admitted candidate contributes in proportion to its probability. A lower order gives a larger bound, as discussed below.

For AGC-ST, we instantiate this family at $\gamma = 0.5$. Starting from Eq. 8,

$$H_{0.5}(\bar{p}_t) = \frac{1}{1-0.5} \log \sum_{v \in C_t} \bar{p}_t(v)^{0.5} \tag{12}$$

$$= 2 \log \sum_{v \in C_t} \sqrt{\bar{p}_t(v)}, \tag{13}$$

Exponentiating this expression gives

$$D_{0.5}(\bar{p}_t) = \exp\left(2 \log \sum_{v \in C_t} \sqrt{\bar{p}_t(v)}\right) \tag{14}$$

$$= \left(\sum_{v \in C_t} \sqrt{\bar{p}_t(v)}\right)^2, \tag{15}$$

where the last step uses $\exp(2 \log x) = x^2$ for $x > 0$. Taking its ceiling gives the integer support cap

$$N_t^{\mathrm{R}} = \lceil D_{0.5}(\bar{p}_t) \rceil \tag{16}$$

$$= \left\lceil \left(\sum_{v \in C_t} \sqrt{\bar{p}_t(v)}\right)^2 \right\rceil. \tag{17}$$

This recovers Eq. 4.

Choice of Order. Three limiting cases locate the order-0.5 count within this family, with D_1 and D_∞ understood as limits in γ. The count $D_0(\bar{p}_t) = |C_t|$ treats every filtered candidate equally, $D_1(\bar{p}_t) =$

$\exp(H_t)$ is the exponentiated gate entropy, i.e., the perplexity of $\bar{p}_t$, and $D_\infty(\bar{p}_t) = 1/\bar{p}_t(v_{t,1})$ depends only on the top candidate. Since D_γ is non-increasing in γ, lower orders give effective numbers no smaller than those at higher orders. At $\gamma = 0$, the cap equals $|C_t|$ and the support rule reduces to the prefix-only variant in Tab. 3(b). In AGC-ST, the cap only bounds the candidate range, and the subsequent prefix rule on the capped distribution q_t can still exclude weak candidates within it. We therefore consider orders below one to retain a broader candidate range before prefix selection, while keeping the order positive so that the effective number remains sensitive to candidate probabilities. We instantiate this choice at $\gamma = 0.5$, for which $\exp(H_t) \le D_{0.5}(\bar{p}_t) \le K_t$. This argument motivates considering $0 < \gamma < 1$ rather than a unique value; the specific default $\gamma = 0.5$ is selected empirically in Sec. 4.3.

C.2 Adaptive Support Range

AGC-ST converts the filtered distribution into a bounded, step-specific support size. Let $K_t = |C_t|$ be the number of candidates in the filtered set from Eq. 1, and let $K_{\max}$ denote the maximum candidate-set size allowed by the decoding filter. In our main experiments, the hard top-k cap is $K_{\max} = 30$, while top-p filtering can make K_t smaller at individual steps.

Candidate and Cap Range. For any filtered distribution $\bar{p}_t$ over C_t, the Rényi-0.5 effective number and its integer cap are

$$D_{0.5}(\bar{p}_t) = \left(\sum_{v \in C_t} \sqrt{\bar{p}_t(v)} \right)^2, \tag{18}$$

$$N_t^{\mathrm{R}} = \lceil D_{0.5}(\bar{p}_t) \rceil . \tag{19}$$

The real-valued effective number is always between 1 and K_t. The lower bound follows from the fact that $\sum_{v \in C_t} \bar{p}_t(v) = 1$ and $\sqrt{x} \ge x$ for $x \in [0, 1]$:

$$\left(\sum_{v \in C_t} \sqrt{\bar{p}_t(v)} \right)^2 \ge \left(\sum_{v \in C_t} \bar{p}_t(v) \right)^2 = 1. \tag{20}$$

The upper bound follows from the Cauchy–Schwarz inequality:

$$\left(\sum_{v \in C_t} \sqrt{\bar{p}_t(v)} \right)^2 \le \left(\sum_{v \in C_t} 1^2 \right) \left(\sum_{v \in C_t} \bar{p}_t(v) \right) = K_t. \tag{21}$$

Together, these bounds give $1 \le D_{0.5}(\bar{p}_t) \le K_t$. Since K_t is an integer, taking the ceiling preserves the upper bound and gives

$$1 \le N_t^{\mathrm{R}} \le K_t \le K_{\max}. \tag{22}$$

The Rényi cap is therefore a distribution-dependent support upper bound within the filtered candidate set, rather than a fixed global support size.

Effect of the Entropy-Margin Gate. The entropy-margin gate in Eq. 3 activates only when

$$H_t \ge \alpha, \qquad m_t = \bar{p}_t(v_{t,1}) - \bar{p}_t(v_{t,2}) \le \beta, \tag{23}$$

where $v_{t,1}$ and $v_{t,2}$ are the two highest-probability candidates. The margin is evaluated only when at least two candidates remain after filtering. One-candidate steps have $H_t = 0$ and cannot pass the entropy gate for $\alpha > 0$. The entropy condition implies a lower bound on the Rényi cap. Rényi entropy is non-increasing in its order, hence

$$H_{0.5}(\bar{p}_t) \ge H_1(\bar{p}_t) = H_t \ge \alpha. \tag{24}$$

Exponentiating gives $D_{0.5}(\bar{p}_t) \ge \exp(\alpha)$, and applying the ceiling operation gives

$$N_t^{\mathrm{R}} = \lceil D_{0.5}(\bar{p}_t) \rceil \ge \lceil \exp(\alpha) \rceil . \tag{25}$$

The margin condition also rules out a strongly dominant top token. Since the candidates are sorted, $\bar{p}_t(v_{t,2}) \ge \bar{p}_t(v_{t,1}) - \beta$. Combining this with $\bar{p}_t(v_{t,1}) + \bar{p}_t(v_{t,2}) \le 1$ gives

$$2\bar{p}_t(v_{t,1}) - \beta \le \bar{p}_t(v_{t,1}) + \bar{p}_t(v_{t,2}) \le 1, \tag{26}$$

$$\bar{p}_t(v_{t,1}) \le \frac{1 + \beta}{2}. \tag{27}$$

Therefore, the entropy condition selects steps whose Rényi effective number is not too small, while the margin condition excludes steps with an overly dominant top token.

Prefix Selection Tightens the Cap. After the Rényi cap, AGC-ST normalizes $\bar{p}_t$ on $C_t^{\mathrm{R}} = \{v_{t,1}, \ldots, v_{t,N_t^{\mathrm{R}}}\}$ to obtain the capped distribution q_t defined in Sec. 3.2, and then selects the smallest prefix whose capped mass reaches ρ as in Eq. 5:

$$n_t = \min\left\{ n \in \{1, \ldots, N_t^{\mathrm{R}}\} : \sum_{i=1}^{n} q_t(v_{t,i}) \geq \rho \right\}. \tag{28}$$

For $\rho \in (0, 1]$, $\sum_{i=1}^{N_t^{\mathrm{R}}} q_t(v_{t,i}) = 1 \geq \rho$, so such a prefix always exists and satisfies

$$1 \leq n_t \leq N_t^{\mathrm{R}} \leq K_t \leq K_{\max}. \tag{29}$$

Combining Eq. 25 with Eq. 29, gated steps have an adaptive intermediate range

$$\lceil \exp(\alpha) \rceil \leq N_t^{\mathrm{R}} \leq K_t \leq K_{\max}, \qquad 1 \leq n_t \leq N_t^{\mathrm{R}}. \tag{30}$$

The final support can therefore be smaller than the Rényi cap when the leading candidates already carry enough mass under q_t.

Uniform Limiting Case. The widest support occurs in the limiting case where the filtered distribution is uniform over its K_t candidates. In that case, $\bar{p}_t(v) = 1/K_t$ for every $v \in C_t$, and

$$\left(\sum_{v \in C_t} \sqrt{\bar{p}_t(v)} \right)^2 = \left(K_t \sqrt{\frac{1}{K_t}} \right)^2 = K_t, \tag{31}$$

so $D_{0.5}(\bar{p}_t) = K_t$ and $N_t^{\mathrm{R}} = K_t$. The capped distribution q_t is also uniform, so prefix selection gives

$$n_t = \lceil \rho K_t \rceil. \tag{32}$$

This limiting case shows how AGC-ST expands support when the filtered distribution is maximally dispersed. With the main experimental values $\alpha = 1.25$, $\beta = 0.10$, $\rho = 0.90$, and $K_{\max} = 30$, an activated step satisfies

$$N_t^{\mathrm{R}} \geq \lceil e^{1.25} \rceil = 4, \qquad \bar{p}_t(v_{t,1}) \leq \frac{1 + 0.10}{2} = 0.55, \tag{33}$$

while the widest uniform case gives $n_t = \lceil 0.90 \times 30 \rceil = 27$. In practice, prefix selection often contracts the final support below the Rényi cap, as shown in Fig. 9.

D Experimental Details

D.1 Model Details

We evaluate on five reasoning LLMs: DeepSeek-R1-Distill-Qwen-7B (DS-Qwen-7B) (Guo et al., 2025), DeepSeek-R1-Distill-Llama-8B (DS-Llama-8B) (Guo et al., 2025), Qwen3-8B (Yang et al., 2025), QwQ-32B (Team, 2025; Yang et al., 2024), and Qwen3-32B (Yang et al., 2025). These models cover Qwen- and Llama-series architectures and span scales from 7B to 32B parameters. DS-Qwen-7B and DS-Llama-8B are distilled from DeepSeek-R1, and Qwen3-8B is distilled from larger Qwen3 models (Yang et al., 2025), while QwQ-32B and Qwen3-32B are post-trained directly for reasoning.

D.2 Benchmark Details

Our benchmark suite spans mathematical, scientific, and coding reasoning:

- **AIME2024 and AIME2025** (Zhang & Math-AI, 2024; 2025). AIME is a competition-style mathematics benchmark derived from the American Invitational Mathematics Examination. Each annual set in our evaluation contains 30 problems from AIME I and AIME II. Every problem requires a short integer answer between 0 and 999, so success depends on deriving the answer rather than selecting from options. These benchmarks stress long-form symbolic reasoning, counting, number theory, algebra, and geometry.

- **GPQA-Diamond (GPQA-D)** (Rein et al., 2024). GPQA-D is the high-quality subset of GPQA, a graduate-level science benchmark written and validated by domain experts. It contains 198 multiple-choice questions in biology, physics, and chemistry. The questions are designed to be difficult for skilled non-experts even with web access, making the benchmark a probe of knowledge-intensive scientific reasoning rather than surface retrieval.
- **MBPP** (Austin et al., 2021). MBPP evaluates Python program synthesis from natural-language task descriptions and tests. Our evaluation split contains 257 entry-level Python programming problems. Each problem asks the model to generate a function that satisfies the provided unit tests, so correctness is measured by executable test outcomes rather than lexical overlap.
- **LiveCodeBenchV5 (LCB)** (Jain et al., 2025). LCB evaluates contest-style code generation on recent programming problems. We use the standard v5 evaluation window, which collects problems released between August 2024 and February 2025; our evaluation split contains 279 problems. Unlike MBPP, LCB follows a standard-input and standard-output format, closer to online programming contests, and evaluates generated programs with hidden tests.

These benchmarks cover mathematical derivation, scientific question answering, function-level programming, and contest-style coding.

D.3 Implementation Details

All experiments are conducted in a unified SGLang 0.4.9.post6 inference framework. All comparison methods share the same base model, prompt, maximum generation length, sampling budget, and answer-extraction pipeline within each model–benchmark pair. Except for CoT Greedy, all methods run with the same decoding parameters in both the thinking phase and the answer phase, with temperature $\tau = 0.6$, top-$p = 0.95$, and top-$k = 30$. These parameters define the filtered distribution used for discrete sampling, and each soft-thinking method forms its mixture from this distribution using the support rule in Tab. 5; the exception is SwiR, which mixes over the full vocabulary. All stochastic methods use seed 0.

We generate at most 32,768 new tokens for each response. AGC-ST only changes the thinking phase before the end-of-thinking token. After the model enters the answer phase, decoding returns to the ordinary discrete-token path.

For AGC-ST, we adopt a single shared hyperparameter setting across all five models and five benchmarks: entropy threshold $\alpha = 1.25$, margin threshold $\beta = 0.10$, Rényi order $\gamma = 0.5$, cumulative-mass prefix $\rho = 0.90$, with candidates limited by the shared top-k cap $K_{\max} = 30$. This shared setting is used for the main comparison across all 25 model–benchmark combinations. It is selected as described in Sec. 4.3; GPQA-D, MBPP, LCB and the two Qwen3 models are held out from that selection.

All experiments are conducted on NVIDIA GH200 120GB GPUs. Runtime varies by model and benchmark. For DS-Qwen-7B and DS-Llama-8B, a 16-sample run on AIME2024 or AIME2025 finishes on a single GPU in about 2–3 hours. GPQA-D, MBPP, and LCB are evaluated with data parallelism on up to four GPUs, with each run finishing within about 5 hours. For QwQ-32B, most runs are performed with four-GPU tensor parallelism and finish within about 5 hours. GPQA-D and LCB require 8 and 16 GPUs, respectively, to keep each run within about 7 hours. Upon acceptance, we will release the complete codebase, including the unified SGLang-based inference framework and our implementations of all comparison methods.

D.4 Baseline Implementation and Hyperparameters

All methods are integrated into our SGLang 0.4.9.post6 codebase. Soft Thinking and Soft-Thinking G follow the SGLang implementation of the Soft-Thinking repository[1], and Multiplex-I follows the public Multiplex-Thinking implementation[2]. For selective baselines without SGLang implementations, we re-implement the published methods in our unified framework, following the routing

[1] https://github.com/eric-ai-lab/Soft-Thinking

[2] https://github.com/GMLR-Penn/Multiplex-Thinking

Table 5: Method-specific settings in the main comparison. All stochastic methods share the same thinking and answer decoding settings, with top-$k = 30$, top-$p = 0.95$, and temperature $\tau = 0.6$.

Method	Gate	Soft Support	Method-Specific Parameters
CoT Greedy	None	Discrete argmax	–
CoT	None	Discrete sample	–
Soft Thinking	All thinking steps	Top-10 mixture	Cold stop: entropy 0.01, patience 256
Multiplex-I	All thinking steps	3-sample mixture	–
Soft-Thinking G	All thinking steps	Top-5 mixture	Gumbel temperature 0.5
SeLaR	Entropy gate	Top-3 mixture	Contrastive weight 1.0
SwiR	Switch gate	Full-vocabulary mixture	Window 512, $\beta = 0.7$
AGC-ST	Entropy-margin gate	Adaptive support from top-30	Entropy 1.25, margin 0.10, Rényi-0.5 cap, prefix 0.90

Table 6: Benchmark- and model-specific hyperparameters for selective soft-thinking baselines.

Method	Scope	AIME2024	AIME2025 / MBPP / LCB	GPQA-D
SeLaR	Normalized entropy threshold	0.40	0.50	0.70
SwiR	DS-Qwen-7B, α	0.65	0.70	0.70
SwiR	DS-Llama-8B, α	0.65	0.70	0.70
SwiR	Qwen3-8B, α	0.90	0.90	1.00
SwiR	QwQ-32B, α	0.90	0.90	1.00
SwiR	Qwen3-32B, α	1.00	0.90	0.60

logic and all method-specific technical details described in the papers and their public reference implementations of SwiReasoning[3] and SeLaReasoning[4].

Tab. 5 reports the method-specific gate and soft-support settings of each baseline in the main comparison. The table separates the shared decoding budget from the mechanism-specific parameters, so differences across rows reflect each method's routing or aggregation design rather than changes to the base sampling setup.

SeLaR and SwiR require additional benchmark- or model-specific routing parameters, which are listed in Tab. 6. For SeLaR, the normalized-entropy thresholds for AIME2024, AIME2025, and GPQA-D follow the dataset-specific values reported by Fu & Luo (2026). SeLaR reports no coding benchmarks, and we use 0.50 for MBPP and LCB. For SwiR, we follow the source-aligned α settings for DeepSeek-R1-Distill-Llama-8B, assign the same DeepSeek-distill settings to DS-Qwen-7B, and follow the Qwen-family settings for Qwen3-8B, QwQ-32B and Qwen3-32B. SwiReasoning reports no settings for Qwen3-32B on MBPP and LCB, so for these two benchmarks we use the corresponding Qwen3-8B settings. For SwiR, we disable switch-count control in the main comparison and keep window size 512 and $\beta = 0.7$.

D.5 Evaluation Protocol

For each question, each stochastic method produces $k = 16$ independent responses on AIME2024 and AIME2025 and $k = 4$ on GPQA-D, MBPP and LCB, while CoT Greedy produces one response per question in each run. Although greedy decoding involves no sampling, its outputs can differ across runs because batched GPU inference is not numerically deterministic (Yuan et al., 2025). We therefore report the mean of three CoT Greedy runs (seeds 0, 1 and 2). We report Avg@k, the mean correctness across the k sampled responses, which is an unbiased Monte Carlo estimate of single-sample accuracy (Pass@1) (Chen et al., 2021). Answer extraction follows the benchmark format: boxed integer extraction for AIME, choice-letter extraction for GPQA-D, unit-test execution for MBPP, and the LiveCodeBench evaluation pipeline for LCB. In Tabs. 1–3, Mean is the unweighted average over the displayed benchmarks or settings. Bold marks the highest accuracy per column within each model group in Tab. 1 and the highest Mean among non-reference configurations in Tabs. 2–3.

[3] https://github.com/sdc17/SwiReasoning

[4] https://github.com/Parker-rfu/SeLaReasoning

Table 7: Summary of licenses and source URLs for experimental assets.

Resource	Type	License	URL
QwQ-32B	Base LLM	Apache-2.0	https://huggingface.co/Qwen/QwQ-32B
DS-Qwen-7B	Base LLM	MIT, base Qwen2.5 Apache-2.0	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
DS-Llama-8B	Base LLM	MIT, base Llama 3.1 license	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B
Qwen3-8B	Base LLM	Apache-2.0	https://huggingface.co/Qwen/Qwen3-8B
Qwen3-32B	Base LLM	Apache-2.0	https://huggingface.co/Qwen/Qwen3-32B
SGLang 0.4.9.post6	Inference framework	Apache-2.0	https://github.com/sgl-project/sglang
Soft-Thinking	Baseline code and benchmark packaging	Not specified in repository; modified SGLang under Apache-2.0	https://github.com/eric-ai-lab/Soft-Thinking
Multiplex-Thinking	Baseline code	MIT	https://github.com/GMLR-Penn/Multiplex-Thinking
SwiReasoning	Baseline code	BSD-3-Clause	https://github.com/sdc17/SwiReasoning
SeLaReasoning	Baseline code	Not specified in repository	https://github.com/Parker-rfu/SeLaReasoning
AIME2024	Benchmark data	Apache-2.0 for the math-ai HF release	https://huggingface.co/datasets/math-ai/aime24
AIME2025	Benchmark data	Apache-2.0 for the math-ai HF release	https://huggingface.co/datasets/math-ai/aime25
GPQA-Diamond	Benchmark data	CC-BY-4.0	https://huggingface.co/datasets/idavidrein/gpqa
MBPP	Benchmark data	CC-BY-4.0	https://huggingface.co/datasets/google-research-datasets/mbpp
LiveCodeBenchV5	Benchmark data and evaluation code	HF data: Creative Commons; code: MIT	https://huggingface.co/datasets/livecodebench/code_generation_lite

D.6 Analysis Traces

For the distributional analysis in Fig. 1, we record one sampled CoT reasoning trace per question for each of the 25 model–benchmark combinations using the shared decoding parameters in App. D.3. At each reasoning step, we store the filtered next-token distribution after top-k/top-p truncation. All statistics in Fig. 1 are computed directly over the recorded thinking steps, with each step included once.

In Fig. 1, panel (a) averages cumulative top-n mass over thinking steps within each model. Panels (b,d) combine step counts across runs: (b) reports region proportions, while (d) normalizes the support-size distributions separately for all and Q1 steps. Panels (c,e) report individual runs. Panel (f) combines Q1 steps across runs and groups them by truncated support size; sizes 4–10 cover 95.5% of Q1 steps. Top-n denotes the first n candidates ranked by $\bar{p}_t$.

For the box plots in Fig. 1(e,f) and Fig. 7(b), boxes span the 25th–75th percentiles and black lines mark medians. Whiskers extend to the most extreme observations within 1.5 interquartile ranges of the box edges; outliers are not displayed.

The same traces are used for Figs. 6 and 7 and Tab. 4. The support-size statistics of AGC-ST in Fig. 4 and App. E.4 use one sampled AGC-ST rollout per question for each of the 25 model–benchmark combinations, recorded in the same way. The same-state interventions use separate DS-Qwen-7B rollouts, described in App. E.3.

D.7 Licenses

Tab. 7 summarizes the licenses and URLs for the base models, framework, baseline code references, and benchmark/evaluation assets used in our experiments. For benchmarks, we use the files or preparation scripts distributed with the Soft-Thinking repository for consistency with prior soft-thinking evaluations, and report the license of the specific public release used when available. The Soft-Thinking and SeLaReasoning repositories do not distribute a license file; we use both only as references when re-implementing the corresponding methods.

E Additional Experimental Results

E.1 Statistical Significance of Gains over Baselines

Sec. 4.2 compares methods by point estimates. We quantify uncertainty in these gains using a paired bootstrap over evaluation questions (Koehn, 2004). For each model–benchmark pair, we resample questions with replacement while keeping the observed per-question scores fixed, and recompute every method's accuracy on the same resampled set, so that each difference is formed on identical questions. The five-benchmark difference averages the per-benchmark differences with equal weight in each bootstrap replicate, matching Mean in Tab. 1. We draw $50{,}000$ resamples and report percentile 95% intervals; a gain is significant at the nominal 5% level when its interval excludes zero.

Tab. 8 lists every baseline in descending order of average accuracy. On the five-benchmark average, the gain of AGC-ST is significant against CoT for all five models and in 34 of the 35 model–baseline comparisons. The gains are largest against CoT Greedy, Soft Thinking, Multiplex-I and SeLaR, ranging from $+1.41$ to $+12.15$ points, and all 20 of these comparisons are significant. The only

Table 8: Paired comparisons of AGC-ST against every baseline on the five-benchmark average. Within each model group, baselines are listed in descending order of average accuracy. Δ is AGC-ST minus the baseline in percentage points, with a 95% paired bootstrap interval over questions.

Model	AGC-ST Mean	Baseline	Baseline Mean	Δ [95% CI]	
DS-Qwen-7B	55.41	CoT	53.76	$+1.65$	$[+0.14, +3.20]$
		Soft-Thinking G	53.72	$+1.69$	$[+0.21, +3.20]$
		SwiR	53.42	$+1.99$	$[+0.24, +3.81]$
		Multiplex-I	49.56	$+5.85$	$[+4.25, +7.50]$
		SeLaR	49.55	$+5.87$	$[+4.24, +7.52]$
		CoT Greedy	49.08	$+6.34$	$[+3.85, +8.91]$
		Soft Thinking	47.73	$+7.68$	$[+3.80, +11.62]$
DS-Llama-8B	49.80	Soft-Thinking G	48.27	$+1.53$	$[+0.02, +3.07]$
		CoT	48.09	$+1.71$	$[+0.52, +2.90]$
		SwiR	48.03	$+1.77$	$[+0.22, +3.34]$
		Multiplex-I	44.22	$+5.58$	$[+4.12, +7.07]$
		SeLaR	42.95	$+6.85$	$[+5.29, +8.42]$
		CoT Greedy	38.72	$+11.08$	$[+8.55, +13.65]$
		Soft Thinking	37.65	$+12.15$	$[+8.71, +15.67]$
Qwen3-8B	72.68	CoT	71.51	$+1.17$	$[+0.13, +2.21]$
		Soft-Thinking G	71.03	$+1.65$	$[+0.54, +2.77]$
		SwiR	70.90	$+1.78$	$[+0.59, +3.01]$
		Multiplex-I	69.62	$+3.06$	$[+1.80, +4.33]$
		SeLaR	69.03	$+3.65$	$[+2.34, +5.02]$
		CoT Greedy	67.04	$+5.64$	$[+3.59, +7.77]$
		Soft Thinking	65.82	$+6.86$	$[+3.76, +10.21]$
QwQ-32B	75.27	SwiR	74.48	$+0.79$	$[-0.26, +1.86]$
		CoT	74.09	$+1.18$	$[+0.17, +2.21]$
		Soft-Thinking G	73.93	$+1.34$	$[+0.25, +2.46]$
		CoT Greedy	73.27	$+2.00$	$[+0.14, +3.86]$
		Multiplex-I	72.72	$+2.55$	$[+1.40, +3.75]$
		Soft Thinking	72.04	$+3.23$	$[+1.41, +5.12]$
		SeLaR	69.03	$+6.24$	$[+4.87, +7.76]$
Qwen3-32B	77.14	CoT	76.05	$+1.09$	$[+0.03, +2.18]$
		SwiR	76.04	$+1.11$	$[+0.08, +2.19]$
		Multiplex-I	75.73	$+1.41$	$[+0.16, +2.69]$
		Soft-Thinking G	75.67	$+1.47$	$[+0.37, +2.59]$
		CoT Greedy	75.13	$+2.01$	$[+0.45, +3.63]$
		SeLaR	73.58	$+3.57$	$[+2.46, +4.70]$
		Soft Thinking	73.44	$+3.70$	$[+1.55, +6.10]$

comparison whose interval includes zero is SwiR on QwQ-32B ($+0.79$ $[-0.26, +1.86]$), where the point estimate remains positive. The leading baselines are close to one another: the top three in each group lie within 0.61 points, and the top-ranked baseline changes across groups, being CoT on DS-Qwen-7B, Qwen3-8B and Qwen3-32B, Soft-Thinking G on DS-Llama-8B, and SwiR on QwQ-32B. AGC-ST is the only method with the highest average in all five groups.

E.2 Inference Latency and Decode Efficiency

We assess two aspects of inference efficiency: fixed-token execution overhead and natural-EOS response latency. The fixed-token evaluation includes a four-method GPU rotation and a CoT/AGC-ST same-GPU crossover. All measurements use DS-Qwen-7B on the 30 AIME2024 problems, GH200 120GB GPUs, batch size 1, seed 0, and the shared decoding settings ($\tau = 0.6$, top-$k = 30$, top-$p = 0.95$). We time synchronous generation after synthetic-prompt warm-up, excluding initialization, evaluation, and result writing. In both tables, seconds per response is total generation time divided by the request count. Tokens per second and milliseconds per token use total completion tokens and total generation time.

Fixed-Token Execution Overhead. CoT, Soft-Thinking G, SwiR, and AGC-ST each generate exactly 512 tokens per request with EOS termination disabled. Four Latin-square rounds rotate the

Table 9: Fixed-512-token decoding efficiency on DS-Qwen-7B and AIME2024. The four-method comparison uses four Latin-square GPU rotations. The same-GPU crossover alternates CoT and AGC-ST on the same two physical GPUs.

Protocol	Method	Requests	s/response ↓	tokens/s ↑	ms/token ↓
Four-method rotation	CoT	120	8.692	58.906	16.976
	Soft-Thinking G	120	8.579	59.680	16.756
	SwiR	120	8.684	58.959	16.961
	AGC-ST	120	8.436	60.694	16.476
Same-GPU crossover	CoT	120	8.531	60.014	16.663
	AGC-ST	120	8.613	59.442	16.823

Table 10: Latency with natural stopping on DS-Qwen-7B and all 30 AIME2024 problems. Length cap counts responses that reach the 32,768-token limit.

Method	Requests	s/response ↓	tokens/response	tokens/s ↑	ms/token ↓	Length cap
CoT	30	214.203	12,958.7	60.497	16.530	3 / 30
Soft-Thinking G	30	212.450	12,584.2	59.234	16.882	1 / 30
SwiR	30	233.496	13,615.9	58.313	17.149	1 / 30
AGC-ST	30	231.465	13,569.6	58.625	17.058	0 / 30

methods across four GPUs, yielding 120 requests per method. Tab. 9 shows similar execution costs. A Latin-square analysis of variance on run-level mean latency, with round and GPU as blocking factors, detects no method effect ($F(3, 6) = 0.32$).

Same-GPU Confirmation. A four-stage crossover assigns CoT and AGC-ST to two GPUs in opposite orders (ABBA and BAAB), with 120 requests per method. The aggregate relative latency difference is $+0.96\%$, with opposite directions on GPU0 ($+4.55\%$) and GPU1 (-2.37%). Together, the fixed-token tests detect no consistent additional overhead in this setting. The numerical differences should not be interpreted as an algorithmic speedup.

Natural-EOS Response Latency. Each method generates one response per problem with a 32,768-token cap. Four disjoint problem chunks rotate each method across the four GPUs. Tab. 10 reports response time alongside output length. Compared with SwiR, AGC-ST has similar mean latency (-0.87%). Compared with CoT, its mean latency is 8.06% higher, accompanied by 4.71% more tokens and 3.19% higher milliseconds per token. These natural-generation measurements reflect both execution cost and differences in generated trajectories.

The mean paired latency difference from CoT is $+17.26$ seconds and from SwiR is -2.03 seconds. These results describe one model and benchmark at batch size 1 and do not characterize batched throughput or latency across other settings.

E.3 Same-State Interventions: Controls and Validation

Sec. 4.4 shows that changing the support size can alter the next-step prediction at gated states. We supplement these interventions with three controls that rescale the top-1 embedding, permute the mixture weights, or replace the non-top-1 candidates, and examine how the unmatched rate responds to changes in input norm, weight assignment, and candidate composition. We then report checks that the intervention itself is faithful. All results use DS-Qwen-7B with the decoding and gate settings of Sec. 4.1, tracing one rollout per problem. The unmatched criterion always uses the single-token inputs from the full retained set C_t as its reference. Tab. 11 defines every substituted input, and Tab. 12 reports unmatched rates for the mixtures and controls.

At the deployed support, the mixture's next-step argmax differs from that of the unscaled top-1 single-token input at 47.18% of gated states on AIME2024 ($[45.00, 49.52]$) and 47.73% on LCB ($[47.24, 48.18]$), with 95% problem-level bootstrap intervals in brackets. The unmatched criterion is stricter, requiring disagreement with every single-token input from C_t, so matching some retained single-token input does not imply matching the top-1 input.

Table 11: Substituted inputs used by the same-state intervention. The soft mixtures share one ranking and renormalization rule for every k.

Input	Construction	Purpose
Intervention		
Soft mixture, $k = 2, 3, 4, 6$	Convex mixture over the first k retained tokens of the truncated distribution, renormalized	
Soft mixture, full	Convex mixture over every retained token after top-k/top-p truncation	
AGC-ST	The width AGC-ST selects per state (Rényi cap, then 0.90 prefix)	
Single tokens	Each retained token on its own	
Controls		
Scaled top-1	$c\,E(v_1)$, with c set so the norm matches the paired soft mixture	Compare with a norm-matched top-1 input
Shuffle	Same tokens, weights permuted among them	Examine sensitivity to weight assignment
Swap	Weights kept; tokens $2 \dots k$ replaced by vocabulary tokens matched on cosine to $E(v_1)$	Examine sensitivity to replacing non-top-1 candidates

Vector Norm. We rescale the top-1 embedding to each mixture's norm, keeping its direction (*Scaled top-1*). The resulting rates fall far below those of the corresponding mixtures, so rescaling along the top-1 direction does not reproduce them.

Weight Assignment. Permuting the weights among the same retained tokens (*Shuffle*) leaves the rate near that of the corresponding mixture. It does not follow that weight assignment is irrelevant: these are rates aggregated over states, and equal rates need not mean equal behavior at a given state.

Candidate Identity. Replacing the non-top-1 candidates with vocabulary tokens matched on cosine similarity to $E(v_1)$ (*Swap*) lowers the rate but leaves it well above the norm control. Because this substitution changes both the candidates and the mixture's geometry, the aggregate rates do not isolate the contribution of candidate identity.

Validation. Two identical inputs evaluated in one intervention forward pass agree bit-exactly over $8{,}127$ comparisons, so the substitution introduces no arithmetic difference of its own. Re-evaluating the deployed mixture inside the intervention reproduces the token the main run emitted on all but 0.685% of comparisons. Changing only the batch composition, with no intervention at all, already changes the token on 0.55%, so the remaining difference is at the level of the engine's own nondeterminism. Running the intervention leaves the main trajectory token-identical over 146 gated steps and does not advance the sampling RNG, and an independent reimplementation of the support selector reproduces the deployed width on $3{,}780$ of $3{,}781$ gated steps.

Scope. These interventions characterize one-step behavior at gated states. They do not measure accuracy; accuracy comparisons among the tested support rules are reported in Tab. 3.

E.4 Support Calibration: Pairwise Structure and Model-Wise Decomposition

Fig. 8 gives the pairwise structure of the two calibration stages. The Rényi-0.5 cap mainly acts on broad supports: it stays close to the truncated support when the support is narrow (panel c), and it reduces the share of gated steps with 10 or more candidates from 11% to 5%. Prefix selection then removes one or two candidates at most gated steps (panel a). The final width is not determined by the cap alone. A cap of 5 yields an adaptive support of 3 at 10% of gated steps and 4 at 23%, and a cap of 6 yields 4 or 5. The retained-mass prefix therefore sets the final width from the distribution within the cap.

Table 12: Unmatched rates (%) at gated states, with 95% problem-level bootstrap intervals. Controls are evaluated at k=4 and full support, both defined at every gated state. Each control is matched to the soft mixture in the same block.

Input	**AIME2024**	**LCB**
$k = 4$		
Soft mixture	8.12 [7.54, 8.66]	8.20 [7.96, 8.45]
Scaled top-1	1.19 [0.78, 1.62]	1.00 [0.91, 1.09]
Shuffle	8.62 [7.67, 9.75]	8.41 [8.16, 8.66]
Swap	7.75 [6.64, 8.64]	7.04 [6.82, 7.26]
full support		
Soft mixture	8.99 [8.17, 9.87]	9.08 [8.83, 9.34]
Scaled top-1	1.38 [1.02, 1.78]	1.03 [0.94, 1.12]
Shuffle	9.60 [8.83, 10.36]	9.72 [9.45, 9.99]
Swap	7.56 [6.63, 8.41]	6.71 [6.48, 6.93]

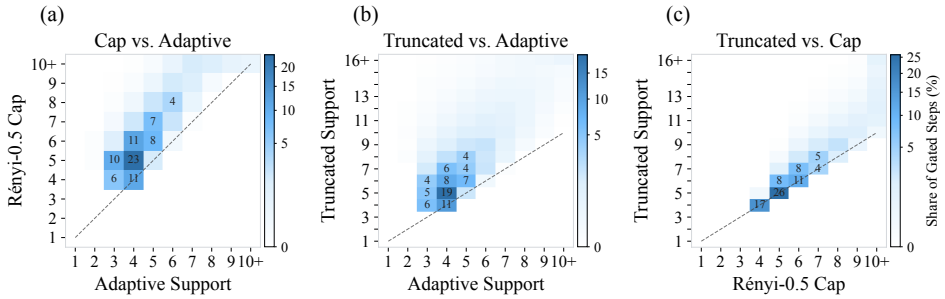

Figure 8: Pairwise support sizes across calibration stages on gated steps, aggregated over all models and benchmarks. Each heatmap sums to 100%.

Fig. 9 breaks down the calibration stages by model. The models differ most in the breadth of their gated distributions: DS-Qwen-7B and DS-Llama-8B place 16% and 14% of gated steps at a truncated support of 10 or more, QwQ-32B 6%, and the two Qwen3 models almost none. After calibration, a final support of 4 candidates is the most common for every model, but its share rises from 43% and 45% for the DeepSeek-R1-distilled models to 51% for QwQ-32B and 60% and 58% for Qwen3-8B and Qwen3-32B. The remaining steps spread over neighboring sizes, and the DeepSeek-R1-distilled models keep more of them at 5 or more candidates. One shared configuration thus yields final support distributions that follow the breadth of each model's gated distributions. Although 4 candidates is the most common final support size, fixed top-4 mixing under the same gating rule yields lower accuracy than AGC-ST in all six AIME ablation settings (Tab. 3).

F Broader Impacts

AGC-ST aims to improve the reliability of reasoning at inference time without modifying model weights. Potential positive impacts include more accurate reasoning on mathematical, scientific, and coding tasks with open-source models, without additional training. Potential negative impacts follow the general dual-use risks of stronger LLM reasoning, including more capable automated code generation or problem solving in harmful contexts. Because our method is an inference-time decoding strategy, appropriate deployment should follow the safeguards, access controls, and monitoring practices of the underlying models and applications.

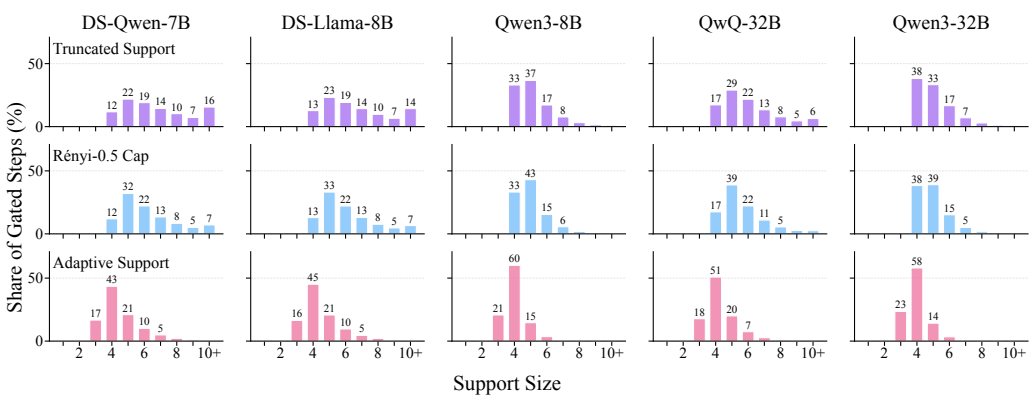

Figure 9: Support-size distributions across calibration stages on gated steps for each model, aggregated over benchmarks. Each row sums to 100% within a model.

G BENCHMARK PROMPTS

In this section, we list the prompt templates used in our experiments before applying the model-specific chat template. We do not add a system message. For each sample, the corresponding template below is filled with the instance-specific fields and placed as the content of a single user message; the resulting one-message chat is then passed to `tokenizer.apply_chat_template(..., add_generation_prompt=True)`. The placeholders `{Question}` and `{TestCases}` denote dataset-provided fields rather than concrete example instances. The prompts are grouped according to the three benchmark categories in Sec. 4.1: mathematical reasoning, scientific reasoning, and coding reasoning.

Mathematical Reasoning. AIME2024 and AIME2025 use the same mathematical-reasoning prompt.

AIME2024 and AIME2025

```
Please reason step by step, and put your final answer within
\boxed{{}}.

{Question}
```

Scientific Reasoning. GPQA-D uses the following multiple-choice prompt.

GPQA-Diamond

```
Please solve the following multiple-choice question. Please show
your choice in the answer field with only the choice letter, e.g.,
"answer": "C".

{Question}
```

Coding Reasoning. MBPP and LCB use task-specific code-generation prompts because MBPP provides test cases while LCB uses standard-input programming tasks.

MBPP

```
Please solve the programming task with test cases below in Python.
Make sure your code satisfies the following requirements:
1. The function name and signature must match exactly as specified
in the test cases.
2. Your code should be wrapped in a markdown code block without
including any test cases.

Task:
{Question}

Test Cases:
```python
{TestCases}
```
```

LiveCodeBench

```
You will be given a question (problem specification) and will
generate a correct Python program that matches the specification
and passes all tests.

Question: {Question}

Read the inputs from stdin solve the problem and write the answer
to stdout (do not directly test on the sample inputs). Enclose your
code within delimiters as follows. Ensure that when the python
program runs, it reads the inputs, runs the algorithm and writes
output to STDOUT.
```python
# YOUR CODE HERE
```
```

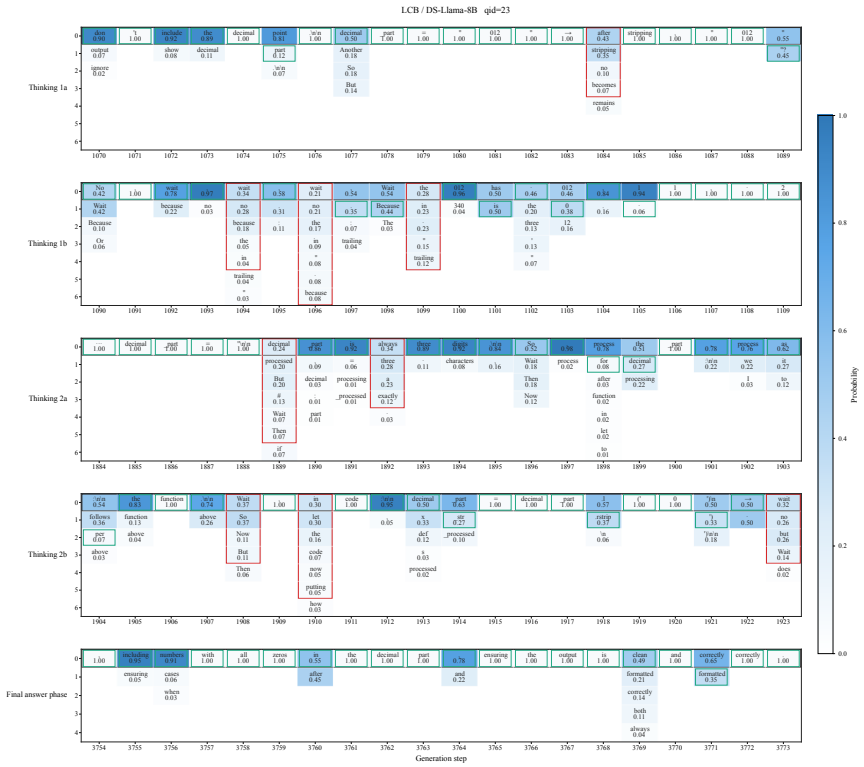

Figure 10: Segment-level heatmap for LCB with DS-Llama-8B, question id 23.

H Qualitative Analysis

H.1 Additional Qualitative Heatmap Cases

Fig. 10 extends the trace in Fig. 5 with two earlier thinking segments and one segment of the final answer phase, using the same display. Nine of the 80 thinking steps shown are gated, with support sizes from 4 to 7. The additional gated steps again occur at transitions; at step 1094, for example, “wait”, “no”, and “because” share the probability mass. The final answer phase remains discrete, as AGC-ST applies soft mixing only during thinking.

Figs. 11 and 12 show AIME2024 traces from DS-Qwen-7B and QwQ-32B with the same display. In both traces, calculation steps and the final answer stay discrete, and gated steps appear at revisions and changes of direction, such as step 3578 for DS-Qwen-7B (“wait”, “let”, “arranged”) and step 2163 for QwQ-32B (“actually”, “direction”, “no”). The two models differ in how often and how widely they mix. In the segments shown, DS-Qwen-7B has nine gated steps and QwQ-32B has five, with support sizes from 4 to 7 in both, and more QwQ-32B positions are near-deterministic. Together with Fig. 9, these cases indicate that how often and how widely AGC-ST mixes varies with the model, whereas in the examples shown, mixing concentrates at transition points.

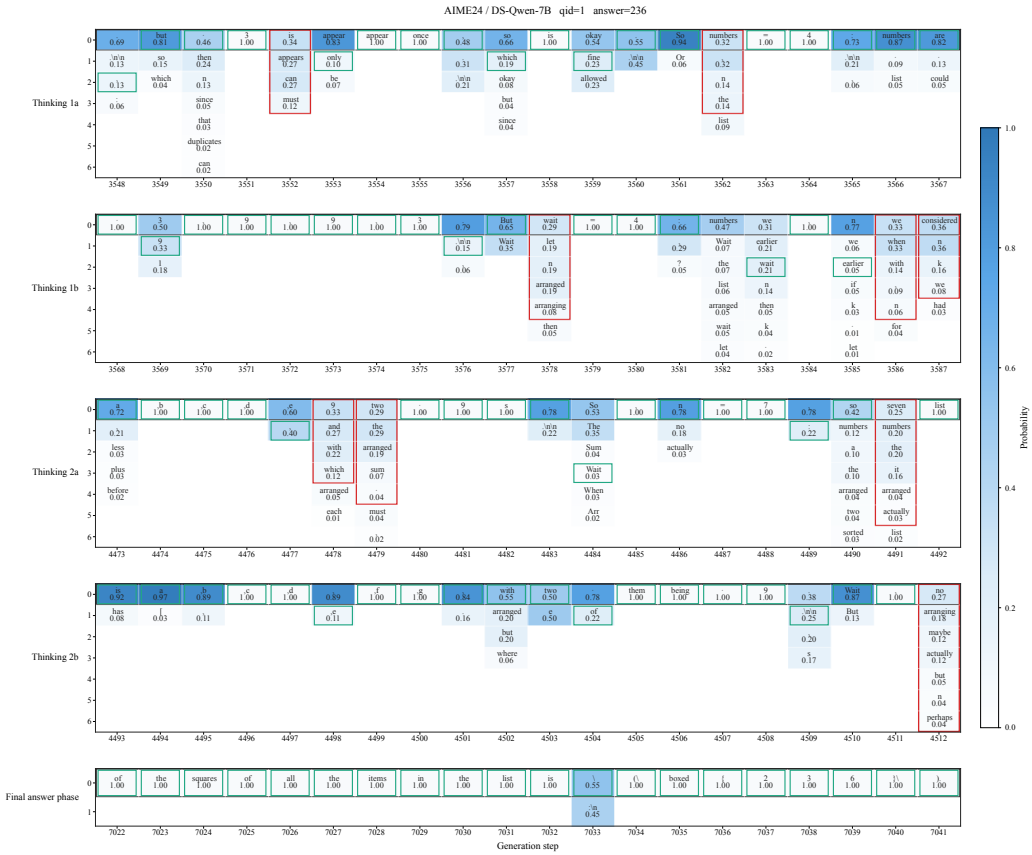

Figure 11: Segment-level heatmap for AIME2024 with DS-Qwen-7B, question id 1.

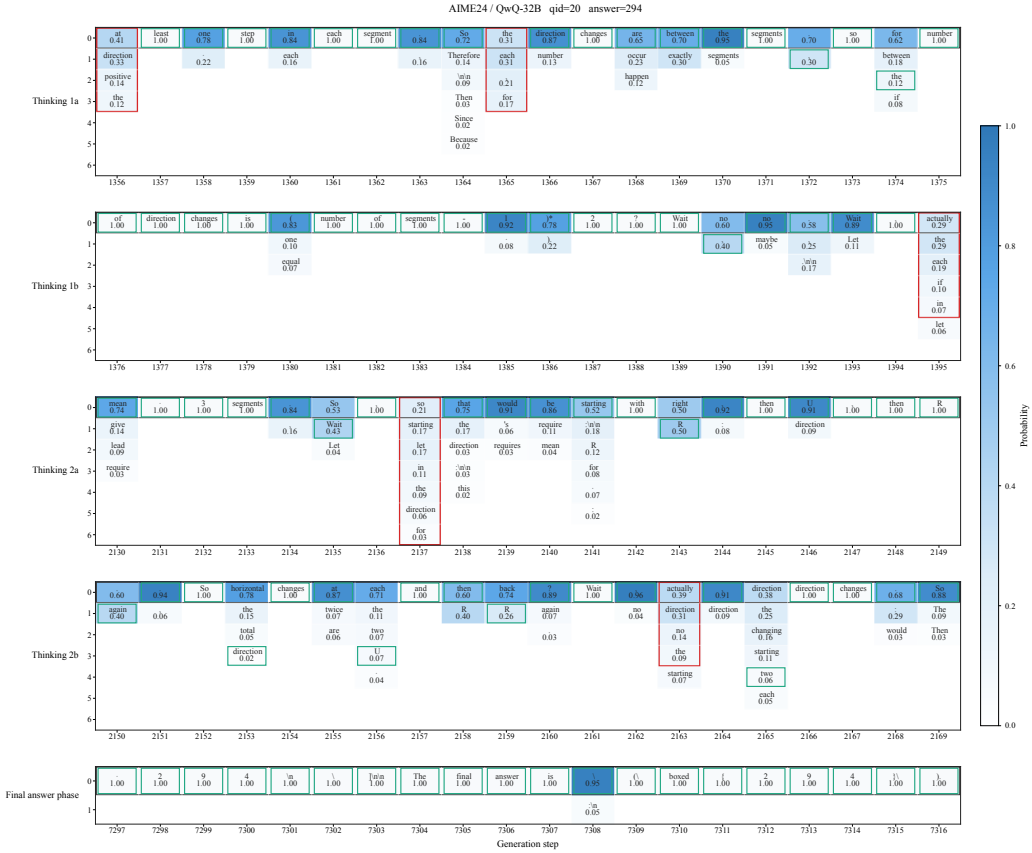

Figure 12: Segment-level heatmap for AIME2024 with QwQ-32B, question id 20.

H.2 Qualitative Output Examples

We provide representative correct responses generated by AGC-ST. Each example reports the benchmark question index used in our evaluation files, the model, the ground-truth answer, and the answer extracted from the model output. For readability, we shorten each raw completion by showing an early thinking excerpt and the final answer excerpt.

AIME2025 / DS-Llama-8B

Question ID. 7

Problem. Let k be a real number such that the system

$$|25 + 20i - z| = 5, \qquad |z - 4 - k| = |z - 3i - k|$$

has exactly one complex solution z. The sum of all possible values of k can be written as m/n, where m and n are relatively prime positive integers. Find $m + n$.

Ground truth. 77 **AGC-ST answer.** $\boxed{77}$

Model output excerpt.

```
Selected thinking excerpt:

Okay, so I have this problem where I need to find all real numbers k such that the
system of equations:

|25 + 20i - z| = 5

and

[Intermediate thinking omitted for readability.]

Distance from (25,20): |8*25 -6*20 -8k -7| /10 = |200 -120 -8k -7| = |73 -8k|,
correct. So, |73 -8k| = 50. So, 73 -8k = 50 gives k = 23/8, and 73 -8k = -50 gives k =
123/8. Sum is 146/8 = 73/4. So, 73 and 4 are coprime, so m + n is 77. Seems seems
correct.

I think that's right. So, the answer is 77.

**Final Answer**
\boxed{77}

Final answer excerpt:

Thus, \( m = 73 \) and \( n = 4 \), giving \( m + n = 77 \).

\[
\boxed{77}
\]
```

GPQA-Diamond / DS-Qwen-7B

Question ID. 119

Problem. Select the suitable reagents for the following reactions:

$$\text{butan-2-one} + \text{NaCN} + A \rightarrow \text{2-hydroxy-2-methylbutanenitrile},$$

$$\text{2-(4-benzylphenyl)-2-hydroxybutanenitrile} + B\ (\text{H}_2\text{O}) \rightarrow \text{2-(4-benzylphenyl)-2-hydroxybutanoic acid}.$$

Options.
A. $A = \text{NaHSO}_3, B = \text{CH}_3\text{COOH}$
B. $A = \text{H}_3\text{O}^+, B = \text{HCl}$
C. $A = \text{NaHSO}_3, B = \text{HCl}$
D. $A = \text{H}_3\text{O}^+, B = \text{CH}_3\text{COOH}$

Ground truth. C **AGC-ST answer.** C

Model output excerpt.

```
Selected thinking excerpt:
```

Okay, so I've got this organic chemistry problem here, and I need to figure out the right reagents for the reactions given. Let me try to break it down step by step.

The first reaction is between butan-2-one and NaCN to form 2-hydroxy-2-methylbutanenitrile. Hmm, I remember that butan-2-one is a ketone. Reacting ketones with nitrile nucleophiles usually involves some kind of substitution or addition. Wait, but butan-2-one has a carbonyl group. Maybe it's a nucleophilic addition? But wait, NaCN is a source of the nitrile ion CN-.

Wait, another butan-2-one is (CH3)2CO. If we add CN- to a ketone, maybe it's a nucleophilic addition, but ketones typically don't undergo that because the carbonyl is not very reactive towards nucleophiles. Oh wait, maybe it's an example of a nucleophilic addition, but under specific conditions. Alternatively, perhaps it's a substitution reaction, but I'm not sure.

[Intermediate thinking omitted for readability.]

Looking at the options, option C says A = NaHSO3, B = HCl. Option B says A = H3O+, B = HCl. Option C is correct.

Wait, let me double. The first reaction uses NaHSO3 as the reagent A, which makes make sense because it's a mild acid that can help in the substitution. The second reaction uses HCl as the reagent B, which would cause the hydrolysis of the nitrile to the carboxylic acid.

So the correct answer would be option C: A = NaHSO3, B = HCl.

Final answer excerpt:

The nitrile group (-CN) in the starting material is converted into a carboxylic acid group (-COOH) in the product. This transformation occurs via hydrolysis of the nitrile group. Hydrolysis of nitriles to carboxylic acids typically requires acidic conditions, often facilitated by HCl or H_2SO_4. Among the options provided, **B = HCl** is the most suitable reagent for this reaction. Therefore, **B = HCl**.

Correct Option:
Combining the results for both reactions, the correct choice is **A = $NaHSO_3$, B = HCl**.

ANSWER: C