Filter by Quality, Stop by Difficulty: Confidence-Aware Self-Consistency with Adaptive Early Stopping

Jooyeong Na[1] **Wansik Jo**[2] **Hyunsouk Cho**[1*]
[1]Ajou University, Suwon, Republic of Korea
[2]Meritz Fire & Marine Insurance, Seoul, Republic of Korea
{sklwndud,hyunsouk}@ajou.ac.kr wnsk.jo@meritzfire.co.kr

Abstract

Self-consistency improves LLM reasoning by majority voting over multiple sampled solutions, but always exhausts the sampling budget and trusts every solution equally. Early-stopping methods cut token cost substantially via convergence monitoring. However, deciding *when* to stop cannot change *what* is voted on, so they save tokens without meaningfully improving accuracy. We introduce ***CASE*** (**C**onfidence-**A**ware **S**elf-consistency with Adaptive **E**arly stopping), which improves both accuracy and efficiency by leveraging token-level confidence normalized at two scales: within-problem to filter low-quality solutions and weight survivors, and cross-problem to adapt the stopping threshold to problem difficulty. Across 5 models and 6 benchmarks, ***CASE*** simultaneously saves 85% of tokens and improves accuracy by +2.24 pp over majority voting, outperforming every early-stopping baseline on both accuracy and efficiency. Component and role-swap analyses confirm that filtering and weighting are complementary, and the two normalization scales are non-interchangeable.

1 Introduction

Self-consistency (Wang et al., 2023) is the dominant strategy for improving the reasoning accuracy of large language models: it samples N solutions with chain-of-thought prompting (Wei et al., 2022) and selects the answer by majority vote (MV). By aggregating diverse reasoning paths, self-consistency mitigates the influence of individual reasoning errors and can substantially boost accuracy over greedy decoding. However, this approach trusts every solution equally regardless of *solution quality*, and exhausts the full sampling budget regardless of *problem difficulty*. In practice, easy problems consume more samples than needed; hard problems, where incorrect solutions often dominate, lock the budget into an incorrect majority.

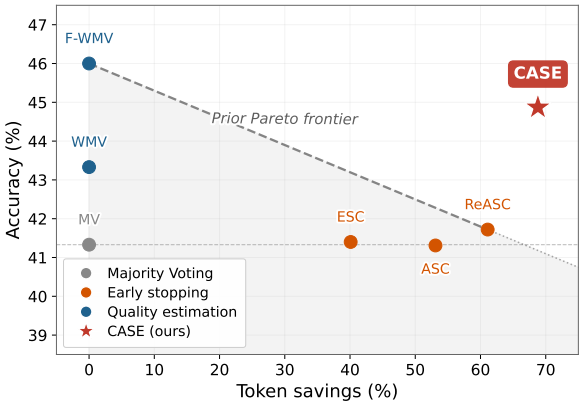


Figure 1: Accuracy vs. token savings, averaged over the 20 competition cells ($N = 64$). Prior early stopping stays within 0.4 pp of MV (ESC 40.1%, ASC 53.1%, ReASC 61.1% savings), while ***CASE*** reaches 68.8% at +3.5 pp; the dashed line is the prior Pareto frontier.

Early-stopping methods address this inefficiency, stopping sampling once sufficient consensus forms. Count-based methods such as ESC (Li et al., 2024) and ASC (Aggarwal et al., 2023) trigger stopping from vote counts alone, while ReASC (Kim et al., 2026) adds a per-solution reliability signal. These methods reduce token cost by 40–61% on competition benchmarks (Figure 1) but cluster at the efficiency end of the prior Pareto frontier, saving tokens without meaningfully improving accuracy over MV. The limitation is structural: deciding *when* to stop cannot alter *what* is voted on, so once incorrect solutions dominate the pool, no stopping rule recovers the correct answer.

Improving accuracy beyond MV requires a different kind of signal: solution quality, so that unreliable solutions can be identified and down-weighted or excluded before voting. Approaches that leverage reward models (Cobbe et al., 2021; Lightman et al., 2024) to score solution correctness,

* Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 42269–42289
October 24-29, 2026 ©2026 Association for Computational Linguistics

or that prompt the model to evaluate its own output (Taubenfeld et al., 2025), achieve strong quality discrimination but require task-specific training data or additional inference calls per solution. A lighter alternative estimates quality from the model's own token log-probabilities (Kang et al., 2025; Razghandi et al., 2025; Zhou et al., 2025; Fu et al., 2026), which are available as a byproduct of generation at negligible additional cost. Applied as a full-budget reference in our setup (F-WMV in §4.1), filtering and weighting by these signals surpass MV by $+4.7$ pp on competition benchmarks (Figure 1), confirming that improving vote pool quality is the key to surpassing majority voting accuracy. However, these methods consume the entire budget, leaving efficiency unaddressed.

To improve accuracy and efficiency jointly, we introduce ***CASE*** (**C**onfidence-**A**ware **S**elf-consistency with Adaptive **E**arly stopping). ***CASE*** estimates solution quality from token log-probabilities produced during generation, leveraging this signal to filter low-quality solutions, weight survivors by quality, and adapt the stopping threshold to problem difficulty in real time as solutions arrive. To serve these distinct roles, ***CASE*** normalizes the confidence signal at two scales: within-problem for filtering and weighting, and cross-problem for difficulty-aware stopping (§3; validated in §4.4). Prior confidence-based methods instantiate strict subsets of these three roles (Appendix A); reading one signal at two scales lets ***CASE*** occupy all three at once.

Across 5 models and 6 benchmarks, ***CASE*** saves 85% of tokens while improving accuracy by $+2.24$ pp over MV ($+3.5$ pp on competition), entering the accuracy–efficiency region beyond the prior Pareto frontier (Figure 1). A factorial analysis (§4.3) reveals filter–weight complementarity: filtering raises accuracy but slows convergence; weighting concentrates vote mass on high-confidence survivors, restoring the sampling efficiency lost to filtering. A role-swap experiment (§4.4) confirms the two normalization scales play non-interchangeable roles; applying the wrong scale for filtering increases sample usage by 38%.

Our contributions are:

- We decompose self-consistency into filtering, weighting, and stopping, and propose ***CASE***, which instantiates all three from a single token-level confidence signal read at two scales.
- Through a systematic component analysis (§4.3), we identify the filter–weight complementarity underlying ***CASE***'s accuracy–efficiency balance.
- A role-swap experiment (§4.4) confirms that the two normalization scales play non-interchangeable roles.

2 Related Work

Early Stopping for Self-Consistency. Early-stopping methods reduce cost by monitoring vote convergence: ESC (Li et al., 2024) stops when an observation window agrees (zero entropy); ASC (Aggarwal et al., 2023) stops once a Beta posterior over the leading answer's majority probability exceeds a threshold; RASC (Wan et al., 2025) leverages reasoning-path features for both weighting and stopping; and ReASC (Kim et al., 2026) adds a GMM gate that returns the first solution if reliable, otherwise applies confidence-weighted Beta stopping without modifying the vote pool. A complementary direction allocates sampling budgets by problem difficulty (Wang et al., 2025; Liu et al., 2026). None of these reliably surpass majority voting accuracy: the available signals (vote counts, reasoning-path features, pre-estimated difficulty) address stopping or budget allocation rather than solution quality, so a low-quality vote pool cannot be improved.

Solution Quality Estimation. A separate line of work improves accuracy by estimating individual solution quality and leveraging these estimates during voting. Trained verifiers (Cobbe et al., 2021; Lightman et al., 2024) require task-specific data, while LLM-as-judge (Zheng et al., 2023) and per-solution confidence queries (Taubenfeld et al., 2025) add inference cost. Lightweight alternatives extract quality signals from token-level distributions or aggregated log-probabilities (Kang et al., 2025; Razghandi et al., 2025; Leang et al., 2026; Zhou et al., 2025), improving accuracy over MV but consuming the full sampling budget. DeepConf (Fu et al., 2026) similarly extracts token-level confidence, with an offline mode at full budget and an online mode that performs token-level termination of individual solutions, requiring modifications to the standard serving pipeline (§4.2 and Appendix I). No prior method integrates these gains with early stopping in a single framework within a standard serving pipeline (Appendix A).

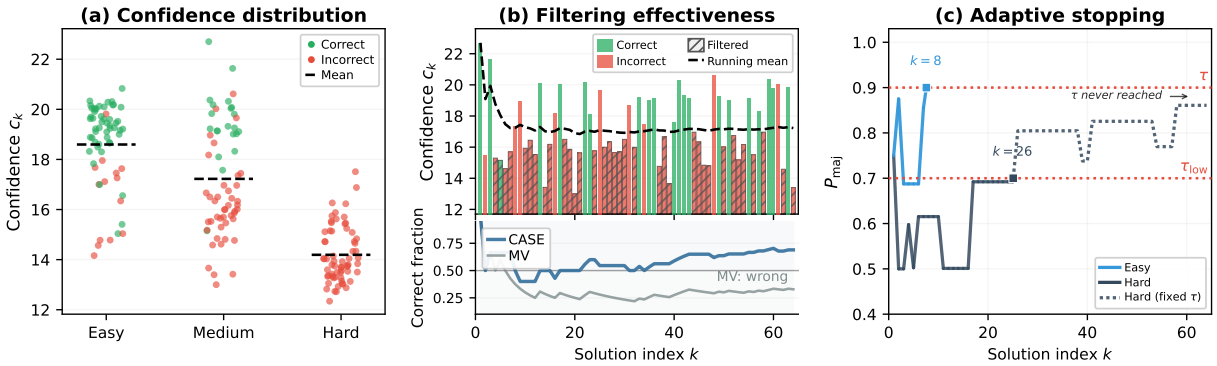


Figure 2: Motivating observations for ***CASE*** (Qwen3-1.7B-Think, AIME25). **(a)** Confidence c_k of 64 solutions on Easy, Medium, and Hard problems; incorrect solutions cluster below the per-problem mean (dashed), and the means decrease with difficulty. **(b)** On a Medium problem, filtering solutions below the running mean (hatched) removes predominantly incorrect solutions; the correct-answer fraction (bottom) shows ***CASE*** stabilizing above 0.5 while MV ends below, confirming that filtering recovers the correct answer. **(c)** P_{maj}, the posterior probability that the current leader remains dominant, serves as a stopping criterion. Easy crosses τ at k=8; Hard never reaches τ, requiring τ_{low} (k=26); the dotted segment shows the Hard trajectory continues below τ even at full budget.

3 CASE: Confidence-Aware Self-Consistency with Adaptive Early Stopping

In this section, we first decompose self-consistency into three functional components (§3.1), then describe the confidence signal shared by all three (§3.2), and finally instantiate each: filtering ($\mathcal{F}$; §3.3), weighting ($\mathcal{W}$; §3.4), and early stopping with difficulty adaptation ($\mathcal{S}$; §3.5).

3.1 A Modular Decomposition of Self-Consistency

Self-consistency (Wang et al., 2023) generates N independent chain-of-thought solutions $\{s_1, \ldots, s_N\}$ and selects the final answer by majority vote. We decompose this aggregation process into three functional components: (1) a *candidate filter* $\mathcal{F}$ that prunes low-quality solutions from the candidate pool, $\mathcal{F}(C) \to C' \subseteq C$; (2) a *weighting function* $\mathcal{W}$ that assigns each retained solution $s_k \in C'$ a vote weight w_k, $\mathcal{W}(C') \to \mathbf{w} \in \mathbb{R}^{|C'|}$; and (3) an *early-stopping criterion* $\mathcal{S}$ that determines whether to stop after each new solution. Let a_k denote the answer extracted from s_k. Each candidate answer a accumulates the weights of all surviving solutions that produced it, and the final prediction $\hat{a}$ is the answer with the largest total weight:

$$\hat{a} = \arg\max_{a} \sum_{s_k \in C'} w_k \cdot \mathbb{1}[a_k = a]. \qquad (1)$$

Based on this formulation, our goal is to design $\mathcal{F}, \mathcal{W}, \mathcal{S}$ that jointly maximize accuracy and minimize sampling cost, up to a budget of N solutions. Existing methods address only a subset of these components (Appendix A): standard self-consistency applies identity filtering ($C' = C$), uniform weighting ($w_k = 1$), and a fixed budget; ASC (Aggarwal et al., 2023) introduces a Beta-posterior $\mathcal{S}$ but retains identity filtering and uniform weighting; confidence-weighted voting (e.g., CER (Razghandi et al., 2025)) assigns non-uniform $\mathcal{W}$ but applies no filtering and no early stopping. Unlike these partial methods, ***CASE*** instantiates $\mathcal{F}$, $\mathcal{W}$, and $\mathcal{S}$ from a shared signal, detailed in the following subsections.

3.2 Solution Quality Estimation

All three components of ***CASE*** ($\mathcal{F}$, $\mathcal{W}$, $\mathcal{S}$) require a measure of solution quality. ***CASE*** estimates this quality with a *confidence score* derived from token log-probabilities, which correlate with solution correctness (Fu et al., 2026; Kang et al., 2025). Since log-probabilities are available as a byproduct of generation, this score requires no additional model calls and can be computed at every sampling step.

Following Fu et al. (2026), ***CASE*** computes this score in two steps: first at the token level, then aggregated to the solution level. For the k-th solution s_k consisting of T tokens, the top-K log-probabilities $\ell_{t,1} \geq \cdots \geq \ell_{t,K}$ are extracted at each position t. The token-level confidence is the negated mean of these top-K values, $c_t = -\frac{1}{K}\sum_{j=1}^{K} \ell_{t,j}$. When the model is confident, probability mass concentrates on a few top-ranked tokens, so most of the top-K entries take strongly

negative log-values and c_t becomes large; when uncertain, mass spreads across the top-K and c_t decreases. The design rationale for this choice is discussed in Appendix C.

CASE then aggregates token scores into a solution-level score over a group of g consecutive tokens (the *group size*; stride 1). Among aggregation strategies, ***CASE*** selects the *tail* group, defined as the mean of the final g token scores: $c_k = \frac{1}{g}\sum_{t=T-g+1}^{T} c_t$. This choice gives the highest average AUROC across models (Appendix D). The *tail confidence* c_k reflects the model's certainty in the concluding phase where reasoning converges to a final answer; it serves as the shared input to $\mathcal{F}$, $\mathcal{W}$, and $\mathcal{S}$.

3.3 Candidate Filtering ($\mathcal{F}$)

With c_k from §3.2, the filter $\mathcal{F}$ prunes low-quality solutions to raise the correct-answer fraction in the vote pool. A natural approach is to exclude solutions whose c_k falls below a fixed threshold, but this is problematic: the model tends to produce higher scores on easier problems (Figure 2(a): mean confidence decreases from Easy to Hard). Any single cutoff therefore over-rejects on Hard problems and under-rejects on Easy ones. ***CASE*** addresses this through within-problem normalization (hereafter *intra-z*): each solution's score is compared to others on the same problem, enabling a single threshold to apply uniformly.

To compute intra-z, each solution's confidence is normalized against the scores observed so far for the same problem: $z_k^{\text{intra}} = (c_k - \hat{\mu}_{<k})/\hat{\sigma}_{<k}$, where $\hat{\mu}_{<k}, \hat{\sigma}_{<k}$ are the running mean and standard deviation of $\{c_1, \ldots, c_{k-1}\}$. Solutions arrive one at a time, so the statistics are updated incrementally with each new solution.

With this score, $\mathcal{F}$ excludes any solution with $z_k^{\text{intra}} < f_z$ from the vote pool, i.e., $C' = \{s_k \in C \mid z_k^{\text{intra}} \geq f_z\}$. Even excluded solutions update $\hat{\mu}, \hat{\sigma}$: otherwise the running mean would drift upward, making the filter progressively stricter. Because intra-z measures quality *relative to each problem*, the filter maintains a balanced pass rate regardless of problem difficulty (f_z is set in §4.1). Figure 2(b) illustrates this on a Medium problem: filtering low-confidence solutions (hatched bars) preferentially removes incorrect ones, recovering the correct answer that MV misses.

3.4 Weighting ($\mathcal{W}$)

After filtering, the surviving solutions are aggregated into a final answer. The simplest choice is uniform majority voting, but this treats all solutions equally regardless of confidence, leaving the fine-grained quality information in c_k unused and delaying consensus. Since intra-z from §3.3 already captures each solution's relative quality, ***CASE*** reuses this score to weight votes.

CASE applies an exponential weight $w_k = \exp(\lambda \cdot z_k^{\text{intra}})$ to each retained solution $s_k \in C'$, following Kim et al. (2026). Here λ controls how sharply vote mass concentrates: $\lambda = 0$ recovers uniform voting, while larger λ progressively concentrates mass on the highest-confidence solutions. The exponential form amplifies high-confidence solutions (those with large z^{intra}), sharpening the vote signal as confidence rises. In practice, this concentrates the surviving pool's mass on high-confidence answers, compensating for the convergence cost that filtering induces (§4.3).

Since running statistics are unreliable when too few c_k have been observed, ***CASE*** applies a brief *warmup*: the first k_w solutions bypass $\mathcal{F}$ and $\mathcal{W}$ with unit weight while still updating $\hat{\mu}$ and $\hat{\sigma}$. After warmup, filtering and weighting activate with these stabilized statistics.

3.5 Difficulty-Aware Early Stopping ($\mathcal{S}$)

While $\mathcal{F}$ and $\mathcal{W}$ improve accuracy, $\mathcal{S}$ addresses the remaining efficiency objective from §3.1. We instantiate $\mathcal{S}$ as two layers: a base stopping rule S (Eq. 2) that terminates once the vote pool reaches sufficient consensus, and a difficulty adaptation D that conditions S's threshold on problem difficulty.

The base rule S follows ASC (Aggarwal et al., 2023): it frames stopping as a comparison between the top two answers, modeling the leader's win rate p with a Beta posterior. ***CASE*** extends this to confidence-weighted votes following ReASC (Kim et al., 2026). Let $V_1 \geq V_2$ denote the weighted votes of the top two answers, accumulated over the filter survivors C'. The leading answer's win rate p follows a $\text{Beta}(V_1 + 1, V_2 + 1)$ posterior, and ***CASE*** defines $P_{\text{maj}} = P(p > \frac{1}{2})$, the probability that the leading answer remains dominant as more solutions are generated. ***CASE*** stops sampling once this probability exceeds a threshold τ:

$$p \sim \text{Beta}(V_1{+}1, V_2{+}1), \quad P_{\text{maj}} = P(p > \tfrac{1}{2}) \geq \tau. \tag{2}$$

A higher τ demands more evidence before stopping; this works well for Easy problems where the pool quickly reaches consensus (Figure 2(c)). However, on Hard problems P_{maj} stays well below τ throughout the full budget (Figure 2(c)), wasting budget for negligible accuracy gain. To avoid this waste, ***CASE*** should terminate earlier on Hard problems with a relaxed threshold.

The remaining question is how to detect problem difficulty at inference time. The intra-z score cannot provide this signal: within-problem normalization removes cross-problem variation by design (§3.3), so Hard and Easy problems yield similar score distributions. ***CASE*** therefore introduces a second normalization that preserves cross-problem variation (hereafter *inter-z*): $z_k^{\text{inter}} = (c_k - \mu_G)/\sigma_G$, where (μ_G, σ_G) are the global mean and standard deviation of confidence scores. Because Easy problems produce higher confidence on average (Figure 2(a)), they yield positive inter-z while Hard problems yield negative values, providing the difficulty signal for adaptive stopping. A problem's mean confidence predicts its solve rate at a median per-cell Spearman correlation of $+0.57$, positive in 24 of the 25 math cells (Appendix D).

CASE precomputes (μ_G, σ_G) once per evaluation setting by sampling a single solution per problem; that same solution is then consumed as each problem's first warmup sample (§3.4), so calibration adds no extra inference cost. Precomputation is not even required: a rolling estimate accumulated over the served stream matches it to within 0.03 pp (Appendix E).

CASE leverages this signal to relax the stopping threshold on Hard problems. Like $\mathcal{F}$ and $\mathcal{W}$, this adaptation engages after warmup, once sufficient inter-z samples have accumulated for a stable running mean. The running mean of inter-z over all observed solutions (including filtered ones, since difficulty is a property of the problem) gives the difficulty estimate. If this running mean falls below a threshold δ, the problem is classified as Hard and the stopping threshold is relaxed from τ to τ_{low} ($\tau_{\text{low}} < \tau$). ***CASE*** then terminates as soon as P_{maj} exceeds τ_{low}, even on problems where it would never reach τ. Figure 2(c) shows this: on Hard, ***CASE*** stops via τ_{low} instead of exhausting the full budget.

Together, the three components address both terms of the objective in §3.1: $\mathcal{F}$ and $\mathcal{W}$ refine the vote pool with intra-z normalization, while $\mathcal{S}$ stops sampling by base consensus (S) and relaxes its threshold on Hard problems (D) via inter-z normalization. Algorithm 1 summarizes the full procedure; all hyperparameters are set in §4.1.

Algorithm 1 ***CASE*** Inference

Require: Budget N, calibration stats (μ_G, σ_G), hyperparameters $\{k_w, f_z, \lambda, \tau, \tau_{\text{low}}, \delta\}$

votes $\leftarrow \{\}$
for $k = 1$ **to** N **do**
  $c_k \leftarrow \text{TailConf}(s_k)$; $z_k^{\text{inter}} \leftarrow (c_k - \mu_G)/\sigma_G$
  $\tau_{\text{eff}} \leftarrow \tau$
  if $k \le k_w$ **then**
    votes[Answer(s_k)] += 1 {Warmup}
  else
    $z_k^{\text{intra}} \leftarrow (c_k - \hat{\mu}_{<k})/\hat{\sigma}_{<k}$
    if $z_k^{\text{intra}} \ge f_z$ **then**
      votes[Answer(s_k)] += $\exp(\lambda \cdot z_k^{\text{intra}})$ {$\mathcal{F}+\mathcal{W}$}
    end if
    if mean of $z_{1:k}^{\text{inter}} < \delta$ **then**
      $\tau_{\text{eff}} \leftarrow \tau_{\text{low}}$ {Hard problem}
    end if
  end if
  Update $\hat{\mu}, \hat{\sigma}$ with c_k {incl. filtered (§3.3)}
  if $P_{\text{maj}} \ge \tau_{\text{eff}}$ **then**
    return $\arg\max_a$ votes[a] {$\mathcal{S}$: Stop; P_{maj} from Eq. 2}
  end if
end for
return $\arg\max_a$ votes[a]

4 Experiments

We evaluate ***CASE*** along its design axes of efficiency and accuracy. After the setup (§4.1), we compare ***CASE*** against self-consistency baselines and analyze when the accuracy gain holds and what drives it (§4.2), attribute the resulting gains to individual components and to the dual-scale normalization design (§4.3 and §4.4), and measure wall-clock cost under a serving workload (§4.5).

4.1 Experimental Setup

Datasets and Baselines. We evaluate on six benchmarks in three categories. *Competition math*: AIME 2024 and 2025 (30 problems each; Zhang and Math-AI Team, 2024, 2025) and HMMT February 2024 and 2025 (30 problems each; Dekoninck et al., 2026). *Standard math*: MATH-500 (500 problems; Hendrycks et al., 2021). *Graduate-level science*: GPQA Diamond (198 problems; Rein et al., 2024).

We compare ***CASE*** against three early-stopping methods that aim to reduce sampling cost (ESC, ASC, ReASC) and three full-budget references that anchor the achievable accuracy at $N = 64$ (MV, WMV, F-WMV).

Stopping baselines. ESC (Li et al., 2024) (win-

dow $w=4$[1]), ASC (Aggarwal et al., 2023) (Beta-posterior, $\tau=0.95$), and ReASC (Kim et al., 2026) (GMM gate + weighted stopping, $\tau=0.95$, $\lambda=0.7$, online variant[2]); ASC and ReASC parameters follow the values reported in their original papers.

Full-budget references. MV anchors standard accuracy; WMV and F-WMV apply ***CASE***'s weighting (and filtering, for F-WMV) at full budget as aggregate references for these components. Methods requiring additional inference (DSC; Wang et al., 2025) or custom decoding pipelines (DeepConf; Fu et al., 2026) are excluded from Table 1; DeepConf is compared separately (§4.2 and Appendix I).

Implementation Details. We evaluate on five models that cover the 1.7B–8B parameter range, two architecture families (Qwen and Gemma), and both thinking-mode and non-thinking-mode variants, distinguished by whether the model was trained to emit chain-of-thought reasoning before its answer: Qwen3-1.7B-Thinking (1.7B-Think), Qwen3-4B-Thinking (4B-Think), Qwen3-4B-Instruct (4B-Inst), Qwen3-8B non-thinking (8B-NT) (Yang et al., 2025), and Gemma3-4B-IT (Gemma Team et al., 2025). Combined with the six benchmarks, this yields 30 model–benchmark combinations.

For each model and benchmark, we sample $N = 64$ solutions per problem once and replay this fixed pool in 20 independent random orders, applying each method's stopping criterion online: at step k the decision considers only the first k solutions. Because all methods see identical solutions in identical orders, any difference reflects the stopping rule, not sampling variance.

Performance is reported on two axes: (1) efficiency, capturing how much of the full sampling budget is consumed, and (2) accuracy. For (1) we report two complementary metrics: **Samp** (sample count, macro-averaged across combinations; includes the calibration solution and filtered-out solutions) for solutions consumed, and **TokSv** ($1-$tokens used/full-budget tokens, micro-averaged) for the fraction of budget tokens saved. For (2) we report **Acc%**, the percentage of problems whose selected answer matches the reference, averaged over the 20 shuffles and macro-averaged across combinations; Δ_{MV} is its change relative to MV in percentage points. Both efficiency metrics are needed because per-solution length varies $\sim 11\times$ across our models (1.6K–18K tokens; Appendix B): Samp alone undercounts the actual compute saved on long-solution models. We report results at $N = 64$ throughout the main text and verify that ***CASE***'s advantage persists and widens across $N \in \{16, 32, 64\}$ (Appendix H).

All six hyperparameters are fixed identically across the 30 combinations: $k_w = 3$, $f_z = 0$, $\lambda = 0.3$, $\tau = 0.90$, $\tau_{\text{low}} = 0.70$, $\delta = -0.7$. Appendix G details the rationale for each default. Sensitivity sweeps reported there confirm that the gain over MV is robust to the specific choice. Group sizes and sampling configurations are listed in Appendix B.

4.2 Experimental Results

Across the 30 model–benchmark combinations, no early-stopping baseline Pareto-dominates ***CASE*** on any cell, while ***CASE*** dominates the best per-cell baseline on 16 of 30 and is Pareto-incomparable on the rest (Table 1). Aggregated, ***CASE*** improves accuracy by $+2.24$ pp over MV at 13.6 samples and 85.3% token savings, while ESC, ASC, and ReASC stay within 0.12 pp of MV at 17 to 26 samples, confirming that prior early-stopping methods save tokens without improving accuracy. On the hardest cells it even exceeds full-budget references: on 8B-NT/HMMT24, $+5.5$ pp at 32.7 samples surpasses F-WMV's $+3.3$ pp at full budget (Table 1).

CASE gains accuracy when two conditions hold: (i) tail confidence must separate correct from incorrect solutions, and (ii) MV's vote pool must leave the room to improve. Competition benchmarks satisfy both: tail confidence is highly discriminative here (AUROC 0.85; Table 11), and MV errors are frequent (13 of 20 cells below 50%), giving $+3.5$ pp. MATH-500 fails (ii): MV already reaches about 85%, leaving little to recover. GPQA fails (i): AUROC falls to 0.58 (weakly discriminative), so the filter cannot tell correct from incorrect even though MV's accuracy near 50% leaves the room. Even so, both benchmarks still save 90% and 87% of tokens. The two conditions also predict the outcome off the grid, on MMLU-Pro and on a 14B model (Appendix K).

Within GPQA that failure is concentrated where answers are recalled rather than derived: tail con-

[1]ESC reports $w = 5$ (general) and $w = 8$ (MATH); we follow the most recent related work (Kim et al., 2026) in adopting $w=4$.

[2]Unlike the offline variant (which requires labeled validation data), the online variant calibrates the GMM threshold from Stage 1 confidence scores across the test set without labeled data, matching ***CASE***'s calibration setting.

Table 1: Per-cell results across 30 model–benchmark combinations (N=64, averaged over 20 shuffles). Δ: accuracy change vs. MV (pp). Samp: average samples consumed. TokSv: token savings (%). Best Δ, Samp, and TokSv per row among early-stopping methods in **bold**. Per-cell standard error across the 20 shuffles is ≤ 0.8 pp accuracy, ≤ 1.0 samples, and ≤ 1.9 pp TokSv across all methods (per-method breakdown in Appendix F).

			Full Budget		ESC			ASC			ReASC			*CASE*		
Model	**Dataset**	**MV**	**WMV**	**F-WMV**	**Δ**	**Samp**	**TokSv**	**Δ**	**Samp**	**TokSv**	**Δ**	**Samp**	**TokSv**	**Δ**	**Samp**	**TokSv**
1.7B-Think	AIME24	70.0	+3.3	+10.0	−0.5	30.1	45.8	−0.3	22.5	59.0	+2.3	17.2	67.3	**+5.2**	**10.9**	**80.4**
	AIME25	46.7	+6.7	+10.0	+1.3	36.7	31.2	0.0	28.3	46.6	+4.2	24.9	51.0	**+7.0**	**16.8**	**68.6**
	HMMT24	30.0	0.0	+6.7	−0.3	38.6	30.8	−0.3	30.7	44.9	−0.7	27.9	48.8	**+4.5**	**16.8**	**68.8**
	HMMT25	26.7	0.0	+20.0	+0.2	41.5	25.7	0.0	29.5	47.3	+2.2	27.4	50.2	**+14.8**	**22.3**	**59.3**
	MATH500	86.4	+0.2	0.0	−0.1	7.9	76.1	**0.0**	6.9	80.8	−0.2	5.5	82.5	−0.5	**3.8**	**92.1**
	GPQA	39.9	+1.5	+4.5	−1.0	20.3	64.2	−0.3	20.0	64.7	+0.6	16.7	70.0	**+2.3**	**11.2**	**79.9**
4B-Think	AIME24	90.0	0.0	0.0	+0.3	10.3	77.3	0.0	10.2	77.3	−0.2	9.2	78.6	**+0.5**	**4.2**	**92.2**
	AIME25	86.7	0.0	0.0	0.0	7.5	85.1	**0.0**	5.4	90.5	−0.3	4.2	92.1	−0.2	**3.7**	**93.7**
	HMMT24	53.3	+3.3	+3.3	+0.8	22.7	54.1	+0.3	18.6	63.4	+0.8	15.1	70.7	**+3.5**	**10.5**	**79.1**
	HMMT25	63.3	0.0	+6.7	+0.3	20.8	58.6	0.0	19.4	62.5	−1.3	17.0	66.6	**+3.2**	**10.2**	**80.3**
	MATH500	88.6	0.0	0.0	0.0	4.7	90.6	**0.0**	4.4	91.9	**0.0**	3.3	92.5	**0.0**	**3.2**	**94.7**
	GPQA	66.2	+1.0	+1.5	−0.5	11.3	78.4	−0.1	11.1	78.9	−0.6	9.6	80.9	**+0.2**	**6.5**	**87.9**
4B-Inst	AIME24	80.0	+3.3	+3.3	−1.5	20.0	61.4	−0.2	15.4	71.0	+2.0	13.1	75.0	**+3.0**	**7.1**	**87.2**
	AIME25	63.3	0.0	−3.3	0.0	26.2	48.2	**+0.2**	17.2	66.8	0.0	15.5	69.0	−2.7	**11.5**	**78.3**
	HMMT24	33.3	0.0	0.0	**+0.2**	30.8	46.6	0.0	22.0	62.8	0.0	19.5	65.8	**+0.2**	**13.2**	**77.2**
	HMMT25	33.3	+3.3	+10.0	+0.5	30.5	46.7	+0.2	21.9	61.9	+2.0	17.6	69.4	**+7.2**	**15.0**	**73.7**
	MATH500	89.2	+0.2	+0.4	0.0	6.2	79.0	0.0	5.8	83.8	−0.1	4.4	85.1	**+0.1**	**3.6**	**91.5**
	GPQA	58.1	−0.5	−1.5	−0.3	9.3	75.3	**0.0**	8.7	78.6	−1.4	6.0	86.4	−1.1	**4.6**	**90.3**
8B-NT	AIME24	46.7	0.0	0.0	**−0.2**	44.0	20.3	**−0.2**	33.3	37.9	−2.0	29.9	43.4	−2.5	**22.6**	**59.3**
	AIME25	33.3	0.0	0.0	−0.3	45.2	23.2	**0.0**	35.7	39.5	−3.3	30.1	50.1	−0.2	**25.4**	**57.0**
	HMMT24	20.0	0.0	+3.3	0.0	50.3	15.4	−0.2	40.9	29.6	−3.3	36.2	39.4	**+5.5**	**32.7**	**40.9**
	HMMT25	13.3	+3.3	+6.7	0.0	50.0	16.6	0.0	40.7	31.0	+1.7	30.5	**54.1**	**+7.2**	**29.9**	45.5
	MATH500	85.8	−0.4	−0.4	−0.4	11.8	63.2	−0.2	9.9	71.6	**−0.1**	8.7	72.5	−1.3	**5.3**	**86.7**
	GPQA	49.0	+0.5	+2.0	−0.4	16.2	68.9	**+0.1**	15.5	71.0	−1.5	10.2	80.6	−0.2	**7.8**	**85.6**
Gemma3-4B	AIME24	20.0	+3.3	+3.3	+0.3	44.2	30.1	0.0	37.9	39.9	−3.8	27.9	57.4	**+1.7**	**24.5**	**60.0**
	AIME25	13.3	+6.7	+6.7	+0.2	46.5	22.8	0.0	37.9	35.9	+3.2	33.8	43.1	**+6.0**	**27.9**	**51.8**
	HMMT24	3.3	+3.3	+3.3	0.0	39.0	34.3	0.0	29.5	49.1	+1.0	22.3	**64.2**	**+3.0**	**21.2**	62.8
	HMMT25	0.0	+3.3	+3.3	0.0	44.2	27.0	0.0	32.8	44.1	+3.3	**21.5**	**65.2**	**+3.7**	23.7	59.8
	MATH500	76.4	+0.6	+0.4	−0.1	15.8	60.0	**0.0**	12.9	67.9	−0.4	11.6	69.2	−0.8	**5.9**	**87.1**
	GPQA	33.3	−0.5	−1.5	−0.6	7.3	85.0	**−0.2**	7.2	85.6	−0.5	5.5	87.8	−2.0	**4.8**	**91.1**

fidence separates correct from incorrect solutions on Physics (AUROC 0.64) but not on recall-heavy Chemistry (0.50). Confidence reflects how certain a reasoning process is, so when an answer is recalled rather than derived there is no process to read: wrong answers carry the same confidence as correct ones, and the samples concentrate on a single wrong answer. On majority-wrong GPQA problems, that one wrong answer takes a median of 0.84 of the votes, against 0.47 on math (Table 9). Filtering then has nothing to separate, but that same convergence is all the stopping rule needs, so ***CASE*** trades the accuracy gain for savings on GPQA, at -2.0 to $+2.3$ pp per cell (Appendix D).

At the problem level, the accuracy gain takes the form of filter recovery. Stratifying all 4,090 model–problem pairs by correct-solution fraction p_c (Table 2; an oracle quantity rather than an inference-time statistic, though its distribution, the gain-regime share, is estimable on a small labeled slice) yields three regimes. In the *gain regime* (MV-wrong, $p_c > 0$; 13% of pairs), filter recovery rises monotonically from 3% to 39% across p_c bins, scaling the per-instance recovery of Figure 2(b) to a population. In the *savings regime* (MV-correct; 70% of pairs), ***CASE*** preserves 97.7% of MV-correct answers at 93% TokSv. The *unrecoverable regime* ($p_c = 0$; 17% of pairs) still saves 83% of tokens by terminating once consensus stalls. Filter recovery thus drives accuracy in the gain regime; stopping drives efficiency across all three.

Table 2: Per-problem ***CASE*** performance by correct-solution fraction (p_c) and MV outcome across 4,090 model–problem pairs. ***CASE*** values represent recovery rates in the Gain regime (MV-wrong, $p_c > 0$) and the preservation rate in the Savings regime (MV-correct).

	N (%)	MV	***CASE***	Samp	TokSv
Gain (MV-wrong, $p_c > 0$)					
$p_c \in (0, 10\%]$	258 (6.3%)	0%	3.4%	15.1	76.5%
$p_c \in (10, 30\%]$	206 (5.0%)	0%	18.6%	14.5	77.4%
$p_c \in (30, 50\%]$	68 (1.7%)	0%	38.5%	15.8	75.2%
Savings (MV-correct)	2868 (70.1%)	100%	97.7%	4.4	93.1%
Unrecoverable ($p_c = 0$)	690 (16.9%)	0%	0%	11.1	82.7%

Comparison with DeepConf. DeepConf (Fu et al., 2026) is the closest confidence-based method. We exclude it from Table 1 because its online mode terminates mid-generation, which requires modifying the serving framework's decoding loop. Repli-

cated on the same solutions and confidence scores, its best online variant reaches $+1.49$ pp at 53.1% token savings against ***CASE***'s $+2.24$ pp at 85.3%, on $3\times$ fewer samples and with no change to the inference engine. Appendix I reports the controlled comparison in full, including an offline-mode decomposition that localizes the difference to the filter axis; results at $N = 64$ may understate DeepConf, which was originally evaluated at $N = 512$.

4.3 Component Analysis

Having attributed ***CASE***'s behavior to problem regimes (§4.2), we now decompose the same aggregate gains into the contributions of $\mathcal{F}$, $\mathcal{W}$, and $\mathcal{S}$. To isolate each component's contribution, we systematically evaluate all eight combinations of Filter (F), Weight (W), and the base stopping rule (S; Eq. 2). Difficulty adaptation (D) extends S by adjusting its threshold via inter-z (§3.5); we therefore evaluate D only as an add-on to FWS (Table 3). Stopping configurations apply running normalization; non-stopping configurations apply offline normalization computed from the full distribution of $N = 64$ solutions per problem (the two cannot be decoupled: online stopping decides at step k based on only $\{c_1, \ldots, c_k\}$, so it cannot reuse offline statistics that include future solutions).

Table 3: Factorial ablation of F (filter), W (weight), and S (stopping) across all 8 combinations on 30 model–benchmark combinations ($N = 64$). Full ***CASE*** adds D: difficulty-aware stopping. ReASC shown for reference.

F	W	S	Acc%	Samp	TokSv	ΔMV
			49.98	64.0	0.0%	–
		✓	49.91	16.7	80.0%	−0.07
	✓		51.40	64.0	0.0%	+1.42
	✓	✓	51.00	14.7	82.4%	+1.02
✓			53.12	64.0	0.0%	+3.14
✓		✓	52.38	20.2	75.6%	+2.40
✓	✓		53.27	64.0	0.0%	+3.29
✓	✓	✓	52.45	16.5	80.6%	+2.47
CASE (FWS+D)			**52.22**	**13.6**	**85.3%**	**+2.24**
ReASC			50.10	17.4	76.8%	+0.12

Adding filter to stopping alone (S→FS) lifts Δ_{MV} by $+2.47$ pp (from -0.07 to $+2.40$), consistent with the recovery-rate pattern in Table 2. However, samples *increase* from 16.7 to 20.2 ($+3.5$): since $f_z = 0$ removes solutions below the running mean, filtering roughly halves the effective vote count, and the Beta posterior (Eq. 2) needs more samples to reach the stopping threshold.

Weighting compensates for this convergence cost. Adding weight (FS→FWS) reduces samples from 20.2 to 16.5 while Δ_{MV} moves from $+2.40$ to $+2.47$ pp. By concentrating vote mass on the highest-confidence survivors, weighting allows the Beta posterior to converge despite the smaller effective pool. The combined F+W+S thus improves accuracy at essentially no efficiency cost over stopping alone ($+2.47$ vs. -0.07 pp at 16.5 vs. 16.7 samples).

This decomposition also locates ***CASE***'s gain over prior early-stopping methods, which weight and stop but leave the vote pool unmodified (§4.1). Adding the filter to W+S buys $+1.45$ pp of accuracy ($+1.02$ to $+2.47$) at 1.8 extra samples and 1.8 pp of token savings (Table 3): most of ***CASE***'s accuracy gain comes from the axis this line does not occupy. ReASC itself further differs in its stopping and weighting constants and its reliability gate, so this reading bounds its gap to ***CASE*** rather than reproducing it in a controlled comparison.

Difficulty-aware stopping adds a final efficiency layer. Adding it (FWS→***CASE***) extracts $+4.7$ pp additional token savings ($80.6\% \rightarrow 85.3\%$) at a minimal 0.23 pp accuracy cost, yielding $+2.24$ pp over MV at 13.6 samples. The threshold δ is robust: accuracy varies by only 0.45 pp across $\delta \in [-2.0, -0.5]$ (Appendix G).

4.4 Intra-z vs. Inter-z: Role Validation

Filtering, weighting, and stopping rely on two normalization scales: intra-z (within-problem) for quality and inter-z (cross-problem) for difficulty. Each encodes information the other cannot recover: intra-z removes cross-problem variation so a single threshold applies uniformly across difficulty, while inter-z preserves it, distorting any within-problem cutoff. We test the scale–role pairing by independently swapping each of the two intra-z roles, filter and weight (Table 4).

Replacing intra-z with inter-z for the filter (swap-F) inflates sample usage from 13.6 to 18.8; TokSv drops from 85.3% to 73.2% and accuracy drops by 0.37 pp. Inter-z assigns uniformly low scores on hard problems by construction, so the filter rejects most candidates and forces ***CASE*** to continue sampling further. Intra-z avoids this by ranking solutions within each problem, maintaining a balanced pass rate regardless of difficulty.

Swap-W trades 2.8 extra samples ($13.6 \rightarrow 16.4$) for a marginal 0.24 pp accuracy gain. Inter-z weights shrink on hard problems (negative z), slowing Beta-posterior convergence and consuming

Table 4: Independent z-score swap ($N = 64$, 30 model–benchmark combinations). Each swap row replaces intra-z with inter-z for filter, weight, or both; ***CASE*** (intra/intra) shown for reference. See §4.4 for analysis.

Config	F / W	D	Acc%	Samp	TokSv	ΔMV
MV	–	–	49.98	64.0	0.0%	–
CASE	**intra / intra**	✓	**52.22**	**13.6**	**85.3%**	**+2.24**
swap-F	inter / intra	✓	51.85	18.8	73.2%	+1.87
swap-W	intra / inter	✓	52.46	16.4	81.2%	+2.48
swap-FW	inter / inter	✓	51.99	22.0	69.4%	+2.01
CASE	intra / intra	✗	52.45	16.5	80.6%	+2.47
swap-F	inter / intra	✗	51.92	20.7	71.0%	+1.94
swap-W	intra / inter	✗	52.61	19.6	75.7%	+2.63
swap-FW	inter / inter	✗	52.01	23.4	67.4%	+2.03

more samples there for marginal accuracy benefit at substantial efficiency cost. ***CASE*** therefore leverages intra-z for both filter and weight.

Swap-FW (22.0 samples) confirms the filter swap dominates the combined loss; beyond these direct costs, difficulty-aware stopping's efficacy is itself tied to the filter scale: it saves 2.9–3.2 samples when paired with intra-z filtering but only 1.4–1.9 with inter-z, because inter-z filtering rejects most candidates on hard problems, consuming most of the budget regardless of D. All three swaps incur efficiency cost: swap-F and swap-FW are strictly Pareto-dominated by ***CASE*** on all three metrics, and swap-W inflates samples by 21%.

4.5 Deployment Cost

CASE must compute a confidence score and evaluate its stopping rule after each solution is generated, so a problem's solutions are produced one after another rather than as one parallel batch, at a latency cost MV does not pay. The gap should be widest under modern serving, where continuous batching and prefix caching let MV issue all N solutions as a single saturated round. Yet ***CASE*** draws on the same two mechanisms: a server generates for many problems at once, so the GPU stays busy while any one problem waits on its decision, and prefix caching lets that problem's next solution rejoin the running batch with no re-prefill (Kwon et al., 2023). Because token savings cannot show how much of the gap survives, we measure wall-clock directly.

We run the 30 problems of a cell concurrently on one GPU against MV at the full budget (Table 5). ***CASE*** finishes in $0.84\times$ and $0.46\times$ MV's wall-clock while spending $0.28\times$ its tokens, at $+3.3$ pp accuracy: the token savings become time savings. The oracle is faster still ($0.36\times$ and $0.30\times$), but no deployment can occupy that point, since it must know each problem's sample count in advance, and it aggregates by majority vote, returning MV-level accuracy (30.00 and 40.00) rather than ***CASE***'s (36.67 and 50.00).

Table 5: Wall-clock, tokens, and accuracy under a served workload: the 30 problems of a cell are run concurrently on one GPU at $N = 64$, single replicate. Acc% is measured on this run and is not the 20-shuffle Acc% of Table 1. †The oracle spends ***CASE***'s token budget in one fully parallel pass; not a deployable schedule.

	4B-Inst/HMMT25			1.7B-T/AIME25		
	Time (s)	Tokens used	Acc%	Time (s)	Tokens used	Acc%
MV	8,469	13.7 M	33.33	39,311	34.4 M	46.67
CASE	7,134	3.87 M	**36.67**	18,077	9.47 M	**50.00**
Oracle†	3,090	3.87 M	30.00	11,799	9.70 M	40.00

Appendix J reports the harsher single-problem setting (Table 19), in which one problem owns one GPU and the serialization has nothing to hide behind. Even there the cost is adjustable: ***CASE*** costs $1.17\times$ and $0.72\times$ MV's median latency on the two cells, and checking the stopping rule once every $b = 4$ solutions brings both below MV ($0.61\times$ and $0.30\times$). Per-sample stopping therefore maximizes token efficiency while blocked stopping suits a latency target.

5 Conclusion

We propose ***CASE***, which extracts per-solution quality and per-problem difficulty from a single token-level confidence trace by normalizing it within and across problems, supporting filtering, weighting, and adaptive stopping without training or modifications to the standard serving pipeline. Across 30 model–benchmark combinations, ***CASE*** saves 85% of tokens while gaining $+2.24$ pp over majority voting, and is, to the best of our knowledge, the first to enter the upper-right region of the accuracy–efficiency plane beyond the prior Pareto frontier (Figure 1). Component analysis reveals filter–weight complementarity, and a role-swap experiment confirms the two normalization scales play non-interchangeable roles, each encoding information the other cannot recover. ***CASE*** illustrates how a single signal can serve multiple roles in self-consistency, a direction we expect future work to expand.

Limitations

Confidence-signal failure modes. ***CASE***'s accuracy gain hinges on the tail-confidence AUROC of the target distribution exceeding 0.5; otherwise the filter cannot separate correct from incorrect solutions and may preferentially retain incorrect ones (e.g., Gemma3-4B/GPQA: AUROC 0.46, -2.0 pp vs. MV; Tables 1 and 11). Verifying it requires labels, since the failure is that confidence carries no correctness signal, so we recommend measuring tail AUROC on a small labeled slice before deploying on a new model–domain combination. Filtering also offers little headroom on benchmarks where MV already exceeds $\sim 80\%$ (e.g., MATH-500 at $\sim 85\%$): filtering removes correct solutions whose tail confidence sits below the mean, yielding a small accuracy cost (-1.3 to $+0.1$ pp); see Table 2 for the flip-rate decomposition. Its largest shortfall across the grid is -2.7 pp (4B-Inst/AIME25; Table 1), on a cell where the signal is discriminative (AUROC 0.83): the precondition indicates where gains are possible, not a floor on per-cell loss. Separately, outside the gain regime ***CASE*** acts as an efficiency method, preserving 97.7% of MV-correct answers grid-wide (Table 2).

Deployment constraints. Several practical constraints arise. First, ***CASE***'s per-solution decisions require sequential (or streaming) generation; this is largely absorbed by continuous batching under a served workload, though a residual scheduling cost remains, and it is exposed when per-problem latency is the target, where a batched variant recovers it (Appendix J). On the few problems where stopping never fires ***CASE*** spends the full budget, so its worst-case latency exceeds full-budget MV. Second, ***CASE*** requires top-K token log-probabilities, available from open-weight serving frameworks and some proprietary APIs but not universally exposed. Third, the per-setting (μ_G, σ_G) used throughout this paper presupposes the problem set in advance; the pooled fallback trades ≈ 0.67 pp accuracy for ≈ 2 pp token savings, while a rolling estimate accumulated over the served stream removes the presupposition without that trade and is our recommended deployment default (Appendix E). Finally, our main grid covers math reasoning and graduate-level science with models up to 8B across two architecture families, and Appendix K extends it to a 14B reasoning model and to MMLU-Pro. Open-ended and agentic tasks remain out of scope because self-consistency requires a discrete answer to vote on. Adapting the warmup k_w to the stability of the running confidence variance, and a formal account of when token-level confidence tracks solution correctness, are left for future work.

Acknowledgments

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2026-RS-2023-00255968) grant funded by the Korea government (MSIT), IITP grant funded by the Korea government (MSIT) (No. RS-2022-II220680, Abductive inference framework using omni-data for understanding complex causal relations; RS-2026-25617480), the Nano & Material Technology Development Program through the National Research Foundation of Korea (NRF) funded by Ministry of Science and ICT (RS-2024-00444182), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2026-25518808). AI assistants were used for code development in the experimental pipeline, for LaTeX formatting, and for editing of the manuscript; all research design, experiments, analysis, and conclusions are the authors' own.

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. *arXiv 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.

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

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and

Martin Vechev. 2026. Beyond benchmarks: MathArena as an evaluation platform for mathematics with LLMs. *arXiv preprint arXiv:2605.00674*.

Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. 2026. Deep think with confidence. In *International Conference on Learning Representations*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

Zhewei Kang, Xuandong Zhao, and Dawn Song. 2025. Scalable best-of-N selection for large language models via self-certainty. In *Advances in Neural Information Processing Systems*.

Junseok Kim, Nakyeong Yang, Kyungmin Min, and Kyomin Jung. 2026. Reliability-aware adaptive self-consistency for efficient sampling in LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 21575–21590.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Joshua Ong Jun Leang, Zheng Zhao, Aryo Pradipta Gema, Sohee Yang, Wai-Chung Kwan, Xuanli He, Wenda Li, Pasquale Minervini, Eleonora Giunchiglia, and Shay B Cohen. 2026. PiCSAR: Probabilistic confidence selection and ranking for reasoning chains. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 31511–31544.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*.

Zhangyi Liu, Huaizhi Qu, Xiaowei Yin, He Sun, Yanjun Han, Tianlong Chen, and Zhun Deng. 2026. PETS: A principled framework towards optimal trajectory allocation for efficient test-time self-consistency. In *International Conference on Machine Learning*.

Michael Mitzenmacher and Rana Shahout. 2025. Queueing, predictions, and large language models: Challenges and open problems. *Stochastic Systems*, 15(3):195–219.

Ali Razghandi, Seyed Mohammad Hadi Hosseini, and Mahdieh Soleymani Baghshah. 2025. CER: Confidence enhanced reasoning in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7918–7938.

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

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*, pages 20090–20111.

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.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025. 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.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*.

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

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. *arXiv preprint arXiv:2505.09388*.

Yifan Zhang and Math-AI Team. 2024. American invitational mathematics examination (AIME) 2024. https://huggingface.co/datasets/math-ai/aime24.

Yifan Zhang and Math-AI Team. 2025. American invitational mathematics examination (AIME) 2025. https://huggingface.co/datasets/math-ai/aime25.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*.

Zhi Zhou, Yuhao Tan, Zenan Li, Yuan Yao, Lan-Zhe Guo, Yu-Feng Li, and Xiaoxing Ma. 2025. A theoretical study on bridging internal probability and self-consistency for LLM reasoning. In *Advances in Neural Information Processing Systems*.

A Component Taxonomy

Table 6 classifies self-consistency enhancement methods by their components. ESC and ASC are pure stopping methods; CISC (Confidence-Improved Self-Consistency) (Taubenfeld et al., 2025) and CER (Razghandi et al., 2025) improve quality but at full budget; ReASC adds confidence weighting without filtering; RPC (Reasoning Path Consistency) (Zhou et al., 2025) applies pruning and probability-weighted voting at full budget. Parenthesized checkmarks denote partial support: PETS stops by budget exhaustion rather than consensus, and DeepConf's online mode stops within token generation rather than after. ***CASE*** is the only method covering all four axes (filtering, weighting, stopping, and difficulty awareness) within a standard serving pipeline; the difficulty axis corresponds to the D layer that ***CASE*** places on top of its base stopping rule S (§3.5).

Table 6: Taxonomy of self-consistency enhancement methods. ***CASE*** is the only method that covers all four axes (filtering, weighting, stopping, and difficulty awareness).

Method	Filt.	Wt.	Stop	Diff.
ASC (Aggarwal et al., 2023)			✓	
ESC (Li et al., 2024)			✓	
CISC (Taubenfeld et al., 2025)		✓		
CER (Razghandi et al., 2025)		✓		
RASC (Wan et al., 2025)		✓	✓	
ReASC (Kim et al., 2026)		✓	✓	
RPC (Zhou et al., 2025)	✓	✓		
DSC (Wang et al., 2025)			✓	✓
PETS (Liu et al., 2026)			(✓)	✓
DeepConf (Fu et al., 2026)	✓	✓	(✓)	
CASE	✓	✓	✓	✓

B Sampling Parameters

Table 7 summarizes the sampling configuration: parameters follow each model family's recommended setup. Qwen3 thinking models adopt $t = 0.6$, $p = 0.95$; Qwen3 non-thinking models adopt $t = 0.7$, $p = 0.8$; Gemma adopts $t = 1.0$, $k = 64$. Group size for confidence computation is chosen based on typical response length: 2048 for thinking models (avg. 9K–18K tokens) and 256–512 for non-thinking models (avg. 1.6K–3K tokens). Phi-4-reasoning follows its own model card ($t = 0.8$, $p = 0.95$, $k = 50$), the same per-model practice, and takes the thinking group size of 2048. Solution sampling was performed with vLLM (Kwon et al., 2023) on NVIDIA H100 and RTX PRO 6000 Blackwell GPUs; confidence post-processing runs on CPU from cached top-K log-probabilities.

Table 7: Sampling parameters per model. All models are evaluated with $N = 64$ samples per problem and the top-20 log-probabilities per token (for confidence computation; distinct from the Top-k sampling parameter in the table). Grp denotes the confidence-aggregation group size g (§3.2), also distinct from sampling. Phi-4-reasoning, used only in Appendix K, is listed below the rule.

Model	Temp	Top-p	Top-k	Grp
Qwen3-1.7B (T)	0.6	0.95	20	2048
Qwen3-4B-Think	0.6	0.95	20	2048
Qwen3-4B-Inst	0.7	0.8	20	512
Qwen3-8B (NT)	0.7	0.8	20	512
Gemma3-4B-IT	1.0	0.95	64	256
Phi-4-reasoning (T)	0.8	0.95	50	2048

C Confidence Signal Design

Following Fu et al. (2026), ***CASE*** derives its confidence signal from the top-K token log-probabilities ($K = 20$), specifically their negated mean (§3.2), rather than from top-1 or full-vocabulary logits. Top-1 measures how strong the leading candidate is, but not how dominant it is relative to competitors: two tokens with top-1 probability 0.5 are treated identically whether the runner-up has probability 0.48 (near-tie, uncertain) or 0.05 (clear leader). The top-K average captures this gap through the runner-up ranks. Full-vocabulary logits contain the complete distribution, but (i) standard serving frameworks do not expose them by default (vLLM caps returned log-probabilities at `max_logprobs` = 20 (Kwon et al., 2023)), (ii) transmitting per-token distributions over a vocabulary of $\sim$150K inflates memory and I/O, and (iii) the long tail of near-zero-probability tokens is numerically noisy and contributes no useful signal. Top-K with $K = 20$ thus captures the informative head of the distribution while remaining compatible with standard serving.

Group aggregation. Token-level scores c_t are aggregated to a solution-level score c_k over groups of g consecutive tokens (the *group size*; stride 1): for each group, c_t values inside are averaged to produce a *group score*, yielding $T-g+1$ group scores per solution. Following Fu et al. (2026), four reduction strategies map these group scores to c_k: (i) *mean*, the average of all group scores (overall confidence); (ii) *bottom-10%*, the mean of

the lowest decile (worst-region focus); (iii) *lowest*, the minimum group score (single weakest region); and (iv) *tail*, the mean of the final group only (certainty as reasoning concludes). Each captures a different aspect of the reasoning trajectory; Table 8 compares their average AUROC, where *tail* achieves the highest discrimination and is selected as ***CASE***'s signal.

D Confidence Discrimination

For each (model, dataset) cell, we compute AUROC over all solutions in the cell (pooled across problems), with raw confidence as the score and correct/incorrect as the label. Table 8 averages these per-cell values over the 6 datasets per model, alongside Cohen's d (standardized mean difference between correct and incorrect solutions). Tail confidence achieves the highest average AUROC (0.79) and Cohen's d (1.21) and wins on 3/5 models (1.7B-Think, 8B-NT, Gemma3-4B); on 4B-Think and 4B-Inst, Lowest edges out Tail by 0.02 AUROC. 1.7B-Think exhibits the strongest discrimination ($d = 1.60$), consistent with its largest accuracy gain. The Tail vs. Bottom-10% gap is small in aggregate (0.79 vs. 0.78), so single-setup rankings reported elsewhere (Kim et al., 2026) can flip depending on model and dataset; the aggregate ordering, not any single cell, motivates our choice of Tail.

Table 8: AUROC and Cohen's d (correct vs. incorrect confidence) per metric, averaged across datasets. Best metric per model in **bold**. Higher values indicate stronger discrimination between correct and incorrect solutions.

Model	Tail AUC / d	Bottom10 AUC / d	Mean AUC / d	Lowest AUC / d
1.7B-Think	**0.84 / 1.60**	0.82 / 1.49	0.80 / 1.35	0.82 / 1.49
4B-Think	0.79 / 1.05	0.80 / 1.09	0.78 / 1.01	**0.81 / 1.15**
4B-Inst	0.76 / 1.11	0.77 / 1.11	0.74 / 0.99	**0.78 / 1.15**
8B-NT	**0.80 / 1.23**	0.77 / 1.08	0.77 / 1.04	0.77 / 1.10
Gemma3-4B	**0.75 / 1.06**	0.75 / 1.01	0.74 / 1.06	0.74 / 0.99
Avg	**0.79 / 1.21**	0.78 / 1.16	0.77 / 1.09	0.78 / 1.18

Table 11 breaks down AUROC by dataset for the tail metric. Discrimination varies substantially: competition benchmarks average 0.83–0.88, MATH-500 0.70, and GPQA only 0.58. This largely explains why filtering yields minimal accuracy change on GPQA. Although 45% of GPQA problems are MV-wrong with $p_c > 0$ (gain regime), the confidence score cannot reliably separate correct from incorrect solutions, so filtering removes both at similar rates. On MATH-500, intermediate discrimination combines with high base MV accuracy (85%), leaving little room for filtering to improve.

Why GPQA fails. The gap between 0.82 on math and 0.58 on GPQA is not a property of the confidence metric but of how the two domains fail. Tail confidence measures how certain the model is while writing a solution, so it separates correct from incorrect solutions only when errors leave a trace. A math error is a slip inside a derivation: samples slip at different points, wrong answers scatter, and the slip itself depresses confidence where it occurs. A GPQA error is more often a gap in recall, which leaves no derivation to disturb, so the model asserts a wrong fact with the certainty it brings to a correct one. Within GPQA the effect tracks content: discrimination is at chance on recall-heavy Chemistry (0.50) and recovers where reasoning contributes (Biology 0.54, Physics 0.64), against 0.86 on the AIME controls for the same five models.

The visible signature of this failure is concentration. On problems whose majority answer is wrong, the most frequent wrong answer takes a median 0.84 of the votes on GPQA against 0.47 on math (Table 9), so the pool agrees on one wrong option instead of scattering. Concentration is near-uniform across the three subjects (0.83 to 0.87) while AUROC is not, which separates the two mechanisms: the four-way choice set supplies one fluent wrong option for every subject, and what varies by subject is whether the confidence signal can tell it from the correct one. On two cells (Gemma3-4B and 4B-Inst) the AUROC sits at 0.46 and 0.50, confidence carries no usable correctness signal, and filtering can retain incorrect solutions preferentially.

Detecting the regime. Concentration is computable online, since it is what the consensus rule already tracks, but a high top-answer share marks confident agreement without distinguishing agreement on a correct answer from agreement on a wrong one, so the two regimes separate only under labels. We therefore recommend measuring tail AUROC on a small labeled slice before deploying on a new model–domain pair. The same concentration that defeats the filter is what the stopping rule needs, and ***CASE*** consequently trades the accuracy gain on GPQA for savings of 80–91% of its tokens, at a realized cost of -2.0 to $+2.3$ pp versus MV across the five cells (mean ≈ 0).

Table 9: Tail AUROC and wrong-answer concentration, averaged over the five models. Concentration is the median vote share of the most frequent wrong answer on majority-wrong problems.

	Tail AUROC	**Concentration**
Math (25 cells)	0.82	0.47
GPQA (5 cells)	0.58	0.84
Within GPQA, by subject		
Chemistry (recall-heavy)	0.50	0.84
Biology	0.54	0.87
Physics	0.64	0.83
AIME control	0.86	0.39

Table 10: Per-cell Spearman correlation between a problem's mean tail confidence and its correct-solution fraction p_c. Medians are over the 25 math cells and the 5 GPQA cells.

Model	AIME24	AIME25	HMMT24	HMMT25	MATH	GPQA
1.7B-Think	+0.84	+0.86	+0.58	+0.83	+0.50	+0.29
4B-Think	+0.52	+0.33	+0.36	+0.74	+0.22	+0.37
4B-Inst	+0.68	+0.83	+0.62	+0.87	+0.36	−0.01
8B-NT	+0.84	+0.57	+0.44	+0.41	+0.52	+0.18
Gemma3-4B	+0.50	+0.69	+0.38	−0.11	+0.60	−0.13
Median			**+0.57** (math)			+0.18

Confidence and problem difficulty. Difficulty-aware stopping (§3.5) presumes that a problem's mean confidence tracks how often the model solves it. Table 10 reports the per-cell Spearman correlation between a problem's mean tail confidence and its correct-solution fraction p_c. The median is $+0.57$ and the correlation is positive in 24 of the 25 math cells; on GPQA it collapses to a median of $+0.18$, positive in 3 of 5, for the same reason the quality signal does.

Table 11: Per-dataset AUROC for tail confidence (raw, unnormalized). Competition benchmarks show strong discrimination (0.83–0.88 avg), while GPQA averages only 0.58, explaining why ***CASE***'s filtering yields minimal accuracy change on GPQA despite a large fraction of gain-regime problems.

Model	AIME24	AIME25	HMMT24	HMMT25	MATH	GPQA
1.7B-Think	0.89	0.92	0.90	**0.93**	0.73	0.65
4B-Think	0.87	0.83	0.70	0.86	0.62	0.67
4B-Inst	0.85	0.83	0.85	0.88	0.64	0.50
8B-NT	0.86	0.87	0.86	0.84	0.73	0.63
Gemma3-4B	0.77	**0.93**	0.83	0.73	0.78	0.46
Avg	0.85	0.88	0.83	0.85	0.70	**0.58**

E Inter-Problem Calibration

CASE leverages per-setting inter-problem z-score normalization with (μ_G, σ_G) pre-computed from k_{cal} calibration solutions per problem (default $k_{\text{cal}} = 1$; §3.5). We ablate two questions: whether $k_{\text{cal}} = 1$ is sufficient, and how pooled calibration compares to per-setting (Table 12).

$k_{\text{cal}} = 1$ sufficient. Per-setting calibration is robust to k_{cal}: $k_{\text{cal}} = 1$ matches the full $k_{\text{cal}} = 64$ oracle to within 0.02 pp on accuracy and TokSv and within 0.1 samples (Table 12, bottom; the intermediate $k_{\text{cal}} \in \{3, 5\}$ coincide as well and are omitted). The saturation is mechanical: a shift in μ_G moves *all* inter-z scores by the same constant and thus preserves the relative difficulty ranking; only problems whose mean inter-z falls within a narrow band of $\delta = -0.7$ can change stopping behavior, and difficulty-aware stopping is itself activated only on this Hard subset. Quantitatively, the standard error of μ_G from $k_{\text{cal}} = 1$ across 30 problems is $\sigma_G/\sqrt{30} \approx 0.18\sigma_G$, so the calibration-induced band is ± 0.18 standard units, a small fraction of the relevant decision boundary. Reusing the calibration solution as each problem's first warmup sample thus incurs no extra cost.

Table 12: Calibration ablation ($N = 64$, 30 combos, 20 shuffles). TokSv is micro-averaged by token count, consistent with the main results. Top: granularity at $k_{\text{cal}} = 1$ (*Global* pools all 30 settings; *Per-model* pools each model's 6 datasets). Bottom: k_{cal} sweep at per-setting granularity. Intermediate values $k_{\text{cal}} \in \{3, 5\}$ also reach 52.20–52.21 % accuracy (within 0.02 pp of both endpoints) and are omitted for brevity.

Calibration	Acc%	ΔMV	Samp	TokSv
Global (all 30 pooled)	51.53	+1.54	10.9	87.4%
Per-model (6 datasets pooled)	51.59	+1.60	10.8	87.2%
Per-setting, $k_{\text{cal}} = 1$ (default)	52.22	+2.24	13.6	85.3%
Per-setting, $k_{\text{cal}} = 64$ (oracle)	52.22	+2.24	13.5	85.3%

Pooled calibration: an accuracy–efficiency trade-off. Replacing per-setting with pooled (μ_G, σ_G) is not strictly worse. Global and Per-model lose ≈0.67 pp accuracy but gain ≈2 pp in token savings (Global: +1.54 pp / 87.4% TokSv vs. Per-setting +2.24 pp / 85.3%; Per-model: +1.60 pp / 87.2%). Pooling is a fallback when neither per-setting nor rolling calibration is available, with the trade-off direction made explicit by the metric; for a served stream the rolling estimate described below is preferable, matching per-setting accuracy with no pre-computed statistics at all.

Mechanism. The trade-off stems from the wide per-setting (μ_G, σ_G) distribution (Table 14): μ_G spans 12.8–24.9 across the 30 settings (1.9×

spread) and σ_G spans 1.5–5.7 (3.8× spread). Pooling fits a single (μ_G, σ_G) to this heterogeneity. On low-μ_G settings, pooled inter-z is biased high, under-triggering τ_{low}; on high-μ_G settings, it is biased low, over-triggering τ_{low}. Over-triggered settings terminate long-token solutions earlier, yielding the $+2$ pp token gain; the ≈ -0.67 pp accuracy concession reflects the resulting premature stops on hard problems. Per-model does not recover per-setting performance because within-model dataset variation (e.g., 4B-Inst μ_G ranges 18.8 [HMMT]–24.9 [MATH-500]) is comparable to between-model variation.

Rolling calibration: the recommended deployment default. Both per-setting and pooled calibration presuppose that the problem distribution is available before inference, which a live stream does not provide, and that presupposition is removable. A *rolling* estimate accumulates (μ_G, σ_G) over the stream as it is served: problems arrive in random order within each evaluation setting, each problem uses only the statistics of the problems that arrived before it (each contributing its first solution's confidence, the same calibration value used throughout), and for the first M problems only difficulty-adaptive stopping is disabled while filtering, weighting, and consensus stopping remain active.

Table 13 shows that a rolling estimate matches pre-computed per-setting calibration under this protocol: at $M = 5$ it lands within 0.03 pp of per-setting ($+2.22$ vs. $+2.24$) at 0.3 extra samples. All rows replay the same pools with the paper's evaluation pipeline, so the per-setting and pooled rows coincide with the $+2.24$ of Table 3 and the Global $+1.54$ of Table 12. The ≈ 0.67 pp penalty that pooled calibration pays does not transfer, because pooling collapses per-dataset differences into one global scale whereas rolling re-learns the scale of the stream it is serving. We therefore recommend rolling calibration as the deployment default and retain per-setting as this paper's reference configuration.

Sensitivity to the warmup M. Accuracy is flat in M to within 0.14 pp, and what larger M costs is samples rather than accuracy. The warmup leaves the first M problems without difficulty-adaptive stopping, so its cost is confined to the smallest evaluation sets: $+2.4$ samples where $M = 20$ covers two-thirds of a 30-problem stream, and at most $+0.1$ where it covers a tenth or less. We use $M = 5$ as the default, which lands within 0.5 pp of per-setting calibration on all 30 cells.

Table 13: Rolling calibration ($N = 64$, 30 combos, 20 arrival orders). The arrival protocol and the warmup M are defined in the text; all rows replay the same pools with the paper's evaluation pipeline.

Calibration	Δ_{MV} (pp)	Samp
Per-dataset stream		
Per-setting (paper default)	+2.24	13.6
Rolling, $M = 5$ (default)	**+2.22**	**13.8**
Rolling, $M = 10$	+2.25	14.3
Rolling, $M = 20$	+2.36	15.2
Pooled global	+1.54	10.9
Fully mixed stream (all six datasets interleaved)		
Pooled global (pre-computed)	+1.54	10.9
Rolling (no pre-computed stats)	**+1.63**	**10.8**

Mixed streams and isolated problems. On a fully mixed, non-stationary stream with all six datasets interleaved, per-setting calibration is not even definable, since a problem's dataset is unknown at arrival. The deployable comparison is then pooled versus rolling, and rolling needs zero pre-computed statistics while matching pooled calibration on both accuracy and sample cost ($+1.63$ vs. $+1.54$ pp at ≈ 11 samples). For a single isolated problem ***CASE*** still samples a pool, so filtering, weighting, and stopping all operate within it and need no stream; only difficulty-adaptive stopping, which ranks a problem against others, switches off, and the remaining three components still deliver $+2.47$ pp (the FWS row of Table 3).

Table 14: Inter-problem calibration statistics (μ_G/σ_G) for tail confidence, computed from the first solution per problem ($k_{\text{cal}}=1$, 20 shuffles). AIME and HMMT average their 2024/2025 editions.

Model	AIME	HMMT	MATH500	GPQA
1.7B-Think	16.0/2.6	15.3/2.5	19.4/2.6	13.6/1.8
4B-Think	17.7/1.5	17.0/2.1	19.2/2.4	14.1/2.0
4B-Inst	21.3/4.2	18.8/4.4	24.9/3.7	20.4/5.4
8B-NT	16.5/2.8	15.2/2.8	20.0/3.2	16.9/5.0
Gemma3-4B	16.8/3.9	15.6/3.4	23.5/5.7	12.8/3.0

F Shuffle Variance

Each method is evaluated on the same precomputed pool of $N = 64$ solutions per problem, replayed in 20 independent random orders (§4.1). Table 15 reports the standard error of per-cell means across these 20 orderings (SE $=$ std$/\sqrt{20}$), averaged across the 30 combinations with the per-cell maxi-

mum in parentheses. All four methods yield precise per-cell means: maximum SE is 0.78 pp accuracy, 0.95 samples, and 1.81 pp TokSv. ***CASE***'s per-cell accuracy wins in Table 1 (typically 2–7 pp on competition cells) exceed its SE by 3–10×, confirming the observed gains are not artifacts of shuffle order. The slightly higher accuracy SE for ***CASE*** (0.40 pp mean vs. 0.09 pp for ASC) reflects its additional decision points (filter, weight, difficulty-aware threshold) rather than higher underlying variance: ***CASE***'s signal-to-noise ratio remains favorable because its Δ_{MV} magnitude (+2.24 pp aggregate) is substantially larger. SE, not per-shuffle standard deviation, is the relevant precision for the Table 1 means; per-shuffle std is larger by $\sqrt{20} \approx 4.5\times$.

Table 15: Per-method standard error across 20 shuffles (30 cells; SE = std/$\sqrt{20}$). Values are "mean (max)" across cells.

Method	Acc (pp)	Samp	TokSv (pp)
ESC	0.16 (0.41)	0.35 (0.68)	0.67 (1.20)
ASC	0.09 (0.34)	0.28 (0.63)	0.54 (0.93)
ReASC	0.32 (0.76)	0.37 (0.95)	0.75 (1.81)
CASE	0.40 (0.78)	0.34 (0.71)	0.69 (1.33)

G Hyperparameter Sensitivity

Table 16 sweeps each of the six hyperparameters univariately around its default. Across every swept value of every parameter, $\Delta_{\text{MV}} \geq 0$ within rounding (the lone exception, $\lambda = 1.0$, sits at −0.02 pp), confirming ***CASE***'s accuracy gain is not contingent on the specific values chosen; the defaults additionally sit on or near the Pareto-efficient end of every frontier. The largest available improvement over the default is +0.45 pp ($\tau = 0.99$, at the cost of +8.1 samples), while downward excursions trace more efficient operating points along each frontier. The analyses below characterize each HP and report the rationale for its default value.

k_w **(warmup).** The running variance requires at least two samples to be defined ($k_w \geq 2$); we sweep $k_w \in [2, 5]$. Reducing k_w from default 3 to 2 costs 0.48 pp at −0.8 samples, reflecting the noisier intra-z from a 2-sample variance. Increasing to 4 or 5 contributes +0.11 pp/+0.08 pp at +0.5/+0.8 samples. Default $k_w = 3$ is the smallest value with stable statistics.

f_z **(filter threshold).** Default $f_z = 0$ is the centered cutoff of the intra-z distribution. $f_z = -0.5$ admits noisier solutions (−0.48 pp at −0.8 samples); $f_z = 0.5$ tightens the filter for +0.06 pp at +1.3 samples (effectively at the same Pareto point as default). $f_z = 1.0$ over-rejects: −0.35 pp accuracy and +2.5 samples vs. $f_z = 0.5$, strictly dominated.

λ **(weight scale).** Default $\lambda = 0.3$ moderately concentrates vote mass on high-confidence solutions: large enough to accelerate the Beta posterior over uniform voting ($\lambda = 0$), but well clear of the $\lambda = 1.0$ collapse where vote mass converges on a single solution and accuracy drops to MV-level. The value sits within ReASC's reported $[0.1, 0.9]$; our sweep extends to $[0, 1.0]$ to characterize the full Pareto frontier. $\lambda = 0.1$ Pareto-dominates uniform voting ($\lambda = 0.0$): +0.19 pp accuracy and −1.2 samples. Increasing λ trades accuracy for sample efficiency monotonically (+2.44 → +2.24 → +1.62 → +0.82 pp at $\lambda = 0.1, 0.3, 0.5, 0.7$); at $\lambda = 1.0$ vote concentration on a single high-confidence solution collapses voting to MV-level (−0.02 pp).

τ **(stopping threshold).** Default $\tau = 0.90$ relaxes ASC/ReASC's 0.95 since filtering improves the pool quality entering the Beta posterior, allowing earlier stopping at the same evidence level. Monotone trade-off across $[0.80, 0.99]$ with diminishing returns: 0.80→0.90 adds 0.93 pp accuracy at +4.8 samples, but 0.95→0.99 adds only 0.18 pp at +4.6 samples. Default $\tau = 0.90$ sits at the knee. At the inherited reference $\tau = 0.95$, ***CASE*** reaches +2.51 pp / 17.1 samples while ASC reaches only −0.04 pp / 21.1 samples, indicating filter-improved pool quality (not the threshold) drives the gain.

τ_{low} **(relaxed stopping threshold).** Monotone trade-off across $[0.55, 0.90]$: $\tau_{\text{low}} = 0.90$ matches non-relaxed accuracy (+2.47 pp) at +2.9 samples. Default $\tau_{\text{low}} = 0.70$ (midpoint of coin-flip 0.5 and standard $\tau = 0.90$) trades 0.23 pp for those samples.

δ **(difficulty threshold).** Monotone trade-off across $[-2.0, -0.5]$: tightening the Hard criterion (lower δ) classifies fewer problems as Hard, raising accuracy (+2.46 pp at $\delta = -2.0$) at +2.8 samples. Default $\delta = -0.7$ (≈ 24th percentile under a standard Gaussian) is the efficient anchor.

$\lambda \times \tau$ **interaction.** λ and τ have the largest individual effects on accuracy and efficiency, so we also sweep them together (Figure 3) to check for joint interactions. The joint sweep mirrors the individual trends, ranging from accuracy-favored ($\lambda = 0.1, \tau = 0.99$) at 52.69% / 24.6 samples to efficiency-favored ($\lambda = 0.5, \tau = 0.80$) at 50.47% /

6.8 samples, with the default ($\lambda = 0.3, \tau = 0.90$) at the best balance (52.22% / 13.6 samples). Only four cells fall below MV, all at high λ (down to -0.91 pp at $\lambda = 1.0$, $\tau = 0.80$); no joint interactions appear beyond the individual trends.

Table 16: Hyperparameter sensitivity ($N = 64$, 30 combos). Each row varies one HP while fixing others at default. **Bold**: default configuration.

HP	Value	Acc%	ΔMV	Samp	TokSv
Warmup k_w (mathematical min: $k_w \geq 2$)					
	2	51.74	+1.76	12.8	86.0%
	3	**52.22**	**+2.24**	**13.6**	**85.3%**
	4	52.33	+2.35	14.1	84.7%
	5	52.31	+2.32	14.4	84.2%
Filter threshold f_z					
	−0.5	51.74	+1.76	12.8	86.0%
	0.0	**52.22**	**+2.24**	**13.6**	**85.3%**
	0.5	52.28	+2.30	14.9	83.9%
	1.0	51.93	+1.95	17.4	80.9%
Weight scale λ					
	0.0	52.23	+2.25	17.4	80.6%
	0.1	52.42	+2.44	16.2	82.2%
	0.3	**52.22**	**+2.24**	**13.6**	**85.3%**
	0.5	51.60	+1.62	11.0	88.0%
	0.7	50.80	+0.82	8.9	90.0%
	1.0	49.96	−0.02	6.6	91.8%
Stopping threshold τ					
	0.80	51.29	+1.31	8.8	90.6%
	0.85	51.54	+1.56	10.3	89.3%
	0.90	**52.22**	**+2.24**	**13.6**	**85.3%**
	0.95	52.50	+2.51	17.1	80.8%
	0.99	52.67	+2.69	21.7	74.8%
Hard-problem threshold τ_{low} ($\delta = -0.7$ fixed)					
	0.55	51.78	+1.79	12.2	87.1%
	0.65	52.01	+2.02	12.7	86.5%
	0.70	**52.22**	**+2.24**	**13.6**	**85.3%**
	0.80	52.40	+2.42	14.7	83.5%
	0.90	52.45	+2.47	16.5	80.6%
Difficulty threshold δ ($\tau_{low} = 0.70$ fixed)					
	−2.0	52.44	+2.46	16.4	80.9%
	−1.0	52.36	+2.38	15.0	83.7%
	−0.7	**52.22**	**+2.24**	**13.6**	**85.3%**
	−0.5	51.99	+2.01	12.4	86.5%

H Budget Sensitivity

The main analyses fix the budget at $N = 64$; we now verify that ***CASE***'s advantage holds across budgets. ***CASE*** improves over MV at all sampling budgets, with the gap widening as N increases (+1.39 pp at $N = 16$ vs. +2.24 pp at $N = 64$; Table 17). Even at $N = 16$, ***CASE*** retains a +1.39 pp gain over MV while consuming only 45% of the budget. ***CASE*** at $N = 64$ consumes only 13.6 samples on average (fewer than MV at $N = 16$) while exceeding MV's accuracy at $N = 64$, demonstrating that confidence-aware early stopping can outperform simply increasing the sampling budget. The growing advantage is consistent with quality-based filtering benefiting from larger pools; ***CASE*** additionally benefits from more reliable running statistics and Beta-posterior estimates.

Table 17: Budget sensitivity (30 combos). WMV and F-WMV are shown at full budget for reference. ***CASE*** improves accuracy and reduces cost at all budgets.

N	Method	Acc%	ΔMV	Samp	TokSv
16	MV	49.27	–	16.0	–
	WMV	50.59	+1.32	16.0	0.0%
	F-WMV	51.56	+2.29	16.0	0.0%
	ESC	49.26	−0.00	9.9	45.1%
	ASC	49.27	+0.00	9.4	47.9%
	ReASC	49.31	+0.04	7.7	54.6%
	CASE	**50.66**	**+1.39**	**7.2**	**62.4%**
32	MV	49.67	–	32.0	–
	WMV	51.01	+1.34	32.0	0.0%
	F-WMV	52.53	+2.86	32.0	0.0%
	ESC	49.63	−0.04	16.2	58.2%
	ASC	49.65	−0.02	14.2	62.8%
	ReASC	49.66	−0.01	11.7	67.4%
	CASE	**51.60**	**+1.93**	**10.0**	**76.2%**
64	MV	49.98	–	64.0	–
	WMV	51.40	+1.42	64.0	0.0%
	F-WMV	53.27	+3.29	64.0	0.0%
	ESC	49.92	−0.06	26.3	68.1%
	ASC	49.94	−0.04	21.1	73.7%
	ReASC	50.10	+0.12	17.4	76.8%
	CASE	**52.22**	**+2.24**	**13.6**	**85.3%**

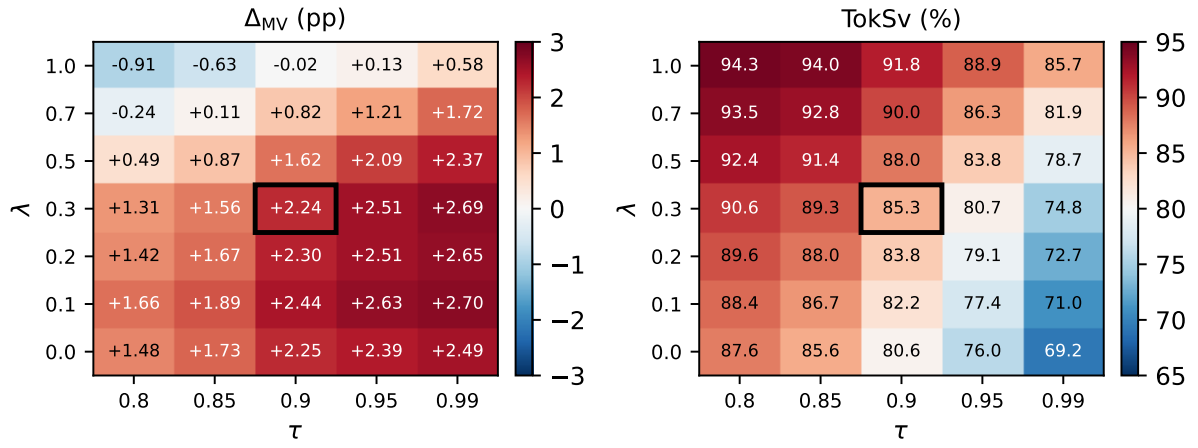


Figure 3: $\lambda \times \tau$ joint sensitivity ($N = 64$, 30 combos). Default ($\lambda = 0.3$, $\tau = 0.90$) marked by black box. Both panels share the RdBu colormap (red = higher, blue = lower). Left: Δ_{MV} (pp). Right: TokSv (%).

I DeepConf Comparison

We replicate DeepConf (Fu et al., 2026) on the same precomputed solutions and confidence scores (Table 18). ***CASE*** and DeepConf differ along three axes; we control for as many of them as each protocol allows. **(i) Confidence metric.** DeepConf's online mode is architecturally locked to the lowest sliding-window score: mid-generation termination evaluates confidence window-by-window during decoding and cannot reference a tail or full-sequence statistic that does not yet exist. In the offline setting, where the full solution is available, we report DeepConf with the same tail confidence as ***CASE*** for a controlled comparison. **(ii) Normalization.** DeepConf leverages raw confidence in both modes; ***CASE*** leverages intra-z (within-problem). **(iii) Termination mechanism.** Mid-generation (DeepConf online) vs. post-solution (DeepConf offline, ***CASE***).

Online setting: serving compatibility. DeepConf's online mode tracks the lowest group confidence across sliding windows within each solution. A warmup of $W = 8$ solutions sets an absolute threshold $\bar{c}$ from the $(1-\eta)$-th percentile of the warmup lowest-group confidences; subsequent solutions are terminated mid-generation as soon as any window's group confidence falls below $\bar{c}$. Survivors contribute weighted votes with weight equal to their lowest group confidence, and consensus stopping ($V(\hat{a})/\Sigma V \geq 0.95$) stops sampling once one answer dominates. Because mid-generation termination evaluates confidence at every decoding step and forces an end-of-sequence token, it requires modifying the serving framework's decoding loop. ***CASE*** instead processes logprobs post-hoc after each complete solution, requiring no changes to the inference engine. In performance terms, DeepConf's best online variant achieves $+1.49$ pp with 53.1% token savings (42.3 samples), while ***CASE*** reaches $+2.24$ pp with 85.3% token savings (13.6 samples), consuming $3\times$ fewer samples while achieving higher accuracy.

Offline setting: filter is the dominant axis. DeepConf's offline mode weights solutions by raw tail confidence and optionally filters by per-problem percentile before voting. Comparing the no-filter WMV baseline ($+0.90$ pp) to the best filtered variant (top 50%, $+2.82$ pp) shows that filtering contributes $+1.92$ pp to DeepConf's offline gains, whereas pure weighting contributes only $+0.90$ pp. Filtering is the dominant axis even within DeepConf's own design.

F-WMV exceeds DeepConf's best by 0.47 pp. F-WMV reaches $+3.29$ pp, a further 0.47 pp over DeepConf's best filtered variant. The factorial ablation (Table 3) localizes this gap to the filter axis: intra-z filtering alone ($z^{\text{intra}} \geq 0$, no weighting) already reaches $+3.14$ pp, exceeding DeepConf's best combined filter-and-weight configuration. Adding exponential weighting on top of F-only contributes only an additional $+0.15$ pp.

Where the 0.47 pp comes from. Both schemes normalize the filter threshold within each problem (DeepConf by per-problem rank, F-WMV by per-problem z-score), so the gap is not attributable to the absence of within-problem normalization in DeepConf. The advantage instead arises from two properties of the continuous z-score formulation:

it preserves confidence magnitude beyond ordinal rank, and it lets the pass rate adapt to each problem's distribution rather than enforcing a uniform top-$X\%$ cut.

Table 18: DeepConf comparison ($N = 64$, 30 combos, 20 shuffles). "No filter, WMV only" is DeepConf's weighted voting without any filtering, included as a reference for isolating filter vs. weight contributions.

	Method	Acc%	ΔMV	Samp	TokSv
	MV	49.98	–	64.0	0.0%
Offline	DeepConf (no filter, WMV only)	50.88	+0.90	64.0	0.0%
Offline	DeepConf (top 50%)	52.81	+2.82	64.0	0.0%
Offline	DeepConf (top 90%)	51.13	+1.14	64.0	0.0%
Offline	DeepConf (top 10%)	51.89	+1.91	64.0	0.0%
Offline	**F-WMV (intra-z norm)**	**53.27**	**+3.29**	64.0	0.0%
Online	DeepConf ($\bar{c} = P_{90}$, more filtering)	51.48	+1.49	42.3	53.1%
Online	DeepConf ($\bar{c} = P_{10}$, less filtering)	50.81	+0.82	40.6	35.3%
Online	***CASE***	**52.22**	**+2.24**	**13.6**	**85.3%**

J Wall-Clock and Deployment Cost

Token savings and wall-clock savings are different quantities, and the second is what a deployment pays. We measure both shapes in which ***CASE*** can be run, expressing every measurement as a multiple of MV run fully in parallel at the full $N = 64$ budget. All timings use vLLM on one NVIDIA RTX PRO 6000 Blackwell 96 GB GPU with the decoding, budget, and hyperparameters of the main grid, on the two cells 4B-Inst/HMMT25 and 1.7B-Think/AIME25.

Two caveats apply throughout. Each configuration was run once, so these are single-replicate timings and carry no standard error, unlike the accuracy results of Appendix F. And because the timed runs generate fresh samples rather than replaying the fixed pool of §4.1, their token savings differ by a few points from Table 1, and their Acc% is not comparable to it either, since the accuracy results elsewhere in the paper average 20 shuffles over the fixed pool. The Acc% in Table 5 identifies which answers each schedule returns and is not a second estimate of accuracy.

Where the serialization cost comes from. ***CASE*** calculates a confidence score for a solution and decides whether to stop only after that solution is complete, so a problem's solutions are submitted one at a time while MV submits all N at once. Two properties of modern serving decide how much this costs. Continuous batching keeps the GPU saturated as long as enough sequences are in flight, and prefix caching removes the prompt cost of every solution after the first; MV exploits both in a single round. ***CASE*** keeps both, since a problem's next solution rejoins the running batch with no re-prefill (Kwon et al., 2023), and gives up only concurrency width: it holds at most one running sequence per unfinished problem, whereas MV is bounded only by KV capacity. The residual gap in Table 5 is therefore a property of the request policy rather than of the token count, since ***CASE***'s concurrency decays as problems stop while a fully parallel pass refills its slots from a queue (Mitzenmacher and Shahout, 2025).

Many problems served at once. Both methods share the engine, the prompts, and prefix caching, and differ only in how solutions are submitted. MV submits every solution of every problem up front. ***CASE*** submits one solution per problem; when a solution finishes, the stopping rule runs on CPU and, if the problem continues, its next solution is queued immediately, with no round barrier and no re-prefill.

We leave MV uncapped so that it runs at its best. Capping it at 30 concurrent sequences to match ***CASE*** slows it on 4B-Inst ($1.26\times$) but leaves it unchanged on 1.7B-Think ($0.99\times$), whose solutions average ≈18K tokens against ≈7K and so leave room for fewer concurrent sequences even uncapped.

Against this uncapped baseline ***CASE*** finishes the workload in $0.84\times$ and $0.46\times$ MV's wall-clock while spending $0.28\times$ its tokens. The oracle finishes in $0.36\times$ and $0.30\times$: with one GPU per *solution*, a single fully parallel round is a schedule no per-solution method can beat. It is not a deployable point, however, since it must know each problem's sample count before the run, and because it aggregates by plain majority vote it returns MV-level accuracy (30.00 and 40.00) rather than ***CASE***'s (36.67 and 50.00).

A single problem served alone. Table 19 removes the cover that other problems provide: one problem owns one GPU, so within-problem serialization has nothing to hide behind and continuous batching can no longer absorb it. Sample-by-sample stopping costs $1.17\times$ and $0.72\times$ MV's median per-problem latency, the schedule that ESC, ASC, and ReASC all share. The stopping rule itself is cheap: it runs on CPU between generations at 0.17% and 0.27% of wall-clock, and the two sequential rows are the same generations replayed with log-probabilities on versus off, a paired median gap of $+2.6\%$ and $+5.4\%$.

Table 19: A single problem served alone (median s/problem), all modes under identical contention, single replicate. Ratios are to parallel MV at the full budget. †Oracle issues the sample count ***CASE*** actually consumed as a single parallel round; batched ***CASE*** checks stopping only every $b=4$ solutions.

Mode	4B-Inst/HMMT25	1.7B-T/AIME25
Parallel MV	338 (1.00×)	1,323 (1.00×)
Oracle, same samples†	99 (0.29×)	330 (0.25×)
Sequential, no logprobs	385 (1.14×)	903 (0.68×)
Sequential ***CASE***	396 (1.17×)	950 (0.72×)
Batched ***CASE*** ($b=4$)	207 (0.61×)	402 (0.30×)
CASE TokSv / $b=4$ TokSv	78.5% / 67.7%	63.8% / 64.7%

The serialization is not inherent, because the stopping rule can run at any granularity. Batched ***CASE*** generates $b=4$ solutions in parallel per problem and checks stopping only at block boundaries; this cuts latency to $0.61\times$ and $0.30\times$, below parallel MV on both cells. Blocking changes only when the stopping rule is checked, not which solutions are filtered or how they are weighted, so relative to per-sample stopping it can only defer the stop. Blocked stopping overshoots slightly on 4B-Inst, where token savings fall from 78.5% to 67.7%, but not on 1.7B-Think, where they rise from 63.8% to 64.7%.

K Generalization Beyond the Main Grid

The main grid spans 1.7B–8B models on math and graduate-level science. Two additional testbeds probe the boundaries of that grid: a larger model and a different domain. In both, what replicates is the paper's rule for *when* gains appear (§4.2), not a uniform gain. Both reuse the pipeline and the ***CASE*** implementation of the main tables with the six hyperparameters at their fixed values (§4.1); no value is retuned for either. Unlike these two testbeds, which generate new solutions, the analyses of Appendices D and E run no new inference: both replay the same cached $N=64$ pools as the main tables and differ only in what is computed from them.

A larger model. Phi-4-reasoning 14B (Abdin et al., 2025) sits above the grid's 8B ceiling. Evaluated on all four competition cells with the same six fixed hyperparameters (Table 20), ***CASE*** gains where the gain regime exists ($+4.2$ pp on HMMT25) and trades at most 1.5 pp for around 78% savings on the two cells where MV is already saturated (83–90%). The gain regime is a necessary condition rather than a guarantee: at comparable regime size the size of the gain tracks the discriminativeness of the confidence signal ($+4.2$ against $+0.3$ pp across the two HMMT cells, at tail AUROC 0.66 against 0.55), so the regime share tells a practitioner where to expect a gain and the AUROC how large it will be.

Table 20: Phi-4-reasoning 14B, above the main grid's 8B ceiling, with the same six fixed hyperparameters. †Gain regime: the share of problems that are MV-wrong with the correct answer present in the pool ($p_c > 0$; §4.2).

Cell	MV	Gain†	Δ_{MV}	Samp	TokSv
HMMT25	60.0	20%	**+4.2**	14.9	73%
HMMT24	53.3	20%	+0.3	16.8	68%
AIME24 *(sat.)*	90.0	3%	−1.3	8.0	78%
AIME25 *(sat.)*	83.3	10%	−1.5	9.7	78%

Table 21: MMLU-Pro (14 categories × 20 problems, two models, same six fixed hyperparameters). Gain regime is defined as in Table 20; the last two rows are defined in the text, the top-answer share as a median over the 14 per-category medians.

MMLU-Pro	4B-Inst	1.7B-Think
Pooled tail AUROC (precondition)	0.61	0.77
CASE Δ_{MV}	−0.1 pp	−1.0 pp
TokSv	88%	84%
MV-correct preservation	98.8%	96.4%
Gain regime	3%	25%
Top wrong-answer share	1.00	0.73

A different domain. MMLU-Pro (Wang et al., 2024) (14 categories × 20 problems, two models) tests a broader knowledge domain (Table 21). The precondition passes (pooled tail AUROC 0.61 and 0.77) yet there is no accuracy gain (-0.1 and -1.0 pp), which is what the regime analysis predicts: what is missing is not the signal but the room to recover.

For 4B-Inst the gain regime is nearly empty (3%), so there is nothing to convert. For 1.7B-Think the nominal regime is larger (25%), but its wrong pools concentrate on one answer (median top-answer share 0.73, and 1.00 for 4B-Inst), so even a discriminative signal has little to invert. What the precondition does deliver is the savings profile, which transfers intact: 98.8% and 96.4% of MV-correct answers preserved at 88% and 84% token savings. Measured on a small labeled slice, the gain-regime share thus predicts where ***CASE*** adds accuracy and where it roughly breaks even at a large token saving.