Funnel of Thoughts: Efficient Test-Time Scaling via Early Voting and Rollout Pruning

Anonymous ACL submission

Abstract

Large Reasoning Models produce diverse, sometimes inconsistent answers across repeated queries on the same problem, so multi-sample inference is a prerequisite for reliable deployment. Majority voting at k rollouts is the standard solution and the de facto accuracy target for this regime, but it is prohibitively expensive at the scale LRMs require. We introduce **Funnel of Thoughts** (FoT), an inference-time method that preserves the full 32-trajectory voted accuracy while halving its attention FLOPs. Across 115K reasoning trajectories from six LRMs, we find that unproductive trajectories often reveal themselves through repeated hesitation markers such as "Wait," "Actually," and "perhaps." These trajectories are less likely to reach the correct answer and consume disproportionate attention FLOPs, degenerating into no-answer loops in the worst case. Built on this zero-cost lexical signal, FoT identifies the vocabulary that captures these pathological patterns and prunes affected trajectories before completion, reducing online generation FLOPs by 56.1% and wall time by 37.6% with no model overhead; the same signal transfers without retuning across held-out architectures and out-of-domain tasks.

1 Introduction

Large Reasoning Models (LRMs) (OpenAI et al., 2024; Guo et al., 2025; Team, 2025) solve problems by generating long chains of thought (Wei et al., 2023). Compared with Instruct models (Ouyang et al., 2022), they explore richer answer trajectories and therefore produce diverse, sometimes inconsistent answers across repeated queries (Brown et al., 2024). This diversity makes multi-sample inference valuable: the correct answer often appears somewhere in the rollout pool, even when no single rollout is reliable. Self-Consistency (SC@k; Wang et al., 2023) exploits this by sampling k rollouts and returning the majority vote, routinely recovering 20–40 percentage points over single-rollout accuracy on competition benchmarks (Snell et al., 2024). The cost is that every rollout must run to completion. When some trajectories spiral into repetitive, counterproductive self-correction (Wang et al., 2025c; Su et al., 2025; Huang et al., 2024), SC pays for thousands of wasted tokens; with quadratic attention, this waste becomes disproportionately expensive. Prior analyses likewise find that 40–50% of LRM tokens can be redundant (Chen et al., 2025).

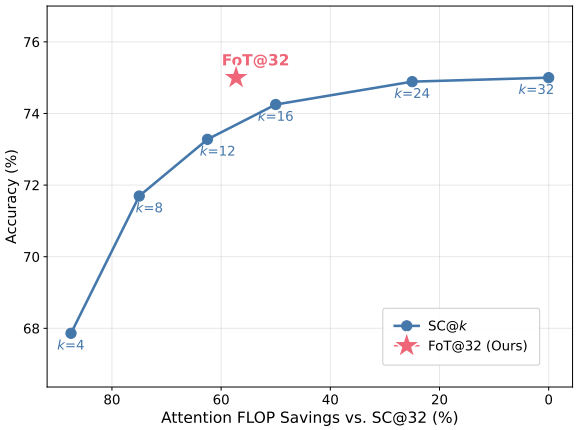

Figure 1: Accuracy vs. compute on AIME24/25. SC@k trades accuracy for compute by varying pool size k. FoT@32 defines a new Pareto point: matching SC@32 accuracy at SC@16-level compute.

To mitigate this inefficiency, we introduce **Funnel of Thoughts** (FoT), an inference-time algorithm that starts with a full pool of k parallel reasoning trajectories and progressively removes unproductive ones in-flight using only the generated text. Unlike efficient-sampling methods that draw fewer trajectories, FoT runs the full pool in parallel and truncates only its spiraling minority, shedding late-stage waste while keeping the productive trajectories that carry the vote. Our key observation is that hesitation markers, such as "Wait," "Actually," and "perhaps", are significantly more fre-

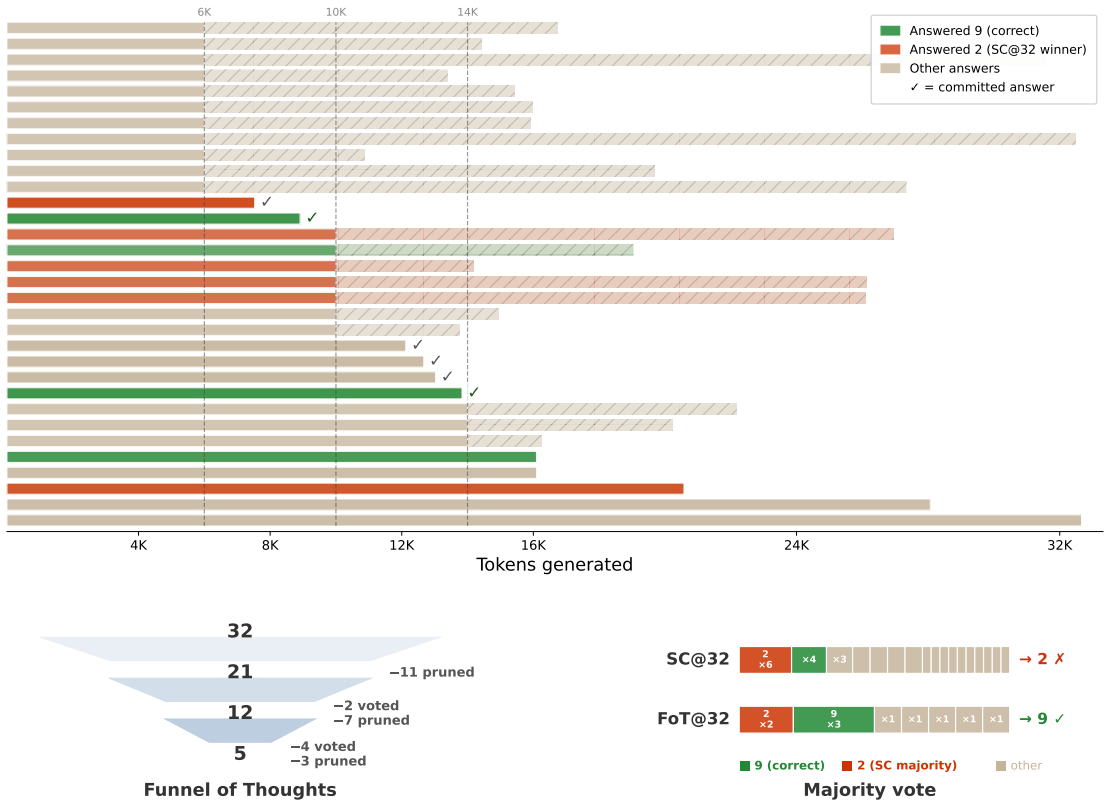

Figure 2: Funnel of Thoughts on AMC23 Problem 20 (DeepSeek-R1-Distill-Qwen-7B, k=32). **Top:** Each bar is one rollout, ordered by sampling seed, and truncated at its pruning or early-voting checkpoint. **Bottom-left:** The pool narrows across three checkpoints. **Bottom-right:** SC@32 elects the wrong plurality (2); FoT@32 lifts the correct answer (9) into the plurality.

quent in incorrect trajectories. At fixed token-count checkpoints, FoT applies two mechanisms: *early voting* banks trajectories that have already committed to an answer; *rollout pruning* removes those exhibiting the highest hesitation marker density. As Figure 1 demonstrates, FoT preserves SC@32 accuracy on AIME24/25, hard competition-math benchmarks where multi-sample inference is most needed, at a 57% reduction in attention FLOPs. Figure 2 traces this on a representative problem, where SC@32's plurality lands on a wrong answer and FoT recovers the correct one.

Our contributions are: (1) a **data-driven analysis** showing that the density of hesitation markers, a zero-cost lexical signal, identifies the unproductive tail of LRM reasoning across 115K rollouts; (2) **Funnel of Thoughts**, an inference-time algorithm exploiting this signal to prune rollouts at inference time in parallel batches without training, reward models, or logit access; (3) **generalizability**: FoT transfers unchanged across architecturally diverse held-out models and out-of-domain tasks, preserving SC@32 accuracy at substantially reduced compute.

2 Related Work

2.1 Reasoning Inefficiency in LRMs

A growing body of work documents systematic inefficiency in LRM reasoning trajectories. Chen et al. (2025) find that about 48% of generated tokens on MATH500 are spent on redundant operations such as repetitive self-verification and unnecessary case exploration, while Su et al. (2025) observe a non-monotonic relationship between trajectory length and accuracy in which both very short and very long rollouts underperform. Huang et al. (2024) show that self-correction without external feedback is counterproductive on average. These findings motivate our approach: the self-correction attempts that manifest as hesitation markers are, on aggregate, wasteful.

Building on this diagnosis, a line of work intervenes at the single-trajectory level to suppress unproductive reasoning. TIP (Wang et al., 2025c) penalizes excessive thought-switching in logits, Wang et al. (2025a) suppress self-reflection keywords to reduce trajectory length, and Wu et al. (2025a) use hidden-state probes to dynamically terminate think-

ing. These approaches focus on shaping the best single rollout. But single-rollout interventions cannot mitigate the fundamental property that LRM inference produces a diverse distribution of final answers, demonstrated by the gap between pass@1 and pass@k (Brown et al., 2024). This diversity grows with trajectory length: every additional decoding step compounds the probability of divergence across rollouts, and LRM trajectories run thousands of tokens.

2.2 Self-Consistency and Sampling Efficiency

SC (Wang et al., 2023) samples k reasoning paths and selects the majority vote, yielding large gains but requiring k full-length trajectories. A line of work reduces cost by deciding when to stop sampling: Adaptive Consistency (Aggarwal et al., 2023) applies a Beta stopping rule on the running majority vote, Certaindex (Fu et al., 2025) uses a learned stability metric, and others refine the criterion further (Wan et al., 2025; Wang et al., 2025b). These methods generate rollouts sequentially: once a trajectory begins, it must run to completion, even when it has entered the wasteful late-stage generation that dominates LRM compute.

Parallel alternatives operate within a k-rollout pool but pursue different goals. Slim-SC (Hong et al., 2025) prunes redundant rollouts during generation via inter-trajectory embedding similarity, requiring a separate embedding model. CISC (Taubenfeld et al., 2025) reweights votes after all rollouts complete using token-level confidence, requiring logprob access and not reducing generation cost. Neither directly targets the pathological trajectories that drive late-stage waste, whereas FoT preemptively identifies and prunes them at inference time using only the generated text.

2.3 Test-Time Compute Scaling

More broadly, test-time compute scaling can be as effective as model scaling: optimal allocation lets smaller models match much larger ones (Snell et al., 2024; Wu et al., 2025b), and the gap between pass@1 and pass@k on hard problems exceeds 50pp (Brown et al., 2024), motivating the multi-sample methods. Process Reward Models offer strong rollout selection in this regime but require a separately trained verifier (Lightman et al., 2023; Wang et al., 2024). The closest prior to our lexical signal is budget forcing (Muennighoff et al., 2025), which appends `Wait` tokens to extend reasoning; we use the *density* of the same token, and others like it, as a signal that extended reasoning has become unproductive.

3 Funnel of Thoughts

3.1 Preliminaries

Pass@k. Pass@k is the probability that at least one of k rollouts is correct. It represents the upper bound of what the model can achieve given its knowledge, but is not attainable in practice without an oracle: selecting the correct rollout from the pool requires knowing the answer a priori. Reaching Pass@k accuracy at low compute cost is therefore the key challenge for effective multi-sampling; SC@k approximates this ceiling via the majority-vote estimator.

Self-Consistency (SC@k). Given a problem q, SC@k (Wang et al., 2023) independently samples k rollouts $\{c_1, \ldots, c_k\}$ from a language model, extracts a candidate answer a_i from each rollout, and returns the majority answer[1]:

$$\hat{a} = \arg\max_{a} \sum_{i=1}^{k} \mathbf{1}[a_i = a]. \tag{1}$$

Attention FLOPs. Each rollout c_i has total sequence length s_i tokens. Because self-attention is quadratic, we measure compute cost as total attention FLOPs across all rollouts: FLOPs $= \sum_{i=1}^{k} 4 \cdot L \cdot d \cdot s_i^2$, where L is the number of layers and d the model dimension.

Setup. We analyze and evaluate FoT on 6 LRMs: DeepSeek-R1-Distill-Qwen-1.5B and 7B (Guo et al., 2025), OpenThinker3-7B (Guha et al., 2025), Qwen3-4B-Thinking-2507, Qwen3-30B-A3B-Thinking-2507, and QwQ-32B (Yang et al., 2025; Team, 2025). The benchmark suite is four competition-math evaluations spanning difficulty levels: AIME24, AIME25, AMC23, and MATH500 (Lightman et al., 2023). For each problem we generate 32 independent rollouts with unique random seeds, yielding a 115,200-rollout pool over 3,600 model-problem pairs and nearly 0.8 billion generated tokens. All rollouts are pre-generated; FoT and baselines operate on the same pool.

[1] We follow standard SC literature convention in using "majority" to denote the most-voted answer regardless of whether it strictly exceeds 50% of votes; on our hardest problems the top answer is often a plurality rather than a strict majority.

	Correct	Incorrect	No Answer
Proportion of pool	69.9%	28.0%	2.1%
Avg. tokens generated	5,557	8,394	31,319
Hesitation density (/1K chars)	3.47	4.17	6.94
Relative attention FLOPs	1.0×	2.3×	17.3×

Table 1: Behavioral profile of 115,200 rollouts across 6 models and 4 benchmarks (AIME24, AIME25, AMC23, MATH500). Correct rollouts commit early, use fewer hesitation tokens, and cost a fraction of the compute. No-answer rollouts consume the maximum generation budget without converging, costing 17× the attention FLOPs of correct rollouts.

3.2 Motivating Observations

Across the 115,200-rollout pool, Table 1 summarizes the behavioral profile; three observations motivate our method.

Observation 1: Early commitment predicts correctness. Correct rollouts produce a final answer (`\boxed{}`) after an average of 5,557 tokens, while incorrect rollouts take 8,394 tokens. A further 2.1% of rollouts never commit at all, consuming the maximum budget. Rollouts reaching an answer early have higher a chance of being correct, motivating an *early voting* mechanism that preserves committed answers before pruning. This suggests that for problems the model can solve quickly, additional reasoning may not improve and can even degrade the answer (Chen et al., 2025; Huang et al., 2024).

Observation 2: Late-stage generation is disproportionately wasteful. Rollouts that fail to commit early enter self-correction spirals (Huang et al., 2024; Wang et al., 2025c). Due to quadratic attention compute, no-answer rollouts, despite only taking 2.1% of the pool, cost 17.3 times the FLOPs of correct ones, consuming a disproportionate share of compute while contributing no useful answer. This asymmetry motivates targeting the wasteful tail via *rollout pruning* rather than uniformly reducing all rollouts.

Observation 3: Hesitation markers signal unproductive reasoning. Hesitation markers—`Wait`, `actually`, `perhaps`, and 18 others—are denser in incorrect rollouts across all 6 LRMs. We measure *hesitation marker density* as marker count per 1,000 characters, computed cumulatively over the generated text. Sorting rollouts into density deciles yields a monotonic accuracy decline ($r = -0.83$ across deciles, $r_{\text{pb}} = -0.30$ per rollout; 25pp gap between D1 and D10), indicating that high-density trajectories are the unproductive tail FoT should prune. We report the details of this process in Appendix A.

Algorithm 1 Funnel of Thoughts (FoT)

Require: Rollouts $\{c_1, \ldots, c_k\}$, checkpoints $\{t_1, \ldots, t_m\}$, keep ratio ρ
$\mathcal{A} \leftarrow \{1, \ldots, k\}$ ▷ active set
$\mathcal{B} \leftarrow \emptyset$ ▷ vote bank
for $j \leftarrow 1$ **to** m **do** ▷ loop over checkpoints
  if $|\mathcal{A}| \leq 2$ **then**
    break
  for each $i \in \mathcal{A}$ **do** ▷ *Early Voting*
    if HasAnswer$(c_i[1{:}t_j])$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\text{Extract}(c_i[1{:}t_j])\}$
      $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i\}$
  if $|\mathcal{A}| \leq 2$ **then**
    continue
  $d_i \leftarrow$ HesitationDensity$(c_i[1{:}t_j])$ for all $i \in \mathcal{A}$ ▷ *Rollout Pruning*
  $n \leftarrow \max(2,\ \lfloor |\mathcal{A}| \cdot \rho \rfloor)$
  $\mathcal{A} \leftarrow$ indices of n lowest-density rollouts
Complete generation for all $i \in \mathcal{A}$
return Plurality$(\mathcal{B} \cup \{\text{Extract}(c_i) : i \in \mathcal{A}\})$

3.3 Algorithm

As illustrated in Algorithm 1, Funnel of Thoughts retains the core SC@k framework, sampling k rollouts and taking the majority vote, but eliminates unproductive rollouts to save compute. The method maintains an *active set* $\mathcal{A}$ of rollouts still being generated and a *vote bank* $\mathcal{B}$ of answers from rollouts that have already committed. At each of m fixed token-count checkpoints $\{t_1, t_2, \ldots, t_m\}$, two mechanisms are applied. After the final checkpoint, the answer is selected by Plurality (argmax over answer counts) across $\mathcal{B}$ and the surviving rollouts in $\mathcal{A}$. The only hyperparameters are the checkpoint schedule and keep ratio ρ, selected via a grid sweep on the pool (§C.2) and held constant across all problems, models, and checkpoints.

Early Voting. At checkpoint t_j, we inspect the text generated so far for each active rollout c_i. Any rollout that has already produced a final answer (`\boxed{}`) is moved to the *vote bank* $\mathcal{B}$: its answer is preserved for the final vote, and the rollout is removed from the active set. This prevents correct early-committed answers from being discarded due to their hesitation marker profile. Early voting requires only substring matching, with no model inference.

Rollout Pruning. Among the remaining active rollouts, we compute hesitation marker density d_i over the entire text generated from rollout start to checkpoint t_j. Rollouts are ranked by density in ascending order; we retain the $\lfloor |\mathcal{A}| \cdot \rho \rfloor$ rollouts

Model	Method	AIME24		AIME25		AMC23		MATH500		Overall (/600)	
		Acc	FLOP	Acc	FLOP	Acc	FLOP	Acc	FLOP	Acc	FLOP
DS-R1-1.5B	SC@32	56.7	—	33.3	—	90.0	—	67.8	—	67.0	—
	AC	56.7	10.4%	33.7	12.7%	90.0	39.4%	67.7	43.3%	66.9	26.4%
	SlimSC	63.3	56.3%	40.0	68.7%	92.5	60.3%	67.4	54.5%	67.5	60.0%
	FoT@32	60.0	57.6%	33.3	54.3%	90.0	55.9%	68.2	50.3%	67.5	54.5%
DS-R1-7B	SC@32	80.0	—	53.3	—	95.0	—	69.0	—	70.5	—
	AC	80.0	31.3%	53.3	28.0%	95.0	60.8%	69.0	56.5%	70.5	44.1%
	SlimSC	83.3	53.8%	53.3	64.9%	97.5	41.0%	69.2	43.5%	71.0	50.8%
	FoT@32	80.0	54.8%	56.7	56.3%	100.0	46.6%	69.2	38.7%	71.2	49.1%
OT3-7B	SC@32	76.7	—	73.3	—	97.5	—	69.6	—	72.0	—
	AC	76.7	55.5%	73.3	49.9%	97.5	67.3%	69.6	71.7%	72.0	61.1%
	SlimSC	73.3	77.0%	50.0	81.6%	97.5	69.5%	68.8	58.0%	70.0	71.5%
	FoT@32	80.0	60.3%	76.7	60.8%	100.0	53.8%	69.6	43.6%	72.5	54.6%
Qwen3-4B	SC@32	86.7	—	86.7	—	100.0	—	72.8	—	76.0	—
	AC	86.7	57.9%	86.7	48.4%	100.0	86.9%	72.8	73.3%	76.0	66.6%
	SlimSC	76.7	65.7%	80.0	61.9%	95.0	43.7%	72.8	51.7%	74.8	55.8%
	FoT@32	86.7	58.0%	80.0	59.4%	100.0	45.3%	72.8	43.8%	75.7	51.6%
Qwen3-30B	SC@32	93.3	—	86.7	—	100.0	—	73.6	—	77.0	—
	AC	93.3	72.2%	88.3	56.8%	100.0	87.3%	73.5	71.7%	77.0	72.0%
	SlimSC	93.3	42.1%	83.3	49.9%	100.0	29.4%	73.4	35.8%	76.7	39.3%
	FoT@32	93.3	54.5%	83.3	59.0%	100.0	38.4%	73.2	37.7%	76.5	47.4%
QwQ-32B	SC@32	90.0	—	83.3	—	100.0	—	70.4	—	74.0	—
	AC	90.0	61.2%	83.3	53.8%	100.0	84.7%	70.4	74.5%	74.0	68.5%
	SlimSC	83.3	69.6%	63.3	75.5%	100.0	50.4%	70.4	45.7%	72.7	60.3%
	FoT@32	86.7	54.0%	83.3	58.1%	100.0	42.8%	70.4	33.5%	73.8	47.1%
Avg.	SC@32	80.6	—	69.4	—	97.1	—	70.5	—	72.8	—
	AC	80.6	48.1%	69.8	41.6%	97.1	71.1%	70.5	65.2%	72.7	56.5%
	SlimSC	78.9	60.8%	61.7	67.1%	97.1	49.1%	70.3	48.2%	72.1	56.3%
	FoT@32	81.1	56.5%	68.9	58.0%	98.3	47.1%	70.6	41.3%	72.9	50.7%

Table 2: Main results across 6 models and 4 competition-math benchmarks (k=32). Acc = accuracy (%); FLOP = attention-FLOP saving vs. SC@32 (%). Accuracy cells are shaded relative to SC@32 (higher , lower , unshaded within $\pm$1pp); FLOP cells are grey -shaded by saving (darker = larger). Accuracy is pooled over problems; Overall FLOP is the per-benchmark mean, so MATH500's size does not dominate.

with the lowest hesitation rates and discard the rest, where $\rho \in (0, 1]$ is a fixed keep ratio. We always retain at least 2 active rollouts.

Online Deployment. FoT integrates directly into parallel inference pipelines. At each checkpoint, the inference server pauses active rollouts, applies early voting and pruning, and resumes survivors. Terminated rollouts release their KV cache immediately, reducing memory pressure and freeing batch slots for the remaining rollouts. Because the pruning decision requires only the generated text, with no separate embedding model, reward model, or logprob access, FoT adds negligible overhead to the serving loop. We validate this integration in Section 4.4.

4 Experiments

Using the same 6-model, 4-benchmark setup from Section 3, we compare FoT against SC@32 and two efficient-SC baselines: Adaptive Consistency (AC; Aggarwal et al., 2023) and Slim-SC (Hong et al., 2025). We then test generalization to held-out models, cross-domain benchmarks, and online deployment, and ablate the contribution of each FoT mechanism. Sampling parameters, baseline configurations, and the full hyperparameter sweep are in Appendix D.

4.1 Main Results

FoT preserves SC@32 accuracy at half the compute. Across the 4 math benchmarks, FoT@32 preserves the accuracy of full self-consistency while reducing attention FLOPs by roughly half. FoT@32 and SC@32 return the same correctness outcome on 3,580 of 3,600 paired model-problem instances, differing on only 20 cases: 12 FoT wins and 8 FoT losses. This yields a non-significant difference under McNemar's exact test, with p=0.50, a net gain of 4 problems, and a paired-bootstrap 95% CI of $[-0.14, +0.36]$pp. As Table 2 demonstrates, FoT holds its FLOP saving in nearly every cell with accuracy preserved, making it a stable operating point on the accuracy–compute tradeoff.

It starts from the same 32-rollout pool as SC@32, banks rollouts that have already committed to an answer, and terminates only the active trajectories whose generated text indicates unproductive hesitation. The savings therefore come from removing late-stage waste, not from giving up the answer diversity that makes multi-sample inference useful.

	Hard regime *AIME24+25 (gap 22.4pp)*		Easy regime *AMC+MATH500 (gap 7.3pp)*	
Method	Acc	FLOP↓	Acc	FLOP↓
Pass@1	60.7	—	70.9	—
Pass@32	83.1	—	78.2	—
SC@32	75.0	—	72.5	—
AC	75.2	44.9%	72.5	65.6%
SlimSC	70.3	64.0%	72.3	48.3%
FoT@32 (Ours)	**75.0**	**57.3%**	**72.7**	**41.7%**

Table 3: Effectiveness of each method split by pass@k gap. FoT preserves SC@32 accuracy while saving more compute in the hard split where consensus is slow; accuracy in %, FLOP saving relative to SC@32.

AC struggles when rollouts diverge. Adaptive Consistency reduces cost by deciding when enough rollouts have been sampled. This is effective when the model reaches answer consensus quickly, but the same condition also means SC@32 was over-provisioned for that benchmark. The pass@1–pass@32 gap reveals the difficulty split as a factor: when the gap is large, the correct answer is often present somewhere in the rollout pool, but completed rollouts do not agree quickly enough for sample-axis early stopping. Table 3 shows that AC is strongest on easier cells with fast consensus, whereas FoT continues to save compute in the hard split because its decision is made within each rollout. Compared with AC, FoT is more robust on hard samples where consensus is slow but individual trajectories still reveal pruning signals.

Slim-SC over-prunes when faced with hard samples. Slim-SC exposes a different failure mode: pruning by inter-trajectory similarity can discard useful diversity. This weakness is most visible on AIME25, where Slim-SC saves 67.1% FLOPs but drops the average accuracy from 69.4 to 61.7 (Table 2). On hard problems, correct solutions often share surface structure even when they provide independently useful evidence for the final vote, so treating similar thought segments as redundant can collapse the pool before enough support accumulates. FoT targets a different property: its hesitation-density signal asks whether a trajectory is spiraling, not whether it resembles another trajectory. Compared with Slim-SC, FoT better preserves the reasoning diversity that gives multi-sample inference its advantage.

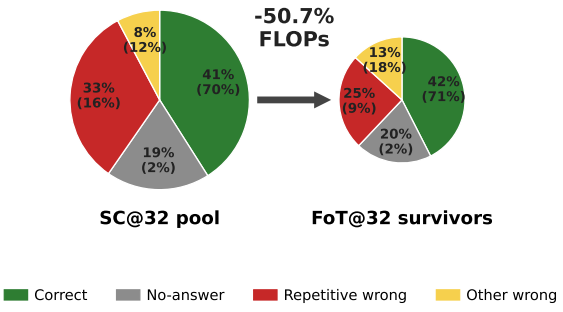

Figure 3: **FoT preserves correct trajectories while pruning repetitive failures.** Pie size indicates total attention FLOPs; slice labels show compute share, with rollout-count share in parentheses.

FoT prunes the failure mode it was designed to catch. Figure 3 decomposes the rollout pool by both compute mass and sample count. The key pattern is that correct trajectories keep nearly the same sample share after pruning, while repetitive wrong trajectories lose compute mass. A checkpoint-level diagnostic on the AIME/AMC stress subset gives the same picture for no-answer rollouts: Table 13 shows that FoT removes 69% of the non-committing tail by the final checkpoint, with the largest reductions in cells where the issue is most acute. This supports the mechanism behind FoT: hesitation-density pruning does not reshape the answer distribution uniformly, but preferentially removes long repetitive loops (Wang et al., 2025c; Su et al., 2025) that consume disproportionate attention cost and add noisy votes. The pie chart makes the motivation explicit: correct paths occupy about 70% of the sample space but only about 40% of the compute, so the savings must come from the wasteful tail.

4.2 Cross-Model Generalization

FoT transfers across model architectures because its pruning rule is relative rather than model-calibrated. We apply the same token set, checkpoints, and keep ratio to four held-out models: Qwen3-14B, Qwen3-32B, Skywork-OR1-7B (He et al., 2025), and Phi-4-reasoning (Abdin et al., 2025). These models differ in their absolute hesitation-marker density (Table 11), but FoT only compares rollouts within the same active pool at the same checkpoint. A model can therefore be more

Benchmark	SC@32	FoT@32	Δ (pp)	Token↓	FLOP↓
Qwen3-14B					
AIME24	83.3	83.3	+0.0	32.5%	53.6%
AIME25	80.0	80.0	+0.0	38.8%	59.6%
AMC23	100.0	100.0	+0.0	18.7%	40.1%
MATH500	70.4	70.6	+0.2	12.7%	35.6%
Qwen3-32B					
AIME24	86.7	83.3	−3.4	32.3%	55.1%
AIME25	80.0	80.0	+0.0	38.3%	60.1%
AMC23	97.5	97.5	+0.0	18.7%	41.0%
MATH500	71.0	71.2	+0.2	12.2%	35.2%
Skywork-OR1-7B					
AIME24	80.0	80.0	+0.0	36.1%	56.6%
AIME25	60.0	60.0	+0.0	39.9%	60.5%
AMC23	95.0	95.0	+0.0	25.7%	53.1%
MATH500	70.8	70.8	+0.0	18.9%	48.6%
Phi-4-reasoning					
AIME24	50.0	56.7	+6.7	35.9%	58.0%
AIME25	36.7	53.3	+16.6	38.9%	61.5%
AMC23	55.0	67.5	+12.5	30.4%	54.4%
MATH500	46.6	47.0	+0.4	39.6%	57.3%
Aggregate	1600→1615 of 2400		+0.6	29.4%	**51.9%**

Table 4: Cross-model transfer with unchanged FoT configuration. FoT preserves SC@32 accuracy on held-out models while saving 51.9% FLOPs on average.

or less verbose overall; what matters is whether a trajectory is unusually hesitation-heavy relative to its peers. Table 4 shows that this criterion preserves SC@32 accuracy with comparable FLOP savings across the held-out models. Phi-4-reasoning is the most architecturally distinct case and also shows the largest transfer gain, further supporting the interpretation. We interpret this gain as arising from Phi-4-reasoning's heavy no-answer tail, reported in Table 12, which FoT can capture through the same relative pruning rule.

4.3 Cross-Domain Generalization

We next ask whether the same signal survives when the answer format changes. GPQA-Diamond (Rein et al., 2023) changes the knowledge domain and final-answer format while retaining natural-language deliberation; LCB-Lite (Jain et al., 2024) is a stronger stress test because the final artifact is executable code and two correct solutions may be syntactically unrelated. If FoT were exploiting math-specific answer syntax or majority-vote structure, we should not expect it to help under execution-based scoring. Instead, Table 6 shows that the unchanged configuration transfers to both settings, and on LCB FoT improves mean per-rollout pass rate while reducing FLOPs. The implication is that hesitation density acts before answer extraction: it identifies low-value continuations in the reasoning process, whether the final output is a boxed number, a multiple-choice option, or code.

Method	Acc (%)	Wall (s)	Wall↓	Gen FLOPs	FLOP↓	KV (M)	KV↓
Pass@1	53.3	4,145	—	3.5×10^9	—	0.37	—
SC@32	76.7	21,561	—	1.29×10^{11}	—	12.88	—
FoT@32	80.0	**13,445**	**37.6%**	$\mathbf{5.67 \times 10^{10}}$	**56.1%**	**8.51**	**33.9%**

Table 5: Online deployment on AIME24 with DS-R1-7B (k=32). Wall time and FLOPs measured end-to-end on a single A100. KV (M) = cumulative KV-cache occupancy in millions of generated tokens; pruned and banked rollouts release their slots at each checkpoint.

Model	SC@32	FoT@32	Δ (pp)	FLOP↓
GPQA-Diamond (accuracy %)				
DS-R1-1.5B	38.9	41.4	+2.5	42.2%
DS-R1-7B	55.6	55.6	+0.0	31.8%
OT3-7B	51.5	53.0	+1.5	56.8%
Qwen3-4B	65.7	67.7	+2.0	35.3%
Qwen3-30B	72.2	72.2	+0.0	38.7%
QwQ-32B	63.6	65.2	+1.6	41.4%
Avg.	57.9	**59.2**	+1.3	**41.0%**
LCB-Lite (mean pass rate %)				
DS-R1-1.5B	15.10	16.29	+1.19	56.6%
DS-R1-7B	36.59	39.49	+2.90	50.3%
OT3-7B	43.63	44.60	+0.97	62.2%
Qwen3-4B	49.23	51.78	+2.55	53.5%
Qwen3-30B	69.36	69.38	+0.03	60.0%
QwQ-32B	61.30	62.15	+0.85	57.9%
Avg.	45.87	**47.28**	+1.42	**56.8%**

Table 6: Cross-domain transfer with unchanged FoT configuration. GPQA reports accuracy; LCB reports mean pass rate. Δ is FoT−SC in percentage points.

4.4 Online Deployment

We deploy FoT in an online inference pipeline using SGLang (Zheng et al., 2024) on a single A100 GPU, generating all rollouts end-to-end for 30 AIME24 problems with DeepSeek-R1-Distill-Qwen-7B. Table 5 shows that FoT@32 solves 24 of 30 problems versus 23 for SC@32, while reducing wall time by 37.6% and generation FLOPs by 56.1%. The same logs show that the cumulative generated-token footprint, and hence KV-cache occupancy, drops from 12.88M to 8.51M tokens. The wall-time savings follow directly from pruning: terminated rollouts release KV cache memory and batch slots, reducing both compute and memory pressure for the remaining rollouts. Because the hesitation marker count is a string operation with negligible overhead, FoT requires no additional model inference beyond the rollouts themselves.

k	SC@k	FoT@k	Δ
8	72.4 ± 0.2	71.9 ± 0.2	-0.5
12	72.6 ± 0.1	72.3 ± 0.2	-0.2
16	72.7 ± 0.1	72.6 ± 0.1	-0.0
24	72.8 ± 0.1	72.9 ± 0.1	+0.1
32	72.8	72.9	+0.1

Table 7: Pool-size scaling on the 4-benchmark pool. For k<32, results are averaged over 5 subset seeds ($\pm$ std).

	w/o Early Voting		**w/ Early Voting**	
Pruning Signal	Correct	FLOP↓	Correct	FLOP↓
None (SC@32)	72.8 (baseline)		—	
Random	72.5 (-0.3)	48.0%	72.8 ($+0.0$)	48.4%
Hesitation density	72.9 ($+0.2$)	51.7%	**72.9 ($+0.1$)**	**48.7%**

Table 8: 2×2 ablation: pruning signal × early voting on the 4-benchmark pool (3,600 model-problem pairs). Δ is relative to SC@32 (72.8). Random pruning results averaged over 10 seeds. FLOP saving here is pooled over all rollouts.

4.5 Ablations

We ablate two properties of FoT in the main text: how its accuracy-compute tradeoff changes with rollout-pool size and the contribution of each mechanism. Kernel-size robustness and additional sensitivity analyses are deferred to Appendix C.

Pool-size scaling. We run FoT@k and SC@k for $k \in \{8, 12, 16, 24, 32\}$ on the full 4-benchmark pool, averaging over 5 random subset seeds for k<32. Table 7 shows that FoT is a large-pool method: pruning is mildly harmful at small k, becomes neutral around k=16, and turns positive once enough rollout diversity is available. This is the expected tradeoff. When the pool is small, pruning removes too much vote diversity; once the pool is large enough, FoT can remove unproductive rollouts while preserving the multi-sample advantage.

Mechanism ablation. Table 8 reports a two-by-two ablation on the same 4-benchmark pool as the main result: random vs. hesitation-density pruning, each with and without early voting, plus SC@32 as the no-pruning baseline. The result is clear: the lexical pruning signal, not early voting, is what preserves SC@32 accuracy under pruning. Random pruning loses accuracy at similar compute savings, while hesitation-density pruning preserves SC@32 accuracy even without banking committed answers. Early voting therefore plays a conservative role in the final method. It protects committed answers and repairs random pruning, but once the pruning signal is already selective, its aggregate effect is small.

Full sweep results, hyperparameter sensitivity, per-model calibration, kernel-size robustness, and the per-difficulty analysis are reported in Appendix C.

5 Discussion

Our findings suggest that predicting per-rollout correctness from hesitation density is fragile, since the same markers appear in both correct and incorrect reasoning and only moderately separate them, with $r_{\text{pb}} = -0.30$. The reliable signal is aggregate: error rate rises monotonically with density, with a decile-level $r = -0.83$, so the high-density tail is disproportionately wrong. This is why FoT works by removing that tail from the answer distribution, pruning the votes least likely to be right and the compute least likely to pay off. It needs no confidence model, reward model, or logit access, and matches SC@32 at roughly half the compute.

The key reason this replacement transfers is that FoT does not rely on a model-specific threshold for how often a hesitation marker should appear. Absolute marker frequency varies across LRMs, but failed long-form reasoning often exposes a shared lexical pattern: repeated revision, backtracking, and non-commitment. FoT uses this pattern comparatively, ranking active rollouts within the same model, problem, and checkpoint. As a result, the signal can survive architectural changes and domain shifts: a Qwen-family model, Phi-4-reasoning, a graduate-level science question, and a code-generation task need not use identical wording for FoT to identify the unusually unproductive tail of the current pool.

This also clarifies why FoT differs from the efficient-SC baselines. Adaptive Consistency is strongest when the answer distribtion stabilizes quickly, and Slim-SC is strongest when similar trajectories are genuinely redundant. Those are precisely the cases where multiple sampling contributes less. In the harder cases, consensus may be slow and superficially similar derivations may still carry useful voting evidence. FoT is designed for this: it preserves the pool-level diversity that makes SC effective, while pruning within-trajectory waste before it dominates the compute budget. This is why its advantage is most persuasive on hard cells, held-out models, and domain shifts, where robustness matters more than early consensus.

6 Limitations

Our evaluation is centered on reasoning tasks with extractable final answers. We include cross-domain tests on GPQA-Diamond and LCB-Lite, but these still have relatively well-defined answer or execution-based evaluation. Open-ended generation, multi-turn interaction, and tasks where correctness cannot be reduced to answer extraction remain outside the scope of this study.

FoT also relies on a commitment detector for early voting. In our math setting this is the `\boxed{}` convention, while other domains require task-specific answer extraction or prompting. The pruning signal itself is independent of the final-answer format, but the vote bank is only as reliable as the commitment detector.

Finally, FoT uses a fixed checkpoint schedule and an English hesitation-marker kernel. The relative-density rule reduces sensitivity to model-level verbosity, but models reasoning in other languages or with substantially different deliberation styles may require revalidating the marker set. Our FLOP accounting assumes standard full attention; under efficient-attention mechanisms the saving scales down toward the token-reduction floor, from 48.7% to 23.0% in the linear limit, which we quantify across window sizes in Appendix D.2. Pruning still releases KV cache and batch slots regardless of the attention regime. Adaptive checkpointing based on per-problem difficulty is another natural extension that could further improve the efficiency–accuracy tradeoff.

References

Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, Piero Kauffmann, Yash Lara, Caio César Teodoro Mendes, Arindam Mitra, Besmira Nushi, Dimitris Papailiopoulos, Olli Saarikivi, Shital Shah, Vaishnavi Shrivastava, and 4 others. 2025. Phi-4-reasoning technical report. *Preprint*, arXiv:2504.21318.

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396, Singapore. Association for Computational Linguistics.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. 2025. Efficiently scaling llm reasoning with certaindex. *Preprint*, arXiv:2412.20993.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *Preprint*, arXiv:2506.04178.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Yang Liu, and Yahui Zhou. 2025. Skywork open reasoner series. https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reaonser-Series-1d0bc9ae823a80459b46c149e41 Notion Blog.

Colin Hong, Xu Guo, Anand Chaanan Singh, Esha Choukse, and Dmitrii Ustiugov. 2025. Slim-sc: Thought pruning for efficient scaling with self-consistency. *Preprint*, arXiv:2509.13990.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. *Preprint*, arXiv:2310.01798.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof qa benchmark. *Preprint*, arXiv:2311.12022.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. 2025. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *Preprint*, arXiv:2505.00127.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, page 20090–20111. Association for Computational Linguistics.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635, Albuquerque, New Mexico. Association for Computational Linguistics.

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025a. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. *Preprint*, arXiv:2506.08343.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025b. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6919–6932, Albuquerque, New Mexico. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025c. Thoughts are all over the place: On the underthinking of o1-like llms. *Preprint*, arXiv:2501.18585.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Menghua Wu, Cai Zhou, Stephen Bates, and Tommi Jaakkola. 2025a. Thought calibration: Efficient and confident test-time scaling. *Preprint*, arXiv:2505.18404.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025b. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *Preprint*, arXiv:2408.00724.

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

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. Sglang: Efficient execution of structured language model programs. *Preprint*, arXiv:2312.07104.

A Motivating Analysis

Token	r_{pb}	p	Prev. (%)	W/C
`perhaps`	−0.417	<1e-300	58.5	2.63×
`Wait,`	−0.249	<1e-268	99.8	1.43×
`Wait`	−0.236	<1e-241	99.8	1.39×
`actually`	−0.191	<1e-156	83.6	1.55×
`no,`	−0.181	<1e-140	81.8	1.46×
`incorrect`	−0.168	<1e-121	16.3	2.74×
`wrong`	−0.145	<1e-90	37.1	1.76×
`Or`	−0.126	<1e-69	19.9	2.07×
`However`	−0.118	<1e-61	32.0	1.67×
`Alternatively`	−0.114	<1e-56	70.1	1.34×
`Maybe`	−0.097	<1e-41	75.4	1.26×
`But wait`	−0.085	<1e-32	57.9	1.32×
`Actually`	−0.080	<1e-29	5.3	2.18×
`instead`	−0.075	<1e-25	45.9	1.28×
`Let me think`	−0.061	<1e-17	50.2	1.20×
`I was wrong`	−0.048	<1e-11	1.3	2.45×
`Let me try`	−0.028	1.1e-4	49.4	1.10×
`Let me reconsider`	−0.024	8.7e-4	0.1	4.95×
`Re-examining`	−0.017	1.7e-2	<0.1	—
`This isn't`	−0.016	2.9e-2	<0.1	8.20×
`I think I made`	−0.015	4.3e-2	4.2	1.19×

Table 9: Operational 21-marker kernel. Tokens are sorted by $|r_{\text{pb}}|$; all have $r < 0$ and $p < 0.05$. Prev. is rollout prevalence; W/C is wrong-to-correct frequency ratio.

Decile	Density range	Accuracy (%)
D1	0.00–1.11	75.7
D2	1.11–1.76	76.1
D3	1.76–2.30	74.1
D4	2.30–2.80	74.0
D5	2.80–3.32	73.2
D6	3.32–3.87	73.1
D7	3.87–4.57	72.3
D8	4.57–5.49	68.6
D9	5.49–6.75	61.4
D10	>6.75	50.7

Table 10: Accuracy by hesitation-marker density decile across 115,200 rollouts. Density is marker count per 1K characters; $r = -0.83$ between decile index and accuracy, with a 25.0pp D1–D10 gap.

Model	All	Correct	Wrong	Wrong/Correct
DS-R1-1.5B	4.05	3.11	5.44	1.75
DS-R1-7B	3.37	2.93	4.28	1.46
OT3-7B	5.11	5.04	5.27	1.04
Qwen3-4B	3.13	3.08	3.25	1.06
Qwen3-30B	2.44	2.44	2.43	1.00
QwQ-32B	4.30	4.18	4.62	1.10

Table 11: Per-model hesitation-marker density on the calibration suite. Density is marker count per 1K characters; wrong/correct reports the ratio between incorrect and correct rollouts.

A.1 Hesitation-Marker Validation

Table 9 shows the 21 hesitation markers used by FoT. We select markers whose density has a significant negative point-biserial correlation with rollout correctness, so higher frequency indicates a greater association with incorrect rollouts. The resulting kernel spans common revision markers such as `Wait,` and `actually`, as well as rarer but highly skewed phrases such as `Let me reconsider`.

We exclude markers that correlate positively with correctness, such as `mistake`, `Oh`, and `Hmm`, because they more often indicate productive self-correction. The aggregate density of the 21 retained markers yields $r_{\text{pb}} = -0.30$, making the composite signal stronger than most individual markers.

Table 10 gives the decile-level diagnostic behind Observation 3. Accuracy declines from 75.7% in the lowest-density decile to 50.7% in the highest-density decile. The signal is not intended to identify the single correct rollout; rather, it separates the unproductive high-density tail that FoT should prune.

A.2 Marker-Selection Validation

A potential concern is that the marker list may overfit to the lexical styles of the models and benchmarks used for calibration. We address this primarily through the held-out model and cross-domain experiments in Section 4.2 and Section 4.3, where the same 21-marker kernel is applied without retuning.

Table 11 reports the absolute marker density by model. The scale varies substantially, but FoT does not apply a fixed density threshold across models; it ranks rollouts within the same active pool at the same checkpoint. This relative rule is why a shared marker kernel can transfer across models with different verbosity levels.

Phi-4-reasoning diagnostic. Table 12 reports the rollout profile behind the Phi-4 cross-model result in Section 4.2. Phi-4 has both a large pass@32–SC@32 gap and an unusually high number of no-answer rollouts, making it a useful diagnostic for whether FoT can remove non-committing trajectories without retuning.

No-answer tail diagnostic. Table 13 reports the checkpoint-level analysis used to interpret Figure 3. FoT does not identify every failed rollout independently; instead, it progressively removes the non-committing tail from the active pool, with the largest reductions in the cells where no-answer behavior is most acute.

Benchmark	Pass@1	SC@32	Pass@32	No-ans./32
AIME24	15.7	50.0	73.3	19.0
AIME25	13.6	36.7	66.7	17.2
AMC23	16.8	55.0	87.5	18.2
MATH500	11.5	46.6	59.6	21.0

Table 12: Phi-4-reasoning rollout profile at k=32. Accuracy columns are percentages; no-answer reports the mean number of rollouts per problem that never produce an extractable final answer.

Subset	Start	Ckpt 1	Ckpt 2	Final
AIME/AMC avg.	2.67	1.77	1.20	0.83
Qwen3-4B/AIME25	9.4	–	–	2.6
OT3-7B/AIME25	5.4	–	–	1.6

Table 13: No-answer rollouts remaining under FoT over the checkpoint sequence on the AIME24/AIME25/AMC23 stress subset. Entries report the mean count per 32-rollout pool; the aggregate removes 69% of no-answer rollouts by the final checkpoint.

		Accuracy (%)			
Model	Dataset	Pass@1	Pass@32	$\text{Gap}_{1\to32}$	SC@32
DS-R1-1.5B	AIME24	29.2	80.0	50.8	56.7
	AIME25	24.3	53.3	29.1	33.3
	AMC23	71.2	100.0	28.8	90.0
	MATH500	62.6	74.0	11.4	67.8
DS-R1-7B	AIME24	54.9	83.3	28.4	80.0
	AIME25	39.1	66.7	27.6	53.3
	AMC23	90.5	100.0	9.5	95.0
	MATH500	68.3	73.0	4.7	69.0
OT3-7B	AIME24	63.6	90.0	26.4	76.7
	AIME25	60.7	80.0	19.3	73.3
	AMC23	93.1	100.0	6.9	97.5
	MATH500	68.8	72.6	3.8	69.6
Qwen3-4B	AIME24	70.6	90.0	19.4	86.7
	AIME25	68.8	86.7	17.9	86.7
	AMC23	99.0	100.0	1.0	100.0
	MATH500	72.3	81.0	8.7	72.8
Qwen3-30B	AIME24	86.8	93.3	6.6	93.3
	AIME25	80.9	90.0	9.1	86.7
	AMC23	99.7	100.0	0.3	100.0
	MATH500	73.0	83.0	10.0	73.6
QwQ-32B	AIME24	80.2	93.3	13.1	90.0
	AIME25	69.7	90.0	20.3	83.3
	AMC23	98.2	100.0	1.8	100.0
	MATH500	70.4	75.2	4.8	70.4
Average		70.7	85.6	15.0	79.4

Table 14: Pass@1 vs. Pass@32 accuracy gap across 6 models and 4 benchmarks (k=32). Pass@1 = average single-rollout accuracy; Pass@32 = oracle accuracy (at least one correct among 32); $\text{Gap}_{1\to32}$ = Pass@32 − Pass@1 (pp). The average gap of 15.0pp confirms that individual rollouts are highly stochastic. SC@32 recovers much of this gap via majority voting, but at full compute cost. Averages here are unweighted means over the 24 model–benchmark cells; the problem-count-weighted aggregate in Table 2 is dominated by MATH500 and is therefore lower.

A.3 Pass@1 vs. Pass@32 Performance Gap

Table 14 reports the per-rollout accuracy, oracle accuracy, and SC@32 majority-vote accuracy for all 6 models across 4 benchmarks. The average pass@1–pass@32 gap is 15.0 percentage points, confirming that individual rollouts are highly stochastic and that correct answers often appear somewhere in the 32-rollout pool. The gap is largest for smaller models on harder benchmarks and smallest for stronger models near ceiling.

B Additional Experiments

This section collects secondary diagnostics behind the main results: per-model behavior, the cases where FoT and SC@32 disagree, and benchmark difficulty effects.

B.1 Per-Model Accuracy Breakdown

Figure 4 breaks down accuracy by model on the 4-benchmark math suite. The pattern mirrors the main table: FoT is most helpful when the base model still has substantial sampling variance, while near-ceiling models leave less room for improvement. SlimSC is less stable on the harder AIME split, consistent with the main-text observation that similarity pruning can remove useful vote diversity.

B.2 Discordant Instance Analysis

Figure 5 examines cases where FoT and SC@32 produce different correctness outcomes on the AIME/AMC subset. FoT wins more often than it loses, and the losses cluster on harder problems with low pass rates. This supports the main-text interpretation that FoT usually preserves the SC answer distribution, while occasionally changing the final vote after removing non-committing or repetitive trajectories.

B.3 Vote-Concentration Diagnostic

Figure 6 asks when FoT changes the final SC decision. We bin each model-problem pair by the fraction of rollouts assigned to the SC@32 plurality answer. Four pairs with no extractable SC answer are excluded from this diagnostic, leaving 3,596 pairs.

The pattern is consistent with FoT's intended role. When SC@32 has a clear majority, FoT almost always preserves it: in the $> 50\%$ bin, the final answer changes in only 0.2% of cases. In contrast, low-consensus pools are both more diverse and less reliable: the $< 25\%$ bin averages 15.2

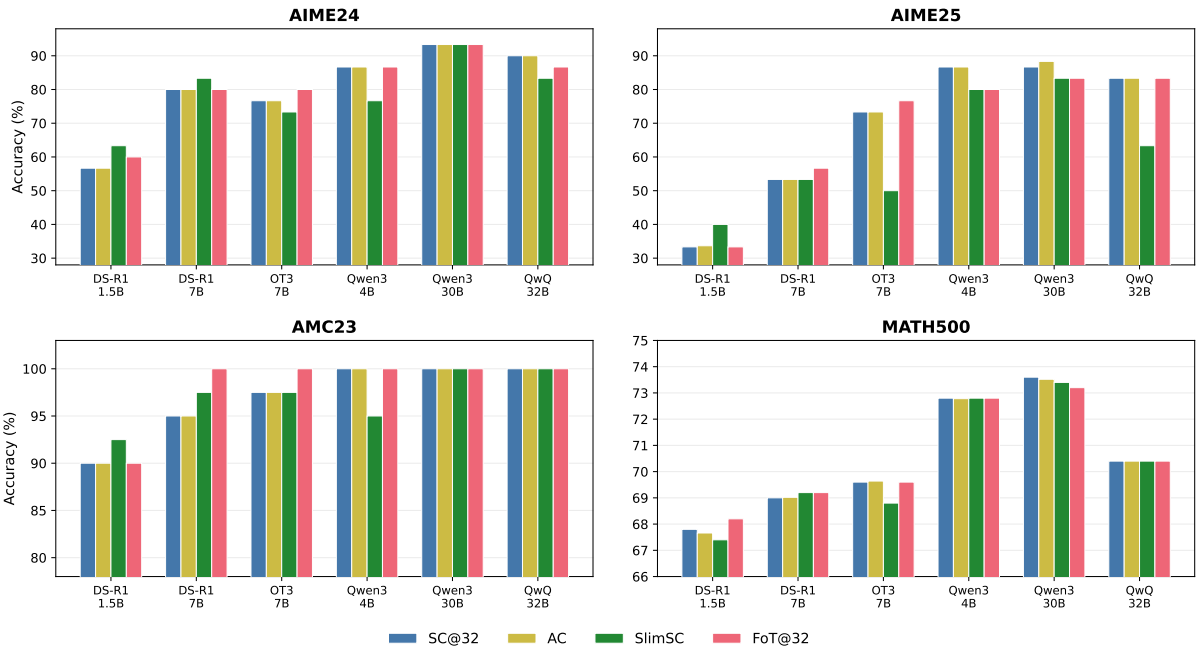

Figure 4: Per-model accuracy breakdown across the 4-benchmark math suite.

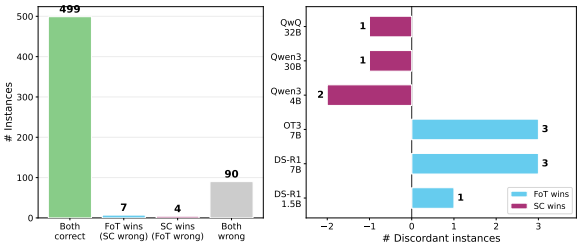

Figure 5: Discordant cases on the AIME/AMC subset. Left: SC@32–FoT@32 concordance. Right: FoT wins and losses by model.

distinct answers, and FoT changes the final answer in 49.1% of cases. This is also where most net corrections arise (+4). The effect is modest, but it explains why FoT can slightly improve aggregate accuracy while primarily serving as a compute-reduction method: it leaves strong SC trends intact and acts mainly on the divergent tail.

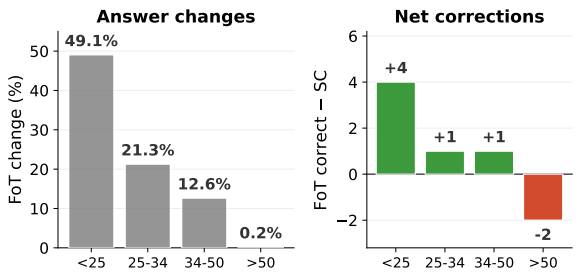

Figure 6: Vote-concentration diagnostic on the 4-benchmark pool. Bins are SC@32 plurality-share ranges. Left: how often FoT changes the SC answer. Right: FoT correct minus SC correct within each bin.

B.4 Difficulty Adaptation

MATH500 (Lightman et al., 2023) sits at the easier end of our math suite. Table 2 shows that FoT maintains accuracy parity on MATH500 while saving less compute than on AIME/AMC. This is expected: easier problems produce shorter rollouts with less hesitation tail to prune, so many trajectories commit early instead of entering the pruning regime. The result clarifies that FoT's savings track late-stage reasoning waste rather than applying a fixed reduction uniformly across benchmarks.

C Additional Ablations

C.1 Hyperparameter Sensitivity

To validate our hyperparameter choices, we conduct a comprehensive sweep over 340 configurations spanning checkpoint count (1–5), checkpoint placement, and keep ratio ($\rho \in \{0.50, 0.60, 0.67, 0.75, 0.85\}$) on the unified 4-benchmark calibration pool (6 models $\times$ 600 problems $\times$ 32 rollouts $=$ 115,200 rollouts, with inverse-frequency stratified weighting so MATH500's larger problem count does not dominate the smaller benchmarks). Figure 7(a) plots all configurations on accuracy vs. FLOP savings axes, color-coded by checkpoint count.

Two natural Pareto candidates emerge. The *math-only* winner ($\{6\text{K}, 11\text{K}, 16\text{K}\}$, ρ=0.67) maximizes accuracy on the math calibration pool (Δ=+9 / 3,600 problems, +0.25pp). The *joint*

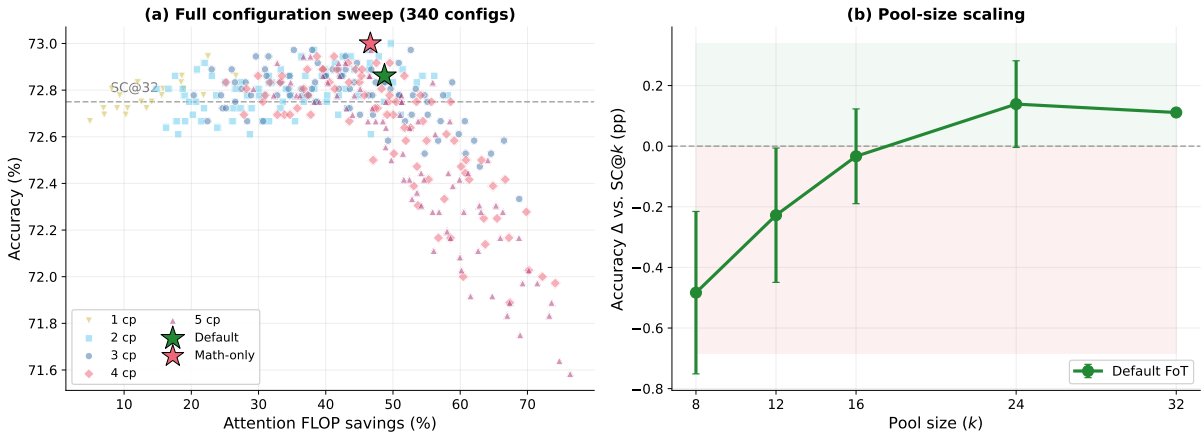

Figure 7: Hyperparameter sensitivity analysis. **(a)** Full 340-configuration sweep on the 4-benchmark calibration pool. Stars mark the default joint-Pareto config and the math-only in-domain maximizer. **(b)** Pool-size scaling for the default config.

winner ($\{6K, 10K, 14K\}$, ρ=0.67; green star), which we use throughout the paper, sacrifices 5 of those math gains in exchange for substantially larger gains on out-of-distribution evaluations: +15 problems on GPQA-Diamond (+1.26pp on 1,188 model-problem pairs) and +14 problems on the cross-architecture Phi-4-reasoning evaluation (+2.33pp on 600 model-problem pairs spanning the same four math benchmarks).

Why not the math-only winner? The math-only schedule is the best choice if the sole objective is maximizing the four-benchmark math aggregate. We instead use the joint configuration because the paper's claim is model- and domain-level transfer: it gives up 5 math problems but recovers larger gains on the held-out Phi-4 and GPQA evaluations. This makes the default configuration a deliberately conservative operating point rather than the in-domain maximizer.

Pool-size behavior. Figure 7(b) shows the expected tradeoff for the default schedule: pruning is too aggressive when the initial pool is small, but the gap closes as k grows and becomes slightly positive at $k \geq 24$. This supports using FoT primarily as a replacement for large-pool SC rather than as a small-k method.

Schedule	Keep ratio ρ 0.50	0.67	0.75
Early 4K, 8K, 12K	2604 (−15)	2617 (−2)	2617 (−2)
Mid 6K, 10K, 14K	2616 (−3)	**2623 (+4)**	2623 (+4)
Math-only 6K, 11K, 16K	2622 (+3)	2628 (+9)	2621 (+2)
Late 8K, 12K, 16K	2617 (−2)	2623 (+4)	2623 (+4)

Table 15: Ablation over checkpoint schedules and keep ratio ρ, aggregated across 6 models and the 4-benchmark pool (3,600 model-problem pairs per cell). Cells show FoT correct count (Δ vs. SC = 2619). The universal joint-Pareto config (Mid $\{6K, 10K, 14K\}$, ρ=0.67; shaded) is our default; Math-only $\{6K, 11K, 16K\}$ at ρ=0.67 wins on math-only accuracy but loses on out-of-distribution generalization.

C.2 Checkpoint and Threshold Sensitivity

We sweep four checkpoint schedules (Early: $\{4K, 8K, 12K\}$, Mid: $\{6K, 10K, 14K\}$, Math-only: $\{6K, 11K, 16K\}$, Late: $\{8K, 12K, 16K\}$) crossed with three keep ratios ($\rho \in \{0.50, 0.67, 0.75\}$) on the unified 4-benchmark pool (3,600 model-problem pairs per cell). Table 15 shows the results.

Two patterns emerge. First, the Early schedule is consistently the worst, hurting accuracy at every keep ratio ($\Delta \leq -2$, down to -15 at ρ=0.50): early checkpoints prune before the hesitation marker signal has accumulated sufficiently, removing rollouts that would have converged correctly given more tokens. Second, the Mid, Math-only, and Late schedules all produce viable configurations ($\Delta \geq 0$) at $\rho \in \{0.67, 0.75\}$, with the math-only winner reaching Δ=+9 on math but giving up the larger out-of-distribution gains discussed in Section C.1. The universal Mid schedule with ρ=0.67, used throughout the paper, is the joint Pareto choice balancing math accuracy (Δ=+4)

against the larger OOD gains.

Model	Best Config	Per-Model		Universal	
		Δ	FLOP↓	Δ	FLOP↓
DS-R1-1.5B	Math-only, ρ=0.50	+10	63.2	+3	52.7
DS-R1-7B	Mid, ρ=0.67	+4	46.5	+4	46.5
OT3-7B	Late, ρ=0.67	+4	45.3	+3	51.2
Qwen3-4B	Late, ρ=0.75	+0	34.1	−2	49.1
Qwen3-30B	Math-only, ρ=0.67	−1	43.0	−3	45.3
QwQ-32B	Math-only, ρ=0.67	+0	42.1	−1	43.9
Agg.		**+17**	**45.7**	**+4**	**48.1**

Table 16: Per-model calibration vs. the universal Mid ρ=0.67 config. Each model's optimal config is selected on its own 600-problem math pool; Δ is relative to SC@32, and Agg. sums across 3,600 model-problem pairs.

C.3 Per-Model Calibration

All main experiments use a single universal config (Mid schedule $\{6\text{K}, 10\text{K}, 14\text{K}\}$, ρ=0.67). We ask: does per-model hyperparameter tuning improve results? For each model, we sweep 12 configurations (4 schedules × 3 keep ratios) on that model's 600 problems across the four math benchmarks independently.

Table 16 shows that per-model calibration yields a small aggregate improvement: Δ=+17 vs. universal +4 on 3,600 model-problem pairs (a marginal +13 problems, +0.36pp). The bulk of this gain concentrates on the weakest model in the suite (DS-R1-1.5B picks Math-only ρ=0.50 for Δ=+10 vs. universal +3); the remaining five models gain ≤ 2 problems each from per-model tuning. Three of six models select the Math-only schedule $\{6\text{K}, 11\text{K}, 16\text{K}\}$, matching the in-domain tradeoff described above, but the preferred schedules are not stable enough to justify model-specific tuning. We therefore keep the universal config as the default.

C.4 Per-Model Config Transfer

To test whether per-model configs transfer, we apply each of the five distinct optimal configs from Section C.3 to all 6 models. Table 17 shows that the math-only Pareto winner, Math-only $\{6\text{K}, 11\text{K}, 16\text{K}\}$ with ρ=0.67, achieves the highest cross-model math aggregate (Δ=+9 on 3,600), but this is the same in-domain tradeoff described in Section C.1. Our universal Mid config (ρ=0.67, Δ=+4 on math) is the joint Pareto choice that balances math performance against out-of-distribution generalization. No per-model-tuned configuration dominates the universal config across both math and OOD evaluations simultaneously.

C.5 Kernel-Size Robustness

We sweep the hesitation-marker kernel from 1 to 21 tokens, adding tokens in descending order of $|r_{\text{pb}}|$ (Table 9). All other hyperparameters are held at the paper default ($\{6\text{K}, 10\text{K}, 14\text{K}\}$, ρ=0.67, hesitation marker density). Table 18 reports the full sweep on the unified 4-benchmark pool (3,600 model-problem pairs).

Two findings. First, accuracy is essentially flat across kernel sizes: Δ ranges from −3 to +6 problems on 3,600, with all 8 settings within 0.17pp of one another and well within single-rollout sampling noise. Even the top-1 kernel (`perhaps` alone) preserves SC@32 accuracy to within 3 problems. Second, FLOP saving is constant to two significant figures (50.59–50.76%), confirming that the pruning behavior is driven by aggregate hesitation density rather than the identity of any individual marker. The three highest-$|r|$ markers (`perhaps`, `Wait,`, `actually`) match the full 21-token kernel within sampling noise (Δ=+5 vs. +4).

This robustness is itself the substantive finding: the lexical signal is concentrated in a small number of canonical English hesitation markers, and the method does not depend on the specific composition of the kernel. We retain the 21-token kernel as the calibrated default for stability, but smaller kernels (down to top-3) are an equivalent operating point.

C.6 Early Voting Analysis

Figure 8 analyzes the early voting mechanism on the AIME/AMC subset under the current FoT configuration. On average, 14.7 of 32 rollouts are banked before pruning begins. The bank contains a correct answer in 78% of all instances, rising to 71% on FoT wins and falling to 25% on SC wins. This confirms that early commitment is a useful correctness signal, while the 4-benchmark ablation in Table 8 shows that the aggregate gain comes primarily from hesitation-density pruning.

D Implementation Details

D.1 Generation Details

All rollouts are generated using SGLang (Zheng et al., 2024) with the following parameters: temperature T=0.6, top-p=0.95, top-k=30, and a maximum generation budget of 32,768 tokens. Each model uses a rolling random seed starting from 42 (i.e., seed = 42 + rep_id) to ensure reproducibility while maintaining independence across rollouts.

Source Config	Target Model (Δ vs. SC@32 on 600 problems)						
	DS-R1-1.5B	DS-R1-7B	OT3-7B	Qwen3-4B	Qwen3-30B	QwQ-32B	Agg.
Mid, ρ=0.67 (Ours)	+3	**+4**	+3	−2	−3	−1	+4
Late, ρ=0.67	+2	+2	**+4**	−2	−1	−1	+4
Late, ρ=0.75	+3	+1	+3	**+0**	−2	−1	+4
Math-only, ρ=0.50	**+10**	+3	+0	−5	−4	−1	+3
Math-only, ρ=0.67 (math-only Pareto)	+5	+4	+3	−2	**−1**	**+0**	+9

Table 17: Cross-model config transfer on the 4-benchmark pool. Cells show Δ vs. SC@32 out of 600 problems per model; Agg. sums across all 3,600 model-problem pairs. Bold marks the self-optimal config from Table 16.

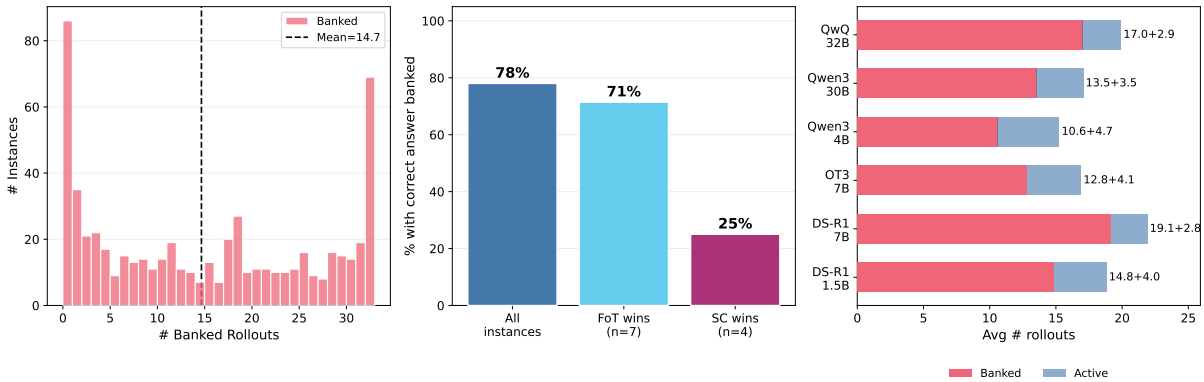

Figure 8: Early voting diagnostic on the AIME/AMC subset. Left: banked rollout count per problem. Center: fraction with a correct answer banked. Right: average banked and active rollouts per model.

Kernel	Δ	Tok↓ (%)	FLOP↓ (%)
`top-1` (perhaps)	−3	29.12	50.59
`top-2`	+6	29.20	50.66
`top-3` (3 markers)	+5	29.22	50.70
`top-5`	+5	29.19	50.65
`top-7`	+6	29.17	50.59
`top-10`	+0	29.17	50.61
`top-15`	+3	29.24	50.76
`top-21` (default)	+4	29.23	50.73

Table 18: Kernel-size sweep on the 4-benchmark pool. Tokens are added by descending $|r_{\text{pb}}|$; Δ is FoT@32 minus SC@32.

Offline rollouts are collected across multiple GPU configurations depending on model size. For the online deployment experiment in Section 4.4, all three conditions (Pass@1, SC@32, FoT@32) are run on a single isolated A100 GPU with a fixed random seed of 2000, ensuring that the same rollout pool is generated across conditions for a controlled comparison.

Checkpoint synchronization overhead. FoT requires pausing all active rollouts at each token-count checkpoint to evaluate the pruning criterion. In batch-mode serving frameworks such as SGLang, this synchronization is natively supported: all rollouts in a batch share the same generation schedule, so pausing at a fixed token count introduces no idle time. In continuous-batching frameworks (e.g., vLLM), rollouts that reach a checkpoint before others may incur brief idle time while the batch synchronizes, a minor scheduling cost. Our online wall-time measurements in Section 4.4 confirm that this overhead is negligible in practice: the pruning decision itself is a string count operation requiring no model inference.

KV-cache footprint. In the online deployment run, FoT reduces the cumulative generated-token footprint from 12.88M under SC@32 to 8.51M under FoT@32, a 33.9% reduction. Since each generated token occupies one KV-cache position per layer, this is the memory-side counterpart of the wall-time and generation-FLOP savings reported in Table 5.

D.2 Efficient-Attention Scaling

Our headline FLOP accounting (Section 3.1) assumes standard full attention, where per-rollout cost scales as s^2 and late tokens dominate, which is precisely the cost FoT removes by terminating spiraling rollouts early. Efficient attention changes this accounting: under sliding-window attention with window w, per-token cost saturates once the position exceeds w, so per-rollout cost grows linearly ($\approx w \cdot s$) rather than quadratically, and sparse

Attention regime	FLOP saving
Full quadratic (headline)	48.7%
Sliding window, w=8K	38.9%
Sliding window, w=4K	31.1%
Sliding window, w=2K	26.9%
Sliding window, w=1K	24.9%
Linear / sparse top-k	23.0%

Table 19: FoT attention-FLOP saving relative to SC@32 under different attention regimes, on the 4-benchmark pool with the universal configuration. The saving interpolates between the full-attention headline and the token-reduction floor as the effective attention window narrows.

or linear variants behave similarly. FoT's saving then tracks its token reduction rather than the larger quadratic figure. Table 19 applies FoT's termination pattern under each regime on the full 4-benchmark pool. The full-attention saving (48.7%, pooled over all rollouts as in Table 8) degrades smoothly toward the token-reduction floor (23.0%) as the window narrows; realistic sliding-window models ($w \approx$ 4–8K) retain roughly 64–80% of it. The accuracy result is unaffected, since it depends only on which rollouts survive pruning, and the KV-cache release at each checkpoint is likewise independent of the attention regime.

D.3 Adaptive Consistency Baseline

We simulate Adaptive Consistency (AC; Aggarwal et al., 2023) on the same pre-generated rollout pool used by all other methods.[2] AC is a sequential early-stopping method: rollouts are consumed one at a time, and at each step a Bayesian stopping criterion determines whether the current majority answer is sufficiently confident. We use the Beta stopping criterion with confidence threshold τ=0.95 (the paper's default), which stops when posterior confidence in the current majority answer exceeds τ.

Because AC assumes sequential generation while our rollouts are generated in parallel, we simulate sequentiality by processing rollouts in index order. To account for ordering effects, we average results over 10 random permutations of the rollout sequence per problem. AC's compute savings are measured as the fraction of rollouts *not consumed* before stopping; FLOP savings are computed identically to other methods (sum of s_i^2 over consumed rollouts only).

[2] https://github.com/Pranjal2041/AdaptiveConsistency

D.4 Slim-SC Baseline Configuration

Our Slim-SC baseline is a faithful reproduction of Hong et al. (2025), using the configuration from their official repository.[3] We use the diversity pruning strategy with cosine similarity threshold τ=0.9, segment-level embeddings via `sentence-transformers/all-mpnet-base-v2`, and a warm-up of 20 thoughts before pruning is enabled. These are the authors' recommended defaults; no threshold tuning was performed on our data.

[3] https://github.com/hyscale-lab/slimsc