VASP: Reasoning Control via Verified Adaptive Stopping Policies

Anonymous ACL submission

Abstract

Test-time scaling improves LLM reasoning but often requires expensive generation budgets or multiple full reasoning traces. We introduce Verified Adaptive Stopping Policy (VASP), a training-free decoding policy that uses sub-argmax probability spikes on control tokens, such as End-of-Thinking, as candidate completion signals. Rather than stopping at every spike, VASP invokes an on-demand self-critique and terminates only after accumulating verified evidence of completion. Across six reasoning-oriented models, VASP substantially improves the accuracy–compute trade-off: on MATH-500, it reduces reasoning tokens by up to 55% and preserves or improves accuracy in 5 of 6 model settings; on AIME 2025, it saves up to 45% tokens while matching accuracy for the strongest Qwen3 models. Our results show that control-token probabilities expose a lightweight interface for adaptive inference-time control without additional training.

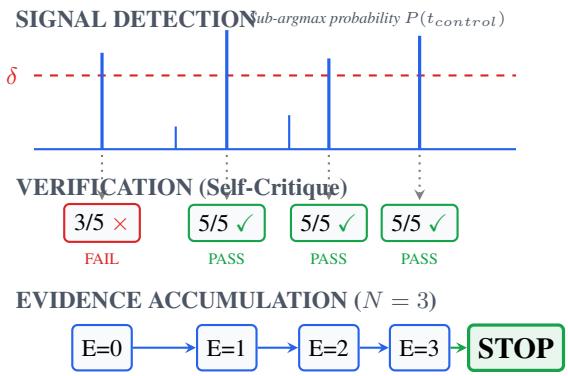


Figure 1: **The VASP Control Loop.** (Top) The model emits control signal spikes at candidate completion points. (Middle) Each spike triggers a semantic verification step. Low scores (< 5) are rejected as hesitation noise; high scores (5) are accepted as valid evidence. (Bottom) The stopping time policy accumulates verified evidence E and terminates generation only when the budget $N = 3$ is met. This stopping policy requires no training. In fact, this minimal cost pipeline can be implemented at the sampler level for an LLM.

1 Introduction

Reasoning performance in Large Language Models (LLMs) is deeply intertwined with the length of the generated rationale. While Chain-of-Thought (CoT) prompting enables models to decompose complex problems, extending these chains introduces a trade-off: longer reasoning paths incur higher latency and are susceptible to error accumulation, a phenomenon known as "overthinking" (Sui et al., 2025). Recent work has formalized this relationship, demonstrating that reasoning performance follows a non-monotonic, inverted U-shaped curve with respect to generation length (Wu et al., 2025). This implies the existence of a dynamic, instance-specific *optimal stopping point*.

Current inference-time scaling strategies for reasoning largely fall into two paradigms: single-trajectory and parallel generation. Parallel strategies, such as Majority Voting (Wang et al., 2022) or Recursive Self-Aggregation (Venkatraman et al., 2025), improve robustness by generating and aggregating multiple reasoning paths. While effective, these methods incur a linear increase in computational cost with the number of samples. Conversely, single-trajectory strategies seek to optimize a linear generation. Recent work such as S1 (Muennighoff et al., 2025) introduced "budget forcing," a technique that behaviorally prolongs reasoning by injecting special wait tokens. However, both paradigms are computationally expensive. Parallel generation relies on greater number of trajectories to improve reasoning whereas S1 style papers rely on increasing the token count within the COT trace. This creates a fundamental tension between computational cost and performance. Consequently, a central question in reasoning efficiency is the following:

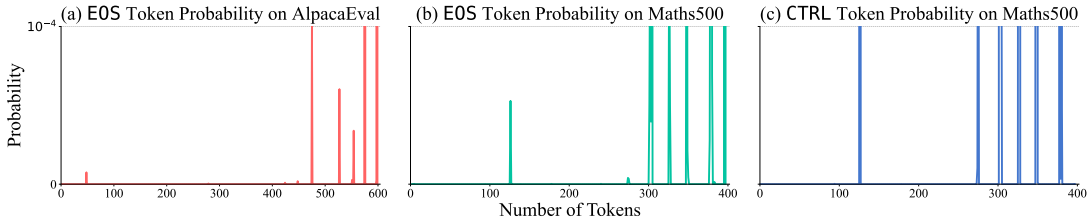


Figure 2: Token-wise probability traces of control tokens during generation. (a) $P(t_{EOS})$ on AlpacaEval exhibits a phase transition from sparse, low-probability spikes to a dense, high-probability cluster, signaling the model's convergence toward termination. (b) Post-thinking $P(t_{EOS})$ and (c) In-thinking $P(t_{EOT})$ on MATH-500 reveal sharp, transient spikes that correspond to semantic boundaries and candidate completion points. The non-monotonic, state-dependent nature of these signals confirms that internal model confidence fluctuates significantly before final termination, motivating the need for adaptive control policies for stopping time over static ones.

Can we achieve high reasoning accuracy while controlling the token count?

In this work, we introduce a paradigm shift by identifying a previously unexploited signal for generation control: the **sub-argmax probability dynamics** of control tokens. While standard decoding discards non-argmax probabilities as noise, we demonstrate that they contain rich, structured information. As illustrated in Figure 2, instruction-tuned models emit sharp probability spikes for control tokens (e.g., `EOT`) well before they become the dominant prediction. These spikes serve as emergent, high-fidelity indicators of the model's internal readiness to conclude reasoning.

To harness this novel signal, we introduce the Critique Verified Adaptive Stopping Policy, or **VASP** for short. This framework casts inference control not as a behavioral forcing problem, but as a principled optimal stopping problem. This represents a fundamental shift from the paradigm of Muennighoff et al. (2025) to a fine-grained probabilistic control model. VASP is notably lightweight and **training-free**, requiring only access to standard token probabilities. Within this framework, the policy π_{VASP} continuously monitors the control signal entropy. When a candidate completion spike is detected, VASP triggers an on-demand semantic verification step, using the model itself to critique the reasoning trace. As illustrated in Figure 1, this effectively gates the termination decision, ensuring that generation stops only when the model is both statistically confident (high spike probability) and semantically consistent (verified critique).

Our comprehensive experiments demonstrate that VASP significantly outperforms standard baselines. On complex reasoning tasks like MATH-500 and AIME 2025, it shifts the pareto frontier for reasoning (see Fig. 4), matching or exceeding the accuracy of Majority Voting while reducing the average token budget by up to **50%**. By replacing static termination heuristics with a signal-driven verification loop, VASP mitigates both over-thinking and under-thinking, establishing a new state-of-the-art for efficient, resource-aware reasoning.

The primary contributions of this work are:

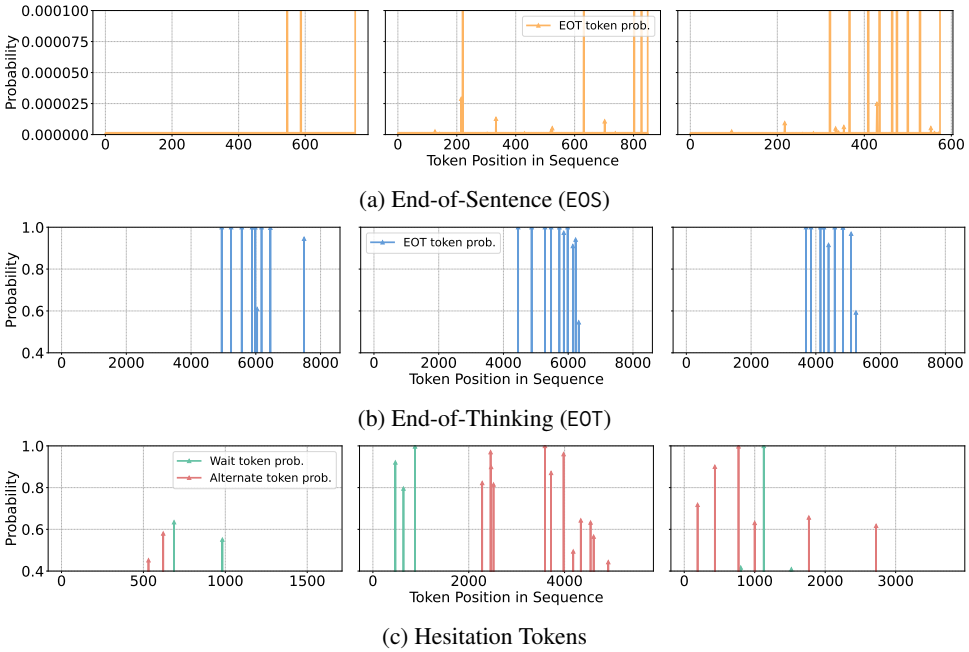
1. Signal Characterization. We characterize the sub-argmax probability dynamics of control tokens (e.g., `EOT`) in instruction-tuned LLMs. We demonstrate empirically that these probability spikes act as emergent indicators of the model's internal readiness to terminate reasoning steps.

2. The VASP Framework. We introduce the **Verified Adaptive Stopping Policy (VASP)**, a control framework that utilizes these signal spikes as triggers for conditional verification. VASP gates the stopping decision behind a model-driven self-critique, dynamically distinguishing between valid termination and hesitation.

3. Empirical Validation. We evaluate VASP across **six diverse model architectures** (ranging from 4B to 32B parameters) on two challenging reasoning benchmarks (MATH-500 and AIME 2025). Our results show consistent computational gains: VASP achieves parity with standard Majority Voting accuracy while reducing the average token consumption by 30–50% on MATH-500 and up to 45% on AIME 2025.

2 Related Work

The challenge of enhancing LLM reasoning efficiency is an active area of research. A primary focus has been on mitigating the "overthinking phenomenon" (Sui et al., 2025), where models generate excessively long Chain-of-Thought (CoT)



(a) End-of-Sentence (EOS)

(b) End-of-Thinking (EOT)

(c) Hesitation Tokens

Figure 3: **Characterizing Control Signals.** We identify three distinct sub-argmax probability dynamics: (a) EOS spikes indicating sentence boundaries, (b) EOT spikes indicating potential reasoning completion, and (c) Hesitation spikes signaling uncertainty. VASP primarily leverages the EOT signal to regulate stopping.

traces. Foundational work has established that reasoning performance follows a non-monotonic, inverted U-shaped curve with CoT length, motivating the search for an optimal, task- and model-dependent reasoning budget (Wu et al., 2025). Current approaches largely fall into two categories: **model-centric** methods that seek to build innately more efficient reasoners through training-time interventions like fine-tuning on optimal-length data (Yang et al., 2025b), and **inference-centric** methods that aim to dynamically control existing models at test-time. Our work, VASP, belongs to the latter category.

Inference-time control strategies have evolved from behavioral interventions to signal-based termination. An important work in this area is the S1 paper (Muennighoff et al., 2025) which introduced budget forcing using special tokens like "Wait" to behaviorally prolong or terminate reasoning. More recent methods have focused on early stopping in sampling-based decoding by leveraging auxiliary signals, such as scores from external reward models (Sun et al., 2024), the consistency of latent embeddings (Wang et al., 2025), or learned confidence scores (Huang et al., 2025). VASP advances this paradigm by identifying and utilizing a more fundamental, natively available signal: the sub-argmax probability of control tokens ($t_{control}$). By framing the control task as a principled optimal stopping problem and designing policies grounded in established theoretical concepts, our work provides a training-free, model-agnostic framework for more precise and efficient test-time control. A more comprehensive discussion of related literature is available in Appendix A.

3 Characterizing Control-Token Dynamics

Before deriving a control policy, we must first establish the statistical reliability of the control signal. We posit that in instruction-tuned models, the probability mass assigned to control tokens (e.g., EOT) is a continuous proxy for the model's internal confidence in semantic completion.

3.1 The Emergence of Sub-Argmax Spikes

Standard decoding treats the EOT token as a stop command only when it achieves the argmax rank. However, analyzing the probability trajectory $P(\texttt{EOT} \mid \mathbf{x}_{<t})$ reveals a richer dynamic structure.

As shown in Figure 2, the signal exhibits distinct "spikes" at semantic boundaries, i.e. end of sentences, completion of sub-steps, or tentative conclusions. These spikes often reach magnitudes of 10^{-2} to 10^{-1} without crossing the decoding threshold. We hypothesize that these sub-argmax spikes on control tokens represent *candidate stopping points* which may not be perfectly calibrated

during training.

Perplexity Displacement and Calibration. We observe that control token spikes mark local minima in semantic entropy. Naively suppressing these spikes to force longer reasoning displaces the model into a higher entropy regime, often leading to repetitive or non-convergent generation. This is a mechanistic signature of *overthinking*. Conversely, accepting every spike at face value risks *underthinking*, as models may emit an EOT token despite logical incompleteness due to miscalibration. VASP resolves this trade-off by using the spike solely as a trigger for verification. This acts as a dynamic calibration filter, distinguishing genuine semantic completion from hesitation without forcing the model into unproductive generation cycles.

4 The VASP Framework

We formulate the problem of inference-time control as an optimal stopping problem over a stochastic sequence. Let a reasoning trace be denoted by a sequence of tokens $\mathbf{x} = (x_1, x_2, \dots)$. At each decoding step t, the policy model $\mathcal{M}$ outputs a probability distribution over the vocabulary $\mathcal{V}$, denoted as $P_{\mathcal{M}}(\cdot \mid \mathbf{x}_{<t})$ or $P(\cdot \mid \mathbf{x}_{<t})$ for short.

Standard decoding terminates deterministically when the `EOS` (End-of-Sequence) or `EOT` (End-of-Thinking) token is the argmax prediction. However, this relies on a binary decision boundary that ignores the model's internal uncertainty. We propose an **Adaptive Stopping Policy (ASP)** that monitors the continuous probability landscape of control tokens to dynamically regulate the stopping time τ.

4.1 Signal Detection: The Sub-Argmax Landscape

We identify a subset of tokens $\mathcal{C} \subset \mathcal{V}$ as *control tokens* (e.g., `EOS`, `EOT`, `Wait`). These tokens serve as markers for semantic boundaries. We define the raw control signal $s_t \in [0, 1]$ at time t as the aggregated probability mass of the control set:

$$s_t = \sum_{v \in \mathcal{C}} P(v \mid \mathbf{x}_{<t}) \tag{1}$$

Empirically, s_t exhibits sparsity; it remains near zero during active reasoning and spikes at candidate completion points. We formalize the detection of these candidate points via an indicator function I_t, which activates when the signal exceeds a sensitivity threshold δ:

$$I_t = \mathbb{I}(s_t > \delta) \tag{2}$$

We refer to a time step where $I_t = 1$ as a *spike*.

4.2 Evidence Accumulation and Verification

Relying on a single spike for termination is brittle due to signal noise. For instance, the control tokens may not be well calibrated. A robust policy must accumulate evidence over time. We generalize this accumulation by introducing a gating mechanism that validates each signal.

We define the **Accumulated Evidence** E_t as the sum of verified spikes up to time t:

$$E_t = \sum_{j=1}^{t} I_j \cdot \Phi(\mathbf{x}_{<j}) \tag{3}$$

Here, $\Phi : \mathcal{V}^* \to \{0, 1\}$ is a **Verification Function**. It acts as a filter: if $\Phi(\mathbf{x}_{<j}) = 1$, the spike is accepted as valid evidence; otherwise, it is rejected as noise.

The Adaptive Stopping Policy π is defined by a budget parameter $N \in \mathbb{Z}_{\geq 1}$ (patience). The generation process terminates at the optimal stopping time τ^*:

$$\tau^* = \min\{t : E_t \geq N\} \tag{4}$$

4.3 Policy Instantiations

This formulation unifies two distinct control regimes based on the choice of the verification function Φ.

Deterministic ASP ($\Phi \equiv 1$). In the low-compute regime, we assume that the signal is inherently trustworthy. We set $\Phi(\mathbf{x}) = 1$ for all $\mathbf{x}$. The evidence function simplifies to a raw count of spikes: $E_t = \sum_{j=1}^{t} I_j$. This policy, which we term **Deterministic ASP (DASP)**, relies solely on signal persistence (patience N) to filter noise. It incurs zero computational overhead but lacks semantic awareness.

Verified ASP (Φ_{critique}). In the high-reliability regime, we recognize that a probability spike does not guarantee semantic correctness. We introduce **Verified ASP (VASP)**, where Φ is parameterized by the model $\mathcal{M}$ itself acting as a critic. When a spike is detected ($I_t = 1$), the policy (acting via the sampler) pauses auto-regressive generation and

inserts a self-critique prompt to evaluate the current trace. Denoting the critique prompt as prompt_c for short:

$$\Phi_{\text{critique}}(\mathbf{x}_{<t}) = \begin{cases} 1 & \text{if } \mathcal{M}(\cdot \mid \mathbf{x}< t, \text{prompt}_c) = \text{“Pass”} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Under VASP, the stopping decision is explicitly gated by the model's semantic self-assessment. While this introduces a small computational cost per spike, it significantly increases the stopping precision, mitigating the risk of underthinking on complex tasks.

5 VASP: Algorithm and Implementation

We provide the concrete implementation of the VASP framework. Algorithm 1 details the unified control loop. The core innovation lies in the `TriggerVerification` step (Lines 6–10), which dynamically switches between deterministic counting and semantic self-reflection based on the chosen verifier Φ.

5.1 Implementation Details

Critique Injection Strategy. Implementing the Verified ASP requires a targeted interruption of the autoregressive loop. When a spike is detected ($I_t = 1$), the sampler intervenes to inject a specific critique template into the context: *“Review the reasoning above on a scale of 1-5. Is it complete and correct?”*

The model is forced to sample a response from the token set $\{1, 2, 3, 4, 5\}$. If the response is **5** ($\Phi = 1$), the spike is validated as a high-confidence stopping point. We increment the evidence counter E. If the response is $<$ **5** ($\Phi = 0$), the spike is treated as hesitation. The generation continues. This mechanism effectively converts the continuous, noisy control signal into a discrete, semantically verified decision process.

Budgeting. The patience parameter N acts as a control knob for the efficiency-accuracy trade-off. Lower values ($N = 1$) prioritize speed, behaving like aggressive early stopping. Higher values ($N \geq 3$) enforce rigorous multi-step verification, essential for complex reasoning tasks.

6 Experimental Setup

Datasets and Tasks We evaluate our method on a diverse suite of benchmarks targeting two key capabilities: reasoning and instruction following. For **reasoning**, we use three benchmarks spanning mathematical and logical problem-solving. First, for arithmetic reasoning, we use **GSM-8K** (Cobbe et al., 2021), a collection of 1,320 grade-school math problems requiring multiple steps of basic arithmetic. Second, we assess performance on more complex mathematical challenges using a 500-problem subset of the **MATH** benchmark (Hendrycks et al., 2021). Third, to test advanced problem-solving, we include 30 competition-level questions from the **AIME 2025**. For **instruction following**, we use **AlpacaEval** (Dubois et al., 2024), an automatic, LLM-based evaluation benchmark consisting of open-ended user queries from real-world scenarios.

Models We conducted experiments across different model families and scales. For **reasoning tasks**, we employ models from two distinct families.

Algorithm 1: Verified Adaptive Stopping Policy

1: **Input:** Model $\mathcal{M}$, Prompt x, Budget N, Threshold δ, Verifier Φ
2: **Output:** Completed reasoning trace $\mathbf{x}$
3:
4: Initialize evidence $E \leftarrow 0$, context $\mathbf{x} \leftarrow [x]$
5: **for each** decoding step $t = 1, 2, \ldots, T_{\max}$ **do**
6: // **Phase 1: Signal Detection**
7: $P_t \leftarrow \mathcal{M}(\mathbf{x})$ ▷ Forward pass
8: $s_t \leftarrow \sum_{v \in \mathcal{C}} P_t(v)$ ▷ Aggregate control signal
9:
10: // **Phase 2: Verification**
11: **if** $s_t > \delta$ **then**
12: score $\leftarrow \Phi(\mathbf{x})$ ▷ Invoke verifier
13: **if** score $= 5$ **then**
14: $E \leftarrow E + 1$ ▷ Accumulate evidence
15: **end if**
16: **end if**
17:
18: // **Phase 3: Termination Check**
19: **if** $E \geq N$ **then**
20: **return** $\mathbf{x}$ ▷ Verified evidence threshold reached, stop
21: **end if**
22:
23: // **Phase 4: Continuation**
24: Sample $x_{t+1} \sim P_t$ ▷ Mask EOT if $E < N$
25: $\mathbf{x} \leftarrow [\mathbf{x}, x_{t+1}]$
26: **end for**

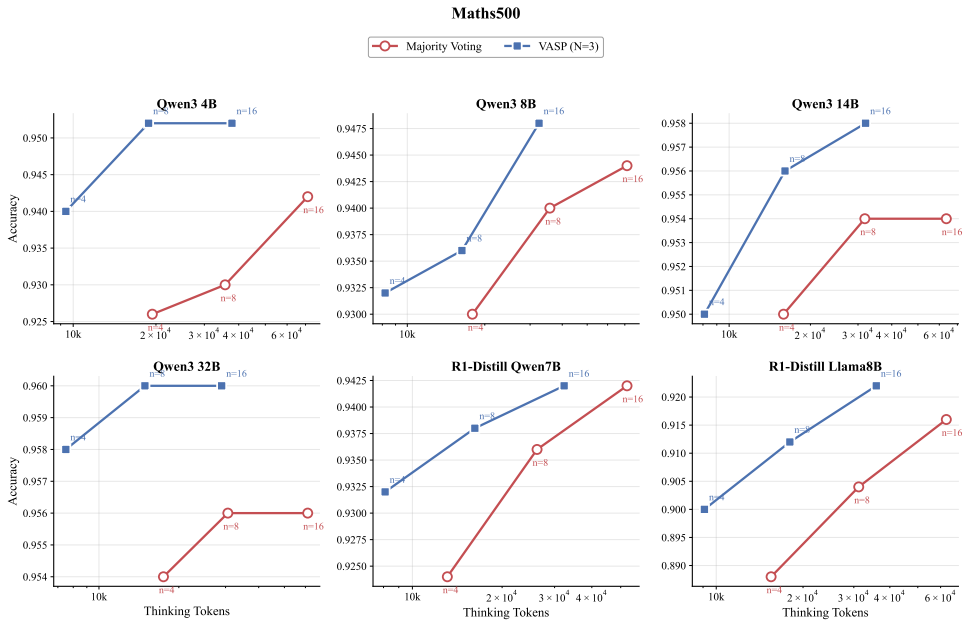


Figure 4: **Test-Time Scaling Efficiency.** Comparison of Majority Voting (Red) vs. VASP ($N = 3$, Blue). The x-axis denotes average thinking tokens (log scale). On MATH-500, VASP achieves parity on accuracy with $\sim$50% fewer tokens.

From the **Qwen3** series, recognized for state-of-the-art performance on public leaderboards (Yang et al., 2025a), we select `Qwen3-4B`, `Qwen3-8B`, and `Qwen3-14B`. Additionally, we use the `R1-Distill Qwen7B` and `R1-Distill Llama8B` models to broaden our evaluation.

Evaluation Metrics Following standard practices for each task, we employ strict and established metrics. For the reasoning benchmarks (GSM-8K, MATH, and AIME), we measure performance using **Accuracy**, i.e. a model's prediction is correct only if it exactly matches the ground-truth solution. For the instruction-following benchmark (AlpacaEval), we report the **win rate** and **length-controlled win rate** against a strong reference model (e.g., GPT-4), as determined by an automated GPT-4-based evaluator.

Compute Accounting. We report average generated reasoning tokens before the final answer. For VASP, we also track verification calls, since each triggered self-critique adds an extra model invocation. This distinguishes token savings from verification overhead.

7 Experimental Results

7.1 Sub-Argmax Signal Characterization

We identify three distinct classes of sub-argmax probability dynamics that serve as actionable control signals. First, **End-of-Sentence (EOS)** spikes (Figure 3a) appear at semantic completion points, providing a soft signal for early termination in instruction following. Second, **End-of-Thinking (EOT)** spikes (Figure 3b) indicate the model's internal readiness to conclude a reasoning chain; identifying these is crucial for mitigating underthinking. Finally, **Hesitation** spikes (Figure 3c), such as `Wait` or `Alternate`, map to moments of high uncertainty, signaling active self-correction during complex problem solving.

Key Takeaway: Control token probabilities are structured, state-dependent signals that accurately reveal the model's internal status regarding termination, continuation, and uncertainty.

7.2 Accuracy vs Compute Trade-off

Figure 4 compares the accuracy-compute curves of VASP ($N = 3$) against standard Majority Voting. We observe consistent **Pareto dominance**: the VASP curve (Blue) lies strictly above and to the left of the baseline (Red) across all architectures.

On MATH-500 (Top), VASP achieves accuracy parity using roughly half the token budget; for instance, Qwen3-14B matches the baseline ($n = 16$) performance using only $\sim 1.6 \times 10^4$ tokens versus 6×10^4. On AIME 2025 (Bottom), VASP prevents the cost explosion typical of hard reasoning tasks. For Qwen3-32B, it matches the peak baseline ac-

Table 1: **Efficiency of Verified Adaptive Stopping (VASP) vs. Standard Majority Voting.** We compare the accuracy and average token cost for standard voting (Base) and our verified policy (VASP with $N = 3$) at a sample size of $n = 8$. **Delta Acc.** denotes the absolute accuracy gain, and **Saved** denotes the percentage reduction in reasoning tokens.

Model	Base Acc.	VASP Acc.	Base Tokens	VASP Tokens	Delta Acc.	Tokens Saved
MATH-500 ($n = 8$)						
Qwen3-4B	0.930	**0.952**	35.7k	**18.8k**	+2.2%	**47.3%**
Qwen3-8B	0.940	0.936	35.8k	**16.3k**	-0.4%	**54.5%**
Qwen3-14B	0.954	**0.956**	31.8k	**16.1k**	+0.2%	**49.4%**
Qwen3-32B	0.956	**0.960**	30.6k	**14.9k**	+0.4%	**51.3%**
R1-Distill Qwen7B	0.936	**0.938**	26.2k	**16.2k**	+0.2%	**38.2%**
R1-Distill Llama8B	0.904	**0.912**	31.2k	**18.0k**	+0.8%	**42.3%**
AIME 2025 ($n = 8$)						
Qwen3-4B	0.734	**0.767**	137.0k	**123.0k**	+3.3%	**10.2%**
Qwen3-8B	0.767	0.667	142.0k	**95.0k**	-10.0%	**33.1%**
Qwen3-14B	0.800	0.800	127.0k	**99.8k**	+0.0%	**21.4%**
Qwen3-32B	0.800	0.800	124.0k	**68.2k**	+0.0%	**45.0%**
R1-Distill Qwen7B	0.600	0.567	114.0k	**96.4k**	-3.3%	**15.4%**
R1-Distill Llama8B	0.467	**0.500**	129.0k	**111.0k**	+3.3%	**14.0%**

curacy (83.4%) while reducing compute by **45%**. Furthermore, VASP effectively mitigates the "overthinking" degradation observed in baselines (e.g., Qwen3-8B on AIME) by pruning low-confidence trajectories before they drift into hallucination.

Key Takeaway: VASP achieves state-of-the-art reasoning performance at approximately **50%** of the standard computational cost.

7.3 Efficiency Summary: Aggregate Performance

Table 1 quantifies the efficiency gains of VASP ($N = 3$) versus Majority Voting ($n = 8$). On MATH-500, VASP achieves universal token reduction (38–54%) while maintaining or improving accuracy in 5 of 6 models. On AIME 2025, the method prevents cost explosion for large models; Qwen3-32B matches the baseline accuracy of 80.0% while saving **45.0%** of the compute budget (∼56k tokens per problem). This confirms VASP as an effective resource governor that prunes unnecessary deliberation without sacrificing reasoning depth.

7.4 Ablation: The Necessity for Verification

Is semantic verification necessary, or is simply forcing the model to "think longer" sufficient? In Table 2, we compare the Deterministic ASP (DASP), which unconditionally forces continuation N times, against the Verified ASP (VASP), which gates continuation via self-critique.

We observe a clear failure mode in the deterministic policy. While setting $N = 3$ improves accuracy over the baseline, pushing to $N = 5$ leads to performance degradation (e.g., S1-32B accuracy drops from 0.567 to 0.537 on AIME). This confirms that blind prolongation induces overthinking. In contrast, VASP achieves the optimal accuracy (0.567) with significantly lower token usage (12.3k vs 14.5k for DASP $N = 3$), showing that the verifier effectively prunes low-value reasoning steps that a deterministic counter would miss.

Table 2: **Ablation of the Verification Function (Φ).** Comparison of Deterministic ASP (DASP), which blindly suppresses the EOT token N times, versus Verified ASP (VASP). DASP suffers from diminishing returns and regression at high N (overthinking), whereas VASP achieves peak accuracy with fewer tokens.

Dataset	Model	Policy Variant	Accuracy ↑	Avg. Tokens ↓
MATH-500	S1-32B	Baseline	0.732	5752
		DASP ($N = 1$)	0.796	6317
		DASP ($N = 3$)	0.808	10659
		DASP ($N = 5$)	0.792	17906
		VASP ($N = 1$)	**0.822**	**8885**
AIME 2025	S1-32B	Baseline	0.400	9552
		DASP ($N = 1$)	0.466	10653
		DASP ($N = 3$)	0.567	14503
		DASP ($N = 5$)	0.537	17786
		VASP ($N = 1$)	**0.567**	**12345**
	Qwen3-8B	Baseline	0.700	18154
		DASP ($N = 3$)	0.700	19007
		DASP ($N = 5$)	0.700	20292
		DASP ($N = 7$)	0.737	20398
		VASP ($N = 1$)	**0.767**	**19269**

Additionally we show ablations with respect to the choice of signal accumulation strategies in Appendix D.

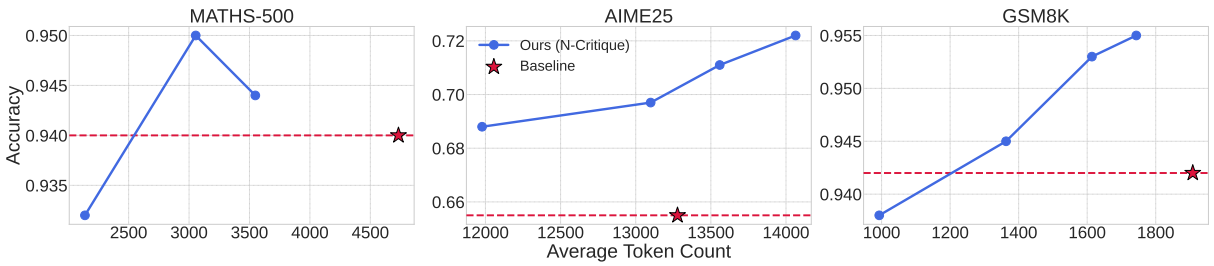


Figure 5: **Effect of Critique Budget (N).** Performance of VASP on Qwen3-14B as N increases. (Left) On MATH-500, $N = 3$ achieves optimal efficiency and pareto. (Center) On AIME 2025, increasing N monotonically improves accuracy. (Right) On GSM8K, VASP achieves pareto than the greedy-baseline at all tested budgets

7.5 Ablation: The Critique Budget (N)

The patience parameter N functions as a tunable control knob for verification rigor. In Figure 5, we analyze the performance of VASP on Qwen3-14B across three datasets (MATH-500, AIME 2025, GSM8K) as we vary $N \in \{1, 3, 5, 7\}$. The red dashed line represents the static baseline performance.

- **Efficiency Mode ($N \leq 3$):** On standard tasks like MATH-500 and GSM8K, a low budget of $N = 3$ is sufficient to surpass baseline accuracy while reducing token consumption by $\sim$35%.

- **Performance Mode ($N \geq 5$):** On the challenging AIME 2025 benchmark, higher verification rigor is necessary. Increasing N to 7 yields a significant accuracy gain (0.72 vs 0.65 baseline), confirming that N acts as an effective control knob for adapting the policy to task difficulty.

7.6 Aggregate Performance Consistency

We conduct a head-to-head comparison of VASP ($N = 3$) against Majority Voting across all 36 experimental configurations (6 models $\times$ 3 sample sizes $\times$ 2 datasets). Figure 6 summarizes the results.

In terms of **computational efficiency** (Accuracy/Token), VASP achieves a **100% win rate**, outperforming the baseline in every single configuration across both MATH-500 and AIME 2025. Regarding **accuracy**, VASP is dominant on MATH-500 (15 wins, 2 ties) and maintains statistical parity on the harder AIME benchmark (5 wins, 7 ties). This confirms VASP as a strictly dominant strategy for efficiency that preserves reasoning performance across diverse tasks.

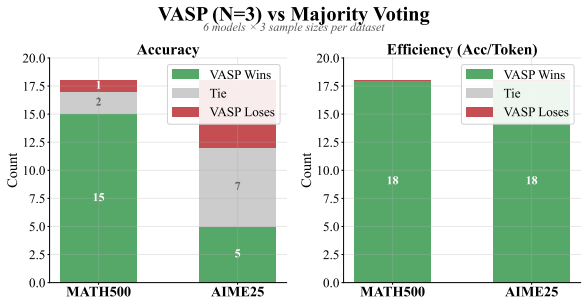


Figure 6: **Head-to-Head Summary.** Comparison of VASP ($N = 3$) vs. Baseline across 36 settings. VASP achieves universal efficiency gains (Right) while maintaining competitive or superior accuracy (Left).

Supplementary Analysis. We provide extended ablations on signal accumulation strategies (Appendix D) and instruction-following performance on AlpacaEval (Appendix E). Additionally, Appendix F contains comprehensive performance tables for our experimental configurations.

8 Discussion.

Our results highlight a new direction for LLM inference. We demonstrate that the entropy of control tokens, which are typically discarded during greedy decoding, can be used as a signal for adaptive control. By combining these stochastic signals with self-verification, VASP creates inference-time strategies that are accurate as well as computationally efficient. Since our stopping time policies can be invoked at inference time at the sampler, they are extremely low cost, and easy to implement. This establishes VASP as a robust, low-cost method for test-time scaling that bridges the gap between static decoding and expensive search-based reasoning.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna

Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

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

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. 2025. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

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. *arXiv preprint arXiv:2501.19393*.

Daniele Paliotta, Junxiong Wang, Matteo Pagliardini, Kevin Y Li, Aviv Bick, J Zico Kolter, Albert Gu, François Fleuret, and Tri Dao. 2025. Thinking slow, fast: Scaling inference compute with distilled reasoners. *arXiv preprint arXiv:2502.20339*.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483. IEEE.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. Fast best-of-n decoding via speculative rejection. *Advances in Neural Information Processing Systems*, 37:32630–32652.

Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, and 1 others. 2025. Recursive self-aggregation unlocks deep thinking in large language models. *arXiv preprint arXiv:2509.26626*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. 2025. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *arXiv preprint arXiv:2503.01422*.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

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

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2025b. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080*.

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

SUPPLEMENTARY MATERIAL

These supplementary materials provide additional details, derivations, and proofs for our paper. The appendix is organized as follows:

- **Section A: Extended Related Work.** A detailed overview of related literature.
- **Section B: Proof of Self-Critique Theory** Demonstrates the benefit of the adaptive self-critique policy.
- **Section C Limitations of this work**
- **Section D Ablations to the choice of signal accumulation strategy**
- **Section E: Model Performance on AlpacaEval, AIME, and Maths-500 Under VASP Policies**
- **Section F: Model Performance for 6 models for n generations in AIME2025 and Math500**

A Extended Related Work

A.1 Test-Time Scaling and Dynamic Inference

A significant line of research has demonstrated that the performance of Large Language Models can be substantially improved by allocating more computational resources at inference time, a paradigm known as test-time scaling (Muennighoff et al., 2025). This approach, however, introduces a critical challenge of efficiency. Our work, VASP, contributes to the growing body of literature on making this scaling more intelligent and resource-efficient.

Foundational Test-Time Scaling Methods. The canonical methods for test-time scaling involve generating multiple candidate sequences and aggregating them. **Best-of-N (BoN)** sampling generates N independent sequences and uses a verifier or a trained reward model to select the highest-scoring output (Lightman et al., 2023). This approach is general-purpose but often relies on the availability of a high-quality, and potentially costly, external verifier. A popular variant, **Self-Consistency**, is designed for tasks with deterministic answers, such as mathematical reasoning (Wang et al., 2022). It generates N sequences and selects the final answer via a majority vote, eliminating the need for an external reward model but limiting its applicability. Both BoN and Self-Consistency are computationally expensive as they require the full generation of all N candidate sequences, creating a linear increase in cost with the number of samples.

Efficient Test-Time Scaling via Early Termination. Recent work has focused on mitigating the high cost of BoN and Self-Consistency by introducing mechanisms for the early termination of unpromising generation paths. These methods differ primarily in the type of signal they use to make termination decisions.

One prominent approach leverages **external verifiers** to score partial sequences. For instance, Speculative Rejection (Sun et al., 2024) periodically queries an external reward model on the partially generated sequences. Trajectories with low partial scores, which are unlikely to yield a high final reward, are pruned, allowing computational resources to be focused on the more promising candidates. While effective, this strategy's performance is contingent on the quality and calibration of the external reward model, which itself can be costly to train and serve.

A second approach utilizes signals from the model's own **latent representations**. Self-Truncation Best-of-N (ST-BON) (Wang et al., 2025) operates on the hypothesis that sequences leading to the same

correct answer will have similar latent embeddings. It monitors the consistency of hidden states across parallel generations and truncates paths that diverge from the main cluster, thereby avoiding the need for an external reward model. This uses a truly internal signal, but one that is high-dimensional and less directly interpretable than the explicit probabilistic outputs of the model.

A third approach uses more general **model confidence scores**. Self-Calibration (Huang et al., 2025) proposes fine-tuning a model to produce a calibrated confidence score for its own generations, often by distilling confidence from Self-Consistency statistics. This learned confidence can then be used to implement early-stopping rules for sampling. This method also uses an internal signal, but it requires a separate training phase to create the calibrated confidence predictor.

Our VASP framework contributes to this line of research by proposing the use of a novel signal that is both *internal* to the model and *natively available* without requiring additional training or complex analysis of latent states: the explicit probability of control tokens, $P(t_{control})$.

Orthogonal Approaches. Other methods seek efficiency through different means. **Structured Search** methods, such as Tree-of-Thought (Yao et al., 2023) and Graph-of-Thoughts (Besta et al., 2024), replace unstructured sampling with a more organized exploration of the reasoning space, often involving backtracking and planning. While powerful, these methods typically introduce significant algorithmic complexity and overhead. Concurrently, **architectural approaches** aim to build faster models from the ground up, for example by using subquadratic architectures like Mamba (Paliotta et al., 2025). These architectural innovations are largely complementary to our work; the principled stopping policies developed within the VASP framework could potentially be applied on top of these faster models to achieve even greater efficiency gains.

A.2 Controlling Reasoning Behavior at Test-Time

Beyond general-purpose sampling efficiency, a specific line of work has focused on directly controlling the behavior of the reasoning process itself at inference time. This is particularly relevant for our work on managing the thinking phase via the t_{EOT} token. A seminal contribution in this area is the **s1** model's "budget forcing" mechanism (Muennighoff et al., 2025). This approach introduced direct, behavioral interventions to control reasoning length: it could forcefully terminate a thinking process that exceeded a token budget, or, crucially, it could prolong a thinking process by appending a special "Wait" token when the model attempted to conclude prematurely. This demonstrated the viability of active, external control over the deliberation process.

Other approaches have used **prompt engineering** to influence reasoning style. For example, Chain-of-Draft (Xu et al., 2025) instructs models to produce concise, draft-like intermediate steps to reduce verbosity. Similarly, works like Renze and Guven (2024) have shown that simply instructing a model to "be concise" can effectively shorten reasoning paths. These methods, while often effective, rely on the model's instruction-following capabilities and may not offer the same level of fine-grained control as algorithmic interventions. VASP builds directly on the legacy of Muennighoff et al. (2025), advancing the paradigm from reactive, behavioral interventions to a predictive, signal-driven control policy. By grounding the decision of *when* to apply interventions like the "Wait" token in the model's underlying probability distribution, VASP offers a more fundamental and fine-grained control mechanism.

A.3 Theoretical Foundations for Policy Design

The design of each VASP stopping policy is a principled application of a concept from established theoretical domains. We ground our methods in optimal stopping theory, stochastic process analysis, and reinforcement learning, allowing us to derive policies from first principles rather than ad-hoc heuristics.

Policies Derived from Optimal Stopping Theory. This field addresses the problem of choosing an optimal time to take an action based on sequential observations. The classic **Secretary Problem**, with its renowned $1/e$ observe-then-commit solution, directly inspires our Adaptive Peak-Threshold Sampler. Similarly, the **Prophet Inequality** setting, which bounds the performance of online algorithms against a 'prophet' with full hindsight, motivates the retrospective logic of our Prophet Lookback policy.

Policies Derived from Stochastic Process Models. We model the control signal $\{s_t\}$ as a time series, allowing us to draw from relevant analytical tools. A **Martingale** is a process where the conditional expectation of the next value is the present value. Modeling the inter-spike interval process as a martingale yields the simple predictive rule in our Last-Interval Budget Sampler. **Change-Point Detection**, which aims to identify shifts in a process's statistical properties, provides the formal basis for our Phase-Shift Sampler, which is designed to detect a change in the rate of spike generation.

A Policy Derived from the Actor-Critic Paradigm. The Actor-Critic framework in reinforcement learning uses a *critic* to estimate the value of an *actor*'s policy. Our Adaptive Self-Critique policy introduces a novel, intra-model instantiation of this concept. The LLM's generative process acts as the actor, and the same LLM, when prompted for self-evaluation, serves as an efficient, on-demand critic, providing a principled, feedback-driven approach to the optimal stopping problem.

B Why is Self-Critique the Superior Policy? (Theorems 1 and 2): Analysis of the Adaptive Self-Critique Policy

In this appendix we give full, self-contained proofs for two versions of the Self-Critique Superiority result: first in the idealized case with no critique cost, and then the general case including a fixed cost C_{crit}.

B.1 Notation and Setup

We consider a decision at a single spike event. Let

$$q \;=\; \Pr\big(\text{spike is premature}\big), \qquad (1-q) = \Pr\big(\text{spike is correct}\big).$$

Upon stopping at a spike, the deterministic "always-stop" policy π_{det} immediately ends generation and obtains utility

$$U_s \;=\; U\big(\text{stop}\big).$$

If one instead *continues* past a premature spike, one realizes an additional utility gain

$$\Delta U \;=\; U\big(\text{continue}\big) \;-\; U\big(\text{stop}\big) \;>\; 0,$$

so that

$$U_c \;=\; U_s + \Delta U$$

denotes the utility of continuing. An LLM-based critic is invoked by the adaptive policy π_{crit} and classifies any spike as either "premature" or "correct." We denote its (symmetric) accuracy by

$$\eta \;=\; \Pr\big(\text{critic correct}\big) \;>\; \tfrac{1}{2},$$

meaning it correctly calls a premature spike "premature" with probability η, and correctly calls a correct spike "correct" with probability η.

B.2 Analysis of Self critique with No Critique Cost

Theorem 1 (Superiority of Adaptive Self-Critique, No Cost)**.** Under the above definitions, and assuming invoking the critic has zero cost, the expected utility difference between π_{crit} and π_{det} at a spike is

$$\mathbb{E}\big[U(\pi_{\text{crit}})\big] - \mathbb{E}\big[U(\pi_{\text{det}})\big] \;=\; \Delta U\;\big[q\,\eta + (1-q)(1-\eta)\big].$$

In particular, since $\eta > 0.5$ and $\Delta U > 0$, this difference is strictly positive for any $q \in [0, 1)$.

Proof. The always-stop policy π_{det} never continues, so it always obtains $\mathbb{E}[U(\pi_{\text{det}})] = U_s$.

The adaptive policy π_{crit} first invokes the critic (with no cost). Two cases arise:

1. **Spike is premature** with probability q:
   (a) Critic correct (prob. η): continue $\rightarrow$ utility U_c.

(b) Critic errs (prob. $1-\eta$): stop $\rightarrow$ utility U_s.

2. **Spike is correct** with probability $1-q$:

   (a) Critic correct (prob. η): stop $\rightarrow$ utility U_s.
   (b) Critic errs (prob. $1-\eta$): continue $\rightarrow$ utility U_c.

Hence the expected utility of π_{crit} is

$$\mathbb{E}[U(\pi_{\text{crit}})] = q\big[\eta\, U_c + (1-\eta)\, U_s\big] + (1-q)\big[\eta\, U_s + (1-\eta)\, U_c\big].$$

Substitute $U_c = U_s + \Delta U$:

$$\mathbb{E}[U(\pi_{\text{crit}})] = q\big[\eta\,(U_s + \Delta U) + (1-\eta)\, U_s\big] + (1-q)\big[\eta\, U_s + (1-\eta)\,(U_s + \Delta U)\big].$$

Collecting terms gives

$$\mathbb{E}[U(\pi_{\text{crit}})] = U_s + \Delta U\ \big[q\,\eta + (1-q)(1-\eta)\big].$$

Subtracting $\mathbb{E}[U(\pi_{\text{det}})] = U_s$ yields the claimed result. □

B.3 Analysis With Critique Cost

Theorem 2 (Superiority of Adaptive Self-Critique, With Cost)**.** Under the same setup, but now assuming each invocation of the critic incurs a fixed expected utility cost $C_{\text{crit}} > 0$, the expected utility difference at a spike is

$$\mathbb{E}\big[U(\pi_{\text{crit}})\big] - \mathbb{E}\big[U(\pi_{\text{det}})\big] \;=\; \Delta U\ \big[q\,\eta + (1-q)(1-\eta)\big] \;-\; C_{\text{crit}}.$$

In particular, whenever $\Delta U\,[\,q\,\eta + (1-q)(1-\eta)\,] > C_{\text{crit}}$, the self-critique policy strictly outperforms always-stop.

Proof. As before, $\mathbb{E}[U(\pi_{\text{det}})] = U_s$. The only change is that invoking the critic now deducts C_{crit} from utility. Thus

$$\mathbb{E}[U(\pi_{\text{crit}})] = \Big\{q\big[\eta\, U_c + (1-\eta)\, U_s\big] + (1-q)\big[\eta\, U_s + (1-\eta)\, U_c\big]\Big\} \;-\; C_{\text{crit}}.$$

Substituting $U_c = U_s + \Delta U$ and collecting terms exactly as in Theorem 1 gives

$$\mathbb{E}[U(\pi_{\text{crit}})] = U_s + \Delta U\ \big[q\,\eta + (1-q)(1-\eta)\big] \;-\; C_{\text{crit}}.$$

Subtracting U_s yields the stated result. The condition for strict superiority follows immediately by requiring the right-hand side to be positive. □

C Limitations and Future Work

Generality of the Control Signal. A key strength of our approach is its reliance on control token probabilities that are *intrinsic* to the model's own generation process. Unlike methods requiring auxiliary reward models or expensive external verifiers, VASP requires no additional training, no labeled preference data, and no modifications to the base model. Our experiments demonstrate that this signal is remarkably consistent across six diverse model families spanning 4B to 32B parameters, suggesting that well-trained reasoning models naturally develop informative stopping signals as a byproduct of standard pre-training objectives.

Training-Free Deployment. VASP operates entirely at inference time with negligible computational overhead—requiring only a comparison of token logits that are already computed during generation. This makes it immediately deployable to any reasoning model without fine-tuning, distillation, or architectural changes. The framework is complementary to, rather than competitive with, training-time improvements to calibrate the control token probabilities.

Unified Framework. Our formulation as an optimal stopping problem with a verification function Φ provides a principled foundation that unifies several distinct approaches (deterministic stopping, self-critique, and majority voting) under a single theoretical lens. This abstraction opens natural avenues for future work: alternative verification functions (e.g., external tool use, retrieval-augmented verification) can be seamlessly integrated by simply instantiating a different Φ.

Scope. Our evaluation focuses on mathematical reasoning benchmarks where ground-truth correctness is unambiguous. The extension of adaptive stopping to open-ended generation (e.g., creative writing, open-domain dialogue) would require domain-specific notions of quality that are beyond the scope of this work. We view this as an orthogonal research direction concerning the definition of utility in subjective domains, rather than a limitation of the VASP mechanism itself.

D Ablation: Signal Accumulation Strategies

In Section 4, we defined the evidence function E_t as a count of discrete spikes. However, alternative accumulation strategies are possible. Here, we evaluate two variants against our chosen N-Spike counting method:

Accumulated Probability: Instead of discretizing spikes via a threshold δ, we integrate the raw probability mass: $E_t = \sum_{j=1}^{t} P(t_{control} \mid \mathbf{x}_{<j})$. Stopping occurs when mass $\geq N$.

Last-Interval: A predictive strategy that estimates the "time to completion" based on the interval between recent spikes.

Figure 7 compares these strategies on the MATH-500 benchmark under two budget constraints (16K and 32K tokens).

Result. We observe that the **N-Spike Count** strategy consistently outperforms the other heuristics.

- **Robustness:** The Accumulated Probability method is sensitive to the "noise floor"—small, non-zero probabilities in the long tail can prematurely trigger termination even without a distinct spike.
- **Efficiency:** The Last-Interval method effectively mitigates underthinking but incurs a disproportionately high token cost (see Right Panel), often exceeding the budget without yielding proportional accuracy gains.

Based on this trade-off, we selected the discrete N-Spike counting method for the core VASP framework, as it provides the optimal balance of noise rejection and signal sensitivity.

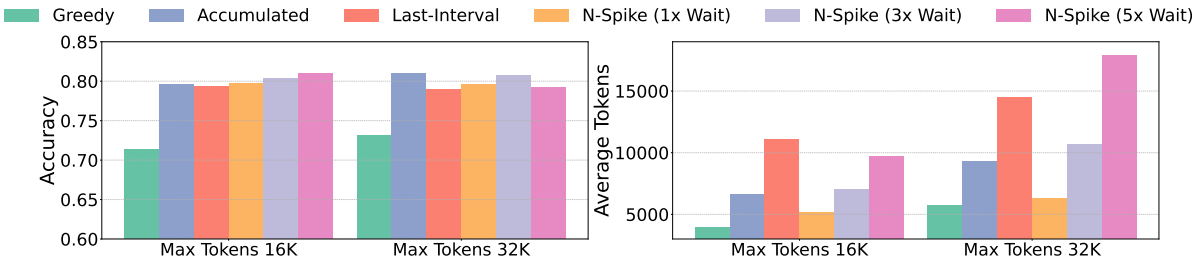


Figure 7: **Comparison of Signal Accumulation Strategies.** We evaluate Greedy decoding vs. three accumulation heuristics: Accumulated Probability, Last-Interval, and N-Spike Counting $(1\times, 3\times, 5\times)$. The N-Spike strategy (specifically $5\times$) achieves the highest accuracy while maintaining a controllable token budget, justifying its selection for the VASP framework.

E Results

Table 3: Performance comparison of adaptive stopping **policies** on `llama3.1-8b-Instruct` under different generation budgets.

Model	Policy	Max Tokens	Threshold	Wait Counter	LC-WR (%)	WR (%)	Average Tokens
Max Tokens = 256							
llama3.1-8b-Instruct	Greedy Policy	256	–	–	16.34	9.47	218.56
llama3.1-8b-Instruct	Accumulated Probability Policy	256	1.00E–02	–	11.20	5.34	137.98
llama3.1-8b-Instruct	Accumulated Probability Policy	256	1.00E–01	–	14.07	6.43	141.87
llama3.1-8b-Instruct	Accumulated Probability Policy	256	5.00E–01	–	15.66	7.45	142.90
llama3.1-8b-Instruct	Last-Interval Budget Policy	256	1.00E–05	–	16.28	8.45	209.61
llama3.1-8b-Instruct	Last-Interval Budget Policy	256	1.00E–03	–	14.25	8.47	215.83
llama3.1-8b-Instruct	Last-Interval Budget Policy	256	1.00E–01	–	15.41	9.09	217.97
llama3.1-8b-Instruct	DASP	256	1.00E–05	1	13.15	6.40	135.94
llama3.1-8b-Instruct	DASP	256	1.00E–03	1	13.40	6.65	138.14
llama3.1-8b-Instruct	DASP	256	1.00E–01	1	13.70	6.82	142.16
llama3.1-8b-Instruct	DASP	256	1.00E–05	3	14.23	6.66	140.55
llama3.1-8b-Instruct	DASP	256	1.00E–03	3	14.67	6.95	142.08
llama3.1-8b-Instruct	DASP	256	1.00E–01	3	13.73	6.80	142.21
llama3.1-8b-Instruct	DASP	256	1.00E–05	5	13.76	6.73	140.92
llama3.1-8b-Instruct	DASP	256	1.00E–03	5	13.45	6.62	142.18
llama3.1-8b-Instruct	DASP	256	1.00E–01	5	13.73	6.80	142.21
Max Tokens = 512							
llama3.1-8b-Instruct	Greedy Policy	512	–	–	22.92	19.91	380.56
llama3.1-8b-Instruct	Accumulated Probability Policy	512	1.00E–02	–	23.36	14.68	296.01
llama3.1-8b-Instruct	Accumulated Probability Policy	512	1.00E–01	–	21.61	14.37	300.91
llama3.1-8b-Instruct	Accumulated Probability Policy	512	5.00E–01	–	25.08	17.25	309.78
llama3.1-8b-Instruct	Last-Interval Budget Policy	512	1.00E–05	–	24.08	18.81	353.31
llama3.1-8b-Instruct	Last-Interval Budget Policy	512	1.00E–03	–	25.13	20.66	365.37
llama3.1-8b-Instruct	Last-Interval Budget Policy	512	1.00E–01	–	23.85	20.21	370.91
llama3.1-8b-Instruct	DASP	512	1.00E–05	1	21.12	12.51	286.63
llama3.1-8b-Instruct	DASP	512	1.00E–03	1	22.05	14.49	296.80
llama3.1-8b-Instruct	DASP	512	1.00E–01	1	22.93	15.95	308.44
llama3.1-8b-Instruct	DASP	512	1.00E–05	3	22.94	15.33	303.06
llama3.1-8b-Instruct	DASP	512	1.00E–03	3	23.93	16.29	306.57
llama3.1-8b-Instruct	DASP	512	1.00E–01	3	23.85	16.84	308.44
llama3.1-8b-Instruct	DASP	512	1.00E–05	5	22.85	15.90	308.10
llama3.1-8b-Instruct	DASP	512	1.00E–03	5	23.85	16.84	308.57
llama3.1-8b-Instruct	DASP	512	1.00E–01	5	23.85	16.84	308.44
Max Tokens = 1024							
llama3.1-8b-Instruct	Greedy Policy	1024	–	–	26.41	28.44	470.42
llama3.1-8b-Instruct	Accumulated Probability Policy	1024	1.00E–02	–	24.92	22.54	397.09
llama3.1-8b-Instruct	Accumulated Probability Policy	1024	1.00E–01	–	26.63	24.98	406.50
llama3.1-8b-Instruct	Accumulated Probability Policy	1024	5.00E–01	–	27.63	26.53	418.35
llama3.1-8b-Instruct	Last-Interval Budget Policy	1024	1.00E–05	–	30.11	29.45	423.27
llama3.1-8b-Instruct	Last-Interval Budget Policy	1024	1.00E–03	–	28.44	28.40	436.80
llama3.1-8b-Instruct	Last-Interval Budget Policy	1024	1.00E–01	–	28.45	28.48	436.80
llama3.1-8b-Instruct	DASP	1024	1.00E–05	1	24.73	21.00	374.77
llama3.1-8b-Instruct	DASP	1024	1.00E–03	1	24.53	22.91	403.34
llama3.1-8b-Instruct	DASP	1024	1.00E–01	1	27.47	26.90	419.77
llama3.1-8b-Instruct	DASP	1024	1.00E–05	3	25.12	23.93	406.38
llama3.1-8b-Instruct	DASP	1024	1.00E–03	3	26.61	25.94	417.71
llama3.1-8b-Instruct	DASP	1024	1.00E–01	3	27.47	26.90	419.77
llama3.1-8b-Instruct	DASP	1024	1.00E–05	5	27.26	26.05	411.97
llama3.1-8b-Instruct	DASP	1024	1.00E–03	5	26.44	25.90	417.76
llama3.1-8b-Instruct	DASP	1024	1.00E–01	5	27.47	26.90	419.77

F Complete Results Table over 6 models on AIME2025 and Math500

We provide the full results table for the experiments in the main paper below.

Table 4: Performance comparison of adaptive stopping **policies** on `s1.1-7B` under different thinking-token budgets, including the VASP.

Model	Policy	Max Thinking Tokens	Threshold	Wait Counter	Accuracy	Average Tokens
Max Thinking Tokens = 2048						
s1.1-7B	Greedy Policy	2048	–	–	0.500	1784.86
s1.1-7B	Accumulated Probability Policy	2048	0.1	–	0.668	2696.68
s1.1-7B	Accumulated Probability Policy	2048	1	–	0.680	2716.13
s1.1-7B	Accumulated Probability Policy	2048	3	–	0.686	2920.95
s1.1-7B	Last-Interval Budget Policy	2048	0.01	–	0.700	3004.00
s1.1-7B	Last-Interval Budget Policy	2048	0.1	–	0.700	2983.52
s1.1-7B	Last-Interval Budget Policy	2048	0.5	–	0.698	3026.28
s1.1-7B	DASP	2048	0.01	1	0.714	2720.42
s1.1-7B	DASP	2048	0.01	3	0.706	2925.70
s1.1-7B	DASP	2048	0.01	5	0.710	2991.65
s1.1-7B	DASP	2048	0.1	1	0.714	2727.15
s1.1-7B	DASP	2048	0.1	3	0.696	2985.12
s1.1-7B	DASP	2048	0.1	5	0.684	3051.00
s1.1-7B	DASP	2048	0.5	1	0.706	2730.81
s1.1-7B	DASP	2048	0.5	3	0.688	2986.16
s1.1-7B	DASP	2048	0.5	5	0.704	3022.09
s1.1-7B	VASP	2048	–	–	0.742	2614.56
Max Thinking Tokens = 4096						
s1.1-7B	Greedy Policy	4096	–	–	0.642	2725.00
s1.1-7B	Accumulated Probability Policy	4096	0.1	–	0.726	3841.05
s1.1-7B	Accumulated Probability Policy	4096	1	–	0.748	3871.68
s1.1-7B	Accumulated Probability Policy	4096	3	–	0.756	4550.30
s1.1-7B	Last-Interval Budget Policy	4096	0.01	–	0.758	5255.40
s1.1-7B	Last-Interval Budget Policy	4096	0.1	–	0.778	5354.85
s1.1-7B	Last-Interval Budget Policy	4096	0.5	–	0.756	5406.34
s1.1-7B	DASP	4096	0.01	1	0.762	3759.02
s1.1-7B	DASP	4096	0.01	3	0.754	4304.67
s1.1-7B	DASP	4096	0.01	5	0.770	4647.54
s1.1-7B	DASP	4096	0.1	1	0.762	3759.02
s1.1-7B	DASP	4096	0.1	3	0.772	4374.37
s1.1-7B	DASP	4096	0.1	5	0.758	5002.43
s1.1-7B	DASP	4096	0.5	1	0.762	3759.02
s1.1-7B	DASP	4096	0.5	3	0.756	4670.00
s1.1-7B	DASP	4096	0.5	5	0.756	5373.00
s1.1-7B	VASP	4096	–	–	0.792	4244.91
Max Thinking Tokens = 8192						
s1.1-7B	Greedy Policy	8192	–	–	0.714	3917.57
s1.1-7B	Accumulated Probability Policy	8192	0.1	–	0.774	5077.64
s1.1-7B	Accumulated Probability Policy	8192	1	–	0.788	5405.76
s1.1-7B	Accumulated Probability Policy	8192	3	–	0.796	6631.65
s1.1-7B	Last-Interval Budget Policy	8192	0.01	–	0.794	11033.53
s1.1-7B	Last-Interval Budget Policy	8192	0.1	–	0.800	10712.60
s1.1-7B	Last-Interval Budget Policy	8192	0.5	–	0.784	10512.50
s1.1-7B	DASP	8192	0.01	1	0.798	5163.36
s1.1-7B	DASP	8192	0.01	3	0.800	5817.96
s1.1-7B	DASP	8192	0.01	5	0.802	6726.14
s1.1-7B	DASP	8192	0.1	1	0.798	5163.36
s1.1-7B	DASP	8192	0.1	3	0.806	5975.91
s1.1-7B	DASP	8192	0.1	5	0.804	8105.31
s1.1-7B	DASP	8192	0.5	1	0.798	5114.54
s1.1-7B	DASP	8192	0.5	3	0.804	7010.83
s1.1-7B	DASP	8192	0.5	5	0.810	9727.63
s1.1-7B	VASP	8192	–	–	0.812	7330.56

Table 5: Performance of Qwen3 Models on the Maths500 Benchmark. This table compares the Baseline performance of Qwen3-8b and Qwen3-14b models against the 'Peak-Sampler-Critique' method under various configurations. The primary metrics are Maths500 accuracy and the average token count per problem. The results show that the Peak-Sampler-Critique method, particularly with k=3 critiques, a spike threshold of 0.25, and a larger context window, achieves the highest accuracy (0.950) while significantly reducing the token count compared to the baseline.

Model	Method	Critiques (k)	Context Window	Spike Threshold	Maths500 Acc.	Avg. Tokens
Qwen3-8b	Baseline	N/A	8k / 16k	N/A	0.924	4683.00
	Baseline	N/A	16k / 32k	N/A	0.934	5312.00
	Peak-Sampler	1	8k / 16k	0.10	0.900	2166.10
	Peak-Sampler	3	8k / 16k	0.10	0.908	2718.62
	Peak-Sampler	5	8k / 16k	0.10	0.922	3108.73
	Peak-Sampler	1	8k / 16k	0.25	0.914	2482.83
	Peak-Sampler	3	8k / 16k	0.25	0.918	2882.83
	Peak-Sampler	5	8k / 16k	0.25	0.922	3220.43
	Peak-Sampler	1	16k / 32k	0.10	0.906	2227.50
	Peak-Sampler	3	16k / 32k	0.10	0.916	2799.19
	Peak-Sampler	5	16k / 32k	0.10	0.928	3370.71
	Peak-Sampler	1	16k / 32k	0.25	0.914	2568.46
	Peak-Sampler	**3**	**16k / 32k**	**0.25**	**0.944**	**3055.18**
	Peak-Sampler	5	16k / 32k	0.25	0.937	3635.18
Qwen3-14b	Baseline	N/A	8k / 16k	N/A	0.933	4286.70
	Baseline	N/A	16k / 32k	N/A	0.940	4732.62
	Peak-Sampler	1	8k / 16k	0.25	0.928	2141.40
	Peak-Sampler	3	8k / 16k	0.25	0.932	2683.92
	Peak-Sampler	5	8k / 16k	0.25	0.930	2883.92
	Peak-Sampler	1	16k / 16k	0.25	0.932	2138.05
	Peak-Sampler	**3**	**16k / 32k**	**0.25**	**0.950**	**3084.97**
	Peak-Sampler	**5**	**16k / 32k**	**0.25**	**0.950**	**3524.97**

Table 6: Performance of Qwen3 Models on the AIME25 Benchmark. This table presents the accuracy and average token consumption for Qwen3 models of varying sizes (4B, 8B, 14B). We compare the standard Baseline generation method against our 'Peak-Sampler-Critique' approach with an increasing number of critiques (k). The Peak-Sampler-Critique method generally improves accuracy over the baseline for all model sizes, with performance scaling with the number of critiques. The Qwen3-14B model with k=7 achieves the highest accuracy of 0.722, a notable improvement over its baseline performance of 0.655. All experiments were conducted using a 16k/32k context window and a spike threshold of 0.25 for the Peak-Sampler method.

Model	Method	Critiques (k)	AIME Accuracy	Average Token Count
Qwen3-4B	Baseline	N/A	0.578	13923.5
	Peak-Sampler	1	0.434	10047.20
	Peak-Sampler	3	0.500	12010.20
	Peak-Sampler	5	0.588	13451.10
	Peak-Sampler	**7**	**0.600**	**14176.20**
Qwen3-8B	Baseline	N/A	0.589 ± 0.056	13793.5 ± 113.77
	Peak-Sampler	1	0.500 ± 0.081	12364.1 ± 320.96
	Peak-Sampler	3	0.622 ± 0.056	13408.4 ± 199.80
	Peak-Sampler	**5**	$\mathbf{0.667 \pm 0.027}$	$\mathbf{13850.0 \pm 462.10}$
	Peak-Sampler	7	0.656 ± 0.032	15192.0 ± 483.86
Qwen3-14B	Baseline	N/A	0.655 ± 0.041	13278.0 ± 201.50
	Peak-Sampler	1	0.688 ± 0.068	11977.7 ± 208.60
	Peak-Sampler	3	0.667 ± 0.072	13099.4 ± 142.80
	Peak-Sampler	5	0.711 ± 0.042	13559.0 ± 378.60
	Peak-Sampler	**7**	$\mathbf{0.722 \pm 0.031}$	$\mathbf{14064.0 \pm 666.58}$

Table 7: Comparison of Accuracy and Average Thinking tokens across different models, sample sizes (n), and critique parameters (N). "Base" denotes standard Majority Voting, while numerical values denote N-Critique. Outputs are format as Accuracy (Avg Thinking Tokens)

Maths500							
n	N	**Qwen3 4B**	**Qwen3 8B**	**Qwen3 14B**	**Qwen3 32B**	**R1-Distill Qwen7B**	**R1-Distill Llama8B**
4	Base	0.926 (19.4k)	0.930 (17.9k)	0.950 (15.9k)	0.954 (17.5k)	0.924 (13.1k)	0.888 (15.5k)
	1	0.926 (–)	0.930 (5.4k)	0.946 (5.4k)	0.946 (4.1k)	0.924 (5.7k)	0.908 (6.7k)
	3	0.940 (9.4k)	0.932 (8.2k)	0.950 (8.1k)	0.958 (7.5k)	0.932 (8.1k)	0.900 (9.1k)
	5	0.935 (–)	0.944 (10.0k)	0.952 (9.9k)	0.950 (9.5k)	0.932 (9.3k)	0.916 (10.1k)
8	Base	0.930 (35.7k)	0.940 (35.8k)	0.954 (31.8k)	0.956 (30.6k)	0.936 (26.2k)	0.904 (31.2k)
	1	0.942 (–)	0.932 (10.6k)	0.954 (10.7k)	0.958 (8.3k)	0.938 (11.3k)	0.914 (13.5k)
	3	0.952 (18.8k)	0.936 (16.3k)	0.956 (16.1k)	0.960 (14.9k)	0.938 (16.2k)	0.912 (18.0k)
	5	0.946 (–)	0.948 (20.1k)	0.958 (19.9k)	0.954 (19.1k)	0.940 (18.6k)	0.918 (20.3k)
16	Base	0.942 (71.3k)	0.944 (71.4k)	0.954 (63.7k)	0.956 (61.2k)	0.942 (52.5k)	0.916 (62.8k)
	1	– (–)	0.938 (21.3k)	0.956 (20.6k)	0.964 (16.6k)	0.934 (22.8k)	0.928 (27.2k)
	3	0.952 (37.8k)	0.948 (32.5k)	0.958 (32.0k)	0.960 (29.0k)	0.942 (32.3k)	0.922 (35.9k)
	5	– (–)	0.948 (41.2k)	0.958 (38.0k)	0.956 (38.2k)	0.942 (37.2k)	0.924 (40.9k)

AIME25							
n	N	**Qwen3 4B**	**Qwen3 8B**	**Qwen3 14B**	**Qwen3 32B**	**R1-Distill Qwen7B**	**R1-Distill Llama8B**
4	Base	0.734 (67.8k)	0.767 (70.6k)	0.800 (65.0k)	0.800 (60.2k)	0.467 (58.0k)	0.434 (62.2k)
	1	0.667 (58.7k)	0.600 (42.8k)	0.767 (42.9k)	0.734 (20.4k)	0.467 (39.5k)	0.467 (49.7k)
	3	0.734 (63.6k)	0.667 (46.8k)	0.734 (49.4k)	0.800 (34.1k)	0.534 (47.4k)	0.400 (54.9k)
	5	0.734 (63.9k)	0.700 (50.6k)	0.767 (55.8k)	0.767 (41.8k)	0.467 (48.5k)	0.367 (56.8k)
8	Base	0.734 (137k)	0.767 (142k)	0.800 (127k)	0.800 (124k)	0.600 (114k)	0.467 (129k)
	1	0.734 (116k)	0.634 (82.5k)	0.800 (82.1k)	0.734 (40.3k)	0.500 (84.5k)	0.500 (94.2k)
	3	0.767 (123k)	0.667 (95.0k)	0.800 (99.8k)	0.800 (68.2k)	0.567 (96.4k)	0.500 (111k)
	5	0.767 (123k)	0.700 (104k)	0.800 (110k)	0.767 (81.2k)	0.534 (101k)	0.500 (113k)
16	Base	0.767 (278k)	0.767 (286k)	0.800 (258k)	0.834 (245k)	0.567 (242k)	0.500 (260k)
	1	0.767 (228k)	0.634 (157k)	0.834 (169k)	0.767 (80.6k)	0.534 (171k)	0.567 (187k)
	3	0.800 (245k)	0.734 (189k)	0.800 (207k)	0.834 (135k)	0.600 (193k)	0.500 (220k)
	5	0.767 (247k)	0.734 (207k)	0.800 (225k)	0.767 (159k)	0.534 (202k)	0.500 (234k)