Intrinsic Selection and Particle Resampling for Inference-Time Scaling Beyond Domain Verifiability

Giorgio Giannone[1]* **Mustafa Eyceoz**[1] **Shabana Baig**[1]
Shivchander Sudalairaj[1] **Anna C. Doris**[2] **Faez Ahmed**[2] **Akash Srivastava**[1,3] **Kai Xu**[1]
[1]Red Hat [2]MIT [3]IBM

Abstract

Inference-Time Scaling (ITS) has largely succeeded in verifiable domains like math and coding, where cheap verification enables scalable output selection. However, extending ITS to tasks prone to systematic failure - driven by faulty initial assumptions or unmet multidimensional constraints - typically relies on costly external solvers or brittle, model-based verifiers. Our key insight is that the intrinsic statistics of parallel sample sets, specifically length-adjusted tail entropy, provide a robust discriminative signal for solution quality without access to ground truth. Crucially, these statistics serve as a difficulty gate for adaptive compute allocation, dynamically routing problems across scaling regimes. First, *Intrinsic Selection* (`iS`) ranks candidates post hoc. It is competitive with consensus-based algorithms across three domains and improves engineering-design selection by 18% in median IoU over a random pick from the same candidate pool. Second, *Intrinsic Particle Filtering* (`iPF`) generalizes this to step-level resampling, guiding generation toward high-confidence reasoning trajectories to improve pass@1 by 6.1 points on average on hard math problems. Finally, *Particle Distillation* (`dPF`) injects privileged guidance via early logit blending and KL-guided resampling, steering generation past systematic reasoning errors to satisfy expert rubrics, yielding up to 26.5% gains on complex clinical responses. Our pipeline applies across broad-purpose, domain-specialized, and multimodal architectures, successfully extending ITS to open-ended domains without requiring trained reward models or exact ground-truth verification.

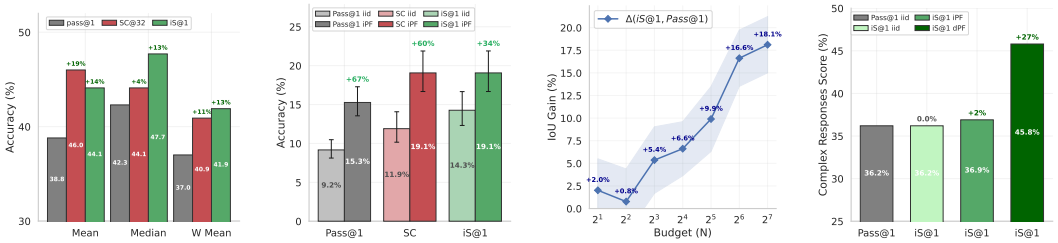
1 Introduction

Recent advances in reasoning and agentic foundation models [Guo et al., 2025, Jaech et al., 2024, Team et al., 2025b] have produced dramatic performance gains. These successes are most prominent in verifiable domains like mathematics and coding, where computationally inexpensive, exact verification, such as symbolic parsing, unit testing, and output matching, enables reinforcement-based tuning at scale [Lambert et al., 2024, Wen et al., 2025]. In these structured settings, building reward models and test-driven verification pipelines is highly feasible, creating a clear path to building high-performing workflows through increased inference-time compute [Snell et al., 2024, Zhang et al., 2025a].

Leveraging this compute, Inference-Time Scaling (ITS) techniques combine parallel and sequential sampling to rank and select promising candidate solutions [Brown et al., 2024, Wu et al., 2025, Wang et al., 2022, Snell et al., 2024]. While ITS has proven remarkably effective when proxy verification is readily available, extending it beyond these domains exposes a fundamental challenge. Existing methods rely heavily on extrinsic metrics, such as reward models, output standardization,

*Correspondence to: `ggiorgio@mit.edu`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) iS vs SC across math, reasoning, and coding domains over seven datasets.

(b) iPF vs iid sampling on hard to solve AIME problems.

(c) OOD Image-to-CAD selection on Fusion360 (IoU).

(d) dPF on HealthBench-Hard complex responses.

Figure 1: Intrinsic inference scaling across domains. **(a)** `iS` is competitive with self-consistency without output verification. **(b)** `iPF` improves the expected accuracy of a random final particle on hard AIME problems. **(c)** On CAD, `iS` improves median IoU over a random pick from the same pool. **(d)** `dPF` improves complex clinical responses through rubric guidance.

or step-level scoring, to rank, select, and steer generation. For domains where external verification is expensive or unavailable at scale, progress has been severely limited [Tang et al., 2025].

The Bottleneck of Extrinsic Verification Relying on extrinsic metrics creates a severe bottleneck: consensus methods fail in open-ended settings [Wang et al., 2022, Chen et al., 2023], reward models are expensive and rarely available [Zhang et al., 2025c, Stiennon et al., 2020], and heuristic parsers or solvers are often fragile and slow (Appendix B, Figure 8). Thus, we ask: *can we guide inference-time scaling using only the intrinsic statistics of generated sample sets, sidestepping external verifiers entirely?*

Our key observation is that parallel sample sets inherently carry strong discriminative signals - specifically, per-token tail entropy and overall length distributions - that encode solution quality and problem difficulty without ground-truth access. By analyzing the proportion of correct solutions n within N samples, we structure our approach around three challenge regimes: identifying the best candidate in *solved-easy* settings (n/N is sizable), concentrating compute on sparse correct trajectories in *solved-hard* tasks ($n/N \ll 1$), and correcting faulty early assumptions to navigate spaces of *systematic failure* ($n/N \approx 0$).

Contributions We propose a framework for inference-time scaling that *bypasses traditional verifiers and process reward models*. It comprises three complementary methods, each addressing a different challenge regime:

- *Intrinsic Selection* (`iS`): ranks candidates using set-level length distributions and adjusted per-token tail entropy, requiring no parsers, canonical forms, or reward models.
- *Intrinsic Particle Filtering* (`iPF`): performs step-level resampling guided solely by intrinsic entropy, steering generation toward high-confidence reasoning trajectories without process reward models.
- *Particle Distillation* (`dPF`): injects privileged information, such as rubric criteria or hints, via logit blending and KL-guided resampling, correcting faulty early assumptions and enforcing constraints without explicit verification.

These three methods form a highly complementary pipeline that operates entirely without trained reward models or ground-truth verification. *While iS and iPF are strictly verifier-free - relying solely on internal token distributions - dPF introduces privileged guidance (e.g., rubrics or hints) to steer generation on highly constrained problems where intrinsic signals alone are insufficient.* Crucially, their success across diverse architectures suggests intrinsic entropy is a fundamental property rather than a model-specific artifact.

Evaluated across math, science, coding, clinical reasoning, and engineering design, the framework integrates into modern inference engines without calibration or post-processing (Figure 1). The verifier-free `iS` is competitive with consensus algorithms without output verification (Figure 1a). On image-to-CAD, where the model writes CadQuery code for a rendered 3D part and scoring requires running a geometric kernel, it improves median IoU over a random pick from the same pool by 18% (Figure 1c). On the hardest math problems, `iPF` raises the expected accuracy of a random final

particle by 6.1 points at a fixed budget of 16 particles (Figure 1b). With privileged guidance but no reward model, dPF achieves up to 26.5% gains on complex clinical responses using 8 particles instead of 32 samples (Figure 1d), and a preliminary agentic study on Kubernetes troubleshooting shows the same trend for iS (Appendix J.12).

2 Background and Related Work

Domain Verifiability Appendix B Table 6 contrasts verifiable domains (e.g., math, coding), which offer cheap automated rewards ($V(x, y) \in \{0, 1\}$), with hard-to-verify domains (e.g., healthcare, large-scale engineering) that require expensive subjective or model-based evaluation. These high costs bottleneck scalability. Furthermore, even in structured fields, heuristic parsers and solvers remain fragile and computationally expensive, effectively rendering them hard-to-verify at scale (Appendix B).

Inference-Time Scaling Allocating additional compute during generation significantly improves reasoning [Snell et al., 2024, Brown et al., 2024, Wu et al., 2025]. Scaling strategies generally divide into *parallel sampling* (generating and selecting from independent trajectories) and *sequential search* (navigating reasoning trees toward high-density solution regions).

Consensus-based Selection Self-Consistency (SC) [Wang et al., 2022] is a highly successful verification-free parallel strategy. It approximates the conditional answer distribution by marginalizing latent trajectories $\{\mathbf{z}\}_{k=1}^{K} \sim p(\mathbf{z} \mid \mathbf{c})$: $p(\mathbf{x} \mid \mathbf{c}) = \sum_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z}, \mathbf{c}) p(\mathbf{z} \mid \mathbf{c})$. These systems extract consensus via majority voting over M choices $\{a_m\}_{m=1}^{M}$, as defined in Eq. 1:

$$x_a^* = \arg\max_a \sum_{m=1}^{M} \mathbb{I}(a_m = a). \tag{1}$$

While scalable and robust to noise, SC relies on cheap syntactic verification, severely limiting its use in open-ended or multi-part tasks. Universal SC [Chen et al., 2023] addresses this via LLM-based judging, but incurs new model-based verification costs.

Confidence-based Selection To bypass external verifiers, confidence-based methods leverage internal model signals. This includes intrinsic verification in RL-trained models [Guo et al., 2025, Jaech et al., 2024], Self-Certainty [Kang et al., 2025], trace-level logit weighting [Fu et al., 2025], and uncertainty-aware generation [Zhang et al., 2025b]. For a probabilistic model p defined over vocabulary V and sequence length T we define: $p(\mathbf{z}) = p(\mathbf{z}_T) \prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{z}_{<t})$, and quantify model uncertainty using token entropy [Shannon, 1948] in Eq. 2:

$$\mathbb{H}[\mathbf{z}_t \mid \mathbf{z}_{<t}] = -\sum_{v \in \mathcal{V}} p(\mathbf{z}_t = v \mid \mathbf{z}_{<t}) \log p(\mathbf{z}_t = v \mid \mathbf{z}_{<t}). \tag{2}$$

Eq. 3 demonstrates that this is equivalent to the negated KL divergence relative to a uniform distribution U:

$$\mathbb{KL}[p, U]_t = \sum_{v \in \mathcal{V}} p(\mathbf{z}_t^v) \log \frac{p(\mathbf{z}_t^v)}{1/V} = -\mathbb{H}[p]_t + \log(V). \tag{3}$$

Sequences with lower entropy (or higher KL divergence) exhibit relative confidence. Our approach complements RL-trained models [Guo et al., 2025, Jaech et al., 2024]; rather than relying solely on step-level verification within individual trajectories, we extract discriminative signals from the collective statistics of an N-sample set.

Particle-based Inference and Guidance Sequential Monte Carlo (SMC) methods [Doucet et al., 2001a, Zhao et al., 2024, Lew et al., 2023] and Particle Filtering (PF) [Puri et al., 2025] approximate the posterior $p(\mathbf{z} \mid \mathbf{c}, \mathbf{o}) \propto p(\mathbf{o} \mid \mathbf{z}, \mathbf{c}) p(\mathbf{z} \mid \mathbf{c})$ via sequential importance resampling [Liu et al., 2001] with a proposal $q(\mathbf{z} \mid \mathbf{c})$, leading to the standard weight update shown in Eq. 4:

$$w_t \propto w_{t-1} \frac{p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c}, \mathbf{o}_t)}{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c})}. \tag{4}$$

Using the model $p(\mathbf{z} \mid \mathbf{c})$ itself as a bootstrap filter proposal [Elfring et al., 2021] reduces this weight update to $w_t \propto w_{t-1}\, p(\mathbf{o}_t \mid \mathbf{z}_t, \mathbf{c})$. Typically, the observation model relies on a Process Reward Model (PRM), i.e., $r_\phi(\mathbf{z}_t, \mathbf{c}) = p(\mathbf{o}_t = 1 \mid \mathbf{z}_t, \mathbf{c})$. However, PRMs are expensive, domain-specific, and prone to overconfidence on out-of-distribution long sequences [Park et al., 2025], limiting their viability in complex or specialized fields. See Appendix A and B for extended related work.

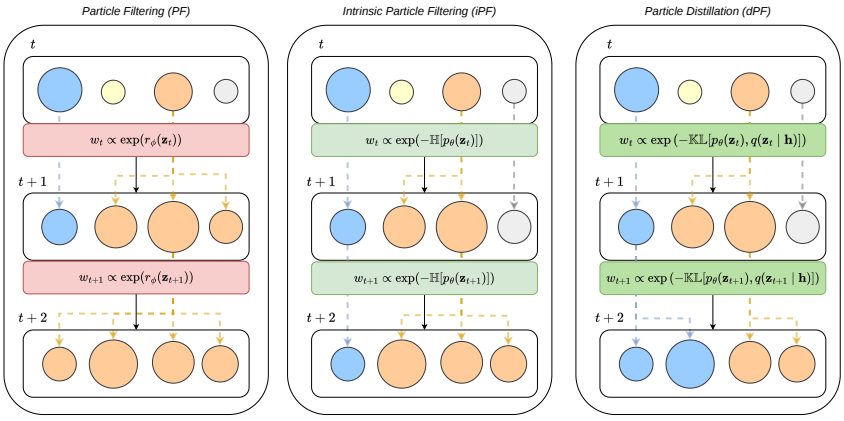


Figure 2: Overview of intrinsic particle filtering (iPF) and particle distillation (dPF). Both extend particle filtering to hard-to-verify domains without requiring calibrated external reward models. iPF resamples particles using intrinsic token-level entropy to concentrate compute on high-confidence trajectories, while dPF uses privileged guidance (e.g., rubrics or hints) through logit blending and KL-guided resampling to correct early errors and enforce task-specific constraints.

3 Method

Our goal is to enable effective inference-time scaling across a broad range of domains, regardless of whether external verification is available. To do so, we exploit intrinsic statistics of generated samples to derive signals for candidate selection, problem difficulty estimation, entropy-guided resampling, and guided steering, as illustrated in our framework overview (Figure 2). See Appendix C for more details.

3.1 Intrinsic Selection (iS)

When the model can generate correct answers among N parallel samples, the main challenge is to identify the best candidate without external verification. Given a problem $\mathbf{c}$, a model $p_\theta(\mathbf{z} \mid \mathbf{c})$, and a set of N sampled trajectories $\{\mathbf{z}_n\}_{n=1}^N$, Intrinsic Selection (`iS`) ranks candidates using adaptive token-level intrinsic statistics:

$$\texttt{iS}(\mathbf{c}) = g\big(\{\mathbf{z}_n\}_{n=1}^N,\, l(\cdot),\, \mathbb{H}[\cdot]\big)\,. \tag{5}$$

The scoring function g in Eq. 5 aggregates per-token entropy over an adaptively chosen tail region determined by sequence length $l(\cdot)$ and token entropy $\mathbb{H}[\cdot]$.

Adaptive Tail Entropy We define an adaptive tail window that balances two competing effects: longer windows capture more of the answer region, while shorter windows reduce noise from the preceding reasoning trace. Using the set-level length distribution $l_n = l(\mathbf{z}_n)$ and entropy distribution $\mathbb{H}[\mathbf{z}_n]$, we define the tail cutoff as

$$t_c = \frac{\sqrt{N^{-1}\sum_{n=1}^N l(\mathbf{z}_n)}}{N^{-1}\sum_{n=1}^N \mathbb{H}[\mathbf{z}_n]}. \tag{6}$$

The sublinear scaling $t_c \propto \sqrt{l}$ in Eq. 6 reflects the empirical observation that answer spans grow sub-linearly with total sequence length. The inverse dependence on mean entropy ensures that, when the model is uncertain across the sample set, a narrower tail is used to avoid uninformative entropy spikes.

Letting the tail index set of sample n be $\mathcal{A}^{(n)} = \{t \in \mathbb{Z} \mid l_n - t_c < t \le l_n\}$, we define the adjusted tail entropy score for sample n as:

$$s_n(\mathbf{c}) = \operatorname*{agg}_{t \in \mathcal{A}^{(n)}} \left(\mathbb{H}[\mathbf{z}_t^{(n)} \mid \mathbf{z}_{<t}^{(n)}]\right). \tag{7}$$

In Eq. 7 we aggregate over tokens using the median, which is robust to outlier spikes caused by punctuation or formatting, and select the sample with the smallest s_n. In the following, we refer to this quantity interchangeably as `iS@1`.

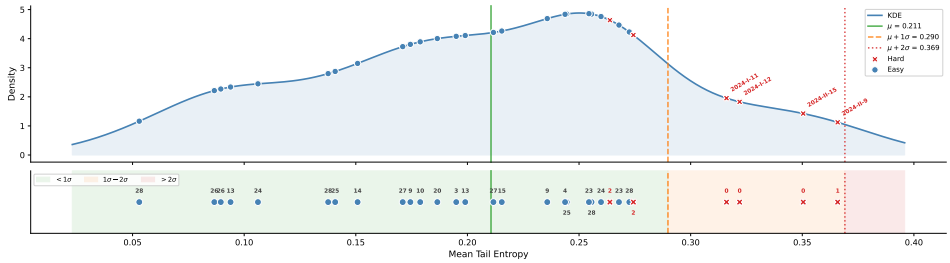


Figure 3: Problem-level entropy and estimated difficulty. Higher adjusted mean per-token tail entropy correlates with harder problems. Instances more than 1σ above the mean are strongly associated with hardness, defined by n/N with $n \in \{0, 1, 2\}$.

Difficulty Estimation and Routing Problem difficulty is inherently compute and model-dependent; what is challenging for a base model may be trivial for an advanced one. Therefore, we propose a data-driven, verifier-free approach to estimate relative problem difficulty using small-scale inference statistics (Appendix J.8 Table 31). By generating a standard budget of N parallel samples from the base policy p_θ, we analyze set-level intrinsic metrics to approximate difficulty. Specifically, after applying length-based trimming to filter outliers, we compute the average token-level tail entropy.

These scores reflect the model's task-specific uncertainty, allowing us to accurately isolate the highest-entropy subset (Figure 3). This mechanism naturally serves as a difficulty routing for downstream compute allocation: problems exhibiting low tail entropy are resolved instantly via `iS`, while tasks triggering the high-entropy gate are dynamically routed to `iPF` or `dPF` for step-level intervention or guidance.

3.2 Intrinsic Particle Filtering (iPF)

When a correct trajectory exists but lies in a low-density region ($n/N \ll 1$), parallel sampling alone is inefficient. We therefore introduce Intrinsic Particle Filtering (`iPF`), which concentrates generation on sparse, high-confidence trajectories using intrinsic entropy signals rather than external reward models.

In standard sequential importance resampling for inference scaling, a reward model is used to approximate the target distribution [Puri et al., 2025, Zhang et al., 2025c]. If we take the model itself as the proposal distribution, i.e.,

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c}) = p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c}),$$

then by Bayes' rule the unnormalized particle weight update reduces to:

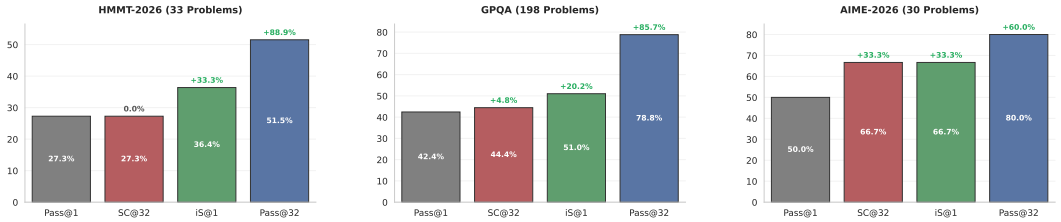
$$w_t \propto w_{t-1}\, p(\mathbf{o}_t \mid \mathbf{z}_t, \mathbf{c}). \tag{8}$$

Rather than using an external reward to define the observation likelihood $p(\mathbf{o}_t \mid \mathbf{z}_t, \mathbf{c})$ in Eq. 8, we posit that particles with low recent entropy correspond to more confident and promising reasoning directions. We therefore define a pseudo-observation likelihood [Karampela and Beeson, 2026, Gloaguen et al., 2022] as an exponential function of intrinsic entropy (as defined in Eq. 2), which yields the `iPF` resampling update in Eq. 9:

$$w_t \propto w_{t-1} \exp(-\mathbb{H}[\mathbf{z}_t]). \tag{9}$$

Building the Resampling Distribution When intrinsic metrics are used for resampling, the resulting unnormalized weights can vary widely in magnitude. This contrasts with standard PRM-based resampling, where scores are typically bounded in $[0, 1]$ and log-weights can be obtained using transformations such as $\log(p/(1-p))$. Without such bounds, intrinsic signals can lead to degenerate resampling, in which a single particle dominates and the particle set collapses. To mitigate this issue, in Eq. 10 we apply a robust normalization step before computing the final weights:

$$\tilde{w}_t \leftarrow \frac{\mathbb{H}[\mathbf{z}_t] - m(\mathbb{H}[\mathbf{z}_t])}{s(\mathbb{H}[\mathbf{z}_t]) + \epsilon}, \tag{10}$$



(a) HMMT 2026 (b) GPQA-Diamond (c) AIME 2026

Figure 4: Selection accuracy on recent math and reasoning benchmarks (Qwen3-4B-Instruct-2507, $N = 32$). SC@N and iS@1 select from the same N candidates and use identical generation compute. pass@1 is the expected accuracy of a uniformly random pick from that pool, and pass@N is the oracle for the pool. The 2026 problems post-date the model's release, which limits training-data contamination.

where ϵ is a small constant for numerical stability, and $m(\cdot)$ and $s(\cdot)$ denote the batch mean and standard deviation, respectively. We then obtain normalized particle weights using the softmax function, $w_t = \sigma(-\tilde{w}_t)$, where $\sigma(x) = \exp(x)/\sum_{x \in X} \exp(x)$.

We use the same normalization strategy for both $\mathbb{H}$-based and $\mathbb{KL}$-based scoring. With the raw weights of Eq. 9, iPF is a bootstrap filter for the base model tilted by an entropy potential, which targets confidence rather than correctness; dividing by $s(\cdot)$ adds adaptive tempering, so standard unbiasedness guarantees do not apply [Beskos et al., 2016]. Unlike twisted SMC [Zhao et al., 2024], iPF replaces learned twist functions with the observed recent entropy, a myopic but training-free proxy (Appendix C).

3.3 Particle Distillation (dPF)

When a problem exhibits *systematic failure* ($pass@N \approx 0$), simply increasing the sample budget N is computationally wasteful. Because the base policy is consistently misled by incorrect early assumptions, standard inference-time scaling yields diminishing returns. To overcome this, we introduce *Particle Distillation* (dPF), an extension of our resampling framework that operates without trained reward models. dPF actively corrects these faulty assumptions using a guide conditioned in-context on privileged information [Liao et al., 2026]—such as hints, expert rubrics, or environmental constraints [Zhao et al., 2026, Agarwal et al., 2024, Hübotter et al., 2026]. By combining early-trajectory logit blending with subsequent KL-guided resampling, dPF steers generation to satisfy complex multidimensional constraints before the model can commit to a flawed reasoning path.

Logits Blending and Guided Divergence When a model systematically fails due to incorrect early assumptions, naively scaling generation is futile. dPF mitigates this by mixing the base model's unconditional generation with a guide conditioned on privileged information $\mathbf{h}$ (e.g., rubrics, constraints, or hints) during the critical first steps:

$$\text{logits}(\mathbf{z}_t^h) = (1-\alpha)\,\text{logits}(\mathbf{z}_t, \mathbf{c}) + \alpha\,\text{logits}(\mathbf{z}_t, \mathbf{c}, \mathbf{h}) \quad \alpha \in (0,1), \tag{11}$$

where $\mathbf{z}_t \sim p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}^h, \mathbf{c})$ in Eq. 11. Inspired by classifier-free guidance in diffusion models [Ho and Salimans, 2022], we apply a linear annealing schedule to the blending parameter α. This directly steers trajectories toward correct assumptions before the model commits to a flawed reasoning path.

After annealing α, we transition to steering the remaining steps via guided step-level resampling [Doucet et al., 2001b, Zhao et al., 2024]. Instead of raw entropy, we dynamically compute particle weights based on the KL divergence between the base model and the guide:

$$w_t \propto w_{t-1} \exp\left(-\mathbb{KL}[p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c}), q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c}, \mathbf{h})]\right). \tag{12}$$

By penalizing divergence using Eq. 12, we effectively distill the guided trajectory back into the unconstrained model, navigating highly constrained problems leveraging guidance.

4 Experiment

Evaluating across the verifiability spectrum (Appendix O Table 40), we show that: *(i)* intrinsic statistics offer a verification-free selection signal rivaling consensus methods (Sec. 4.1); *(ii)* they reliably gauge problem difficulty for adaptive compute allocation (Sec. 4.1); and *(iii)* verifier-free resampling (iPF) and guided distillation (dPF) enhance inference scaling (Sec. 4.2, 4.3).

Models We evaluate Qwen3-4B-Instruct-2507 [Yang et al., 2025], gemma-4-E2B-it [Team et al., 2025a], medgemma-1.5-4b-it [Sellergren et al., 2026] (medical), a CAD-fine-tuned Qwen2.5-VL-7B-Instruct [Bai et al., 2025, Doris et al., 2025] (vision-language), and, for a preliminary agentic study, Qwen2.5-72B-Instruct [Yang et al., 2024].

Table 1: Level of Domain Verifiability.

Domain	Verification	Datasets
Mathematics	Symbolic	AIME, HMMT
Reasoning	Output	GPQA-Diamond
Coding	Execution	LiveCodeBench-v6
Healthcare	Semantic	HealthBench-Hard
Engineering	Solver	Fusion360

Datasets We benchmark across five domains using AIME and HMMT [Balunović et al., 2025], GPQA-Diamond [Rein et al., 2023], LiveCodeBench-v6 [Jain et al., 2024], HealthBench-Hard [Arora et al., 2025], and image-to-CAD on DeepCAD [Wu et al., 2021] and Fusion360 [Willis et al., 2021]. These span symbolic, output, execution, semantic, and solver-based verification. Details in Appendix O.

Evaluation For each problem $\mathbf{c}$, every method selects one candidate from the same pool $\{\mathbf{z}^{(n)}\}_{n=1}^{N}$, whose candidates have scores $r^{(n)} = r(\mathbf{z}^{(n)}, \mathbf{c})$. As references for a selected candidate's score (iS@1, SC@N), we report:

$$\text{pass@1} = \frac{1}{N}\sum_{n=1}^{N} r^{(n)}, \qquad \text{pass@}N = \max_{n} r^{(n)}, \tag{13}$$

averaged over problems. With *hard* (binary) scores, $r^{(n)} \in \{0, 1\}$, as in math, GPQA, and LiveCodeBench, pass@1 $= c/N$ for c correct candidates and pass@$N = 1[c > 0]$. These are the standard unbiased pass@k estimates [Chen et al., 2021] at $k = 1$ and $k = N$. With *soft* (real-valued) scores, as in image-to-CAD (IoU) and HealthBench (rubric score), Eq. 13 gives the mean and the maximum score in the pool of N samples, with no pass/fail threshold. In both cases, pass@1 is the expected score of a uniformly random pick from the pool and pass@N is the oracle for that pool. We keep the same names so that results read the same way across domains. Because iS@1 and SC@N select from the same N samples, they use identical generation compute. pass@1 is therefore a random-selection reference that isolates the value of the selection signal, not a compute-matched single-sample baseline. We generate up to 128 samples for iS and up to 32 particles for iPF/dPF. Logit blending is applied to short sequences, and KL-guided resampling is triggered for longer outputs (average length above ten 128-token steps). Step-level methods select their final particle with iS. Baselines include Self-Certainty [Kang et al., 2025], Self-Consistency [Wang et al., 2022], DeepConf [Fu et al., 2025], CISC [Taubenfeld et al., 2025], and RPC [Zhou et al., 2026].

4.1 Intrinsic Selection (iS) with Adjusted Tail Entropy

Table 2 reports accuracy on seven benchmark datasets. We summarize performance using three aggregate metrics: the unweighted mean (avg), the median (med), which is less sensitive to outliers, and the dataset-size weighted average (wt avg). For iS, we apply a single estimator across all datasets, without symbolic verification or output matching. Bold values indicate the best practical estimator, excluding the pass@32 upper bound. iS performs robustly across benchmarks, achieving the best median (47.7%) and weighted average (41.9%) among practical estimators, while Self-Consistency (SC) obtains the best unweighted average (46.0%). Both methods substantially outperform the pass@1 baseline. As shown in Figure 4, iS consistently narrows the gap between pass@1 and the pass@32 upper bound across the considered domains.

These results establish intrinsic selection via tail entropy as a strong verification-free alternative to SC. Unlike SC, which relies on output parsing and canonical-form matching, iS operates directly on

Table 2: Evaluation accuracy (%) across seven benchmark datasets, reporting mean performance over four independent seeds. Aggregate metrics include the unweighted average (avg), median (med), and dataset-size weighted average (w avg). For iS, we use a single estimator—adjusted tail entropy—across all datasets, without symbolic or output verification. iS remains competitive with a strong Self-Consistency (SC) baseline.

Method	A2024	A2025	A2026	H2025	H2026	GPQA	LCB	avg	med	wt avg
pass@1	$56.5_{\pm 0.4}$	$45.3_{\pm 0.5}$	$50.0_{\pm 0.1}$	$30.2_{\pm 0.3}$	$28.5_{\pm 0.4}$	$42.3_{\pm 0.1}$	$19.0_{\pm 0.9}$	38.8	42.3	37.0
SC	$72.5_{\pm 1.6}$	$60.8_{\pm 2.8}$	$65.8_{\pm 1.6}$	$30.8_{\pm 0.8}$	$29.5_{\pm 0.9}$	$44.1_{\pm 0.6}$	$18.2_{\pm 1.2}$	**46.0**	44.1	40.8
iS@1 (ours)	$67.5_{\pm 2.1}$	$49.9_{\pm 2.4}$	$54.2_{\pm 1.6}$	$35.0_{\pm 0.9}$	$34.8_{\pm 0.7}$	$47.7_{\pm 0.5}$	$21.5_{\pm 1.2}$	44.4	**47.7**	**42.0**
pass@32	$87.5_{\pm 0.8}$	$75.8_{\pm 2.1}$	$83.3_{\pm 1.4}$	$56.7_{\pm 3.3}$	$52.3_{\pm 1.5}$	$79.3_{\pm 0.7}$	$35.0_{\pm 0.4}$	67.1	75.8	66.6

Table 3: Image-to-CAD selection on in-distribution (DeepCAD) and out-of-distribution (Fusion360) test sets (IoU; VSR = valid-sample rate). pass@1 and pass@128 are the mean and maximum IoU of the 128-candidate pool used by `iS@1`.

	DeepCAD (in-distro)			Fusion360 (out-distro)		
Method	Mean	Median	VSR	Mean	Median	VSR
pass@1	0.62	0.75	97.5%	0.41	0.38	94.4%
`iS@1` (ours)	**0.66**	**0.76**	**100%**	**0.45**	**0.48**	**97.0%**
pass@128	0.89	0.98	100%	0.83	0.88	100%

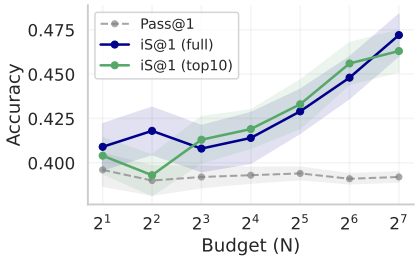


Figure 5: Inference Scaling behavior for image-to-CAD generation using `iS@1` on Fusion360 test set.

token-level statistics. This makes it applicable wherever sampling is possible, including open-ended settings in which SC is difficult to apply, providing evidence that the tail-entropy signal reflects a fundamental property rather than an architecture-specific artifact.

Engineering Design While generation scaling (pass@128) improves image-to-CAD performance (Table 3), output verification becomes a major computational bottleneck at scale. Executing geometric kernels and computing Intersection over Union (IoU) can account for up to 40% of the generation loop. In addition, the absence of lightweight reward models in engineering makes standard inference-time scaling methods, such as Best-of-N or step-level reward guidance, especially challenging to apply.

To address these limitations, `iS` uses set-level length distributions and adjusted tail entropy to identify high-quality designs. This yields a reliable scaling mechanism without the substantial overhead of external geometric solvers (Table 3). Unlike our other evaluations, the CAD experiments use a custom fine-tuned Qwen2.5-VL model [Doris et al., 2025]. The strong performance of `iS` on this specialized multimodal architecture suggests that the selection mechanism is largely agnostic to model family and fine-tuning procedure. As shown in Figure 5, `iS` also scales predictably on Fusion360 test set. As the sample budget N increases, `iS` steadily narrows the gap to the pass@N oracle, indicating that the intrinsic signal becomes more informative with additional samples, even without a domain-specific reward model. Appendix J.7 for more details.

Table 4: Accuracy on Hard 25% Subset (Budget $N = 16$) for AIME datasets using Intrinsic Particle Filtering (`iPF`) as resampling mechanism. We run 10 seeds and report Min, Avg, Max performance.

		A24			A25			A26		
Metric	Sampling	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Gen Quality										
pass@1	parallel	6.7	17.0	29.2	1.6	4.9	9.4	2.0	5.6	10.5
pass@1	`iPF`	10.7	22.8	36.6	2.0	8.3	16.1	4.9	14.7	26.3
Δ pass@1			**+5.8**			**+3.3**			**+9.2**	
Self-Consistency										
SC	parallel	7.1	21.4	35.7	0.0	10.7	25.0	0.0	3.6	10.7
SC	`iPF`	14.3	28.6	46.4	0.0	10.7	25.0	3.6	17.9	32.1
Δ SC			**+7.1**			0.0			**+14.3**	
Entropy Selection										
`iS`	parallel	10.7	25.0	42.9	0.0	7.1	17.9	0.0	10.7	21.4
`iS`	`iPF`	10.7	25.0	42.9	3.6	17.9	32.1	3.6	14.3	28.6
Δ `iS`			0.0			**+10.7**			**+3.6**	

Problem Difficulty Estimation We evaluate whether intrinsic statistics can reliably estimate problem difficulty to dynamically route compute (Appendix J.8). As shown in Table 31, length-trimmed tail entropy is a highly reliable difficulty estimator for structured reasoning. It achieves strong ranking quality (AUROC > 0.79) and $> 90\%$ precision for adaptive compute allocation on AIME-2026 and HMMT-2026.

Ablations The appendix provides extensive ablations of `iS`, showing that the adaptive tail beats a fixed token window on five of seven datasets, median beats mean aggregation on six of seven, and length trimming has a small effect (Appx J.6). Tail entropy beats Self-Certainty on all five math datasets and is competitive with DeepConf, CISC, and RPC (Appx J.3). Accuracy grows with the budget (Appx J.1), the estimator transfers to Gemma without retuning (Appx I), and selection holds when the final answer is removed (Appx J.5); reward-model and hybrid comparisons are in Appx H.

4.2 Intrinsic Particle Filtering (`iPF`) on Hard Problems

We evaluate whether intrinsic, entropy-guided step-level resampling (`iPF`) improves generation quality on problems where the base model rarely succeeds. We compare `iPF` against a standard i.i.d. base-

line on the hardest 25% of the AIME (2024–2026) datasets—where iS underperforms SC (Sec 4.1 Table 2)—as well as on clinical reasoning tasks from HealthBench-Hard. For iPF, we deploy 16 particles for up to 120 steps (128 tokens per step) and apply systematic resampling with z-score normalization, yielding an approximate step resampling rate of 20%.

As detailed in Table 4, iPF significantly outperforms the 16-sample baseline on structured math tasks, achieving an average +6.1 percentage point gain in pass@1 across the AIME datasets, highlighted by a notable +9.2 point increase on AIME-2026.

By iteratively refining trajectories, iPF successfully navigates problems where the standard sampling policy occasionally identifies, but fails to execute, correct approaches. Furthermore, because this resampling mechanism concentrates particles around promising reasoning paths, self-consistency (SC@16) effectively amplifies the correct signal once it is found, driving SC accuracy from 3.6% to 17.9% on AIME-2026.

Crucially, this capability extends to open-ended, hard-to-verify domains. As shown in Figure 6, iPF consistently improves problem-weighted rubric scores over the i.i.d. baseline on HealthBench-Hard, excelling specifically on reasoning-heavy clinical themes. In contrast, Particle Distillation (dPF, discussed next) provides complementary steering on this same dataset for tasks that demand strict adherence to complex, rubric-specific criteria.

4.3 Steering Resampling with Particle Distillation (dPF)

We evaluate dPF in two complementary settings prone to *systematic failure*: competition mathematics, where the model systematically errs due to incorrect assumptions or problem complexity, and clinical healthcare, where quality is multidimensional and requires strict adherence to expert rubrics and safety constraints rather than binary verification. In both cases, dPF injects privileged information to steer sampling past these structural bottlenecks.

Mathematical Reasoning (Teacher Guidance) For mathematics, we generate hints (e.g., procedures or specific sub-problem strategies) from a strong teacher model using trajectories with the final answers removed. Crucially, neither the hint generator, the guide distribution $q(\mathbf{z} \mid \mathbf{c}, \mathbf{h})$, nor the sampling model $p_\theta(\mathbf{z} \mid \mathbf{c})$ accesses the ground truth answer.

Table 5: Math particle distillation results comparing dPF ($N = 8$, guided resampling) against $N = 32$ i.i.d. samples. Guidance hints are generated via Qwen3.5-27B [Team, 2026]. Crucially, the hint generator, guide q, and sampling model p access only $\mathbf{h}$, without ground-truth answers or extrinsic verification. See Appx J Table 34 for ablations.

Metric	Sampling	Dataset				
		A24	A25	A26	H25	H26
pass@1	parallel	56.7	44.3	50.1	30.0	28.2
pass@1	dPF	**64.6**	**47.9**	**62.5**	**32.5**	**37.1**
iS@1	parallel	66.7	53.3	53.3	**33.3**	36.4
iS@1	dPF	**76.7**	**63.3**	**63.3**	**33.3**	**45.5**
pass@32	parallel	86.7	70.0	80.0	56.7	51.5

As detailed in Tables 5 and 34, distilling this privileged guidance allows the model to correct faulty procedural assumptions early in the trajectory. With just 8 particles, dPF consistently matches or exceeds the 32-sample i.i.d. baseline across all metrics, achieving a 4–5 point improvement in weighted average pass@1. We find that logit blending is particularly effective on the hardest datasets, driving the largest performance gains (e.g., +8.9 points on HMMT-2026).

Furthermore, critique-based hints and teacher demonstrations offer complementary benefits depending on the domain's specific failure mode: critiques excel on AIME, where identifying faulty assumptions is critical, while teacher demonstrations perform better on HMMT, where the solution paths themselves are highly non-obvious.

Finally, Figure 7 illustrates this steering mechanism on an unsolved AIME problem (pass@$N = 0$). By applying guided resampling, dPF shifts the KL distribution from a diffuse regime (mean 0.111) to a concentrated, low-divergence one (mean 0.042), effectively forcing particles toward trajectories where the base model and the guide agree.

Clinical Healthcare (Rubric Guidance) To test transferability, we evaluate HealthBench-Hard - comprising 100 problems across seven clinical themes - using MedGemma-4B-IT for generation and MedGemma-27B as the judge. Despite the shift in domain and architecture, both iPF and dPF yield substantial gains without requiring any retuning of the particle filtering hyperparameters. We

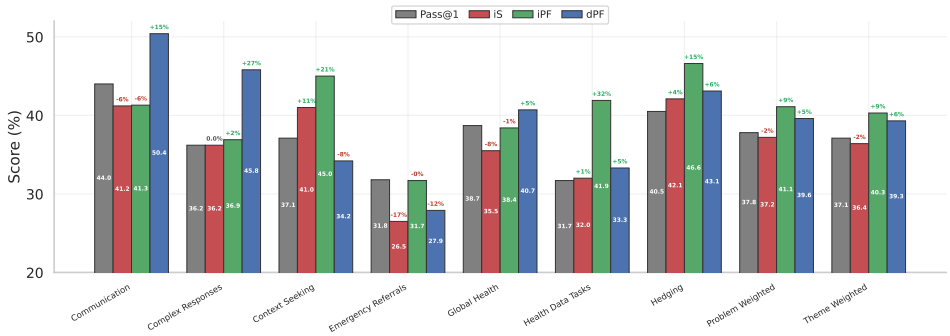


Figure 6: HealthBench Results using iPF and dPF.

compare a 32-sample i.i.d. baseline against iPF ($N = 16$) and dPF ($N = 8$). In this setting, dPF employs a rubric-conditioned KL divergence strategy: rather than using teacher hints, a guide model conditioned solely on the problem's rubric criteria directs resampling and logit blending over the first 25% of steps.

As shown in Figure 6 and Appendix Table 35, iPF emerges as the strongest overall method, achieving a 0.411 mean score (+8.7% over pass@1). It specifically excels on themes requiring deep reasoning - such as health data tasks (+32.2%) and context seeking (+21.3%) - where sustained low entropy strongly correlates with successfully completing a logical chain. Meanwhile, dPF serves as the second-best overall strategy (+4.8% over pass@1), but dominates on themes with concrete, checkable criteria. For instance, the rubric-conditioned guide provides a highly targeted signal for communication (+14.5%) and complex responses (+26.5%), successfully steering generation to include mandatory pharmacological or guideline-based content. Conversely, standard iS degrades performance by -1.5% compared to pass@1.

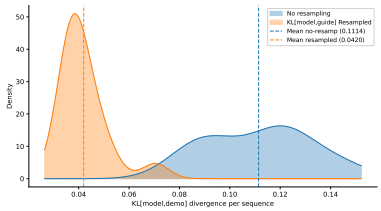


Figure 7: KL divergence between the base and hint-guided models on the unsolved AIME problem 2024-II-15. Guided resampling (orange) shifts particles toward low-divergence regions (mean KL: 0.042 vs. 0.111), indicating that dPF steers generation toward trajectories consistent with the privileged hint.

In HealthBench, single-forward-pass entropy tends to select the most "confident" response, but confidence alone does not guarantee multidimensional rubric coverage. This contrast perfectly highlights the complementary nature of our framework: while iS serves as a highly effective selection and routing mechanism for scalar quality metrics, iPF and dPF are essential for driving generation quality in complex, open-ended spaces.

5 Conclusion and Limitations

We present a framework for inference-time scaling that uses intrinsic model statistics - in particular, length distributions and tail entropy - to estimate problem difficulty and rank candidates without process-level rewards. For more challenging tasks, Intrinsic Particle Filtering (iPF) performs step-level resampling using internal signals, while Particle Distillation (dPF) steers generation with privileged information. Together, these components recast inference-time compute as a practical and robust mechanism driven by intrinsic signals.

Limitations Tail entropy measures confidence, not correctness: it can pick a confidently wrong answer when a misconception is frequent and fluent, and it misses multidimensional quality. Low recent entropy can also reflect premature commitment, so iPF may discard particles that would have resolved correctly, and its heuristic, adaptive weights carry no SMC guarantees. Difficulty estimation costs N samples up front, dPF needs privileged information, and our evidence covers the evaluated models and domains. Future work includes intrinsic statistics as training signals, a set-level probabilistic formulation of iS and iPF, and self-generated critiques in place of external hints.

References

R. Agarwal, N. Vieillard, Y. Zhou, P. Stanczyk, S. R. Garea, M. Geist, and O. Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

R. K. Arora, J. Wei, R. S. Hicks, P. Bowman, J. Quiñonero-Candela, F. Tsimpourlas, M. Sharman, M. Shah, A. Vallone, A. Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

M. Balunović, J. Dekoninck, I. Petrov, N. Jovanović, and M. Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

A. Beskos, A. Jasra, N. Kantas, and A. Thiery. On the convergence of adaptive sequential monte carlo methods. 2016.

B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

X. Chen, R. Aksitov, U. Alon, J. Ren, K. Xiao, P. Yin, S. Prakash, C. Sutton, X. Wang, and D. Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

N. Chopin, O. Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.

A. C. Doris, M. F. Alam, A. Heyrani Nobari, and F. Ahmed. Cad-coder: An open-source vision-language model for computer-aided design code generation. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 89220, page V03AT03A031. American Society of Mechanical Engineers, 2025.

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001a.

A. Doucet, N. De Freitas, N. J. Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001b.

E.-M. El-Mhamdi and L.-N. Hoang. On goodhart's law, with an application to value alignment. *arXiv preprint arXiv:2410.09638*, 2024.

J. Elfring, E. Torta, and R. Van De Molengraft. Particle filters: A hands-on tutorial. *Sensors*, 21(2):438, 2021.

Y. Fu, X. Wang, Y. Tian, and J. Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

L. Gao, J. Schulman, and J. Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

W. R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

P. Gloaguen, S. Le Corff, and J. Olsson. A pseudo-marginal sequential monte carlo online smoothing algorithm. *Bernoulli*, 28(4):2606–2633, 2022.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

J. Hübotter, F. Lübeck, L. Behric, A. Baumann, M. Bagatella, D. Marta, I. Hakimi, I. Shenfeld, T. K. Buening, C. Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Z. Kang, X. Zhao, and D. Song. Scalable best-of-n selection for large language models via self-certainty. *arXiv preprint arXiv:2502.18581*, 2025.

T. Karampela and R. Beeson. A variational pseudo-observation guided nudged particle filter. *arXiv preprint arXiv:2603.16705*, 2026.

L. Kuhn, Y. Gal, and S. Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, S. Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

A. K. Lew, T. Zhi-Xuan, G. Grand, and V. K. Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

X. L. Li, A. Holtzman, D. Fried, P. Liang, J. Eisner, T. Hashimoto, L. Zettlemoyer, and M. Lewis. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*, 2022.

B. Liao, H. Dong, X. Xu, C. Monz, and J. Bian. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*, 2026.

H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

J. S. Liu, R. Chen, and T. Logvinenko. A theoretical framework for sequential importance sampling with resampling. In *Sequential Monte Carlo methods in practice*, pages 225–246. Springer, 2001.

K. Misaki, Y. Inoue, Y. Imajuku, S. Kuroki, T. Nakamura, and T. Akiba. Wider or deeper? scaling llm inference-time compute with adaptive branching tree search. *arXiv preprint arXiv:2503.04412*, 2025.

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

Y.-J. Park, K. Greenewald, K. Alim, H. Wang, and N. Azizan. Know what you don't know: Uncertainty calibration of process reward models. *arXiv preprint arXiv:2506.09338*, 2025.

I. Puri, S. Sudalairaj, G. Xu, K. Xu, and A. Srivastava. A probabilistic inference approach to inference-time scaling of llms using particle-based monte carlo methods. *arXiv preprint arXiv:2502.01618*, 2025.

D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

G. Sanchez, H. Fan, A. Spangher, E. Levi, P. S. Ammanamanchi, and S. Biderman. Stay on topic with classifier-free guidance. *arXiv preprint arXiv:2306.17806*, 2023.

S. Sanyal, T. Xiao, and X. Ren. Mixing inference-time experts for enhancing llm reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21246–21260, 2025.

A. Sellergren, C. Gao, F. Mahvar, T. Kohlberger, F. Jamil, M. Traverse, A. Tono, B. Sadjad, L. Yang, C. Lau, et al. Medgemma 1.5 technical report. *arXiv preprint arXiv:2604.05081*, 2026.

C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Y. Tang, S. Wang, L. Madaan, and R. Munos. Beyond verifiable rewards: Scaling reinforcement learning for language models to unverifiable data. *arXiv preprint arXiv:2503.19618*, 2025.

A. Taubenfeld, T. Sheffer, E. Ofek, A. Feder, A. Goldstein, Z. Gekhman, and G. Yona. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20090–20111, 2025.

G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025a.

K. Team, Y. Bai, Y. Bao, Y. Charles, C. Chen, G. Chen, H. Chen, H. Chen, J. Chen, N. Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025b.

Q. Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

H. Wang, C. Du, K. Kawaguchi, and T. Pang. Think in parallel, answer as one: Logit averaging for open-ended reasoning. *arXiv preprint arXiv:2512.02874*, 2025a.

P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Y. Wang, P. Ji, C. Yang, K. Li, M. Hu, J. Li, and G. Sartoretti. Mcts-judge: Test-time scaling in llm-as-a-judge for code correctness evaluation. *arXiv preprint arXiv:2502.12468*, 2025b.

X. Wen, Z. Liu, S. Zheng, S. Ye, Z. Wu, Y. Wang, Z. Xu, X. Liang, J. Li, Z. Miao, et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

K. D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J. G. Lambourne, A. Solar-Lezama, and W. Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4):1–24, 2021.

R. Wu, C. Xiao, and C. Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021.

Y. Wu, Z. Sun, S. Li, S. Welleck, and Y. Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025.

A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

J. Zenn and J. Geiping. When are likely answers right? on sequence probability and correctness in llms. *arXiv preprint arXiv:2606.27359*, 2026.

Q. Zhang, C. Hu, S. Upasani, B. Ma, F. Hong, V. Kamanuru, J. Rainton, C. Wu, M. Ji, H. Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025a.

T. Zhang, H. Shi, Y. Wang, H. Wang, X. He, Z. Li, H. Chen, L. Han, K. Xu, H. Zhang, et al. Tokur: Token-level uncertainty estimation for large language model reasoning. *arXiv preprint arXiv:2505.11737*, 2025b.

Z. Zhang, C. Zheng, Y. Wu, B. Zhang, R. Lin, B. Yu, D. Liu, J. Zhou, and J. Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025c.

S. Zhao, R. Brekelmans, A. Makhzani, and R. Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024.

S. Zhao, Z. Xie, M. Liu, J. Huang, G. Pang, F. Chen, and A. Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Z. Zhou, T. Yuhao, Z. Li, Y. Yao, L.-Z. Guo, Y.-F. Li, and X. Ma. A theoretical study on bridging internal probability and self-consistency for llm reasoning. *Advances in Neural Information Processing Systems*, 38: 87380–87413, 2026.

Contents

1 Introduction **1**

2 Background and Related Work **3**

3 Method **4**

3.1 Intrinsic Selection (iS) 4

3.2 Intrinsic Particle Filtering (iPF) 5

3.3 Particle Distillation (dPF) 6

4 Experiment **6**

4.1 Intrinsic Selection (`iS`) with Adjusted Tail Entropy 7

4.2 Intrinsic Particle Filtering (`iPF`) on Hard Problems 8

4.3 Steering Resampling with Particle Distillation (`dPF`) 9

5 Conclusion and Limitations **10**

A Extended Related Work **17**

B Verifiability **19**

B.1 The Bottleneck of Extrinsic Verification 19

B.2 Challenges in Verifiable Domains 20

C Perspective on Intrinsic Scoring and Resampling **21**

D Algorithms **23**

D.1 Intrinsic Selection 23

D.2 Intrinsic Particle Filtering 23

D.3 Particle Distillation 24

E Code for Intrinsic Selection **25**

F Intrinsic Selection and Resampling: Workflow Analysis **26**

F.1 Graphical Model 26

F.2 Workflow 26

F.3 Parallel Sampling 27

F.4 Intrinsic Selection 28

F.5 Intrinsic Selection for Hard Problems 29

F.6 Hint Generation 30

F.7 Particle Distillation for Hard Problems 30

G Problem Complexity **31**

G.1 Compute and Wall-Clock Overhead 32

H Extrinsic Reward Ablation **33**

H.1 Model-Free Selection 33

H.2 Reward Model and Hybrid Selection . . . 33

I Gemma Ablation **35**

J Additional Ablations and Experiments **36**

J.1 Budget . . . 37
J.2 Tail Entropy as a Correctness Signal . . . 38
J.3 Consensus and Confidence . . . 40
J.4 Confidence-Weighted Consensus: RPC and CISC . . . 41
J.5 Selection without the Answer . . . 42
J.6 Entropy Tail and Aggregation Ablations . . . 43
J.7 Image-to-Code CAD Scaling . . . 44
J.8 Difficulty Estimation . . . 45
J.9 Intrinsic Particle Filtering . . . 47
J.10 Particle Distillation . . . 48
J.11 Healthcare . . . 49
J.12 Agentic Tool Use: Kubernetes Troubleshooting . . . 51

K Entropy Correlation **52**

L Qualitative Analysis of Token-Level Metrics **53**

M Entropy and KL Guidance Visualizations **56**

N Examples **58**

N.1 Hard Math Problems Examples . . . 58
N.2 Confidently Wrong Selections . . . 60
N.3 HealthBench Examples . . . 61

O Details **63**

O.1 Methodological Details . . . 63
O.2 Models . . . 63
O.3 Datasets . . . 63
O.4 Evaluation Metrics and Experimental Setup . . . 64

A Extended Related Work

Domain Verifiability Appendix Table 6 contrasts verifiable and hard-to-verify domains. Verifiable fields, such as mathematics and coding, offer canonical outputs that provide cheap, automated reward signals ($V(x, y) \in \{0, 1\}$) for reinforcement learning and inference-time filtering. In contrast, extending inference-time scaling to hard-to-verify or open-ended domains - such as healthcare and engineering design - presents major challenges [Jaech et al., 2024]. These settings lack cheap verification, forcing reliance on expensive subjective rubrics, human intervention, LLM-as-a-judge pipelines [Wang et al., 2025b], or computationally heavy geometric simulators. High verification costs bottleneck scalability, and in these environments, external reward models are prone to over-optimization [Gao et al., 2023, El-Mhamdi and Hoang, 2024] and generalize poorly [Park et al., 2025]. Furthermore, practical verification remains fragile even in structured fields: heuristic parsers break easily, and solver-based validation can take hours per sample, effectively rendering them hard-to-verify at scale (Appendix B).

Inference-Time Scaling Recent advancements have demonstrated that allocating additional compute during generation, often termed test-time or inference-time scaling, significantly improves LLM reasoning performance [Snell et al., 2024, Brown et al., 2024, Wu et al., 2025]. Scaling strategies typically fall into two categories: parallel sampling, where multiple independent trajectories are generated to form a consensus, and sequential step-level searches, such as Tree of Thoughts [Yao et al., 2023] or Monte Carlo Tree Search (MCTS [Misaki et al., 2025]), which systematically explore reasoning trees. While parallel scaling is straightforward and highly effective for standard problems, step-level methods navigate high-density regions of the solution space to tackle harder tasks where correct trajectories lie in low-probability regions.

Consensus-based Selection Self-Consistency (SC) [Wang et al., 2022] is the most successful verification-free approach in modern parallel inference, marginalizing over diverse reasoning trajectories via majority voting to achieve strong results whenever answers admit a canonical form. It marginalizes latent trajectories $\{\mathbf{z}\}_{k=1}^{K} \sim p(\mathbf{z} \mid \mathbf{c})$ to approximate the conditional answer distribution:

$$p(\mathbf{x} \mid \mathbf{c}) = \sum_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z}, \mathbf{c}) p(\mathbf{z} \mid \mathbf{c})$$

In practice, systems extract consensus via majority voting over a discrete set of M choices $\{a_m\}_{m=1}^{M}$:

$$x_a^* = \arg\max_a \sum_{m=1}^{M} \mathbb{I}(a_m = a)$$

While inexpensive and robust to noise, SC is heavily confined to domains with cheap syntactic verification, which can be non-trivial even in mathematics.

Universal SC [Chen et al., 2023] addresses this via LLM-based judging, but incurs new model-based verification costs. Confidence-weighted variants such as CISC [Taubenfeld et al., 2025] and RPC [Zhou et al., 2026] weight or prune votes by model probability, but still need answers parsed into a canonical label, and how well probability and consensus track correctness varies across tasks [Zenn and Geiping, 2026].

Other works group answers by meaning to compute semantic entropy [Kuhn et al., 2023], which helps measure uncertainty but requires expensive clustering. Fundamentally, these approaches depend on output-level comparison or LLM-based parsing, limiting their applicability when outputs are long-form, multi-part, or lack a single correct answer. Our intrinsic selection sidesteps this by scoring at the token-probability level, requiring no output parsing.

Confidence-based Selection Building on early findings that base LLMs are often well-calibrated [Kadavath et al., 2022], confidence-based methods leverage the model's internal signals to rank solutions, bypassing external verifiers. Reasoning models trained via reinforcement learning [Guo et al., 2025, Jaech et al., 2024] intrinsically verify their reasoning during generation. Orthogonally, approaches like Self-Certainty [Kang et al., 2025] measure the divergence from a uniform distribution to the model's predictive distribution to gauge intrinsic confidence. For a probabilistic model defined over a discrete vocabulary $\mathcal{V}$ and a token sequence of length T, denoted as $p(\mathbf{z}) = p(\mathbf{z}_T) \prod_{t=1}^{T-1} p(\mathbf{z}_t \mid \mathbf{z}_{<t})$, we define the entropy at token t as [Shannon, 1948]:

$$\mathbb{H}[\mathbf{z}_t \mid \mathbf{z}_{<t}] = -\sum_{v \in \mathcal{V}} p(\mathbf{z}_t = v \mid \mathbf{z}_{<t}) \log p(\mathbf{z}_t = v \mid \mathbf{z}_{<t})$$

This can also be expressed as a negated KL divergence relative to a uniform distribution U:

$$\mathbb{KL}[p, U]_t = \sum_{v \in \mathcal{V}} p(\mathbf{z}_t^v) \log \frac{p(\mathbf{z}_t^v)}{1/|\mathcal{V}|} = -\mathbb{H}[p]_t + \log(|\mathcal{V}|)$$

A sequence demonstrates relative confidence when its entropy is lower, or its KL divergence from uniform is larger, compared to competing sequences. More complex pipelines, such as DeepConf [Fu et al., 2025] and

TokUR [Zhang et al., 2025b], utilize trace-level scores from token probabilities and entropy-aware generation to identify promising candidates without extrinsic rewards. Our approach complements RL-trained models; while they internalize verification, our work advances this shift by introducing a purely intrinsic, length-adjusted tail entropy metric that robustly isolates reasoning quality, generalizing these ideas to step-level resampling and extracting discriminative signals from the collective statistics of an N-sample set.

Particle-based Inference and Guidance Guided decoding seeks to iteratively steer generation using dynamic feedback. Drawing from Bayesian inference, Sequential Monte Carlo (SMC) [Doucet et al., 2001a, Zhao et al., 2024, Lew et al., 2023] methods leverage external rewards to define tilting distributions and step-level resampling. Particle Filtering (PF), a subset of SMC, aims to approximate the posterior $p(\mathbf{z} \mid \mathbf{c}, \mathbf{o}) \propto p(\mathbf{o} \mid \mathbf{z}, \mathbf{c})p(\mathbf{z} \mid \mathbf{c})$ via sequential importance resampling [Liu et al., 2001], introducing a proposal $q(\mathbf{z} \mid \mathbf{c})$:

$$w_t \propto w_{t-1} \frac{p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c}, \mathbf{o}_t)}{q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{c})}$$

Choosing the model itself as the proposal, the weight update reduces to $w_t \propto w_{t-1}\, p(\mathbf{o}_t \mid \mathbf{z}_t, \mathbf{c})$, where the observation model can be parameterized by a Process Reward Model (PRM), i.e., $r_\phi(\mathbf{z}_t, \mathbf{c}) = p(\mathbf{o}_t = 1 \mid \mathbf{z}_t, \mathbf{c})$ [Lightman et al., 2023, Puri et al., 2025]. However, training PRMs is expensive, domain-specific, and requires calibration, as they often degrade and become overconfident on long sequences outside their training distribution [Zhang et al., 2025c, Park et al., 2025]. To overcome this, recent approaches steer generation using intrinsic metrics [Wang et al., 2025a, Sanyal et al., 2025] or Contrastive Decoding [Li et al., 2022, O'Brien and Lewis, 2023]. Our logit blending approach draws a direct parallel to classifier-free guidance for language models [Sanchez et al., 2023, Ho and Salimans, 2022], interpolating an unconditional base policy p_θ and a hint-guided policy $q(\mathbf{z} \mid \mathbf{c}, \mathbf{h})$ to steer trajectories away from incorrect early assumptions without relying on external PRMs.

`iPF` allocates a finite budget among partial trajectories, as beam search [Snell et al., 2024] and Monte Carlo tree search [Misaki et al., 2025] do. Unlike beam search, it resamples stochastically instead of deterministically keeping the top-k prefixes, so a low-scoring prefix survives with non-zero probability and the population stays more diverse. Unlike MCTS, it keeps no explicit tree, visit counts, or value estimates, and it needs no external value function or rollouts. All particles advance in lockstep, which matches batched inference engines.

B Verifiability

The distinction between verifiable and non-verifiable tasks fundamentally dictates a model's capacity to scale during post-training. In verifiable domains, the availability of "cheap approximate verification"—such as executing unit tests or running mathematical solvers—provides a low-cost, high-speed mechanism for evaluating outputs. Because grading each sample costs mere fractions of a cent and milliseconds of compute, researchers can feasibly execute millions of rollouts for Reinforcement Learning (RL) or build vast search trees at inference time. Conversely, non-verifiable tasks rely on expensive and slow evaluation methods, such as human annotators or large LLM-as-a-Judge systems. This high cost acts as a strict bottleneck, making it computationally and financially prohibitive to scale post-training optimization to the same massive extent.

Table 6: Conceptual Map: Domain Verifiability.

Feature	Verifiable Domains	Hard-to-Verify Domains
Domain Examples	Engineering, Math, Reasoning, Code	Healthcare, Enterprise
Verification Methods	Solvers, Symbolic Output, Test Execution.	LLM-as-Verifier, Human Evaluation. Model-based
Core Characteristics	Output Canonicalization, Reference Solutions	No Canonical Form (Open-ended), No References (relies on rubrics)
Cost & Scalability	Cheap Approx Verification $\pi(x)$. Scaling computationally viable. Reward Models.	Expensive verification creates a strict bottleneck, severely limiting scalability.
Ranking & Selection	Allows for absolute ranking and selection based on exact correctness.	No absolute ranking or selection without verification; strictly relative preferences.
Training Dynamics	$f(x) \leftrightarrow y$. Exact verifier $V(x, y) \in \{0, 1\}$ provides the high-throughput reward signal needed for Reinforcement Learning (RL).	Optimization relies on proxy rewards. Highly challenging to scale post-training without cheap verification.
Inference Dynamics	$\pi(x) \rightarrow \hat{y}$. Allows for inference-time scaling and filtering using $V(x, \hat{y})$.	$\pi(x) \rightarrow ?$ Exact correctness is undefined or subjective at test time.

Training, Validating and Evaluating foundation models fundamentally diverges based on the intrinsic verifiability of the target task. In domains such as mathematics, software engineering, and formal reasoning, tasks are highly verifiable because they possess canonical forms and definitive reference solutions. This structure allows for inexpensive, scalable verification methods like symbolic solvers or direct test execution, where a model trained on a deterministic mapping $f(x) \leftrightarrow y$ produces a readily testable inference output $\pi(x) \rightarrow \hat{y}$. Conversely, complex applications in healthcare and enterprise contexts push evaluation beyond traditional verifiability. These hard-to-verify tasks typically lack exact reference solutions or canonical forms, relying instead on open-ended generation or relative preferences where the definitive correctness of an inference output remains ambiguous ($\pi(x) \rightarrow ?$). Because these subjective domains preclude cheap, programmatic approximate verification, evaluating model performance necessitates highly expensive, rubric-driven mechanisms, primarily relying on expert human evaluators or LLM-as-a-Judge frameworks.

B.1 The Bottleneck of Extrinsic Verification

Relying on extrinsic verification presents significant limitations for scaling. Training PRMs and ORMs is expensive [Zhang et al., 2025c], require calibration [Park et al., 2025], and become increasingly difficult to build as the underlying models and reasoning budgets grow stronger. Furthermore, rubric-driven reward models generalize poorly and fail entirely on the hardest problems where the base success rate is zero. Most importantly, extrinsic verification is largely unavailable for open-ended or non-verifiable domains.

To understand this bottleneck, we categorize solution domains by their verification complexity (Table 40). Traditionally, domains are considered verifiable if a post-processing routine can extract a canonical representation for the generated output and if a reference solution for the given problem exists, or non-verifiable/unverifiable if no such routine exists. However, verifiability is ultimately a function of available resources and compute [2].

A more practical distinction is between easy-to-verify domains at scale, where checking a solution is much cheaper than generating it, and hard-to-verify domains at scale, where the verification process itself is complex, computationally expensive, or requires expert judgment. Doing so, we can think of domain verification as a

[2] Notice that in practice almost nothing is strictly unverifiable if enough time, human expertise, and computing power are available. The true constraint for scaling is the cost, automation level, and availability of verification at training and validation time. We will see examples of such problem in engineering.

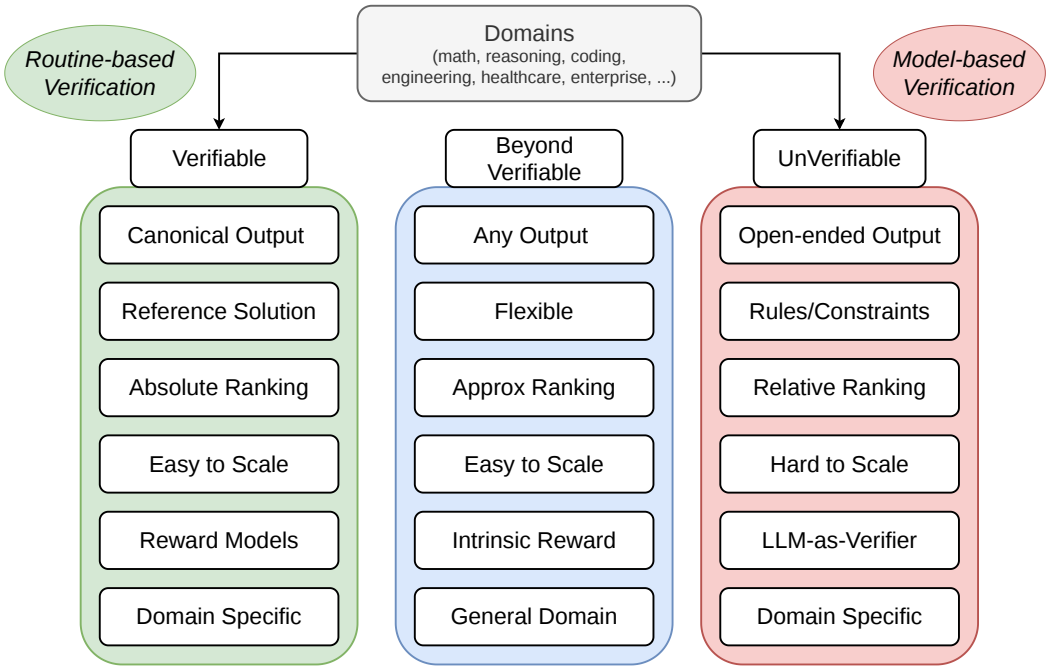


Figure 8: Beyond Domain Verifiability. We would like to build ITS methods that can approximate verifiability for non-verifiable domains, without requiring human intervention or expensive extrinsic reward.

spectrum. For example, evaluating the aerodynamics of an engineering design is technically verifiable (and deterministic) through a fluid dynamics simulation, making it easy-to-verify at the prototype level. However, running a high-fidelity simulation takes hours and massive compute, making it act as a hard-to-verify domain at scale, when there is need to evaluate thousand parallel designs s of generated by a custom foundation model. Such compute requirements make large-scale RL training or model validation hard to scale, making extrinsic verification a bottleneck.

B.2 Challenges in Verifiable Domains

Even among domains traditionally considered verifiable, practical implementation can be challenging. Consensus algorithms [Wang et al., 2022, Fu et al., 2025] rely heavily on symbolic and output verification, meaning solutions must be parsed and matched at the string level. In mathematics, this requires post-processing tools like `math_verify`[3] to normalize solutions into canonical forms. These heuristic-based parsers are often brittle, and differing implementations, model behaviors or edge cases can drastically alter results, forcing practitioners to rely on custom solutions.

Other verifiable domains present their own distinct hurdles. Coding requires access to executable test environments, while engineering requires expensive solvers or slow simulators that limit large-scale verification. Among hard-to-verify domains like healthcare and enterprise tool calling, progress has been even slower. Verification in these areas requires checking for correctness at the trajectory level, rubric-informed scoring, safety requirements, and open-ended nuances, which demands expensive human-in-the-loop systems or complex LLM-judge pipelines. Finally, even when output verification is cheap and easy, verifying intermediate steps (reasoning trajectories, mathematical proofs) can still be challenging and require large amount of compute and specialized post-processing pipelines.

For these reasons, a general method that can approximate verification at inference time would be of great utility, irrespective of the level of verifiability of the underlying domain.

[3] `https://github.com/huggingface/Math-Verify`

C Perspective on Intrinsic Scoring and Resampling

This section provides a compact interpretation of the intrinsic scoring and resampling rules used by iS, iPF, and dPF. The goal is not to derive a formal guarantee of correctness, but to clarify the statistical intuition behind the surrogate quantities used in the main method.

Let $\mathbf{c}$ denote a problem instance and let $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_T)$ be a sampled trajectory from the base model $p(\mathbf{z} \mid \mathbf{c})$. Recall that the token-level entropy is:

$$\mathbb{H}_t(\mathbf{z}) = \mathbb{H}(\mathbf{z}_t \mid \mathbf{z}_{<t}, \mathbf{c}). \tag{14}$$

For `iS`, the method scores each sampled trajectory using an aggregated entropy over an adaptively chosen suffix region. Writing $A(z) \subseteq \{1, \ldots, T\}$ for the tail index set selected by the procedure in Section 3, the resulting score is

$$S(\mathbf{z}) = \mathrm{agg}_{t \in A(\mathbf{z})} \mathbb{H}_t(\mathbf{z}), \tag{15}$$

where agg is the same robust aggregation operator used in the manuscript.

The underlying intuition is that, for many reasoning tasks, the final answer region is more informative than the earlier exploratory portion of the trajectory. Under this view, lower suffix entropy serves as a proxy for local confidence on the part of the sequence most relevant to the final output. This is precisely the quantity used by `iS` to rank candidates.

The adaptive tail construction can be interpreted as follows: if the tail window is too long, the score is contaminated by earlier reasoning tokens that are weakly related to the final answer. If the window is too short, the resulting estimate of suffix confidence becomes noisy. The heuristic cutoff used in the paper therefore aims to select a region that is long enough to stabilize the score while remaining focused on the answer-bearing suffix.

This same perspective also helps explain the benefit of response-length filtering and robust aggregation. In practice, sampled trajectories may include pathological short or long traces, for example due to truncation, formatting failures, or runaway reasoning. Trimming such outliers before estimating set-level statistics reduces distortion in the tail estimate, while median-style aggregation reduces sensitivity to localized entropy spikes.

For `iPF`, the method replaces an external reward model with entropy-based pseudo-observations during resampling. A useful way to interpret this update is as sampling from a confidence-tilted trajectory distribution of the form

$$\tilde{p}(\mathbf{z} \mid \mathbf{c}) \propto p(\mathbf{z} \mid \mathbf{c}) \exp\left(-\lambda \sum_{t=1}^{T} \mathbb{H}_t(\mathbf{z})\right), \tag{16}$$

where $\mathbb{H}_t(\mathbf{z})$ denotes the standardized last step entropy statistic used in the resampling rule, and $\lambda > 0$ is an implicit temperature induced by normalization.

This interpretation should be read as heuristic rather than exact. The paper does not claim that iPF recovers a true posterior over correct trajectories, and the resampling weights are not calibrated likelihoods. Instead, iPF biases computation toward continuations that remain locally stable under the base model, which is consistent with its empirical role on problems where standard sampling occasionally finds the correct direction but fails to sustain it.

What `iPF` targets If the entropy weights were applied with a fixed strength, `iPF` would be a standard bootstrap particle filter [Chopin et al., 2020]. It would sample from the base model tilted toward low-entropy trajectories, and its normalizing constant could be estimated without bias. That constant measures how confident the base model is on the problem, not how likely it is to be correct, because entropy is not a likelihood of correctness. In practice, `iPF` standardizes the entropy scores within each particle batch (Eq. 10). This sets the strength of the tilt from the current population, similar to adaptive tempering: it keeps one particle from dominating when entropies are spread out and sharpens selection when they are close. Like other adaptive SMC schemes, this choice gives up the standard unbiasedness guarantee [Beskos et al., 2016]. We therefore present `iPF` as a practical heuristic that moves compute toward lower-entropy prefixes. It does not estimate task success or sample from a calibrated posterior over correct trajectories.

Relation to twisted SMC Zhao et al. [2024] view language-model inference as sampling from the base model reweighted by a reward on complete sequences. They learn twist functions that estimate, at each step, how promising a partial sequence is for that final reward. `iPF` has neither a final reward nor learned twists: it reweights each step using only the entropy of the most recent tokens. This makes `iPF` cheap and training-free, since it needs no reward model and reuses logits the sampler already computes. The cost is that its intermediate steps carry no guarantee of anticipating a correct final answer. A more principled alternative would define the target over sets of candidates rather than individual trajectories, matching the set-level statistics that `iS` and the difficulty gate already use. We leave this to future work.

For dPF, the same logic applies with privileged guidance. After the initial logit-blending phase, the method performs resampling using KL divergence between the base policy and a guide conditioned on hints or rubric

information. This can be interpreted as favoring trajectories that stay close to a guide-aware policy during the most consequential part of generation:

$$\tilde{p}_h(\mathbf{z} \mid \mathbf{c}, \mathbf{h}) \propto p(\mathbf{z} \mid \mathbf{c}) \exp\left(-\lambda \sum_{t=1}^{T} D_{\mathrm{KL}}(p_t(\cdot \mid \mathbf{z}_{<t}, \mathbf{c}) \,\|\, q_t(\cdot \mid \mathbf{z}_{<t}, \mathbf{c}, \mathbf{h}))\right), \tag{17}$$

where q_t denotes the guide policy conditioned on privileged information $\mathbf{h}$. Notice that the guide q_t is the policy model p_θ conditioned on in-context information.

Again, this expression is best understood as an interpretation of the steering effect rather than a formal target distribution. In particular, dPF does not attempt exact imitation of the guide at every step. Rather, it preferentially retains trajectories whose local decisions remain compatible with useful guidance early enough to avoid systematic failure.

Taken together, these interpretations clarify the intended operating regime of the framework. Intrinsic selection is most natural when suffix-level uncertainty is informative about answer quality, while entropy-guided or KL-guided resampling is most useful when additional inference-time compute can concentrate mass on promising reasoning paths. Conversely, when confidence is poorly aligned with quality or when trajectories have little variation in their answer-bearing region, the intrinsic signal may be weaker, which is consistent with the empirical limitations discussed in the manuscript.

Answer Region Scaling The adaptive tail cutoff, t_c, is designed to isolate the concluding answer region without incorporating noise from the preceding exploratory reasoning trace. For a generated trajectory of total length L, two scaling properties emerge: the total Shannon entropy grows linearly, $\mathcal{O}(L)$, reflecting the cumulative information volume, whereas the geometric dispersion in the latent space scales sublinearly, $\|\mathbf{z}_{\text{answer}}\|_2 \approx \sigma\sqrt{L}$, following standard concentration bounds for independent increments [Vershynin, 2018]. Because our intrinsic scoring mechanism ranks samples based on their relative separability within the entropy landscape - a geometric property rather than a strictly volumetric one - we adopt the sublinear scaling factor, establishing $t_c \propto \sqrt{L}$.

Preventing Premature Collapse in `iPF` Standard sequential particle filtering is notoriously susceptible to sample impoverishment, wherein resampling dynamics greedily trap the generation process in sub-optimal local states [Doucet et al., 2001a,b]. This risk is particularly acute in our verifier-free setting, as intrinsic entropy signals are inherently noisier than explicit, calibrated reward models. A single transient drop in entropy could theoretically force the particle set to collapse entirely around an incomplete or flawed reasoning path. To maintain broad exploration over long generation horizons, we modify the resampling distribution to dynamically normalize weights and explicitly detect absorbing configurations. By intervening before particle diversity catastrophically collapses, our intrinsic particle filtering (`iPF`) preserves alternative reasoning pathways without requiring computationally expensive Markov Chain Monte Carlo (MCMC) rejuvenation steps [Hastings, 1970] or rigid thresholding algorithms [Gilks et al., 1995], thus fully retaining the verifier-free benefits of intrinsic scoring [Lightman et al., 2023, Wang et al., 2024].

D Algorithms

D.1 Intrinsic Selection

Algorithm 1 Intrinsic Selection (`iS`)

Require: Problem $\mathbf{c}$, model p_θ, sample budget N
Ensure: Selected solution $\mathbf{z}^*$

// Parallel generation
for $n = 1, \dots, N$ **do**
  Generate $\mathbf{z}_n \sim p_\theta(\cdot \mid \mathbf{c})$
  Record $l_n \leftarrow l(\mathbf{z}_n)$ and $\{\mathbb{H}[\mathbf{z}_{n,t} \mid \mathbf{z}_{n,<t}]\}_{t=1}^{l_n}$
end for
// Adaptive tail window
$\bar{l} \leftarrow N^{-1} \sum_n l_n$ ▷ mean length
$\bar{h} \leftarrow N^{-1} \sum_n \big(l_n^{-1} \sum_t \mathbb{H}[\mathbf{z}_{n,t}]\big)$ ▷ mean entropy
$t_c \leftarrow \lfloor \sqrt{\bar{l}} \,/\, \bar{h} \rfloor$ ▷ tail cutoff (Eq. 6)
// Score and select
for $n = 1, \dots, N$ **do**
  $\mathcal{A}_n \leftarrow \{t \in \mathbb{Z} \mid l_n - t_c < t \le l_n\}$ ▷ tail index (Eq. 7)
  $s_n \leftarrow \mathrm{agg}_{t \in \mathcal{A}_n} \; \mathbb{H}[\mathbf{z}_{n,t} \mid \mathbf{z}_{n,<t}]$
end for
return $\mathbf{z}^* \leftarrow \mathbf{z}_{n^*}$ where $n^* = \arg\min_n \; s_n$ ▷ lowest tail entropy

D.2 Intrinsic Particle Filtering

Algorithm 2 Intrinsic Particle Filtering (`iPF`)

Require: Problem $\mathbf{c}$, model p_θ, particles N, max steps S, tokens per step B
Ensure: Refined particle set $\{\mathbf{z}_n\}_{n=1}^N$

$w_n \leftarrow 1/N$ for all n
for $t = 1, \dots, S$ **do**
  for $n = 1, \dots, N$ **do**
    Generate B tokens: $\mathbf{z}_n^{(t)} \sim p_\theta(\cdot \mid \mathbf{z}_n^{(<t)}, \mathbf{c})$
    $h_n \leftarrow \mathrm{agg}_{b \in [B]} \; \mathbb{H}[\mathbf{z}_{n,b}^{(t)} \mid \mathbf{z}_{n,<b}^{(t)}]$ ▷ step entropy
  end for
  $\tilde{h}_n \leftarrow (h_n - \bar{h})/(\mathrm{std}(h) + \epsilon)$ for all n ▷ z-score
  $w_n \leftarrow \sigma(-\tilde{h}_n)$ ▷ weight update (Eq. 9)
  Resample indices $\{n'\}$ from $\mathrm{Cat}(w_1, \dots, w_N)$
  $\mathbf{z}_n^{(\le t)} \leftarrow \mathbf{z}_{n'}^{(\le t)}$ for all n
end for
return $\{\mathbf{z}_n\}_{n=1}^N$;

D.3 Particle Distillation

Algorithm 3 Particle Distillation (dPF)

Require: Problem $\mathbf{c}$, hint $\mathbf{h}$, model p_θ, guide $q(\cdot \mid \mathbf{c}, \mathbf{h})$, particles N, steps S, blend steps S_b, tokens per step B
Ensure: Steered particle set $\{\mathbf{z}_n\}_{n=1}^N$

Initialize N particles; $w_n \leftarrow 1/N$ for all n
for $t = 1, \ldots, S$ **do**
  if $t \leq S_b$ **then** ▷ Phase 1: logit blending
    $\alpha_t \leftarrow 1 - t \,/\, S_b$ ▷ anneal from guide toward base (Eq. 11)
    for $n = 1, \ldots, N$ **do**
      $\ell_n \leftarrow (1 - \alpha_t) \cdot \mathrm{logits}_{p_\theta}(\mathbf{z}_n, \mathbf{c}) + \alpha_t \cdot \mathrm{logits}_q(\mathbf{z}_n, \mathbf{c}, \mathbf{h})$
      Sample B tokens from $\mathrm{softmax}(\ell_n)$
    end for
  else ▷ Phase 2: KL-guided resampling
    for $n = 1, \ldots, N$ **do**
      Generate B tokens from $p_\theta(\cdot \mid \mathbf{z}_n^{(<t)}, \mathbf{c})$
      $d_n \leftarrow \mathbb{KL}\big[p_\theta(\mathbf{z}_n^{(t)} \mid \mathbf{z}_n^{(<t)}, \mathbf{c}) \,\|\, q(\mathbf{z}_n^{(t)} \mid \mathbf{z}_n^{(<t)}, \mathbf{c}, \mathbf{h})\big]$
    end for
    $\tilde{d}_n \leftarrow (d_n - \bar{d})/(\mathrm{std}(d) + \epsilon)$ for all n
    $w_n \leftarrow \sigma(-\tilde{d}_n)$ ▷ cumulative weight update (Eq. 12)
    Resample from $\mathrm{Cat}(w_1, \ldots, w_N)$
  end if
end for
return $\{\mathbf{z}_n\}_{n=1}^N$

E Code for Intrinsic Selection

```python
import numpy as np

def adaptive_tail_window(
    entropies: list[list[float]], included: list[int],
    tail_min: int = 64, tail_max: int = 2048,
) -> int:
    lengths: list[int] = [len(entropies[i]) for i in included]
    mean_len: float = float(np.mean(lengths))
    mean_ent: float = float(np.mean([
        np.mean(entropies[i])
        for i in included if entropies[i]]))
    mean_ent = max(mean_ent, 1e-6)
    return int(np.clip(
        np.sqrt(mean_len) / mean_ent, tail_min, tail_max))

def trim_length_outliers(
    lengths: list[int], trim_pct: float = 0.05,
) -> list[int]:
    n: int = len(lengths)
    if n < 16:
        return list(range(n))
    lo = float(np.percentile(lengths, trim_pct * 100))
    hi = float(np.percentile(lengths, (1-trim_pct) * 100))
    inc = [i for i in range(n) if lo <= lengths[i] <= hi]
    return inc if len(inc) >= 2 else list(range(n))

def tail_scores(
    entropies: list[list[float]], included: list[int],
    tail: int, agg: str = "median",
) -> list[float]:
    inc: set[int] = set(included)
    fn = np.median if agg == "median" else np.mean
    scores: list[float] = []
    for i in range(len(entropies)):
        if i not in inc or not entropies[i]:
            scores.append(float("inf"))
        else:
            scores.append(float(fn(entropies[i][-tail:])))
    return scores

def select_by_tail_entropy(
    entropies: list[list[float]],
    tail_min: int = 64, tail_max: int = 2048,
    agg: str = "median",
) -> tuple[int, list[float]]:
    lengths: list[int] = [len(e) for e in entropies]
    included = trim_length_outliers(lengths)
    tail = adaptive_tail_window(
        entropies, included, tail_min, tail_max)
    scores = tail_scores(entropies, included, tail, agg)
    return int(np.argmin(scores)), scores
```

Listing 1: iS@1 selection. Given N candidate responses with per-token entropy, select the one with lowest aggregated entropy in an adaptive tail window.

F Intrinsic Selection and Resampling: Workflow Analysis

F.1 Graphical Model

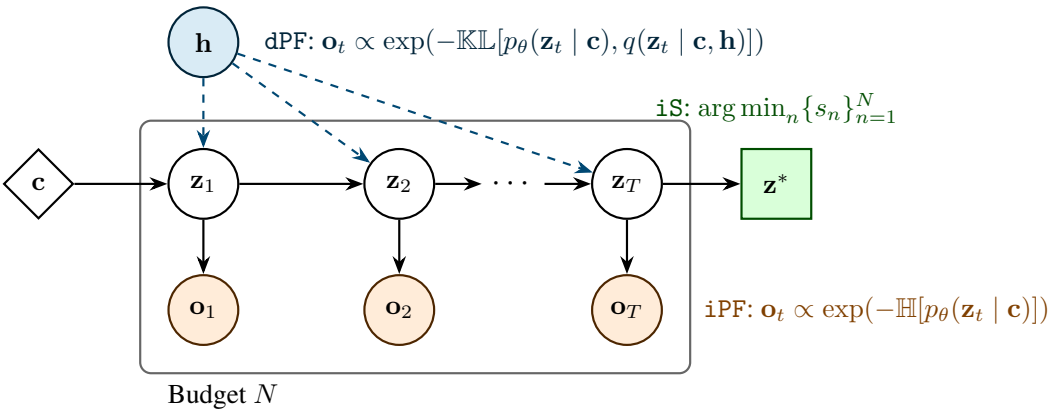


Figure 9: Probabilistic graphical model for our intrinsic selection and resampling methods. Step trajectories $\mathbf{z}_1, \ldots, \mathbf{z}_T$ are generated conditioned on the prompt $\mathbf{c}$. At each step t, intrinsic pseudo-observations $\mathbf{o}_t$ (gray nodes) are attached to the latents, representing step-wise intrinsic signals used by **iPF**. The hint node $\mathbf{h}$ (blue) injects privileged information that guides early latent variables through dashed arrows, corresponding to **dPF**. The selection node $\mathbf{z}^*$ (green) denotes the chosen trajectory at the end of the chain, as in **iS**.

F.2 Workflow

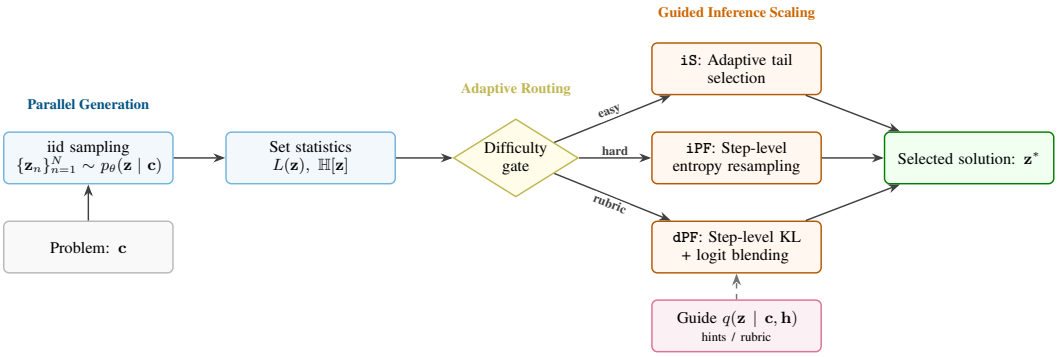


Figure 10: **Adaptive Intrinsic Inference Scaling Pipeline.** Given a problem $\mathbf{c}$, set-level statistics from an initial parallel pass provide a difficulty gate that dynamically routes compute. Easy problems are resolved instantly via Intrinsic Selection (iS). Hard problems trigger step-level Intrinsic Particle Filtering (iPF) driven by entropy. Tasks requiring specific adherence to multidimensional criteria utilize Particle Distillation (dPF) guided by privileged hints or rubrics. All paths yield a refined solution $\mathbf{z}^*$ without external reward models.

F.3 Parallel Sampling

Table 7: AIME 2024 Statistics (Hard problems: 4/30) sorted by number of correct solutions. Notice how methods based on Self-Consistency are not robust to syntactic output diversity, and cases where $pass@N = 1$ but $pass@1 \rightarrow n/N$ with $n << N$ and $n \in [1, N/2]$. In these cases, most selection methods break and $top@k \rightarrow 0$ for $k = \{1, 2, 3\}$. We define as hard problems where $pass@1 \rightarrow n/N$, with $n = \{0, 1, 2\}$. Notice how the hardness of a problem (under the model distribution) is strongly correlated with $pass@1$ (extrinsic metric) and inversely correlated with adjusted tail entropy (intrinsic metric). When verification is not available or hard, we can use the tail entropy as a proxy for hardness, without the need or external verifiers or ground truth.

uid	n_correct	n_samples	pass@1	pass@N	is_hard
2024-I-12	0	32	0.000	0	True
2024-I-11	0	32	0.000	0	True
2024-II-15	0	32	0.000	0	True
2024-II-9	1	32	0.031	1	True
2024-II-10	3	32	0.094	1	False
2024-II-8	3	32	0.094	1	False
2024-I-8	3	32	0.094	1	False
2024-I-9	5	32	0.156	1	False
2024-I-10	10	32	0.312	1	False
2024-I-13	10	32	0.312	1	False
2024-II-11	11	32	0.344	1	False
2024-I-14	13	32	0.406	1	False
2024-II-12	14	32	0.438	1	False
2024-II-7	15	32	0.469	1	False
2024-II-14	18	32	0.562	1	False
2024-II-3	22	32	0.688	1	False
2024-II-1	25	32	0.781	1	False
2024-II-13	26	32	0.812	1	False
2024-II-2	26	32	0.812	1	False
2024-II-5	28	32	0.875	1	False
2024-I-5	28	32	0.875	1	False
2024-I-7	28	32	0.875	1	False
2024-II-6	29	32	0.906	1	False
2024-I-1	30	32	0.938	1	False
2024-I-6	31	32	0.969	1	False
2024-I-3	31	32	0.969	1	False
2024-I-4	31	32	0.969	1	False
2024-I-2	32	32	1.000	1	False
2024-I-15	32	32	1.000	1	False
2024-II-4	32	32	1.000	1	False

F.4 Intrinsic Selection

Table 8: Detailed Problem Results. We rank each problem by adjusted mean tail entropy. We can see that there is strong correlation between low pass@1 (pass_rate), hardness of the problem, and increase in mean tail entropy. So we can use such quantity (for example selecting the worst 10/20% of the samples) as proxy for problem complexity, select the hard problems and use more compute for inference-time scaling (larger N, step-level steering, or using demonstrations).

				Entropy			Certainty			Length			
uid	is_hard	n_correct	pass_rate	Mean	Std	Sum Mean	Mean	Std	Sum Mean	Mean	Std	CV	n_samples
2024-II-9	True	1	0.03	0.36	0.10	506.08	28.27	2.14	39345.19	24162.72	5319.21	0.22	32
2024-II-15	True	0	0.00	0.35	0.06	486.94	28.51	1.86	39690.78	23192.78	3515.64	0.15	32
2024-I-12	True	0	0.00	0.32	0.08	451.34	28.94	1.49	40288.83	19258.34	1208.11	0.06	32
2024-I-11	True	0	0.00	0.29	0.10	406.70	28.10	3.07	39119.66	25822.25	3908.03	0.15	32
2024-I-8	False	3	0.09	0.27	0.07	379.02	28.12	2.44	39149.96	26822.72	2417.06	0.09	32
2024-I-6	False	31	0.97	0.27	0.04	355.44	34.18	1.47	45486.80	4343.81	649.49	0.15	32
2024-II-1	False	25	0.78	0.26	0.04	316.24	33.26	1.43	40237.73	8421.97	4786.78	0.57	32
2024-I-15	False	32	1.00	0.26	0.05	359.33	29.52	2.42	41094.30	18204.16	1822.06	0.10	32
2024-II-8	False	3	0.09	0.26	0.07	357.10	28.24	4.04	39316.17	26011.81	3266.31	0.13	32
2024-II-2	False	26	0.81	0.26	0.05	356.77	31.01	2.78	43168.00	16381.31	5945.18	0.36	32
2024-II-5	False	28	0.88	0.25	0.08	354.26	33.52	2.54	46657.12	13180.72	4949.95	0.38	32
2024-I-5	False	28	0.88	0.24	0.10	339.43	34.07	3.38	47428.58	12911.13	3516.81	0.27	32
2024-I-9	False	5	0.16	0.24	0.08	177.87	33.98	2.69	25441.03	3821.31	4808.08	1.26	32
2024-II-11	False	11	0.34	0.23	0.05	326.24	32.72	2.68	45550.70	17319.03	2986.65	0.17	32
2024-II-14	False	18	0.56	0.22	0.05	306.68	30.29	1.42	42165.72	19931.22	1763.68	0.09	32
2024-I-4	False	31	0.97	0.21	0.03	256.61	35.09	0.79	41620.87	3603.72	611.90	0.17	32
2024-II-7	False	15	0.47	0.20	0.05	275.74	29.95	3.02	41689.50	19006.72	3434.10	0.18	32
2024-II-3	False	22	0.69	0.19	0.07	266.54	34.73	2.86	48350.23	11397.84	6344.92	0.56	32
2024-II-10	False	3	0.09	0.19	0.07	266.02	31.44	3.31	43762.68	17632.31	3025.67	0.17	32
2024-I-13	False	10	0.31	0.18	0.05	253.45	35.12	1.59	48670.15	11595.38	4451.38	0.38	32
2024-I-3	False	31	0.97	0.17	0.03	239.67	34.97	0.70	48684.48	12489.47	2042.10	0.16	32
2024-I-10	False	10	0.31	0.17	0.07	234.81	33.48	4.42	46601.31	20580.13	2871.57	0.14	32
2024-II-12	False	14	0.44	0.15	0.07	206.22	33.24	4.95	46274.16	18409.88	5885.25	0.32	32
2024-II-6	False	29	0.91	0.14	0.03	195.71	39.17	1.06	54521.82	5705.25	1973.39	0.35	32
2024-I-2	False	32	1.00	0.13	0.05	172.35	38.52	2.38	46360.82	5611.31	3042.39	0.54	32
2024-II-13	False	26	0.81	0.11	0.03	152.72	37.99	2.34	52887.98	9952.16	3460.27	0.35	32
2024-I-14	False	13	0.41	0.11	0.06	149.41	38.58	3.29	53706.37	11407.22	4367.01	0.38	32
2024-I-1	False	30	0.94	0.09	0.02	119.32	40.58	0.73	54157.97	3566.91	989.76	0.28	32
2024-I-7	False	28	0.88	0.09	0.02	118.09	40.32	0.82	53401.82	3110.28	565.81	0.18	32
2024-II-4	False	32	1.00	0.06	0.02	78.75	42.47	0.81	58968.78	3775.69	559.02	0.15	32

F.5 Intrinsic Selection for Hard Problems

Table 9: Entropy statistics and correct answer tracking across particles for problem `2024-II-9`. For this problem, extrinsic and intrinsic selection methods (self-consistency, self-certainty, entropy, output reward models) fail. This problem is hard to solve using inference-time scaling alone.

				Entropy			
particle	correct	answer	n_tokens	Mean	Tail Mean	Tail Median	Tail p90
0	False	1022	8032	0.3721	0.2417	0.0015	0.9007
1	False	2	8039	0.4095	0.2727	0.0092	0.9706
2	False	252	3667	0.3079	0.2819	0.0002	1.0730
3	False	32282	9359	0.3766	0.2870	0.0050	1.0191
4	False	900	8850	0.4270	0.3048	0.0090	1.0742
5	False	1022	6710	0.4122	0.3098	0.0042	1.1130
6	False	34	7300	0.4253	0.3152	0.0121	1.0785
7	False	900	6151	0.4498	0.3191	0.0066	1.0749
8	False	570	7051	0.4462	0.3211	0.0039	1.1135
9	False	1022	9017	0.4052	0.3249	0.0597	1.0323
10	False	512	10050	0.4236	0.3262	0.0423	1.0472
11	True	902	7397	0.4526	0.3388	0.0217	1.1343
12	False	1142	8570	0.4218	0.3445	0.0149	1.1611
13	False	900	6942	0.4267	0.3465	0.0294	1.1598
14	False	1022	8525	0.4308	0.3594	0.0754	1.1023
15	False	122	7711	0.4096	0.3639	0.0841	1.1132
16	False	482	5072	0.4187	0.3850	0.0637	1.2463
17	False	2	7035	0.5032	0.3983	0.1442	1.1851
18	False	126	6832	0.4476	0.4052	0.0963	1.2312
19	False	2	8636	0.3920	0.4109	0.1091	1.2209
20	False	2	6848	0.4688	0.4329	0.1629	1.2490
21	False	2	5104	0.4701	0.4463	0.1606	1.2983
22	False	1922	5162	0.4756	0.4521	0.1641	1.3111
23	False	1022	5315	0.4892	0.4535	0.1660	1.3235
24	False	1024	7999	0.4447	0.4540	0.2047	1.2958
25	False	2	5939	0.4547	0.4559	0.1638	1.3046
26	False	2	5525	0.5062	0.4654	0.2123	1.3222
27	False	1024	6274	0.4634	0.4743	0.2272	1.3574
28	False	2	6083	0.4666	0.4900	0.2468	1.3810
29	False	2	5819	0.5136	0.4939	0.2955	1.3238
30	False	2	4633	0.4789	0.4945	0.2297	1.4065
31	False	2	4186	0.5228	0.5481	0.3036	1.5093

F.6 Hint Generation

The guide $q(\mathbf{z}|\mathbf{c}, \mathbf{h})$ sees the problem $\mathbf{c}$ as conditioning (exactly as the model we are sampling)

Problem AIME `2024-II-9`

There is a collection of 25 indistinguishable white chips and 25 indistinguishable black chips. Find the number of ways to place some of these chips in the 25 unit cells of a 5×5 grid such that: each cell contains at most one chip all chips in the same row and all chips in the same column have the same colour any additional chip placed on the grid would violate one or more of the previous two conditions.

and privileged information $\mathbf{h}$ concatenated, in the form of an hint or procedure for critical issues in the problem:

Hint for Problem AIME `2024-II-9`

<hint> The key insight is that maximality forces every row and every column to be non-empty, meaning each is assigned exactly one color. A cell gets a chip if and only if its row and column share the same color. So the problem reduces to choosing subsets of rows and columns to be 'white' vs 'black', with edge cases when all rows are the same color.

Each row must be uniformly white or black (not empty, by maximality), and similarly for columns. Consider how choosing which rows are white and which columns are white determines the entire chip placement. </hint>

F.7 Particle Distillation for Hard Problems

Table 10: Entropy statistics and correct answer tracking for particle distillation. We use logits blending for the first 25% of the sampling steps with $\alpha = 0.7$ and linear annealing schedule: $\alpha\ \texttt{logits}(\mathbf{z}, \mathbf{c}, \mathbf{h}) + (1 - \alpha)\ \texttt{logits}(\mathbf{z}, \mathbf{c})$.

				Entropy			
particle	correct	answer	n_tokens	Mean	Tail Mean	Tail Median	Tail p90
0	True	902	8189	0.3866	0.2576	0.0024	0.9026
1	True	902	8189	0.3866	0.2576	0.0024	0.9026
2	False	1022	9710	0.4293	0.2896	0.0260	0.9853
3	False	1022	4899	0.4882	0.4379	0.1297	1.2943
4	False	1022	4899	0.4882	0.4379	0.1297	1.2943
5	False	1022	4899	0.4882	0.4379	0.1297	1.2943
6	False	1022	4899	0.4882	0.4379	0.1297	1.2943
7	False	1022	4899	0.4882	0.4379	0.1297	1.2943

G Problem Complexity

To accurately model how inference-time scaling (ITS) behaves across diverse tasks, we must understand the relationship between a model's inherent capacity and its specific failure modes. We categorize problem complexity into three distinct scenarios to better allocate computational resources and determine the most effective scaling strategy. Assuming p_θ represents the language model sampler, N is the maximum parallel sampling budget, and n is the number of correct solutions generated within that budget, we classify tasks as follows:

- **Level I: Solved-Easy** ($n/N \approx k$, pass@N=1). In this regime, the correct solution resides in high-density regions of the model's probability space. Finding the answer requires minimal exploration because the model's default policy naturally aligns with the correct reasoning path. Consensus-based algorithms, such as self-consistency or majority voting, perform optimally here. Because the success rate k represents a sizable portion of the compute budget, standard parallel sampling is highly efficient without requiring intermediate interventions.
- **Level II: Solved-Hard** ($n/N \ll k$, pass@N=1). Correct trajectories exist for these tasks, but they are sparse and buried in low-density regions of the probability space. While a solution is theoretically reachable, relying on blind parallel scaling is computationally inefficient and akin to finding a needle in a haystack. These cases necessitate step-level guidance, tree-search algorithms, and intermediate resampling (such as `iPF`). By evaluating partial trajectories, these methods dynamically concentrate the compute budget on high-confidence reasoning paths rather than wasting resources on dead ends.
- **Level III: Systematic Failure** ($n/N \approx 0$, pass@$N \approx 0$). Under this condition, the model completely fails to generate correct solutions due to **faulty initial assumptions** or an inability to satisfy **multidimensional constraints**. The model's policy becomes trapped in logically flawed or hallucinated reasoning regions, rendering both parallel scaling and internal verification entirely ineffective. Overcoming systematic failure requires privileged steering via `dPF` or external verifiers to actively realign the generation process. Domain-specific rubrics, expert hints, or environmental feedback are necessary to break the model out of its incorrect prior distributions.

G.1 Compute and Wall-Clock Overhead

A major advantage of using intrinsic model statistics over external reward models or geometric solvers is drastically reduced verification latency. We quantify the wall-clock overhead of our intrinsic methods against a standard unguided parallel generation baseline (`pass@`N). Both methods rely solely on the base model's top-10 log-probabilities natively provided by vLLM. Neither requires an external reward model, verifier, or additional GPU.

As shown in Table 11, Intrinsic Selection (`iS`) adds nearly zero overhead ($< 1\%$). It computes adaptive tail entropy in microseconds using highly optimized, CPU-only vectorized operations on vLLM's existing token log-probabilities. Intrinsic Particle Filtering (`iPF`) introduces an approximately $2\times$ overhead (105%). Crucially, this stems from the sequential nature of step-level resampling rather than an increased token budget. Both intrinsic approaches remain substantially faster than execution-based verification, which can consume up to 40% of pipeline time in domains like engineering design.

Token budget equivalence `iPF@`N, `dPF@`N, and `pass@`N share the same total token budget. For $N = 16$ particles averaging 64 rounds of 128 tokens, the effective budget is roughly 100K tokens (accounting for early termination at an average of $\bar{n}_{\text{active}} = 12.2$ out of 16). This matches the 96K tokens used by `pass@16`. Thus, the overhead for step-level intrinsic methods is purely architectural.

iPF $\approx$ pass@$2N$ in wall-clock Because `iPF@16` requires roughly the same wall-clock time as `pass@32`, a fair practical evaluation compares `iPF@16` against `iS@32` to determine if particle filtering offers benefits beyond simply doubling the sample budget.

Table 11: Estimated wall-clock latency per problem on math benchmarks (avg. response length 6000 tokens). Model: Qwen3-4B on a single H100. Timings reflect the full generation and selection loop. Token budgets for iPF@16, dPF@16, and pass@16 are matched (100K tokens). iPF overhead arises from sequential step generation, while dPF incurs additional KL scoring overhead.

Method	Budget (N)	Avg. Wall-Clock	Overhead	Primary Cost
pass@128 (Baseline)	128	268 s	—	Parallel token generation
`iS@128`	128	268 s	$< 1\%$	Vectorized entropy (CPU)
pass@32 (Baseline)	32	67 s	—	Parallel token generation
`iS@32`	32	67 s	$< 1\%$	Vectorized entropy (CPU)
pass@16 (Baseline)	16	33 s	—	Parallel token generation
`iS@16`	16	33 s	$< 1\%$	Vectorized entropy (CPU)
`iPF@16`	16	68 s	105%	Sequential step generation
`dPF@16`	16	85 s	155%	Sequential step generation + KL scoring

Overhead decomposition for iPF and dPF The 105% overhead of `iPF@16` compared to `pass@16` decomposes into three sources, estimated from 1,890 generation rounds. The primary bottleneck is short-batch inefficiency (60%), as vLLM optimizes for long-sequence parallel decoding, meaning 128-token batches underutilize prefill and attention kernels. Step synchronization accounts for another 30%, since each of the 64 steps requires an API roundtrip and a resampling barrier before proceeding. Finally, KV cache management contributes 10% due to the memory copies required for particle duplication during the 22% of steps that trigger resampling (ESS/$N < 0.5$).

Assuming `dPF@16` is executed entirely within vLLM and utilizes only KL-guided resampling (omitting the early logit blending phase), it inherits all of `iPF`'s sequential overheads. However, `dPF` incurs an additional 50% relative latency penalty on top of the baseline because the guide model must perform a forward scoring pass on the generated step tokens at each resampling barrier to compute the KL divergence ($D_{\text{KL}}(p \parallel q)$).

H Extrinsic Reward Ablation

In this section, we evaluate various selection strategies for choosing the final response from $N = 32$ sampled solutions per problem. All evaluations are performed using `Qwen3-4B-Instruct`, with accuracy (in %) averaged over five independent seeds ($\pm$ standard error of the mean).

Baselines *Random* selects a solution uniformly at random. *Oracle* (best-of-N) selects a correct solution if at least one exists in the sampled set, providing an upper bound on selection performance.

H.1 Model-Free Selection

Model-free approaches (Table 13) do not rely on an external reward model:

- *Entropy@1*: Selects the response with the lowest adaptive tail entropy, denoted as H_{tail}. The adaptive tail window is scaled by $\sqrt{\bar{L}}/\bar{H}$, incorporating percentile-based sequence length outlier trimming and median aggregation.
- *Self-Consistency (SC)*: Applies majority voting over raw string generations without task-specific normalization.
- *Entropy-Weighted SC (EW-SC)*: Extends SC by weighting each vote by $\exp(-H_{\text{tail}})$.
- *Entropy-Filtered SC (Ek→SC)*: Restricts the majority vote to the top-k candidates with the lowest tail entropy, explicitly filtering out highly uncertain responses prior to aggregation.

H.2 Reward Model and Hybrid Selection

These strategies (Table 14) leverage an external model, `Qwen2.5-Math-PRM-7B`, to score each candidate. Scores are derived from the final 2048 tokens via single-step aggregation to estimate the probability of correctness:

- *ORM@1*: Greedily selects the candidate with the highest ORM score.
- *ORM-SC*: Uses the predicted ORM probabilities as weights for SC voting.
- *Rank Fusion*: Computes the sum of the ranks derived independently from entropy and ORM scoring.
- *Product*: Selects the response maximizing the joint heuristic $P(\text{correct}) \times \exp(-H_{\text{tail}})$.
- *Ek→O / Ok→E*: Filters the candidate pool to the top-k responses based on lowest entropy (or highest ORM score), then selects the argmax ORM score (or argmin entropy) from the filtered subset.
- *Ek→OSC*: First filters to the top-k responses by entropy, followed by ORM-weighted majority voting on the filtered subset.

Table 12: Selection accuracy (%) ($N = 32$, Qwen3-4B-Instruct). **Bold** indicates the best performance among selection methods. Ek→SC denotes entropy top-k filtering followed by SC. Ek→OSC denotes entropy top-k filtering followed by ORM-weighted SC. Product refers to $P(\text{correct}) \times \exp(-H_{\text{tail}})$.

	Bounds		*No external model*					*With Qwen2.5-Math-PRM-7B*				
	Rand.	Oracle	Ent@1	SC	EW-SC	E8→SC	E16→SC	ORM@1	ORM-SC	E8→OSC	E16→OSC	Product
AIME 24	$56.5_{\pm 0.4}$	$87.5_{\pm 0.8}$	$66.7_{\pm 4.9}$	$72.5_{\pm 1.6}$	$73.3_{\pm 0.0}$	$75.8_{\pm 0.8}$	$75.8_{\pm 0.8}$	$75.0_{\pm 1.7}$	$78.3_{\pm 2.2}$	$79.2_{\pm 1.6}$	**80.0**$_{\pm 1.4}$	$75.8_{\pm 2.5}$
AIME 25	$45.3_{\pm 0.5}$	$75.8_{\pm 2.1}$	$52.5_{\pm 2.1}$	$60.8_{\pm 2.8}$	$59.2_{\pm 2.5}$	$54.2_{\pm 3.2}$	$60.0_{\pm 2.4}$	$51.7_{\pm 3.2}$	$60.8_{\pm 2.5}$	$55.8_{\pm 1.6}$	**61.7**$_{\pm 2.2}$	$55.0_{\pm 1.7}$
AIME 26	$50.0_{\pm 0.1}$	$83.3_{\pm 1.4}$	$52.5_{\pm 0.8}$	$65.8_{\pm 1.6}$	$67.5_{\pm 2.1}$	$60.8_{\pm 3.2}$	$61.7_{\pm 2.2}$	$59.2_{\pm 2.1}$	**68.3**$_{\pm 1.0}$	$67.5_{\pm 1.6}$	**68.3**$_{\pm 1.0}$	$61.7_{\pm 2.2}$
HMMT 25	$30.3_{\pm 0.3}$	$56.7_{\pm 3.3}$	$34.2_{\pm 2.8}$	$30.8_{\pm 0.8}$	$30.8_{\pm 0.8}$	$36.7_{\pm 1.4}$	$35.0_{\pm 1.7}$	$36.7_{\pm 1.4}$	$36.7_{\pm 1.9}$	**39.2**$_{\pm 1.6}$	**39.2**$_{\pm 0.8}$	$32.5_{\pm 1.6}$
HMMT 26	$28.5_{\pm 0.4}$	$52.3_{\pm 1.5}$	$34.1_{\pm 0.8}$	$28.8_{\pm 0.9}$	$30.3_{\pm 2.1}$	$32.6_{\pm 1.5}$	$30.3_{\pm 1.2}$	$34.1_{\pm 0.8}$	$30.3_{\pm 1.2}$	$33.3_{\pm 1.2}$	$32.6_{\pm 0.8}$	**35.6**$_{\pm 0.8}$
LCB-100	$18.9_{\pm 0.1}$	$35.0_{\pm 0.7}$	**22.0**$_{\pm 1.2}$	$19.0_{\pm 1.2}$	$19.2_{\pm 1.0}$	$19.5_{\pm 1.7}$	$18.2_{\pm 1.1}$	$14.8_{\pm 0.9}$	$18.5_{\pm 1.6}$	$20.2_{\pm 1.0}$	$20.0_{\pm 1.9}$	$16.2_{\pm 1.3}$
GPQA	$42.3_{\pm 0.1}$	$79.3_{\pm 0.7}$	$46.3_{\pm 0.4}$	$43.4_{\pm 0.6}$	$44.2_{\pm 0.9}$	$46.7_{\pm 0.6}$	$44.7_{\pm 0.6}$	$47.2_{\pm 1.0}$	**47.9**$_{\pm 1.2}$	$46.8_{\pm 1.0}$	$46.8_{\pm 0.7}$	$47.7_{\pm 0.5}$
Math	$42.1_{\pm 0.2}$	$71.1_{\pm 0.7}$	$48.0_{\pm 1.5}$	$51.8_{\pm 0.4}$	$52.2_{\pm 0.4}$	$52.0_{\pm 1.0}$	$52.6_{\pm 0.6}$	$51.3_{\pm 0.9}$	$54.9_{\pm 0.7}$	$55.0_{\pm 1.0}$	**56.3**$_{\pm 0.4}$	$52.1_{\pm 0.3}$
Code	$18.9_{\pm 0.1}$	$35.0_{\pm 0.7}$	**22.0**$_{\pm 1.2}$	$19.0_{\pm 1.2}$	$19.2_{\pm 1.0}$	$19.5_{\pm 1.7}$	$18.2_{\pm 1.1}$	$14.8_{\pm 0.9}$	$18.5_{\pm 1.6}$	$20.2_{\pm 1.0}$	$20.0_{\pm 1.9}$	$16.2_{\pm 1.3}$
Reason.	$42.3_{\pm 0.1}$	$79.3_{\pm 0.7}$	$46.3_{\pm 0.4}$	$43.4_{\pm 0.6}$	$44.2_{\pm 0.9}$	$46.7_{\pm 0.6}$	$44.7_{\pm 0.6}$	$47.2_{\pm 1.0}$	**47.9**$_{\pm 1.2}$	$46.8_{\pm 1.0}$	$46.8_{\pm 0.7}$	$47.7_{\pm 0.5}$

The results across the evaluated selection strategies reveal three key takeaways. First, cascaded approaches that integrate both intrinsic signals and external reward models - specifically, entropy-filtered ORM-weighted Self-Consistency (E_k→OSC) - consistently achieve the highest accuracy across most benchmarks, demonstrating the complementary strengths of internal model confidence and external verification. Second, in the absence of an external reward model, using intrinsic tail entropy to filter out highly uncertain responses prior to aggregation (E_k→SC) generally outperforms raw Self-Consistency (SC), indicating that pruning noisy, low-confidence reasoning traces cleans up the consensus signal. Finally, greedily selecting the highest-scored response from

Table 13: Model-free selection methods ($N = 32$). Ek→SC denotes entropy top-k filtering followed by majority voting. **Bold** indicates the best performance per row (excluding bounds).

	Bounds		*Base*			*Entropy-filtered SC*		
	Random	Oracle	Ent@1	SC	EW-SC	E4→SC	E8→SC	E16→SC
AIME 24	$56.5_{\pm 0.4}$	$87.5_{\pm 0.8}$	$66.7_{\pm 4.9}$	$72.5_{\pm 1.6}$	$73.3_{\pm 0.0}$	$71.7_{\pm 1.0}$	**75.8**$_{\pm 0.8}$	**75.8**$_{\pm 0.8}$
AIME 25	$45.3_{\pm 0.5}$	$75.8_{\pm 2.1}$	$52.5_{\pm 2.1}$	**60.8**$_{\pm 2.8}$	$59.2_{\pm 2.5}$	$52.5_{\pm 3.9}$	$54.2_{\pm 3.2}$	$60.0_{\pm 2.4}$
AIME 26	$50.0_{\pm 0.1}$	$83.3_{\pm 1.4}$	$52.5_{\pm 0.8}$	$65.8_{\pm 1.6}$	**67.5**$_{\pm 2.1}$	$56.7_{\pm 2.7}$	$60.8_{\pm 3.2}$	$61.7_{\pm 2.2}$
HMMT 25	$30.3_{\pm 0.3}$	$56.7_{\pm 3.3}$	$34.2_{\pm 2.8}$	$30.8_{\pm 0.8}$	$30.8_{\pm 0.8}$	**37.5**$_{\pm 0.8}$	$36.7_{\pm 1.4}$	$35.0_{\pm 1.7}$
HMMT 26	$28.5_{\pm 0.4}$	$52.3_{\pm 1.5}$	**34.1**$_{\pm 0.8}$	$28.8_{\pm 0.9}$	$30.3_{\pm 2.1}$	$30.3_{\pm 1.2}$	$32.6_{\pm 1.5}$	$30.3_{\pm 1.2}$
LCB-100	$18.9_{\pm 0.1}$	$35.0_{\pm 0.7}$	**22.0**$_{\pm 1.2}$	$19.0_{\pm 1.2}$	$19.2_{\pm 1.0}$	$20.0_{\pm 1.7}$	$19.5_{\pm 1.7}$	$18.2_{\pm 1.1}$
GPQA	$42.3_{\pm 0.1}$	$79.3_{\pm 0.7}$	$46.3_{\pm 0.4}$	$43.4_{\pm 0.6}$	$44.2_{\pm 0.9}$	$46.1_{\pm 1.1}$	**46.7**$_{\pm 0.6}$	$44.7_{\pm 0.6}$
Math	$42.1_{\pm 0.2}$	$71.1_{\pm 0.7}$	$48.0_{\pm 1.5}$	$51.8_{\pm 0.4}$	$52.2_{\pm 0.4}$	$49.7_{\pm 0.8}$	$52.0_{\pm 1.0}$	**52.6**$_{\pm 0.6}$
Code	$18.9_{\pm 0.1}$	$35.0_{\pm 0.7}$	**22.0**$_{\pm 1.2}$	$19.0_{\pm 1.2}$	$19.2_{\pm 1.0}$	$20.0_{\pm 1.7}$	$19.5_{\pm 1.7}$	$18.2_{\pm 1.1}$
Reason.	$42.3_{\pm 0.1}$	$79.3_{\pm 0.7}$	$46.3_{\pm 0.4}$	$43.4_{\pm 0.6}$	$44.2_{\pm 0.9}$	$46.1_{\pm 1.1}$	**46.7**$_{\pm 0.6}$	$44.7_{\pm 0.6}$

Table 14: ORM-based selection methods ($N = 32$, `Qwen2.5-Math-PRM-7B`). **Bold** indicates the best performance per row.

	Base		*Simple comb.*		*Ek→ORM*			*Ok→Ent*			*Ek→ORM-SC*		
	ORM@1	ORM-SC	Rank fus.	Product	E4→O	E8→O	E16→O	O4→E	O8→E	O16→E	E4→OSC	E8→OSC	E16→OSC
AIME 24	$75.0_{\pm 1.7}$	$78.3_{\pm 2.2}$	$75.0_{\pm 1.7}$	$75.8_{\pm 2.5}$	$70.8_{\pm 1.6}$	$74.2_{\pm 2.1}$	$75.0_{\pm 2.2}$	$78.3_{\pm 1.0}$	$74.2_{\pm 1.6}$	$70.0_{\pm 1.4}$	$71.7_{\pm 1.7}$	$79.2_{\pm 1.6}$	**80.0**$_{\pm 1.4}$
AIME 25	$51.7_{\pm 3.2}$	$60.8_{\pm 2.5}$	$54.2_{\pm 0.8}$	$55.0_{\pm 1.7}$	$51.7_{\pm 3.2}$	$50.8_{\pm 1.6}$	$56.7_{\pm 2.7}$	$53.3_{\pm 3.0}$	$52.5_{\pm 1.6}$	$50.0_{\pm 1.4}$	$54.2_{\pm 2.1}$	$55.8_{\pm 1.6}$	**61.7**$_{\pm 2.2}$
AIME 26	$59.2_{\pm 2.1}$	**68.3**$_{\pm 1.0}$	$60.8_{\pm 3.7}$	$61.7_{\pm 2.2}$	$58.3_{\pm 2.2}$	$59.2_{\pm 0.8}$	$61.7_{\pm 2.9}$	$61.7_{\pm 4.0}$	$64.2_{\pm 3.9}$	$57.5_{\pm 2.8}$	$59.2_{\pm 2.1}$	$67.5_{\pm 1.6}$	**68.3**$_{\pm 1.0}$
HMMT 25	$36.7_{\pm 1.4}$	$36.7_{\pm 1.9}$	$34.2_{\pm 3.4}$	$32.5_{\pm 1.6}$	$35.0_{\pm 2.2}$	$34.2_{\pm 0.8}$	$35.0_{\pm 1.0}$	$35.8_{\pm 2.5}$	$35.0_{\pm 2.9}$	$32.5_{\pm 2.8}$	$35.8_{\pm 2.5}$	**39.2**$_{\pm 1.6}$	**39.2**$_{\pm 0.8}$
HMMT 26	$34.1_{\pm 0.8}$	$30.3_{\pm 1.2}$	$32.6_{\pm 0.8}$	**35.6**$_{\pm 0.8}$	$32.6_{\pm 1.5}$	$33.3_{\pm 0.0}$	$34.1_{\pm 0.8}$	$31.8_{\pm 0.9}$	$31.1_{\pm 0.8}$	$33.3_{\pm 1.2}$	$31.1_{\pm 1.5}$	$33.3_{\pm 1.2}$	$32.6_{\pm 0.8}$
LCB-100	$14.8_{\pm 0.9}$	$18.5_{\pm 1.6}$	$18.8_{\pm 1.3}$	$16.2_{\pm 1.3}$	$20.0_{\pm 1.8}$	$20.0_{\pm 1.2}$	$19.2_{\pm 1.8}$	$15.2_{\pm 0.6}$	$17.8_{\pm 1.3}$	$20.0_{\pm 1.1}$	**20.5**$_{\pm 1.8}$	$20.2_{\pm 1.0}$	$20.0_{\pm 1.9}$
GPQA	$47.2_{\pm 1.0}$	$47.9_{\pm 1.2}$	$48.4_{\pm 0.9}$	$47.7_{\pm 0.5}$	$48.2_{\pm 0.4}$	$47.5_{\pm 0.7}$	$47.5_{\pm 0.9}$	$47.7_{\pm 1.3}$	$47.0_{\pm 0.8}$	**48.5**$_{\pm 0.5}$	$47.1_{\pm 0.3}$	$46.8_{\pm 1.0}$	$46.8_{\pm 0.7}$
Math	$51.3_{\pm 0.9}$	$54.9_{\pm 0.7}$	$51.3_{\pm 0.7}$	$52.1_{\pm 0.3}$	$49.7_{\pm 1.1}$	$50.3_{\pm 0.6}$	$52.5_{\pm 0.9}$	$52.2_{\pm 0.8}$	$51.4_{\pm 0.5}$	$48.7_{\pm 0.7}$	$50.4_{\pm 0.7}$	$55.0_{\pm 1.0}$	**56.3**$_{\pm 0.4}$
Code	$14.8_{\pm 0.9}$	$18.5_{\pm 1.6}$	$18.8_{\pm 1.3}$	$16.2_{\pm 1.3}$	$20.0_{\pm 1.8}$	$20.0_{\pm 1.2}$	$19.2_{\pm 1.8}$	$15.2_{\pm 0.6}$	$17.8_{\pm 1.3}$	$20.0_{\pm 1.1}$	**20.5**$_{\pm 1.8}$	$20.2_{\pm 1.0}$	$20.0_{\pm 1.9}$
Reason.	$47.2_{\pm 1.0}$	$47.9_{\pm 1.2}$	$48.4_{\pm 0.9}$	$47.7_{\pm 0.5}$	$48.2_{\pm 0.4}$	$47.5_{\pm 0.7}$	$47.5_{\pm 0.9}$	$47.7_{\pm 1.3}$	$47.0_{\pm 0.8}$	**48.5**$_{\pm 0.5}$	$47.1_{\pm 0.3}$	$46.8_{\pm 1.0}$	$46.8_{\pm 0.7}$

a reward model (ORM@1) is frequently suboptimal; instead, utilizing ORM probabilities as weights for majority voting (ORM-SC) yields noticeably more robust performance across both mathematical and reasoning domains.

I Gemma Ablation

In our evaluation of selection strategies, entropy-based methods demonstrate significant performance improvements over standard self-consistency (SC) on complex reasoning tasks (where pass@1 $< 30\%$) and code generation benchmarks. This advantage arises because SC struggles to find exact structural matches for code or lengthy reasoning chains at higher sampling temperatures. Conversely, SC remains the most robust choice for easier datasets, such as AIME-2024, where models naturally converge on the correct solution. Furthermore, our results indicate that adaptive tail formulas transfer effectively across diverse model architectures.

Practitioners should leverage entropy-based aggregation for high-complexity or open-ended generation tasks, while defaulting to standard self-consistency for highly constrained, convergent problem-solving.

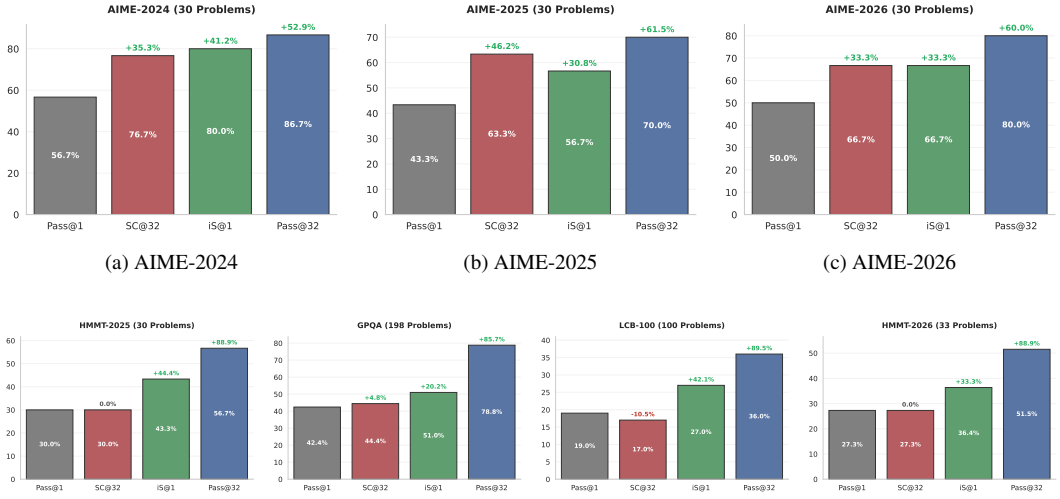
Table 15: Selection accuracy (%) for `Qwen3-4B-Instruct` across mathematics, reasoning, and code generation benchmarks (N=32). The `iS@1` metric evaluates both median and mean aggregations. The best-performing selection method per aggregated column is **bolded** (excluding the pass@32 oracle).

	Dataset							Aggregated		
Method	A24	A25	A26	H25	H26	GPQA	LCB	Mean	Median	w Mean
pass@1	56.5	45.3	50.0	30.2	28.5	42.3	19.0	38.8	42.3	37.0
pass@32	87.5	75.8	83.3	54.2	52.3	79.3	35.0	66.8	75.8	66.4
SC@32	**72.5**	**60.8**	**65.8**	30.8	28.8	44.1	18.2	**45.9**	44.1	40.8
iS@1 (median)	66.7	52.5	52.5	**34.2**	**34.1**	**46.3**	**22.0**	44.0	**46.3**	**41.4**
iS@1 (mean)	66.7	45.0	53.3	30.0	31.8	46.6	21.0	42.1	45.0	40.4

Table 16: Selection accuracy (%) for `gemma-4-E2B-it` evaluated on identical benchmarks (N=32). Method configurations and tail hyperparameters mirror those detailed in Table 15. The best-performing selection method per aggregated column is **bolded** (excluding the pass@32 oracle).

	Dataset							Aggregated		
Method	A24	A25	A26	H25	H26	GPQA	LCB	Mean	Median	w Mean
pass@1	44.8	29.3	39.6	19.2	21.6	40.4	19.6	30.6	29.3	32.5
pass@32	76.7	63.3	63.3	53.3	45.5	84.3	36.0	60.3	63.3	65.3
SC@32	60.0	36.7	50.0	20.0	**33.3**	45.5	20.0	37.9	**36.7**	37.9
iS@1 (median)	60.0	36.7	**56.7**	20.0	21.2	46.0	**25.0**	37.9	**36.7**	38.8
iS@1 (mean)	60.0	36.7	53.3	**23.3**	24.2	**47.5**	24.0	**38.4**	**36.7**	**39.4**

J Additional Ablations and Experiments



(a) AIME-2024 (b) AIME-2025 (c) AIME-2026

(d) HMMT-2025 (e) GPQA-Diamond (f) LiveCodeBench-v6 (g) HMMT-2026

Figure 11: Accuracy pass@1, Self-Consistency, `iS@1`, and pass@N. We provide validation results for our method evaluated over complex math datasets, general reasoning, and coding. The domains require different levels or verification.

J.1 Budget

Table 17: Intrinsic selection accuracy (%) across sample budgets — **mean** of 10 bootstrap trials. Pool of 512 samples per problem. Model: Qwen3-4B-Instruct-2507. Best selection method per dataset in **bold** (excluding pass@N).

N	Method	AIME 2024	AIME 2025	AIME 2026	HMMT 2025	HMMT 2026	GPQA-D	LCBv6-100	Wt. Avg
8	pass@1	56.5	45.4	50.0	30.1	28.6	42.2	19.0	37.0
	iS@1 (ours)	**67.6**	47.5	55.7	**33.8**	**31.6**	**44.9**	**22.4**	**40.6**
	SC	67.6	**56.1**	**60.1**	32.1	28.9	42.6	18.9	39.4
	pass@N	81.8	66.8	75.2	46.5	42.5	69.2	29.9	58.1
16	pass@1	56.6	45.4	49.9	30.2	28.5	42.3	19.0	37.0
	iS@1 (ours)	68.8	47.0	55.2	**34.5**	**31.7**	**45.5**	**22.4**	**40.9**
	SC	**70.1**	**58.2**	**61.9**	32.0	28.1	42.5	18.3	39.5
	pass@N	85.3	71.3	79.3	50.9	47.9	74.7	32.3	62.6
32	pass@1	56.5	45.3	50.0	30.1	28.5	42.3	18.9	36.9
	iS@1 (ours)	68.7	46.4	54.7	**35.4**	**32.0**	**45.6**	**21.8**	**40.8**
	SC	**71.1**	**58.8**	**63.1**	31.6	28.3	42.6	17.6	39.6
	pass@N	87.7	75.4	82.2	55.0	53.3	79.4	34.7	66.4
64	pass@1	56.5	45.3	50.0	30.2	28.5	42.3	18.9	36.9
	iS@1 (ours)	69.5	46.0	54.4	**36.8**	**31.5**	**45.5**	**21.1**	**40.7**
	SC	**71.9**	**59.9**	**64.1**	30.8	27.8	42.7	16.8	39.6
	pass@N	89.1	79.0	85.2	60.7	59.5	83.4	37.2	70.1
128	pass@1	56.5	45.3	50.0	30.2	28.5	42.3	19.0	37.0
	iS@1 (ours)	70.0	46.7	56.7	**40.0**	**33.3**	**44.9**	**20.0**	**40.8**
	SC	**73.3**	**60.0**	**63.3**	30.0	27.3	41.9	16.0	39.0
	pass@N	90.0	80.0	86.7	70.0	63.6	86.9	39.0	73.2

Table 18: Intrinsic selection accuracy (%) across sample budgets — **median** of 10 bootstrap trials. Pool of 512 samples per problem. Model: Qwen3-4B-Instruct-2507. Best selection method per dataset in **bold** (excluding pass@N).

N	Method	AIME 2024	AIME 2025	AIME 2026	HMMT 2025	HMMT 2026	GPQA-D	LCBv6-100	Wt. Avg
8	pass@1	56.7	45.4	50.0	30.0	28.4	42.2	19.0	36.9
	iS@1 (ours)	66.7	46.7	**56.7**	**33.3**	**30.3**	**44.9**	**22.0**	**40.4**
	SC	**66.7**	**56.7**	60.0	33.3	27.3	42.4	19.0	39.2
	pass@N	83.3	66.7	76.7	46.7	42.4	69.2	30.0	58.3
16	pass@1	56.5	45.4	50.0	30.2	28.4	42.3	18.9	36.9
	iS@1 (ours)	**70.0**	46.7	56.7	**33.3**	**30.3**	**45.5**	**23.0**	**41.0**
	SC	70.0	**56.7**	**63.3**	31.7	27.3	42.4	18.0	39.4
	pass@N	86.7	70.0	80.0	50.0	48.5	74.7	32.0	62.5
32	pass@1	56.5	45.3	49.9	30.1	28.5	42.3	18.9	36.9
	iS@1 (ours)	**70.0**	46.7	53.3	**36.7**	**30.3**	**45.5**	**22.0**	**40.8**
	SC	70.0	**60.0**	**63.3**	30.0	27.3	42.4	18.0	39.5
	pass@N	86.7	76.7	83.3	53.3	54.5	79.3	35.0	66.5
64	pass@1	56.5	45.3	49.9	30.2	28.5	42.3	19.0	37.0
	iS@1 (ours)	70.0	46.7	53.3	**36.7**	**30.3**	**45.5**	**21.0**	**40.6**
	SC	**73.3**	**60.0**	**63.3**	30.0	27.3	42.4	17.0	39.5
	pass@N	90.0	80.0	86.7	60.0	60.6	83.3	37.0	70.3
128	pass@1	56.5	45.3	50.0	30.2	28.5	42.3	19.0	37.0
	iS@1 (ours)	70.0	46.7	56.7	**40.0**	**33.3**	**44.9**	**20.0**	**40.8**
	SC	**73.3**	**60.0**	**63.3**	30.0	27.3	41.9	16.0	39.0
	pass@N	90.0	80.0	86.7	70.0	63.6	86.9	39.0	73.2

J.2 Tail Entropy as a Correctness Signal

Table 19 pools all generated samples per benchmark. Table 20 computes the AUROC separately within each eligible problem (at least one correct and one incorrect candidate), using negative tail entropy as the score, and averages over problems. Table 21 reports accuracy by tail-entropy quartile, and Table 22 reports selection accuracy in the sparse-success regime.

Table 19: Pooled sample-level separation between correct (C) and incorrect (I) samples. $\bar{s}$: mean tail entropy. AUROC uses negative tail entropy as the score.

Model	Benchmark	#C	#I	$\bar{s}$ (C)	$\bar{s}$ (I)	Cohen's d	AUROC
Qwen3-4B	AIME 2024	2,169	1,671	**0.1774**	0.3049	0.97	0.764
	AIME 2025	1,739	2,101	**0.1391**	0.2516	0.88	0.724
	AIME 2026	1,919	1,921	**0.1620**	0.3282	1.14	0.797
	HMMT Feb 2025	1,159	2,681	**0.1898**	0.3579	1.14	0.791
	HMMT Feb 2026	1,169	3,055	**0.1984**	0.3254	0.90	0.741
	Math pooled	8,155	11,429	**0.1704**	0.3169	1.03	0.773
Gemma4-E2B	AIME 2024	430	530	**0.0659**	0.1268	1.31	0.837
	AIME 2025	281	679	**0.0480**	0.1277	1.73	0.906
	AIME 2026	379	581	**0.0557**	0.1371	1.59	0.879
	HMMT Feb 2025	157	803	**0.0682**	0.1516	1.36	0.861
	HMMT Feb 2026	110	946	**0.0762**	0.1302	0.87	0.750
	Math pooled	1,357	3,539	**0.0605**	0.1352	1.39	0.856

Table 20: Within-problem discrimination. For each eligible problem we rank its candidates by ascending tail entropy and compute the AUROC for separating correct from incorrect candidates. Values above 0.5 mean that lower tail entropy ranks correct candidates ahead of incorrect ones.

Model	Benchmark	Mean AUROC	Median AUROC	Problems AUROC > 0.5
Qwen3-4B	AIME 2024	0.676	0.724	**83%**
	AIME 2025	0.593	0.622	**68%**
	AIME 2026	0.624	0.667	**75%**
	HMMT Feb 2025	0.686	0.700	**81%**
	HMMT Feb 2026	0.603	0.659	**69%**
	Math pooled	0.638	0.675	**76%**
Gemma4-E2B	AIME 2024	0.706	0.696	**80%**
	AIME 2025	0.704	0.733	**88%**
	AIME 2026	0.673	0.695	**82%**
	HMMT Feb 2025	0.768	0.871	**92%**
	HMMT Feb 2026	0.609	0.662	**80%**
	Math pooled	0.695	0.723	**84%**

Table 21: Accuracy (%) by tail-entropy quartile (Q1 = lowest entropy). Accuracy falls sharply from Q1 to Q4, but Q1 never reaches 100%, and on Qwen3-4B AIME 2025 Q2 exceeds Q1. Low entropy raises the likelihood of correctness but does not rule out confident errors.

Model	Benchmark	Q1 (lowest)	Q2	Q3	Q4 (highest)
Qwen3-4B	AIME 2024	**84.2**	69.3	44.9	27.6
	AIME 2025	61.4	**69.9**	33.4	16.5
	AIME 2026	**85.1**	64.2	31.2	19.4
	HMMT Feb 2025	**62.2**	32.8	18.2	7.6
	HMMT Feb 2026	**54.3**	28.8	17.9	9.8
Gemma4-E2B	AIME 2024	**82.1**	60.0	28.3	8.8
	AIME 2025	**77.1**	33.3	3.7	3.3
	AIME 2026	**83.3**	55.8	14.2	4.6
	HMMT Feb 2025	**47.1**	10.0	6.8	1.6
	HMMT Feb 2026	**26.5**	6.1	5.3	3.8

Table 22: Selection accuracy (%) in the sparse-success regime ($0 < c/N \leq 0.25$, with c correct samples), aggregated over the five math datasets. The regime is defined with oracle correctness for analysis only. All methods select from the same pool, and uniform selection equals pass@1.

Model	`iS@1`	SC	Uniform (pass@1)
Qwen3-4B	**25.0**	14.3	11.2
Gemma4-E2B	**23.1**	3.8	8.2

Together these analyses separate two questions. Is tail entropy informative within a problem? Yes: the mean within-problem AUROC exceeds 0.5 on every benchmark, and lower tail entropy ranks correct candidates first on 68–92% of eligible problems. Is it sufficient? Not always. When a correct trajectory exists but is rare, `iS@1` selects it 1.7 times as often as SC on Qwen3-4B (25.0% vs. 14.3%) and 6 times as often on Gemma4-E2B (23.1% vs. 3.8%), where SC falls below uniform selection. When the pool contains no correct trajectory (systematic failure), no selection rule over the pool can succeed, and high set-level tail entropy is the deployment-time cue for routing to `dPF`.

J.3 Consensus and Confidence

To further contextualize our findings, Table 23 compares validation accuracy across math datasets for Self-Consistency, `iS`, and state-of-the-art confidence-based selection methods, notably Self-Certainty [Kang et al., 2025] and DeepConf [Fu et al., 2025] ($N = 32$, max length 16,000 tokens). These baselines represent different paradigms for inference-time scaling:

- Self-Consistency: Relies purely on output consensus by marginalizing reasoning trajectories via majority voting over a canonical answer format.
- Self-Certainty: Depends entirely on intrinsic confidence, measuring the divergence from a uniform distribution to the model's predictive distribution to bypass external verifiers.
- DeepConf: Hybridizes the two by weighting majority voting consensus with trace-level confidence scores derived from token log probabilities.
- `iS` (Ours): Evaluates solution quality intrinsically by ranking candidates using set-level length distributions and adjusted per-token tail entropy, entirely bypassing external verifiers and consensus matching.

Table 23: Validation accuracy across math datasets using various baselines involving confidence-based, consensus-based, and refining-based inference scaling. DeepConf merges consensus and confidence metrics for selection. For the metrics involving confidence, we use the selection variant performing the best over each dataset.

Method	AIME 2024	AIME 2025	AIME 2026	HMMT 2025	HMMT 2026
Self-Consistency [Wang et al., 2022]	$72.5_{\pm 1.5}$	$60.8_{\pm 2.8}$	$65.8_{\pm 1.6}$	$30.8_{\pm 0.8}$	$28.5_{\pm 0.4}$
Self-Certainty [Kang et al., 2025] (val)	$67.5_{\pm 2.2}$	$50.0_{\pm 0.4}$	$51.7_{\pm 0.4}$	$35.0_{\pm 1.1}$	$34.8_{\pm 1.9}$
DeepConf [Fu et al., 2025] (val)	$72.7_{\pm 1.3}$	$62.7_{\pm 2.5}$	$70.7_{\pm 1.3}$	$30.7_{\pm 2.5}$	$25.5_{\pm 4.5}$
`iS@1` (our, val)	$74.2_{\pm 1.7}$	$52.5_{\pm 2.1}$	$64.2_{\pm 1.5}$	$38.3_{\pm 0.8}$	$37.1_{\pm 1.1}$

The empirical results highlight the complementary strengths of consensus- and confidence-based approaches, as well as the unique advantages of `iS`. DeepConf performs exceptionally well on the AIME datasets, where the model successfully produces a sizable proportion of correct solutions and majority voting is already highly effective. By leveraging trace-level confidence to upweight reliable consensus clusters, DeepConf noticeably improves upon the baseline, notably reaching 70.7% on AIME 2026 compared to the 65.8% Self-Consistency baseline.

However, this reliance on consensus becomes a bottleneck on harder datasets like HMMT. On HMMT datasets, where the frequency of correct answers within the sample set drops significantly ($n/N \ll 1$), extracting a reliable consensus frequently fails. In these sparse regimes, pure consensus degrades, and hybrid methods like DeepConf struggle to compensate because the underlying answer clusters are too weak.

Conversely, `iS` achieves the highest overall accuracy on the HMMT datasets (38.3% and 37.1%). By selecting the candidate with the lowest adjusted tail entropy, `iS` efficiently isolates the best solution even when correct trajectories are sparse and lack a strong consensus cluster. Crucially, `iS` achieves these gains without requiring any output parsing, canonical-form matching, or explicit answer clustering. This lack of dependency on extrinsic output formats means that, unlike Self-Consistency or DeepConf, `iS` can seamlessly extend to open-ended and hard-to-verify domains.

J.4 Confidence-Weighted Consensus: RPC and CISC

We compare with two recent methods that combine model probability with consensus. RPC [Zhou et al., 2026] prunes low-probability trajectories with a Weibull-mixture fit and then votes with probability weights. CISC [Taubenfeld et al., 2025] weights each vote by a per-path confidence score. These runs use the official RPC evaluation setup and average over 3 seeds. Within each table, all methods select from the same 32 samples. The avg and med columns are over the seven datasets, and wt avg weights each dataset by its number of problems.

Table 24: SC@32, RPC@32, and `iS@1` on Qwen3-4B-Instruct-2507. All methods select from the same 32 samples. Best practical estimator in bold.

Method	A2024	A2025	A2026	H2025	H2026	GPQA	LCB	avg	med	wt avg
pass@1	56.5	45.3	50.0	30.2	28.5	42.3	19.0	38.8	42.3	37.0
SC@32	72.5	60.8	65.8	30.8	29.5	44.1	18.2	46.0	44.1	40.8
RPC@32	**75.6**	**62.2**	**70.0**	25.6	25.2	43.3	6.0	44.0	43.3	37.7
`iS@1` (ours)	67.5	54.4	58.9	**36.7**	**34.8**	**50.0**	**21.5**	**46.3**	**50.0**	**43.7**
pass@32	87.5	75.8	83.3	56.7	52.3	79.3	35.0	67.1	75.8	66.6

Table 25: SC@32, RPC@32, and `iS@1` on gemma-4-E2B-it. All methods select from the same 32 samples. Best practical estimator in bold.

Method	A2024	A2025	A2026	H2025	H2026	GPQA	LCB	avg	med	wt avg
pass@1	54.5	25.6	37.8	23.3	22.2	38.9	18.0	31.5	25.6	32.1
SC@32	60.0	34.4	46.7	18.9	27.3	44.1	18.0	35.6	34.4	36.0
RPC@32	**62.2**	37.8	**47.8**	16.7	10.1	44.8	3.0	31.8	**37.8**	32.0
`iS@1` (ours)	45.6	**38.9**	36.7	**31.1**	**28.3**	**47.8**	**26.3**	**36.4**	36.7	**39.0**
pass@32	80.0	61.1	60.0	48.9	39.4	84.3	34.7	58.3	60.0	64.2

Table 26: CISC vs. SC on Qwen3-4B-Instruct-2507 ($N = 32$; mean $\pm$ std over 3 seeds, same protocol as Table 24). avg is over the five math datasets.

Method	A2024	A2025	A2026	H2025	H2026	avg
SC@32	$72.5_{\pm 1.6}$	$60.8_{\pm 0.0}$	$65.8_{\pm 1.6}$	$30.8_{\pm 0.0}$	$29.5_{\pm 0.0}$	51.9
CISC@32	$73.2_{\pm 3.1}$	$62.8_{\pm 3.1}$	$68.0_{\pm 2.7}$	$30.8_{\pm 0.0}$	$29.5_{\pm 0.0}$	52.9
CISC $-$ SC	$+0.7$	$+2.0$	$+2.2$	$+0.0$	$+0.0$	$+1.0$

The pattern matches Table 2. Probability-weighted consensus is strongest on AIME, where answers are integers and the correct answer is often modal. RPC is best on all three AIME datasets for Qwen3-4B and on two of three for Gemma4-E2B, and all of CISC's $+1.0$ average gain over SC comes from AIME, with no change on HMMT. Where answers are harder to canonicalize or correct trajectories are sparse (HMMT, GPQA, LCB), RPC can fall well below SC (e.g., 6.0 vs. 18.2 on LCB with Qwen3-4B), and `iS@1` is best. As a result, `iS@1` has the best average, median, and weighted average on Qwen3-4B, and the best average and weighted average on Gemma4-E2B. All methods remain well below pass@32, so selection still has room to improve.

J.5 Selection without the Answer

A fundamental limitation of consensus-based algorithms, such as Self-Consistency, is their strict reliance on a clearly defined, extractable final answer. These methods must parse a canonical output format (e.g., a `\boxed{}` string in mathematics) to group identical solutions and compute majority votes. This dependency effectively precludes their use in open-ended domains - such as clinical drafting, creative writing, or complex engineering design - where solutions are highly variable and no single definitive "answer" string exists.

To demonstrate that Intrinsic Selection (`iS`) completely bypasses this limitation, we designed an experiment to evaluate the robustness of our entropy-based scoring when the final answer is entirely inaccessible. Specifically, we stripped the `\boxed{}` answer region from every generated sequence prior to evaluation, forcing the `iS` scoring function to operate solely on the intrinsic statistics of the preceding reasoning trace.

Table 27: Performance comparison without and with answer stripping using Intrinsic Selection. Overall, our method is highly robust to answer presence and even sees marginal gains when the final answer is removed.

Method	A24	A25	A26	H25	H26	GPQA	wt avg
w/o strip	67.5	49.2	52.5	35.0	31.1	46.5	46.96
w/ strip	67.5	50.0	55.0	35.8	32.6	47.6	48.08
Δ	+0.0	+0.8	+2.5	+0.8	+1.5	+1.1	+1.06

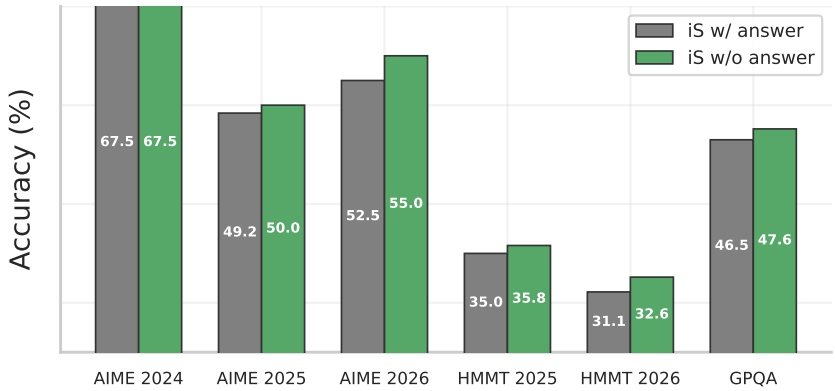


Figure 12: Answer Stripping. `iS` remains robust without the final answer, demonstrating its effectiveness regardless of domain verifiability.

As shown in Figure 12 and detailed in Table 27, the performance of `iS` remains stable or slightly improves across all six math and reasoning datasets when the answer is stripped. We observe consistent positive deltas, ranging from neutral (+0.0 on AIME 2024) to noticeable gains (+2.5 on AIME 2026), yielding an average dataset-size weighted gain of +1.06 percentage points overall.

This marginal improvement suggests that formatting tokens and standardized answer wrappers may occasionally introduce artificial entropy spikes that add noise to confidence estimations. By evaluating only the reasoning steps, `iS` captures a purer signal of the model's internal certainty. Most importantly, these results empirically confirm that the discriminative signal captured by adjusted tail entropy resides entirely in the latent structure of the reasoning trajectory rather than the final answer tokens. Consequently, `iS` functions effectively as a verifier-free inference scaling method applicable even in open-ended domains lacking canonical outputs.

J.6 Entropy Tail and Aggregation Ablations

The core premise of Intrinsic Selection (`iS`) is that not all tokens in a reasoning trajectory are equally informative for assessing confidence. Early steps in a chain-of-thought sequence often contain exploratory or boilerplate text, while the final tokens encapsulate the model's conclusive stance. However, the length of the relevant "answer region" varies significantly across problems and domains. To validate our design choices for isolating this discriminative signal, we conduct ablation studies on both the tail window definition and the token-level aggregation function.

Table 28: Ablation: tail window and aggregation for iS@1 ($N=32$, with outlier trimming). L: all tokens. $\sqrt{L}$: tail $= \lfloor\sqrt{L}\rfloor$. $\sqrt{L}/\bar{H}$ (ours): tail $= \lfloor\sqrt{L}/\bar{H}\rfloor$ where $\bar{H}$ is the mean per-token entropy.

	$L = 2048$		$\sqrt{L}$		$\sqrt{L}/\bar{H}$ (ours)	
	Mean	Median	Mean	Median	Mean	Median
AIME 24	$65.8_{\pm 1.6}$	$66.7_{\pm 4.3}$	$\mathbf{71.7}_{\pm 2.2}$	$70.8_{\pm 3.7}$	$65.8_{\pm 0.8}$	$69.2_{\pm 4.4}$
AIME 25	$46.7_{\pm 3.0}$	$50.0_{\pm 2.4}$	$43.3_{\pm 4.7}$	$49.2_{\pm 3.4}$	$45.0_{\pm 2.2}$	$\mathbf{53.3}_{\pm 3.0}$
AIME 26	$53.3_{\pm 3.6}$	$\mathbf{57.5}_{\pm 3.7}$	$49.2_{\pm 2.8}$	$52.5_{\pm 2.1}$	$52.5_{\pm 3.7}$	$50.8_{\pm 2.1}$
HMMT 25	$36.7_{\pm 1.4}$	$35.0_{\pm 2.2}$	$35.0_{\pm 1.0}$	$\mathbf{39.2}_{\pm 0.8}$	$30.0_{\pm 1.4}$	$34.2_{\pm 2.1}$
HMMT 26	$30.3_{\pm 2.1}$	$29.5_{\pm 1.5}$	$31.1_{\pm 1.9}$	$30.3_{\pm 1.2}$	$31.8_{\pm 0.9}$	$\mathbf{32.6}_{\pm 1.5}$
LCB-100	$19.8_{\pm 1.1}$	$21.0_{\pm 1.1}$	$\mathbf{22.2}_{\pm 1.3}$	$21.5_{\pm 1.0}$	$21.5_{\pm 1.7}$	$21.8_{\pm 0.8}$
GPQA	$44.6_{\pm 1.1}$	$43.9_{\pm 0.6}$	$44.9_{\pm 1.2}$	$\mathbf{46.0}_{\pm 1.5}$	$45.5_{\pm 0.7}$	$\mathbf{46.0}_{\pm 0.5}$

Table 28 contrasts different tail formulations and aggregation strategies across mathematics, coding, and reasoning tasks. A fixed full-sequence window ($L = 2048$) typically yields sub-optimal performance because it incorporates noise from the early problem-solving phases. By shifting to a sublinear scaling factor, specifically the square root of the mean length ($\sqrt{L}$), the scoring mechanism effectively zeroes in on the concluding steps, leading to measurable improvements across multiple domains.

Our proposed adaptive tail strategy, $\sqrt{L}/\bar{H}$, further scales this window inversely by the mean per-token entropy of the sample set. The underlying intuition is that when the model exhibits high overall uncertainty across trajectories, a narrower tail is necessary to avoid integrating uninformative entropy spikes. The results show that the adaptive tail consistently rivals or outperforms the static $\sqrt{L}$ approach. Furthermore, when comparing aggregation functions, the median is demonstrably superior to the mean. The mean is highly sensitive to isolated entropy spikes caused by punctuation, formatting tokens, or minor syntactic uncertainties, whereas the median provides a statistically robust summary of the trajectory's structural confidence.

Table 29: Effect of length outlier trimming on iS@1 ($\sqrt{L}/\bar{H}$, median). Trimming excludes responses outside the $[5\%, 95\%]$ length percentile before computing the adaptive tail window.

	No trim	Trim (ours)
AIME 24	$68.3_{\pm 4.2}$	$\mathbf{69.2}_{\pm 4.4}$
AIME 25	$\mathbf{53.3}_{\pm 3.0}$	$\mathbf{53.3}_{\pm 3.0}$
AIME 26	$\mathbf{52.5}_{\pm 2.1}$	$50.8_{\pm 2.1}$
HMMT 25	$\mathbf{34.2}_{\pm 2.1}$	$\mathbf{34.2}_{\pm 2.1}$
HMMT 26	$\mathbf{32.6}_{\pm 1.5}$	$\mathbf{32.6}_{\pm 1.5}$
LCB-100	$21.5_{\pm 0.6}$	$\mathbf{21.8}_{\pm 0.8}$
GPQA	$45.7_{\pm 0.4}$	$\mathbf{46.0}_{\pm 0.5}$

Additionally, we evaluate the impact of length-based outlier trimming. Since our adaptive tail is a function of the set's mean length ($\bar{L}$) and mean entropy ($\bar{H}$), extreme pathological behaviors - such as infinite generation loops, repeating tokens, or premature stopping - can heavily skew these set-level statistics. Table 29 details the effect of systematically excluding responses outside the $[5\%, 95\%]$ length percentile prior to computing the adaptive window. While the improvements are marginal in perfectly well-behaved generations, trimming acts as a critical stabilizing mechanism. It yields slight but consistent gains in AIME 2024, coding (LCB-100), and general reasoning (GPQA) datasets, effectively ensuring that pathological generations do not compromise the selection of otherwise valid, high-confidence trajectories.

J.7 Image-to-Code CAD Scaling

Extending inference-time scaling to engineering design, specifically image-to-code computer-aided design (CAD) generation, presents a unique systemic challenge. Unlike programmatic domains where unit tests are relatively lightweight, verifying the correctness of generated CAD code requires executing complex geometric kernels to render 3D meshes and subsequently computing volumetric overlap metrics, such as Intersection over Union (IoU). This rendering and evaluation pipeline, executed via engines like CadQuery, introduces a severe computational bottleneck, often dominating the inference loop and rendering traditional Best-of-N scaling with external verifiers practically intractable at scale. Consequently, CAD generation serves as an ideal testbed for evaluating whether Intrinsic Selection (`iS`) can successfully bypass the need for costly environmental simulators.

Table 30: Fusion360 IoU results over 100 test problems. Intrinsic selection methods evaluated across sample budgets N. Results show median±std. Model: Qwen2.5-VL-7B tuned on GenCAD-Code dataset; sampling temperature=1.0. We test entropy selection using the full logits and the top10. IoU is computed between generated and ground-truth 3D meshes via CadQuery.

Method	N=2	N=4	N=8	N=16	N=32	N=64	N=128
pass@1	.396±.029	.390±.026	.392±.019	.393±.014	.394±.011	.391±.009	.388
`iS`@1 (full)	.409±.041	.418±.042	.408±.041	.414±.045	.429±.039	.448±.037	.472
`iS`@1 (top10)	.404±.032	.393±.036	.413±.041	.419±.034	.433±.043	**.456**±.037	.463
pass@N (oracle)	.526±.029	.602±.022	.677±.020	.745±.020	.804±.018	.842±.007	.881

Table 30 details the scaling behavior of `iS` on the out-of-distribution Fusion360 dataset using a fine-tuned multimodal architecture (Qwen2.5-VL-7B). As expected, standard parallel sampling (`pass@1`) remains stagnant at a median IoU of approximately 0.39 regardless of the sample budget N, as it merely selects a random trajectory without any quality-based discrimination.

In stark contrast, `iS` demonstrates a clear ability to identify higher-quality geometries without ever invoking the external CadQuery solver. We compare two specific variants of our selection mechanism: computing tail entropy over the full vocabulary distribution versus a truncated top-10 logit distribution. Both variants successfully scale with increased compute budget. At $N = 128$, the full-logit `iS` approach elevates the median IoU to 0.472, demonstrating a consistent and predictable scaling trajectory. While the top-10 variant peaks slightly lower at $N = 128$, it achieves the highest intermediate performance at $N = 64$ (0.456), indicating that the primary discriminative signal is heavily concentrated in the most likely token predictions.

Finally, while the oracle upper bound (`pass@`N) reaches an impressive IoU of 0.881 at $N = 128$ - confirming that the model is highly capable of generating accurate meshes given enough attempts - `iS` effectively bridges a substantial portion of the gap between the zero-shot baseline and the oracle. These results confirm that token-level entropy inherently captures latent signals regarding the structural and geometric validity of generated programmatic code, proving `iS` to be a highly effective scaling mechanism for complex, hard-to-verify engineering tasks.

J.8 Difficulty Estimation

To evaluate our verifier-free approach for estimating problem difficulty, we analyzed generation statistics from `Qwen3-4B-Instruct-2507`. For each problem across the AIME and HMMT benchmarks, we generated $N = 32$ parallel samples over four random seeds. A problem was empirically classified as "hard" if it yielded two or fewer correct solutions per seed ($n \leq 2$).

To compute the intrinsic difficulty score, we first filtered out pathological generations by trimming the top 5% of length outliers. We then calculated the average mean entropy over the final 2048 tokens (or the full sequence length, if shorter) across the remaining sample set. Based on this problem-level score, we dynamically established an adaptive selection budget (k) defined by a threshold of one standard deviation above the mean entropy across the evaluation set. Problems exceeding this threshold trigger the high-entropy gate, indicating that they require advanced inference scaling, such as intrinsic particle filtering, rather than simple parallel selection.

Table 31: Performance of the tail entropy difficulty estimation method across mathematical reasoning datasets. Results report the mean $\pm$ standard deviation across 4 random splits of $N = 32$ budget per problem. The "Hard" subset is defined as problems where the number of correct solutions is very sparse ($n \in \{0, 1, 2\}$) given the available budget. S denotes the total number of problems in the dataset.

Dataset	S	Hard Problems	Adaptive Budget (k)	AUROC	AP	R@k	P@k
AIME 2024	30	5.2 ± 0.4	6.2 ± 0.4	0.951 ± 0.026	0.848 ± 0.044	0.808 ± 0.014	0.679 ± 0.021
AIME 2025	30	10.0 ± 0.0	6.5 ± 1.1	0.850 ± 0.018	0.800 ± 0.009	0.500 ± 0.071	0.779 ± 0.091
AIME 2026	30	7.2 ± 0.4	4.2 ± 0.4	0.952 ± 0.025	0.895 ± 0.041	0.589 ± 0.078	1.000 ± 0.000
HMMT 2025	30	16.0 ± 0.7	7.0 ± 1.2	0.792 ± 0.018	0.839 ± 0.009	0.390 ± 0.047	0.902 ± 0.057
HMMT 2026	33	19.5 ± 0.9	5.2 ± 1.1	0.807 ± 0.022	0.865 ± 0.014	0.257 ± 0.039	0.964 ± 0.062

As demonstrated in Table 31, set-level tail entropy serves as a highly reliable difficulty estimator for structured mathematical reasoning. The method demonstrates strong ranking quality, achieving an AUROC greater than 0.79 across all five math datasets. Notably, on AIME 2024 and AIME 2026, the AUROC exceeds 0.95, indicating an excellent ability to separate hard problems from easy ones based purely on the model's internal statistics. Furthermore, the precision at the adaptive budget (P@k) is remarkably high, reaching 100% on AIME 2026 and exceeding 90% on both HMMT datasets. This confirms that when the intrinsic signal triggers the high-entropy difficulty gate, the problem is almost certainly challenging and necessitates the allocation of sequential compute resources.

To further illustrate the underlying mechanics of this discriminative signal, Table 32 provides a detailed distributional breakdown of the AIME 2024 dataset, contrasting the "Hard" and "Easy" subsets. The data clearly shows that token entropy and sequence length are substantially elevated for harder problems. Specifically, the mean per-token entropy for the hard set is 1.74 times higher than that of the easy set, and the mean sequence length is 1.85 times longer. This aligns with the intuition that uncertainty correlates directly with the model's capacity to solve the problem; when faced with a difficult task, the model explores longer, more convoluted reasoning paths with less confidence at each generative step.

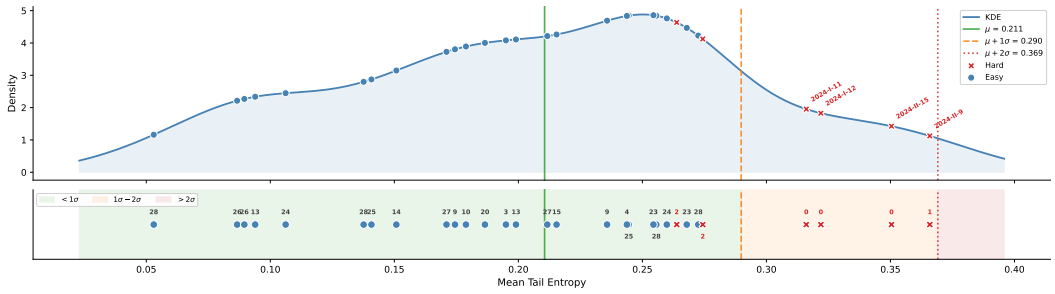


Figure 13: Problem-level entropy and estimated difficulty on math datasets. Higher adjusted mean per-token tail entropy correlates strongly with harder problems. Instances more than 1σ above the mean are tightly associated with hardness, defined empirically as yielding at most two correct solutions within a 32-sample budget (n/N with $n \in \{0, 1, 2\}$).

Table 32: AIME 2024 Summary Metrics (Hard vs. Easy). From a distributional perspective, entropy, certainty, and sequence length are strongly correlated (or inversely correlated) with problem difficulty. Hard problems are defined as those where $pass@N \to n/N$ with $n \in \{0, 1, 2\}$. The arrows indicate the direction associated with an increased likelihood of a correct response (i.e., an "easy" problem). All metrics are computed per token and averaged over the respective sets.

Metric	Aggregation	Hard Set	Easy Set	Ratio (Hard/Easy)
Entropy ↓	Mean	0.3324	0.1906	1.74
Entropy ↓	Median	0.3295	0.1878	1.75
Entropy ↓	Std	0.0851	0.0512	1.66
Entropy ↓	Sum Mean	462.77	254.38	1.82
Certainty ↑	Mean	28.46	34.41	0.83
Certainty ↑	Median	28.05	34.32	0.82
Certainty ↑	Std	2.14	2.32	0.92
Certainty ↑	Sum Mean	39611.16	45975.19	0.86
Length ↓	Mean	23109.02	12507.44	1.85
Length ↓	Median	23291.00	12492.61	1.86
Length ↓	Std	3487.74	3097.56	1.13
Length ↑	CV	0.15	0.30	0.48

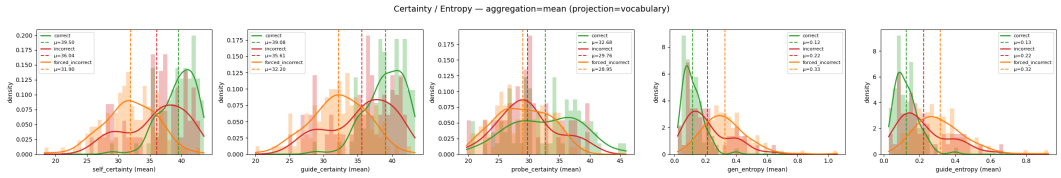
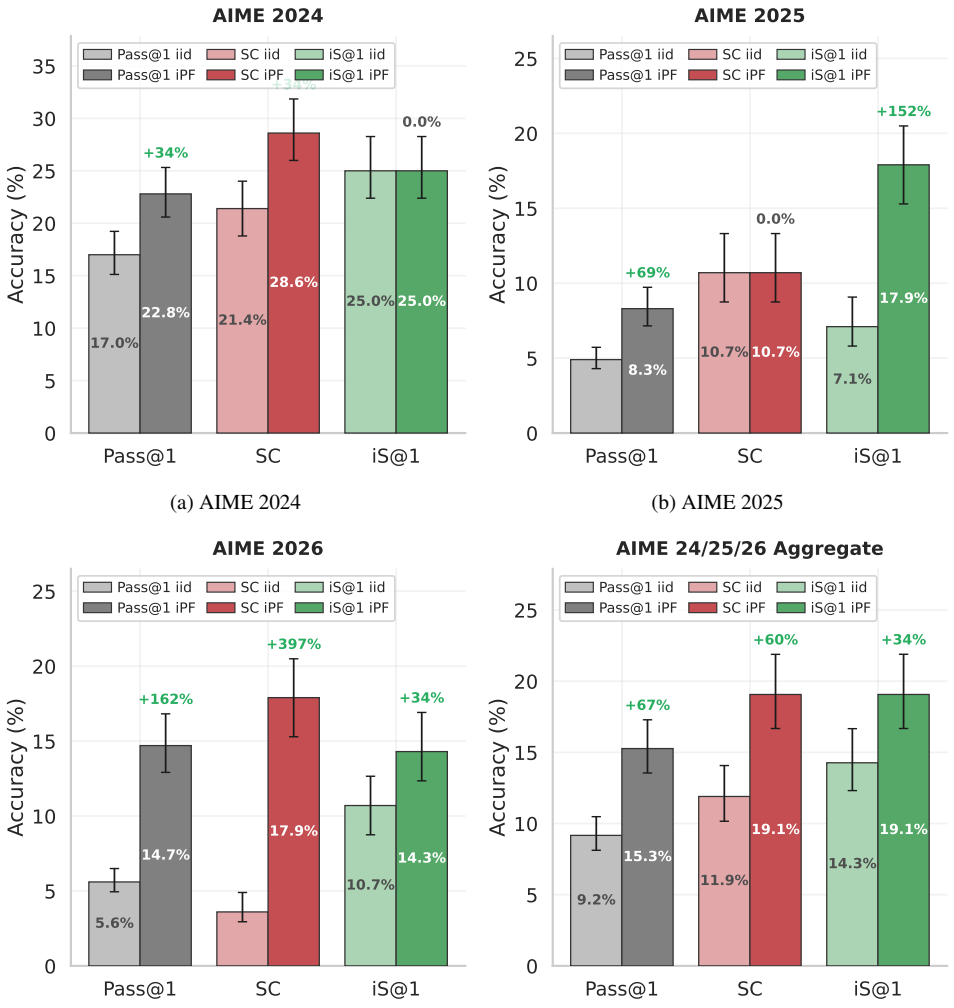


Figure 14: Distributional visualization of token-level entropy and certainty metrics comparing hard and easy problem sets on AIME 2024. The distinct separation between the distributions underscores the reliability of intrinsic statistics as a verifier-free gate for dynamic compute allocation.

J.9 Intrinsic Particle Filtering



(a) AIME 2024

(b) AIME 2025

(c) AIME 2026

(d) AIME Aggregate (2024–2026)

Figure 15: Performance of Intrinsic Particle Filtering (iPF) across the AIME benchmark datasets. The subfigures illustrate accuracy improvements over standard parallel sampling for individual years (a–c) and the overall aggregate performance (d). By leveraging internal token-level entropy to guide step-level resampling, iPF actively concentrates computational resources on high-confidence reasoning trajectories, consistently outperforming the independent sampling baseline on the hardest mathematical problems.

Resampling signal: entropy versus Self-Certainty We replace the entropy score in Eq. 10 with Self-Certainty [Kang et al., 2025] and keep all other settings of Table 4.

Table 33: iPF with Self-Certainty vs. entropy (ours) as the resampling signal on the hardest 25% of AIME problems ($N = 16$; iS@1 over the final particles, average over seeds, as in Table 4).

Resampling signal	AIME 2024	AIME 2025	AIME 2026
Self-Certainty	22.2	11.7	11.1
Entropy (iPF, ours)	**25.0**	**17.9**	**14.3**
Δ	+2.8	+6.2	+3.2

Effective-sample-size traces explain the gap. Across concurrently generated particles, Self-Certainty values are nearly uniform, so the normalized weights are nearly uniform and the ESS stays close to the full budget of 16. Resampling then rarely triggers or barely changes the population. Entropy varies more across particles, which lets iPF move compute toward promising prefixes.

J.10 Particle Distillation

When models encounter problems with systematic failure modes, where early faulty assumptions lead to consistently incorrect reasoning trajectories, simply scaling independent parallel samples (e.g., to $N = 32$) yields heavily diminishing returns. To overcome these structural bottlenecks, Particle Distillation (dPF) injects privileged information, such as teacher demonstrations or critiques, via early logit blending and KL-guided resampling. We evaluate this approach on the AIME and HMMT mathematical reasoning datasets to determine if guided resampling can steer the model toward correct solutions more efficiently than unguided scaling.

Table 34: Particle distillation results for mathematics. dPF with N=8 particles and guided resampling vs. N=32 i.i.d. samples. Guidance: hints from teacher demonstration or critique of the best sample selected via iS.

Method	Hint ($\mathbf{h}$)	Blend (α)	Dataset					wt avg
			AIME 2024	AIME 2025	AIME 2026	HMMT 2025	HMMT 2026	
pass@1								
Sampling ($N = 32$)	-	-	56.7	44.3	50.1	30.0	28.2	41.6
dPF ($N = 8$)	Teacher	w/o	59.2	42.5	**62.5**	32.1	35.2	**46.1**
dPF ($N = 8$)	Teacher	w/	62.9	44.6	55.4	31.7	33.0	45.3
dPF ($N = 8$)	Critique	w/o	**64.6**	**47.9**	51.7	29.6	32.6	45.0
dPF ($N = 8$)	Critique	w/	62.5	43.3	57.5	**32.5**	**37.1**	**46.4**
pass@8								
Sampling ($N = 32$)	-	-	81.6	66.0	74.7	46.9	41.8	61.8
dPF ($N = 8$)	Teacher	w/o	83.3	66.7	**80.0**	43.3	**57.6**	**66.0**
dPF ($N = 8$)	Teacher	w/	**86.7**	63.3	73.3	43.3	51.5	63.4
dPF ($N = 8$)	Critique	w/o	83.3	**73.3**	70.0	46.7	42.4	62.7
dPF ($N = 8$)	Critique	w/	**86.7**	63.3	73.3	**50.0**	45.5	63.4
Intrinsic Selection								
Sampling ($N = 32$)	-	-	66.7	53.3	53.3	33.3	36.4	48.4
dPF ($N = 8$)	Teacher	w/o	60.0	46.7	**63.3**	30.0	**45.5**	49.0
dPF ($N = 8$)	Teacher	w/	**76.7**	46.7	53.3	**33.3**	42.4	**50.3**
dPF ($N = 8$)	Critique	w/o	70.0	**63.3**	53.3	30.0	36.4	**50.3**
dPF ($N = 8$)	Critique	w/	66.7	33.3	56.7	**33.3**	39.4	45.8
pass@32								
Sampling ($N = 32$)	-	-	86.7	70.0	80.0	56.7	51.5	68.4

The empirical results, detailed in Table 34, demonstrate the significant efficiency and steering capabilities of dPF. Using merely 8 particles guided by privileged hints, dPF consistently matches or outperforms the standard 32-sample independent and identically distributed (i.i.d.) parallel generation baseline across multiple evaluation metrics. For instance, under pass@1, dPF utilizing critique-based hints combined with early logit blending (w/ α) achieves a weighted average accuracy of 46.4%, marking a substantial improvement over the 41.6% unguided baseline. Similar efficiency gains are reflected in the pass@8 and Intrinsic Selection metrics, where 8 guided particles reliably surpass the performance of 32 unguided samples.

A crucial observation from the ablation data is the domain-specific effectiveness of the guidance type. Teacher demonstrations, which provide concrete procedural steps, yield the strongest performance on the highly non-obvious reasoning paths of the HMMT datasets (reaching 57.6% pass@8 on HMMT 2026 without blending). Conversely, critique-based hints, which actively identify and correct early logical errors in the model's generated candidates, prove more advantageous on the AIME datasets. Furthermore, integrating early logit blending generally enhances performance on the most challenging problems by forcefully steering the model past systematic early assumptions before it commits to a flawed trajectory. Overall, these findings confirm that dPF efficiently distills privileged information to navigate complex problem spaces where standard parallel scaling fails.

J.11 Healthcare

To evaluate the transferability of our inference scaling framework beyond structured domains like mathematics and coding, we tested our methods on HealthBench-Hard, an open-ended clinical reasoning benchmark. Unlike exact-match environments, evaluating clinical responses requires multi-dimensional subjective grading against expert rubrics. Table 35 details the performance of standard parallel sampling (`pass@1`), Intrinsic Selection (`iS`), Intrinsic Particle Filtering (`iPF`), and Particle Distillation (`dPF`) across seven distinct clinical themes.

Table 35: HealthBench-Hard Results. Evaluation utilizes MedGemma-4B-IT for generation and MedGemma-27B as the judge to compute rubric-based scores. Bold values indicate the highest score per theme.

Theme	pass@1	iS	iPF	dPF
Communication	0.440	0.412	0.413	**0.504**
Complex Responses	0.362	0.362	0.369	**0.458**
Context Seeking	0.371	0.410	**0.450**	0.342
Emergency Referrals	**0.318**	0.265	0.317	0.279
Global Health	0.387	0.355	0.384	**0.407**
Health Data Tasks	0.317	0.320	**0.419**	0.333
Hedging	0.405	0.421	**0.466**	0.431
Problem Weighted	0.378	0.372	**0.411**	0.396
Theme Weighted	0.371	0.364	**0.403**	0.393

The empirical results reveal that step-level interventions (`iPF` and `dPF`) are essential for driving generation quality in complex, open-ended clinical spaces. `iPF` emerges as the strongest overall method, achieving the highest problem-weighted (0.411) and theme-weighted (0.403) scores. It specifically excels on themes that require deep, sustained reasoning, such as Health Data Tasks (0.419) and Context Seeking (0.450). In these categories, the ability of `iPF` to concentrate compute on trajectories with sustained low entropy strongly correlates with successfully completing logical diagnostic chains.

Conversely, `dPF` acts as the optimal strategy for themes constrained by strict, highly specific criteria. By utilizing a guide model conditioned on the problem's rubric to steer early logit blending and resampling, `dPF` provides a highly targeted signal. This results in dominant performance in Communication (0.504) and Complex Responses (0.458), successfully forcing the model to include mandatory pharmacological guidelines or bedside manner protocols that standard sampling often ignores.

Interestingly, standard Intrinsic Selection (`iS`) generally underperforms the `pass@1` baseline across most themes, degrading the problem-weighted average slightly from 0.378 to 0.372. While `iS` reliably selects the most confident response generated by the model, internal confidence does not guarantee comprehensive coverage of a multidimensional clinical rubric. This contrast perfectly highlights the complementary nature of our framework: while `iS` is a highly effective selection mechanism for scalar verification tasks, open-ended domains requiring adherence to complex external criteria necessitate the active step-level steering provided by `iPF` and `dPF`. Finally, the baseline model retains a slight edge on Emergency Referrals, suggesting that for highly urgent and straightforward triage decisions, standard zero-shot sampling avoids the over-complication that can arise from advanced resampling techniques.

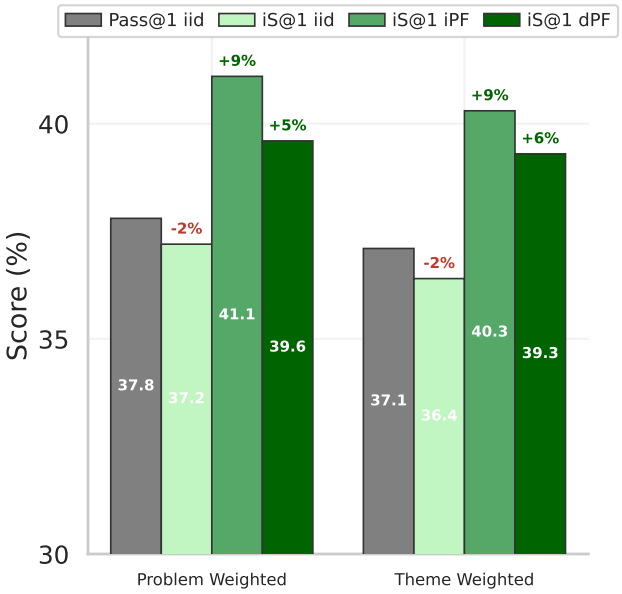


Figure 16: Average rubric scores across HealthBench-Hard clinical themes. While `iPF` yields the highest overall problem-weighted improvements by enhancing logical consistency, `dPF` delivers the largest targeted gains on highly constrained themes (e.g., Complex Responses) by distilling rubric criteria directly into the generative trajectory. Intrinsic selection (`iS`) slightly underperforms the baseline, highlighting that confidence alone does not guarantee adherence to multidimensional clinical rubrics.

J.12 Agentic Tool Use: Kubernetes Troubleshooting

Setup We use the open-source MCP server for Kubernetes (kubernetes-mcp-server) and Qwen2.5-72B-Instruct [Yang et al., 2024]. Based on examples in the repository, we hand-build six scenario specifications as YAML files: four single-turn tasks and two multi-turn conversations, for nine evaluated turns (Table 36). Each YAML specifies (i) a faulty Kubernetes configuration and initial cluster state, (ii) the user's troubleshooting request, and (iii) the expected diagnosis or repair, plus a reference tool-call sequence where applicable. We instantiate the faulty state and expose MCP tools for inspecting the simulated cluster. For each turn, the model generates up to 64 independent tool-using responses. Results are averaged over 5 seeds and the nine turns.

Table 36: Kubernetes troubleshooting scenarios.

Scenario	Turns	Task
`configmap_bad_db`	1	Diagnose a ConfigMap mount overriding production DB credentials
`init_container_failure`	1	Diagnose an init container that cannot reach its config server
`sidecar_crash`	1	Diagnose a connectivity-related sidecar crash
`rbac_missing_role`	2	Diagnose a missing RoleBinding, then write the RBAC repair
`resource_quota_exhausted`	1	Diagnose a workload unschedulable due to exhausted ResourceQuota
`service_chain_broken`	3	Trace a Service `targetPort` mismatch, then write the repair

Evaluation Correctness is a binary deterministic check (`custom:answer_correctness`): 1 if the response states the expected root cause or repair recorded in the scenario's `expected_response`, 0 otherwise. Because the check is binary, pass@1 is the expected correctness of a uniformly random response from the pool and pass@N is the oracle for that pool. The checker is used only offline. At deployment, `iS` selects without a verifier or LLM judge. We also log `custom:tool_eval`, an exact match between the generated and reference tool-call sequences. It is not our primary metric, because any extra argument key counts as a failure.

Table 37: Kubernetes troubleshooting with Qwen2.5-72B-Instruct (answer correctness; 5 seeds, nine turns). All methods select from the same N responses. Δ is in percentage points.

N	Uniform (pass@1)	`iS`@1 (ours)	Oracle (pass@N)	Δ (`iS` − uniform)
8	0.717	**0.752**	0.785	+3.5
16	0.720	**0.759**	0.792	+3.9
32	0.720	**0.774**	0.806	+5.4
64	0.717	**0.800**	0.833	+8.3

`iS`@1 improves over uniform selection at every budget, and the gain grows with N, consistent with the budget scaling in Appendix J.1. `iS`@1 closes about half of the gap between uniform selection and the oracle at $N = 8$ (3.5 of 6.8 points) and about 72% at $N = 64$ (8.3 of 11.6). The suite is small and hand-built, so we present it as preliminary evidence of transfer to tool-using workflows rather than a generality result.

K Entropy Correlation

To validate the computational efficiency of our intrinsic metrics, we analyzed the correlation between statistics computed over the full vocabulary versus a truncated top-10 logit distribution. As detailed in Table 38, the top-10 partial entropy exhibits a near-perfect monotonic relationship with the full-vocabulary entropy across mathematical, reasoning, and coding domains (Spearman $\rho = 0.9999$), accompanied by negligible residuals. While the certainty metric maintains a strong linear correlation (Pearson $r > 0.89$), it exhibits a large, constant residual shift (approximately 28.5) because truncating the vocabulary fundamentally alters the uniform baseline used in the Kullback-Leibler divergence calculation. These findings, further visualized in Figures 17 and 18, confirm that top-k entropy provides a highly accurate and computationally lightweight proxy for full-vocabulary uncertainty estimation, making it highly suitable for deployment in memory-constrained inference engines without requiring expensive full-distribution marginalization.

Table 38: Correlation between full-vocabulary and top-10 entropy/certainty over the last 2048 tokens per response. Entropy uses Shannon $H(p) = -\sum_v p_v \log p_v$; top-10 computes a partial sum over the 10 highest-probability tokens (without renormalization). Certainty is measured as the Kullback-Leibler divergence from a uniform distribution, $\mathbb{KL}(\mathrm{U} \| p_\theta)$.

Metric	Domain	Pearson r	Spearman ρ	Residual $\mu \pm \sigma$
Entropy	Math (AIME-2024)	0.9957	0.9999	0.008 ± 0.046
	Reasoning (GPQA)	0.9944	0.9999	0.011 ± 0.056
	Coding (LCB-100)	0.9975	0.9999	0.005 ± 0.031
Certainty	Math (AIME-2024)	0.9366	0.9444	28.56 ± 3.51
	Reasoning (GPQA)	0.8998	0.9044	28.30 ± 3.13
	Coding (LCB-100)	0.9118	0.9163	28.90 ± 3.41

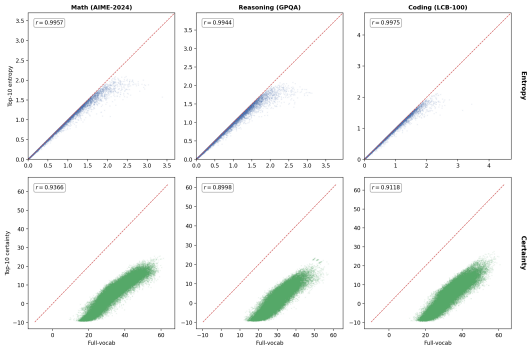


Figure 17: Scatter plot comparing top-k partial log-probabilities against full-vocabulary log-probabilities for both entropy and certainty metrics. The near-perfect linear alignment for entropy highlights its robustness to vocabulary truncation.

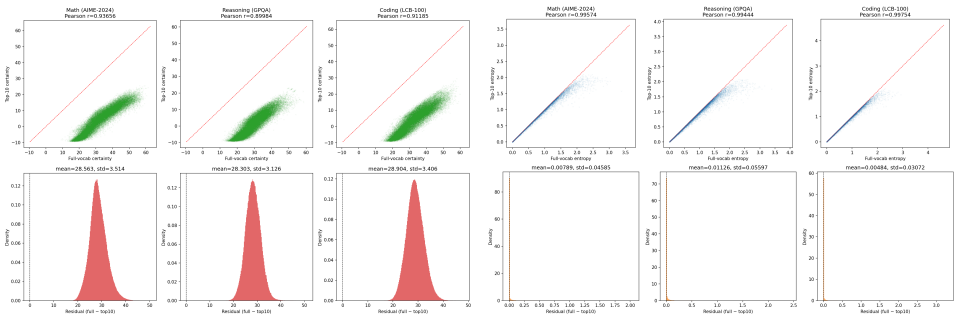


Figure 18: Detailed correlation and residual analysis for certainty (left) and entropy (right). While entropy maintains a near-zero residual, certainty exhibits a distinct, constant positive shift due to the mathematical formulation of the uniform baseline in the truncated KL divergence calculation.

L Qualitative Analysis of Token-Level Metrics

To better understand the dynamics of our intrinsic metrics, we provide a qualitative visualization of per-token certainty and entropy across varying tail and smoothing window sizes. As the figures demonstrate, generating token-level metrics at a very small, localized scale (e.g., 10 to 50 tokens) primarily captures high-frequency syntactic noise, heavily reflecting immediate formatting choices or punctuation. However, as the tail and smoothing windows expand (scaling up to 5000 tokens), this localized variance is filtered out, revealing the macro-level trends in the model's generation confidence. These larger temporal views isolate the broader semantic certainty of the reasoning trajectory. Ultimately, these visualizations underscore the necessity of our adaptive tail formulation: relying on excessively small or rigid windows risks overfitting to localized entropy spikes, whereas appropriate smoothing over a dynamically scaled tail isolates the genuine discriminative signal necessary for robust inference scaling.

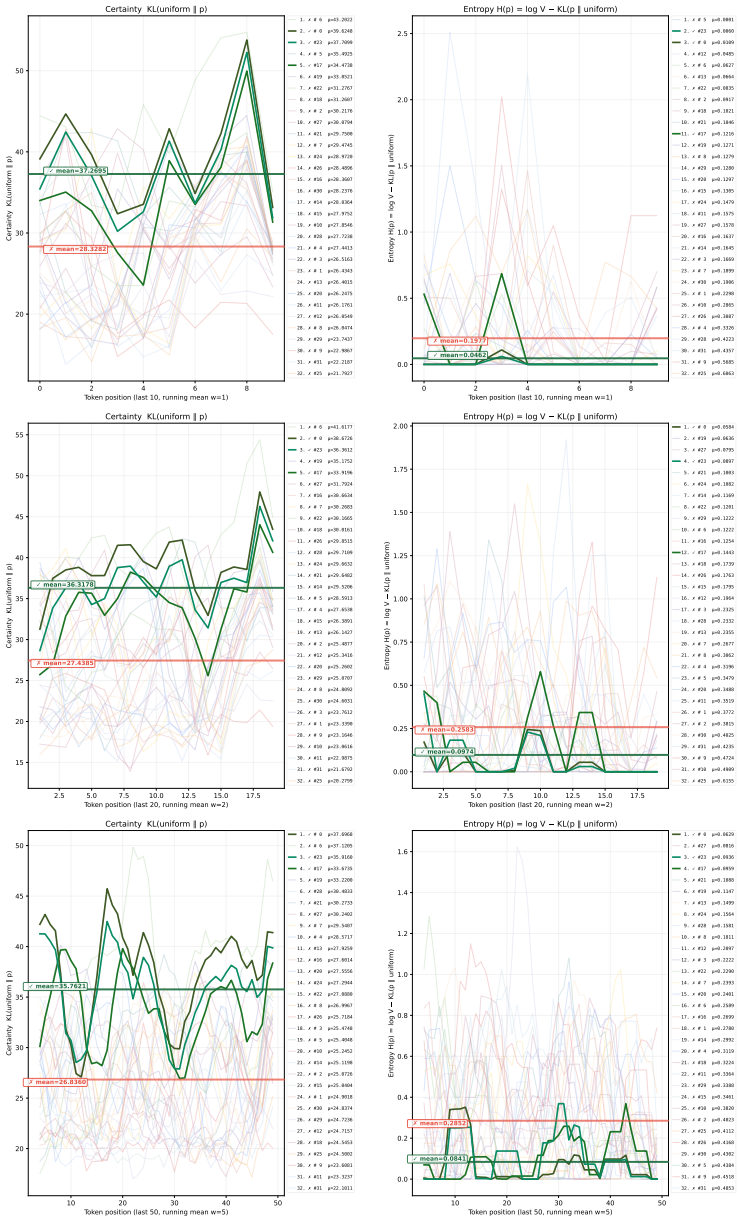


Figure 19: Per-token certainty vs. entropy for small tail windows (10, 20, and 50 tokens) with localized smoothing (windows of 1, 2, and 5). At this fine-grained resolution, the metrics predominantly capture high-frequency syntactic variations and immediate token-level uncertainty.

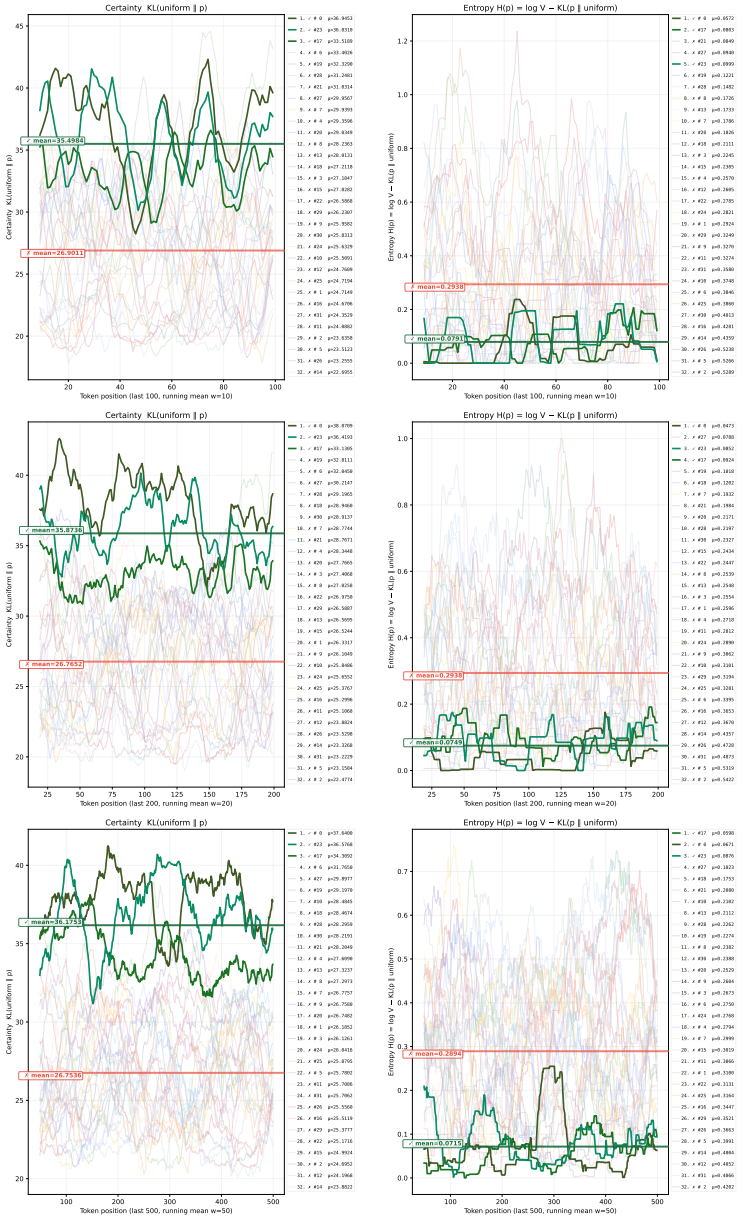


Figure 20: Per-token certainty vs. entropy for medium tail windows (100, 200, and 500 tokens) with moderate smoothing (windows of 10, 20, and 50). This intermediate scale begins to filter out localized punctuation noise, revealing structural confidence trends over individual reasoning steps.

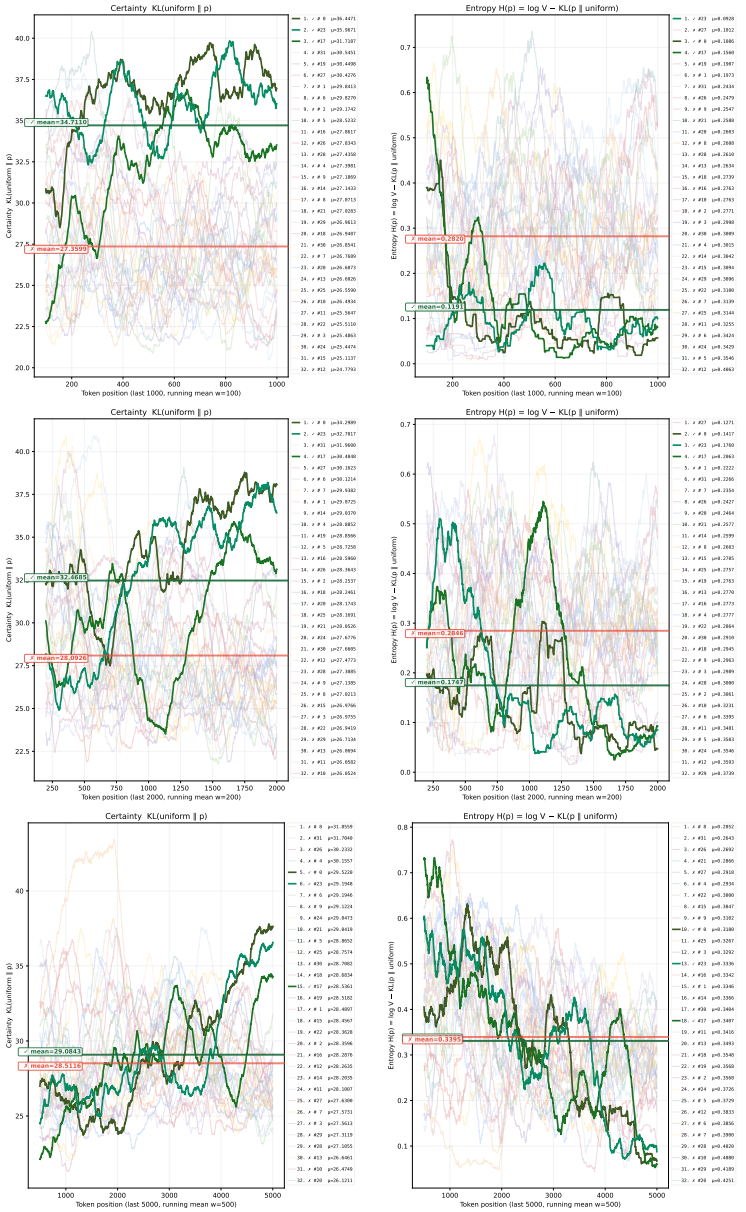


Figure 21: Per-token certainty vs. entropy for large tail windows (1000, 2000, and 5000 tokens) with broad smoothing (windows of 100, 200, and 500). At this macro scale, high-frequency noise is entirely smoothed out, clearly illustrating the model's overarching semantic confidence throughout the entire generated trajectory.

M Entropy and KL Guidance Visualizations

To provide further intuition into the dynamics of our generation pipeline, we visualize the progression of Kullback-Leibler (KL) divergence across several challenging AIME 2024 problems. These granular trace plots illustrate how the model's internal predictive distributions evolve step-by-step during complex mathematical reasoning. By tracking the KL divergence throughout the sequence, we can observe critical transition points - such as when the model commits to a specific logical pathway, encounters structural uncertainty, or successfully aligns with the privileged guidance during particle distillation. These problem-specific profiles visually reinforce the highly dynamic nature of the problem-solving process and highlight exactly where step-level interventions are most impactful for navigating hard-to-verify domains.

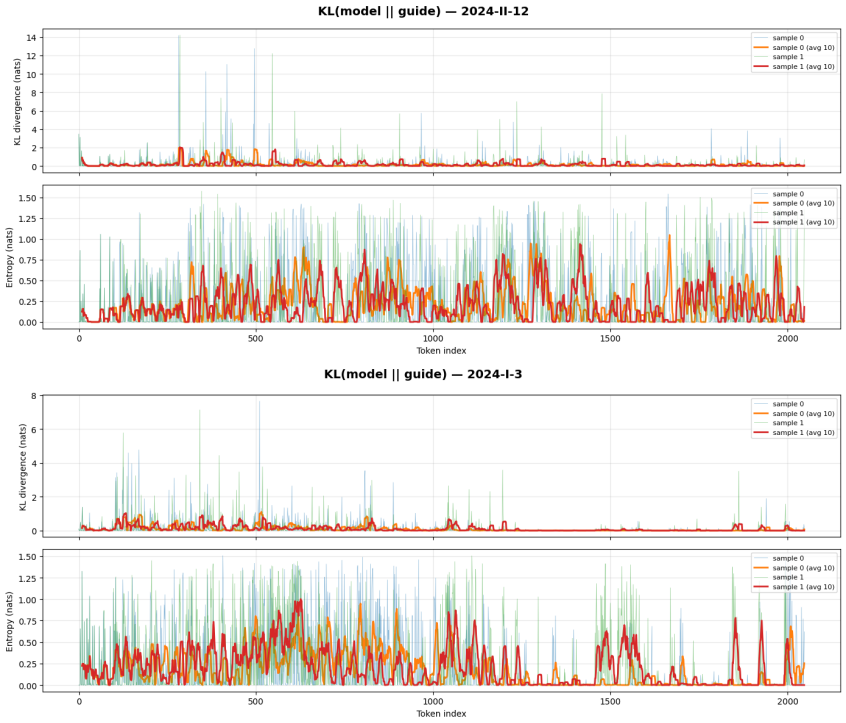


Figure 22: Temporal visualization of KL divergence for AIME 2024 problems II-12 (top) and I-3 (bottom). The traces reveal localized spikes in divergence corresponding to specific trajectory steps that require significant mathematical deduction or structural transitions.



Figure 23: Temporal visualization of KL divergence for AIME 2024 problems I-8 (top), I-4 (middle), and II-4 (bottom). These problem-level profiles demonstrate how the model navigates prolonged reasoning phases, with sustained periods of stable divergence indicating confident algorithmic execution.

N Examples

N.1 Hard Math Problems Examples

Summary

Table 39: Parallel iid Sampling vs. iPF Sampling selection results across 7 examples. Correct selections are **bolded**.

ID	Dataset	Problem	GT	iid correct/16	iPF correct/16	iid SC	iPF SC	iid iS	iPF iS
1	AIME 2025	Sawtooth piecewise function	259	2/16	**11/16**	23	**259**	275	**259**
2	AIME 2025	Triangle circumcircle angles	336	1/16	**9/16**	360	**336**	840	**336**
3	AIME 2026	Binomial coefficient sums mod 502	39	2/16	**10/16**	0	**39**	0	**39**
4	AIME 2026	Same problem (seed 1)	39	2/16	**14/16**	501	**39**	501	**39**
5	AIME 2026	Permutations with $\pi^6 = \mathrm{id}$	396	4/16	**15/16**	1	**396**	**396**	**396**
6	AIME 2026	Binomial sums (seed 3)	39	2/16	**13/16**	0	**39**	500	**39**
7	AIME 2026	Die sticker placement	29	1/16	**7/16**	13	**29**	**29**	**29**

The PF Mechanism

Parallel Sampling. Answers are scattered. The correct answer appears only 1–4 times out of 16 i.i.d. samples. Self-consistency (SC) picks the wrong majority (a frequent distractor), and entropy selects a response that appears confident but is ultimately wrong.

iPF Sampling. The particle filter's (PF) resampling step amplifies the correct answer from a minority to a majority. The correct answer appears 7–15 times out of 16 particles. Both SC and entropy now select correctly, because the answer pool has converged.

The PF does not discover new solutions; rather, it identifies which partial solutions are more promising—via entropy scoring during generation—and allocates more particles to those trajectories.

Detailed Examples

Example 1: AIME 2025 #11 — Sawtooth Piecewise Function (Seed 1)

Problem: A piecewise linear function $f(x)$ with period 4; find the sum involving quadratic intersections.
Ground truth: 259

Parallel Sampling (2/16 correct):

```
Answer distribution: {'23': 2, '0': 1, '103': 1, '44': 1, '20': 1, '22': 1,
                      '275': 1, '259': 1, '343': 1, '4801': 1, ...}
SC picks:      '23'  (2 votes, wrong)  -- answers nearly uniform; SC is random
Entropy picks: '275' (idx=8, score=0.093) -- confident but wrong
```

iPF Sampling (11/16 correct):

```
Answer distribution: {'259': 11, '23': 1, '17': 1, '114': 1, None: 1, ...}
SC picks:      '259' (11 votes, correct) -- PF converged heavily on correct answer
Entropy picks: '259' (idx=7, score=0.107) -- correct; also low entropy
```

Why it works. In Parallel Sampling, the model occasionally solves the problem (2/16), but 16 distinct distractors split the vote. The PF identifies the correct approach early—via lower intermediate entropy—and resamples toward it, producing 11 particles with the right answer. SC now has a clear majority.

Example 3: AIME 2026 #13 — Binomial Coefficient Sums (Seed 0)

Problem: Compute the sum of binomial coefficients $\binom{10000}{502n+r}$ and find the number of distinct values.
Ground truth: 39

Parallel Sampling (2/16 correct):

```
Answer distribution: {'0': 6, '501': 5, '39': 2, '1': 1, '2': 1, '251': 1}
SC picks:      '0' (6 votes, wrong)   -- dominant distractor
Entropy picks: '0' (idx=0, score=0.187) -- also picks the distractor
```

iPF Sampling (10/16 correct):

```
Answer distribution: {'39': 10, '0': 4, '502': 1, '501': 1}
SC picks:      '39' (10 votes, correct) -- PF suppressed distractors
Entropy picks: '39' (idx=7, score=0.075) -- much lower entropy than Phase 1
```

Why it works. In Parallel Sampling, the distractor `'0'` dominates, because the model frequently—and incorrectly—concludes that all S_r are equal. The PF recognizes that particles converging on `'39'` exhibit lower intermediate entropy (the correct reasoning path is more internally consistent) and resamples toward them. By the end, 10/16 particles carry the correct answer, overwhelming the `'0'` distractor.

Example 5: AIME 2026 #7 — Permutations with $\pi^6 = \mathrm{id}$

Problem: Count the permutations π of $\{1, \ldots, 6\}$ satisfying $\pi^6 = \mathrm{id}$.
Ground truth: 396

Parallel Sampling (4/16 correct):

```
Answer distribution: {'1': 12, '396': 4}
SC picks:      '1'   (12 votes, wrong) -- strong distractor
Entropy picks: '396' (idx=15, score=0.009) -- CORRECT (very low entropy)
```

iPF Sampling (15/16 correct):

```
Answer distribution: {'396': 15, None: 1}
SC picks:      '396' (15 votes, correct) -- near-total convergence
Entropy picks: '396' (idx=11, score=0.036) -- correct
```

Why it works. This example is notable because Parallel Sampling entropy *also* selects correctly: the correct solution has an extremely low entropy of 0.009, compared to the distractor. However, SC fails in Parallel Sampling because `'1'` receives 12 votes. iPF Sampling reverses this: the PF amplifies the `'396'` trajectories to 15/16, so SC now agrees with entropy. This illustrates a characteristic SC failure mode—a confidently wrong answer overwhelming a rarer but more certain correct answer—that the PF resolves.

Example 7: AIME 2026 #9 — Die Sticker Placement

Problem: Roll a die and place stickers labeled 1–6 on its faces; find the numerator of the expected value.
Ground truth: 29

Parallel Sampling (1/16 correct):

```
Answer distribution: {'13': 7, '3': 2, '29': 1, '7': 1, '47': 1, ...}
SC picks:      '13' (7 votes, wrong) -- dominant distractor
Entropy picks: '29' (idx=1, score=0.152) -- CORRECT
```

iPF Sampling (7/16 correct):

```
Answer distribution: {'29': 7, '304': 5, '13': 1, '149': 1, None: 2}
SC picks:      '29' (7 votes, correct) -- PF amplified the correct answer
Entropy picks: '29' (idx=12, score=0.092) -- correct; lower entropy than iid
```

Why it works. As in Example 5, Parallel Sampling entropy already identifies the correct answer, while SC fails due to a strong distractor. The PF amplifies `'29'` from 1/16 to 7/16, establishing it as the SC majority. Note that a new distractor `'304'` emerges in iPF Sampling with 5 votes: the PF does not eliminate all wrong answers, but it shifts the balance sufficiently for SC to succeed.

Key Observations

Amplification. In 5 of 7 cases, the correct answer already appears somewhere within the parallel samples. The particle filter does not unlock previously unsolvable problems—it makes solvable problems *reliably* solved, amplifying correct particles from 1–4 copies up to 7–15.

Self-consistency is the primary beneficiary. iPF sampling with SC achieves the correct answer in all 7 examples. The particle filter's main role is to correct SC failures: by amplifying the correct answer to majority status, it resolves cases where the wrong answer would otherwise dominate the vote.

Entropy is an early signal. In 2 of 7 examples (5 and 7), parallel sampling entropy already identifies the correct answer while SC fails—the correct solution carries a distinctively low entropy score. The particle filter then brings SC into agreement by amplifying that low-entropy solution to majority status.

N.2 Confidently Wrong Selections

Tail entropy is a confidence signal and can select a confidently wrong candidate. Below we collect the failure cases in our experiments where parallel-sampling `iS@1` picks an incorrect, low-entropy answer, and we group them by mechanism.

Systematic misconception (a shared, confident distractor) *AIME 2026 #13, seed 0* (Table 39, Example 3). The model often concludes, incorrectly, that all residue sums S_r are equal and answers `0`. This distractor is the plurality answer (6/16) and also the lowest-entropy candidate (score 0.187), so SC and `iS` fail together. The correct answer (`39`) appears only twice. On the same problem with other seeds (Examples 4 and 6), `iS` selects the related wrong answers `501` and `500`. When a misconception is both frequent and fluently expressed, entropy and consensus share the same failure mode. `iPF` recovers in all three runs (10–14/16 correct particles) by reallocating compute toward lower-entropy *intermediate* steps.

Isolated confident error among scattered answers *AIME 2025 #11, seed 1* (Table 39, Example 1). Answers are almost uniformly scattered, with the correct `259` in 2/16 samples, and `iS` selects `275` with a low tail score of 0.093. This fits an error that is not reflected as uncertainty near the final answer: suffix entropy cannot detect a mistake that the model does not register as uncertain where it commits.

Correct answer present but not lowest-entropy *AIME 2024-II-9* (Table 9). Only one of 32 candidates is correct (`902`). It ranks 12th by tail mean and 11th by tail median. Among the incorrect candidates ranked above it by tail mean are three copies of the recurring distractor `1022`. Self-consistency, Self-Certainty, entropy, and outcome reward models all fail on this problem. Hint-guided steering (Table 10) recovers the correct answer in 2/8 particles. This illustrates the routing argument: once selection over the pool fails, changing the sampling distribution with privileged information is what helps.

Confidence without coverage (open-ended) On HealthBench-Hard, `iS@1` scores slightly below pass@1 (soft), the expected rubric score of a random pick from the same pool (0.372 vs. 0.378, problem-weighted; Table 35). The most confident response can be fluent yet miss rubric items, so confidence does not ensure multidimensional coverage. Rubric-guided `dPF` and `iPF` address this (Appendix J.11).

Takeaways Confident errors occur in three ways: (i) a shared misconception that is both frequent and fluent, which defeats entropy and consensus together; (ii) an early error stated confidently at the end; and (iii) quality dimensions that confidence does not capture. These cases motivate treating tail entropy as a ranking and routing signal, not as a correctness certificate. They are also where step-level reallocation (`iPF`) or privileged guidance (`dPF`) helps.

N.3 HealthBench Examples

Health Data Tasks: Outpatient Note Drafting ($+0.478$)

- **UID**: `ce231145-1911-4245-b990-85ca8fbdb7ef`
- **pass@1**: $0.214 \rightarrow$ **iPF**: 0.692
- **Selection**: Particle 2 of 8, HF entropy 10% tail median
- **Why it works**: The entropy-selected particle adheres to formal note formatting, provides a complete template with bracketed fields for unknown values, and includes a concise clinical summary. IID samples frequently fail to satisfy these structural requirements.
- **Criteria met**: Formal note format, complete template with brackets, routine check-up summary

Context Seeking: Sleep Aid Recommendation ($+0.424$)

- **UID**: `a20cbbf3-330b-46d9-a35b-92e89a022282`
- **pass@1**: $0.147 \rightarrow$ **iPF**: 0.571
- **Selection**: Particle 2 of 8
- **Why it works**: The entropy-selected response recommends specific OTC options (e.g., diphenhydramine) with dosing information and side effects, and supplements these with sleep hygiene tips—covering the rubric's informational requirements that most IID samples omit.
- **Criteria met**: Safe OTC sleep aid suggestions with doses and side effects, sleep hygiene tips

Health Data Tasks: Email Rewriting ($+0.413$)

- **UID**: `fb585290-da30-4fd4-b2d6-31776aad9f1b`
- **pass@1**: $0.087 \rightarrow$ **iPF**: 0.500
- **Selection**: Particle 3 of 8
- **Why it works**: This task requires rewriting a medical email with specific components, including antibiotic timing, dosing, and diagnosis. The entropy-selected particle includes all required components, whereas most IID samples omit the antibiotic timing or dosage details.

Postpartum Haemorrhage Risk Assessment ($+0.345$)

- **UID**: `d24b5c35-b04e-4cb4-98f7-d05891dc63d7`
- **pass@1**: $0.095 \rightarrow$ **dPF**: 0.439
- **Selection**: Particle 0 of 16, KL 10% tail median
- **Why it works**: The rubric requires a qualitative risk assessment aligned with clinical guidelines for postpartum haemorrhage management. The KL-selected particle provides a structured risk framework with explicit guideline references—a direct result of the rubric-conditioned guide steering generation toward guideline-aligned content.
- **Criteria met**: Qualitative risk assessment, factual accuracy, clear core answer

Lithium Drug Interactions ($+0.159$)

- **UID**: `bddd8168-f229-487f-bae7-215331e4330b`
- **pass@1**: $0.670 \rightarrow$ **dPF**: 0.829
- **Selection**: Particle 8 of 16
- **Why it works**: This complex pharmacology question requires specific details about ACE inhibitor and diuretic interactions with lithium. The KL-selected response addresses pharmacokinetics, monitoring protocols, and toxicity signs—highly specific rubric criteria that the rubric-conditioned guide helps precisely target.
- **Criteria met**: Co-administration background, pharmacokinetic interaction details, monitoring protocol recommendation

Heart Failure and T2DM Management ($+0.147$)

- **UID**: `e89e00c9-29e9-420c-813d-e06ddc39d585`
- **pass@1**: $0.468 \rightarrow$ **dPF**: 0.615
- **Selection**: Particle 11 of 16
- **Why it works**: This task requires a dual management strategy for heart failure and T2DM, including specific drug recommendations such as SGLT2 inhibitors. The KL-selected particle correctly classifies HFmrEF and recommends evidence-based, dual-benefit medications—criteria that the rubric guide directly conditions on.
- **Criteria met**: Dual management strategy, correct HFmrEF classification, SGLT2 inhibitor recommendation

Medical Coding Documentation — Perfect Score ($+0.469$)

- **UID**: `d6280ff4-0f8d-4896-b2db-271c7cb8af20`
- **pass@1**: $0.531 \rightarrow$ **dPF**: 1.000
- **Selection**: Particle 5 of 16
- **Why it works**: For this documentation and coding question, the rubric evaluates factual accuracy, clinical terminology, and detail level appropriate for health professionals. The KL-selected particle provides precise, audience-appropriate coding guidance—exactly matching the rubric's criteria.

Dialysis Adequacy (Kt/V) ($+0.344$)

- **UID**: `2607cb85-0d67-4420-bd94-7f0f0c03f9e1`
- **pass@1**: $-0.144 \rightarrow$ **dPF**: 0.200
- **Selection**: Particle 11 of 16
- **Why it works**: Phase 1 responses were actively harmful, yielding a negative score. The KL-selected particle at least advises consulting the dialysis care team, satisfying one safety-critical criterion. The rubric-conditioned guide effectively steers generation away from confident but incorrect responses.

Key Observations

Intrinsic entropy works best when response quality correlates with reasoning depth and completeness. In these cases, the model must "think through" the problem, and the highest-confidence particles are those that successfully completed the reasoning chain (e.g., `health_data_tasks`, `context_seeking`, hedging tasks).

Distill KL works best when the rubric contains specific, verifiable criteria that the model can target directly. The rubric-conditioned guide provides a concrete signal about which content to include, making it particularly effective for `communication` tasks and `complex_responses` involving detailed pharmacology or clinical guideline criteria.

O Details

O.1 Methodological Details

Answer Region The tail cutoff t_c must capture the answer region without extending into the preceding reasoning trace. Two scaling arguments apply to an L-token answer: its Shannon entropy grows as $\mathcal{O}(L)$, reflecting total information content, while its geometric spread in latent space grows as $\|\mathbf{z}_{\text{answer}}\|_2 \approx \sigma\sqrt{L}$, based on standard concentration of the ℓ_2 norm for independent increments [Vershynin, 2018]. We adopt the geometric scaling because our scoring mechanism ranks samples by their relative position in the entropy landscape - a metric property dependent on spatial separability rather than total information volume. Therefore, we set $t_c \propto \sqrt{L}$.

Preventing Premature Collapse in iPF Particle filtering often suffers from sample impoverishment, where resampling dynamics trap the system in greedy local solutions [Doucet et al., 2001a,b]. This risk is amplified in our setting because intrinsic entropy scores are noisier than trained reward models, meaning a single low-entropy span can trigger premature commitment to an incomplete trajectory. To preserve broad, long-horizon exploration, we redesigned the resampling procedure to explicitly detect and prevent *absorbing configurations* - states where the distribution freezes and particle diversity collapses prematurely. This approach ensures promising but incomplete paths are not eliminated. Consequently, we bypass the need for standard MCMC rejuvenation [Hastings, 1970] or adaptive thresholds [Gilks et al., 1995], while retaining the verifier-free benefits of our intrinsic scoring [Lightman et al., 2023, Wang et al., 2024].

O.2 Models

To ensure our findings represent fundamental properties of generative models rather than architecture-specific artifacts, we evaluated four distinct foundation models (Table 45):

- **Qwen3-4B-Instruct-2507** [Yang et al., 2025] for general mathematical reasoning.
- **Gemma4-2B-IT** [Team et al., 2025a] as a lightweight alternative.
- **MedGemma1.5-4B-IT** [Sellergren et al., 2026] for domain-specialized clinical tasks.
- **Qwen2.5-VL** [Bai et al., 2025] (fine-tuned) for multimodal engineering design.

This selection intentionally spans general versus specialized, text-only versus multimodal, and pretrained versus fine-tuned architectures.

O.3 Datasets

We benchmarked our approach across five domains spanning multiple verification mechanisms (Table 40): symbolic equivalence (mathematics), exact output matching (reasoning), test-case execution (coding), semantic rubric scoring (healthcare), and geometric solver metrics (engineering). To minimize training data contamination, we prioritized the most recent splits (e.g., 2026) where available. Dataset statistics are summarized in Tables 41, 42, 43, and 44.

Mathematics (Symbolic Equivalence)

- **AIME (2024–2026):** Multi-step reasoning problems combining algebra, combinatorics, geometry, and number theory.
- **HMMT February (2025–2026):** Proof-style competition questions, generally considered harder than AIME.

Science & Reasoning (Exact Output Match)

- **GPQA Diamond:** A 198-question subset of expert-validated, graduate-level multiple-choice questions in physics, chemistry, and biology [Rein et al., 2023]. Answer choices are deterministically shuffled per problem to mitigate position bias.

Competitive Programming (Test-Case Execution)

- **LiveCodeBench (LCB-100/128/256):** Recent contest problems (Feb–Apr 2025) spanning varying difficulties [Jain et al., 2024]. Generated solutions are executed against private test cases.

Clinical Medicine (Semantic Rubric Scoring)

- **HealthBench (Hard/Easy):** Patient-doctor conversation scenarios requiring open-ended medical advice [Arora et al., 2025]. Correctness is determined via an external LLM judge using fine-grained rubric coverage scores rather than exact string matching.

Engineering Design (Geometric Solver)

- **DeepCAD & Fusion360:** Multimodal computer-aided design (CAD) tasks evaluated by computing the Intersection over Union (IoU) between generated and ground-truth 3D meshes using the CadQuery geometric kernel [Wu et al., 2021, Willis et al., 2021].

O.4 Evaluation Metrics and Experimental Setup

Metrics We use pass@1 and pass@N as defined in Eq. 13: the mean score of the N-candidate pool, which is the expected score of a uniformly random pick, and the maximum score in the pool, which is the oracle for that pool. Scores are *hard* (binary) for AIME, HMMT, GPQA-Diamond, LiveCodeBench, and the Kubernetes study, and *soft* (real-valued) for image-to-CAD (volumetric IoU between generated and reference solids) and HealthBench (judge rubric score).

For hard scores with c correct candidates, pass@1 $= c/N$ and pass@$N = \mathbb{1}[c > 0]$. These coincide with the unbiased pass@k estimator of Chen et al. [2021], $1 - \binom{N-c}{k}/\binom{N}{k}$, at $k = 1$ and $k = N$. For intermediate k (e.g., pass@8 from 32 samples in Table 34) we use that estimator directly. Soft scores have no pass/fail outcome, so we use the natural extension: the mean score, and the maximum score in the pool, whose expectation over pools is the expected best of N draws. Both reduce to the hard definitions for binary scores, since the maximum of 0/1 scores indicates whether any candidate is correct. Per-problem HealthBench scores can be negative when penalty criteria are triggered, so soft pass@1 can be negative for individual problems. In image-to-CAD, a program that fails to execute receives an IoU of 0 in every row of a table. Table 30 aggregates per-problem scores over problems by the median rather than the mean.

Selection methods are reported as M@1, the score of the single candidate chosen by method M (e.g., `iS`@1, SC@N). Table 17 (F.3) also uses top@k: 1 if any of the k candidates ranked highest by a method is correct. Thus top@1 is the method's own accuracy, top@N equals pass@N, and under a uniformly random ranking $\mathbb{E}$[top@k] equals pass@k.

Budgets and sub-pools When results are reported for budgets smaller than the generated pool (Tables 17, 18, 30), we draw sub-pools of size N without replacement from the full pool, compute every metric on each sub-pool, and aggregate over trials as stated in each caption. All methods in a row group use the same sub-pools.

Particle methods For `iPF` and `dPF`, resampling makes particles dependent. Here pass@1 and pass@N describe the final particle population (the expected score of a random final particle and the best particle). They do not estimate the base model's single-sample success rate.

Setup For `iS` we generate up to 128 parallel samples per problem (Table 46). For `iPF` (Table 47) and `dPF` (Table 48) we use up to 32 particles. We apply logit blending to short sequences and trigger KL-guided resampling for longer outputs (average length above ten 128-token steps). Baselines are Self-Certainty [Kang et al., 2025] for confidence-based selection, Self-Consistency [Wang et al., 2022] for output consensus, DeepConf [Fu et al., 2025], CISC [Taubenfeld et al., 2025], and RPC [Zhou et al., 2026] for confidence-weighted consensus, and an outcome reward model for extrinsic selection (Appendix H). Table 43 summarizes the configuration of each domain.

Table 40: Levels of Domain Verifiability.

Domain	Verification Mechanism	Benchmark Datasets
Mathematics	Symbolic Equivalence	AIME, HMMT
Reasoning	Exact Output Match	GPQA-Diamond
Coding	Test-Case Execution	LiveCodeBench-v6
Healthcare	Semantic Rubric Scoring	HealthBench-Hard
Engineering	Geometric Solver (IoU)	DeepCAD, Fusion360

Table 41: Detailed overview of the evaluation datasets. To minimize training data contamination for recently released models, we prioritize the latest available problem splits.

Dataset	Domain	Year / Split	# Problems	Prompting Setting
AIME	Mathematics	2024, 2025, 2026	30 + 30 + 30	Zero-shot, CoT
HMMT (Feb)	Mathematics	2025, 2026	30 + 33	Zero-shot, CoT
GPQA-Diamond	Sci/Reasoning	v1.0 (Test)	198	Zero-shot, CoT
LiveCodeBench	Coding	v6 (Recent)	100	Zero-shot
HealthBench-Hard	Healthcare	Hard Split	100	Zero-shot, Rubric-guided
DeepCAD	Engineering	Test Split	100	Image-to-CAD
Fusion360	Engineering	Out-of-distro	100	Image-to-CAD

Table 42: Empirical generation statistics across evaluation datasets. N indicates the sample budget for Phase 1 selection versus Phase 2 resampling algorithms.

Dataset	Problems	N (P1/P2)	Mean	Median	Std	Min	Max	Exposure
AIME-2024	30	128/32	5,739	5,876	3,184	137	16,384	High
AIME-2025	30	128/32	6,127	6,513	3,447	242	16,384	High
AIME-2026	30	128/32	5,402	5,204	3,029	428	16,384	Low
HMMT-2025	30	128/32	6,287	6,817	2,594	346	16,384	High
HMMT-2026	33	128/32	6,392	7,110	3,091	470	16,384	Low
GPQA-Diamond	198	128/-	429	397	322	197	16,384	High
LiveCodeBench	100	128/-	2,266	1,518	2,382	64	16,384	Low
HealthBench	100	128/32	1,252	1,249	510	24	16,384	Low
GenCAD-Code	100	128/-	612	503	340	223	4096	Low
Fusion360	100	128/-	736	644	385	222	4096	Low

Table 43: Summary of experimental setups, models, and hyperparameters across all evaluated domains.

Domain	Datasets	Verification	Primary Model(s)	Methods Evaluated	Key Experimental Settings
Mathematics	AIME (2024–2026) HMMT (2025–2026)	Symbolic	Qwen3-4B-Instruct-2507	`iS` ($N \leq 128$) `iPF` ($N = 16$, 120 steps) `dPF` ($N = 8$, guided)	Evaluated on all problems. Evaluated on hardest 25% (tail entropy). Hints generated by teacher model.
Reasoning	GPQA-Diamond	Output	Qwen3-4B-Instruct-2507 Gemma4-2B-IT	`iS` ($N \leq 128$)	Includes answer-stripping ablation.
Coding	LiveCodeBench-v6	Execution	Qwen3-4B-Instruct-2507 Gemma4-2B-IT	`iS` ($N \leq 128$)	Evaluated most recent (2026) problems to minimize training-data contamination.
Healthcare	HealthBench-Hard	Semantic	MedGemma-4B-IT	`iS` ($N = 32$) `iPF` ($N = 8$, max 30 steps) `dPF` ($N = 16$, max 50 steps)	100 problems across 7 clinical themes. Evaluation judge: MedGemma-27B. Guided entirely by rubric criteria.
Engineering	DeepCAD (In-distro) Fusion360 (Out-distro)	Solver	Qwen2.5-VL (fine-tuned)	`iS` ($N \leq 128$)	Multimodal image-to-CAD task; robust to absence of lightweight reward models.

Table 44: Datasets used for evaluation. LCB and HealthBench subsets are stored as local JSONL files curated from their respective benchmarks.

Short name	HuggingFace path	Config	Split	Domain
`aime-2024`	`Maxwell-Jia/AIME_2024`	—	train	Math
`aime-2025`	`MathArena/aime_2025`	—	train	Math
`aime-2026`	`MathArena/aime_2026`	—	train	Math
`hmmt-feb-2025`	`MathArena/hmmt_feb_2025`	—	train	Math
`hmmt-feb-2026`	`MathArena/hmmt_feb_2026`	—	train	Math
`gpqa-diamond`	`Idavidrein/gpqa`	`gpqa_diamond`	train	Multi-choice
`lcb-100`	`livecodebench/code_generation_lite`	—	—	Code
`healthbench-hard`	`openai/healthbench`	—	—	Healthcare

Table 45: Generation models and sampling hyperparameters. All models are served via vLLM.

HuggingFace path	temperature	top_p	top_k	max_tokens
`Qwen/Qwen3-4B-Instruct-2507`	0.7	0.8	20	16384
`google/gemma-4-E2B-it`	0.7	0.9	—	16384
`google/medgemma-1.5-4b-it`	1.0	0.95	64	16384
`Qwen/Qwen2.5-VL-7B-Instruct`	0.7	0.9	—	16384

Table 46: iS hyperparameters.

Parameter	Value
N samples	32-128
Tail aggregation	median
Tail window bounds	$[64, 2048]$ tokens
Trim outliers	5% each end
Max score tokens	2500

Table 47: iPF hyperparameters.

Parameter	Value
N particles	16
Step tokens	128
Max steps	120
ESS threshold (early)	0.5
ESS threshold (late)	0.7
Resampling	Systematic
Score	Entropy
Aggregation	mean (z-score)
Hard fraction	1.0
Warmup steps	1

Table 48: dPF hyperparameters.

Parameter	Value
N particles	8
Score	KL
Demo budget	16
Hint mode	demo/critique
Blending α	0.7

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide methodology and results for each contribution.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We report limitations in the Conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper focuses on novel algorithms and empirical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We report experimental details in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [N/A].

   Justification: Datasets are obtained from public benchmarks. We rely on open-source libraries for the algorithmic implementations.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We report details in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars and confidence intervals over 3/10 seeds for most of the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We use standard GPU hardware (H100).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conform to the code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our contribution is methodological.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our contribution is methodological.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: We use open-source models and public datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not introduce new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not run experiments with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not run experiments with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer:[N/A]

    Justification: We use LLMs for standard editing and formatting.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.