ReSET: Accurate Latency-Critical NVFP4 Reasoning via Step-Aware Temperature Scaling

Sihwa Lee[1*] **Janghwan Lee**[1*] **Donghoon Yoo**[2] **Jae Gon Kim**[2]
Hanyul Ryu[2] **Soojung Ryu**[2] **Jungwook Choi**[1†]
[1]Hanyang University, Seoul, Republic of Korea
[2]Xenoscube Korean Inc., Seoul, Republic of Korea
{macto94, hwanii0288}@hanyang.ac.kr
{donghoon.yoo, jay.kim, hanyul.ryu, sue.ryu}@xenoscube.ai
choij@hanyang.ac.kr

Abstract

Large reasoning models (LRMs) improve complex problem-solving by generating long intermediate reasoning traces, but this substantially increases inference costs. NVFP4 inference offers a promising approach to reduce both computational and memory costs through hardware-supported low-precision execution. However, directly applying NVFP4 to LRMs introduces two practical limitations: reasoning accuracy degrades under quantization, and existing NVFP4 kernels do not fully realize latency benefits in small-batch autoregressive decoding. In this work, we analyze the effect of NVFP4 quantization on token-level uncertainty during reasoning. We show that quantization increases incorrect sampling at low-entropy symbolic tokens, while causing over-concentration on a small set of tokens in high-uncertainty reasoning steps. Based on this observation, we propose **ReSET**, a reasoning-step entropy-based temperature-scaling method that estimates step-level uncertainty online and adapts the decoding temperature using both token-level and step-level entropy signals. To address the latency gap, we further design a CUDA-core small-M NVFP4 kernel for latency-critical autoregressive decoding. ReSET consistently improves NVFP4 reasoning accuracy across benchmarks and model scales, by 2.6 points on average on AIME-120. Our CUDA-core small-M kernel further improves latency-critical decoding, delivering up to $2.5\times$ kernel-level speedup over NVFP4 vLLM and approximately $2\times$ end-to-end decoding speedup over BF16. Code is available at https://github.com/aiha-lab/ReSET.

1 Introduction

Large reasoning models (LRMs) have emerged as a powerful paradigm for solving complex multi-step mathematical and logical problems by generating extended chain-of-thought (CoT) traces [1–4]. Their accuracy gains rely on *inference-time scaling* — allocating additional compute during sequential decoding to handle harder instances — which makes the dominant cost dimension *per-token decode latency*, not aggregate throughput. This cost grows along two compounding dimensions per generated token — (i) memory bandwidth for repeated weight loading during autoregressive decoding [5] and (ii) aggregate floating-point operations [6] — and is amplified by reasoning traces that routinely exceed 10K tokens under inference-time scaling [7].

Of the levers available to compress these costs, low-precision execution directly targets these two axes at the hardware level: each low-precision element reduces both the bytes loaded per weight access and the per-element compute cost in proportion to the bit width. NVFP4 [8], the most recent low-precision

*Equal contribution. †Corresponding author.

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

format with native Blackwell Tensor-Core support, delivers $\sim 4\times$ higher peak throughput than BF16 and a $\sim 4\times$ smaller weight footprint, making it an attractive lever for low-latency LRM serving.

Realizing this benefit on LRMs, however, is non-trivial: two obstacles stand between NVFP4's headline numbers and a deployable system.

Accuracy. Reasoning involves intermediate symbolic tokens — digits, operators, structural markers — where small perturbations in token probabilities propagate through subsequent steps and break the final answer [9–11]. Quantization-aware training mitigates this but is prohibitively expensive [12–14]; existing NVFP4 PTQ methods [15–18] target weight quantization error, which is already small at NVFP4's group size [19, 18] and does not correlate cleanly with reasoning accuracy.

Latency. The $4\times$ peak throughput above is delivered by Blackwell's block-scaled `tcgen05.mma` instruction (`.kind::mxf4nvf4`), whose smallest supported tile is $M{=}128$ along the token dimension — a structural requirement of block-scaled Tensor-Core execution. Production reasoning serving operates an order of magnitude below this: TPOT SLOs, KV-cache pressure, prefill–decode interference, and generation stalls [20–22] cap the SLO-feasible decode batch at $M \leq 8$ on B200 (Sec. 2.3). The result is below-6.25% tile occupancy and below-1% measured Tensor-Core utilization in the regime that governs production — the headline $4\times$ advantage *collapses* where it matters most.

We address both obstacles with a unified, deployment-oriented design that pairs a near-zero-cost decoding-time control policy with a kernel matched to the actual decode shape. **ReSET** reads quantization-induced sampling errors at *step-level* granularity — the granularity at which reasoning-model uncertainty actually fluctuates — and adapts decoding temperature accordingly, recovering $+2.6$ accuracy points on average on AIME-120 at $\sim 1.5\%$ per-decode-step overhead. On the system side, we engineer a **CUDA-core NVFP4 GEMV kernel** for the small-M decoding regime that Tensor-Core paths cannot serve, achieving 1.57–$2.49\times$ projection latency reduction over vLLM-CUTLASS at $M{=}1$–8 on B200 and up to $1.97\times$ end-to-end latency reduction over BF16 on Qwen3-32B. We are not aware of a prior public CUDA-core NVFP4 GEMV implementation; code is available at `https://github.com/aiha-lab/ReSET`.

We make the following contributions:

- We characterize NVFP4 reasoning failures at *step-level* granularity (Sec. 3), showing that token-level entropy — the signal used by prior entropy-aware decoding — is dominated by the uncertainty of the surrounding reasoning step and is therefore unreliable as a fixed-threshold control signal. We propose **ReSET** (Sec. 4), a step-aware temperature scaling policy whose per-token overhead is a constant-time scalar update.
- We design a CUDA-core small-M NVFP4 GEMV kernel (Sec. 5) tailored to the latency-critical decoding regime where Tensor-Core paths are tile-bound, with no prior public implementation we are aware of.
- We demonstrate $+2.6$ accuracy points on average on AIME-120, 1.57–$2.49\times$ kernel-level speedup over vLLM-CUTLASS at $M{=}1$–8, and up to $1.97\times$ E2E speedup over BF16 on B200 across five reasoning-capable open models (Sec. 6).

2 Background

NVFP4 deployment for LRMs is bounded along two axes: reasoning accuracy under quantization (Sec. 2.1–2.2) and decode-phase latency at small M (Sec. 2.3). The first axis motivates the entropy-based decoding analysis of Sec. 3 and the policy of Sec. 4; the second axis motivates the kernel design of Sec. 5.

2.1 NVFP4 Reasoning Models

NVFP4 is NVIDIA's microscaled FP4 format [23, 24], combining FP4 values with fine-grained shared scaling (group size 16, detailed configs are in Table 8). Recent hardware such as NVIDIA B200 [8] natively supports NVFP4 Tensor Cores, achieving up to 9 PFLOPs and $\sim 4\times$ higher throughput than BF16. NVIDIA's ModelOPT [25] framework further enables practical deployment of NVFP4 reasoning models [26, 27]. However, applying FP4 quantization to LRMs remains challenging because quantization errors can accumulate over long reasoning trajectories [11]. Prior work addresses

this through either training-based approaches, such as quantization-aware training and distillation [12–14], or post-training quantization (PTQ) methods that reduce quantization error through techniques such as rotation, channel-wise scaling, and dynamic block-scale selection [15, 17, 18, 16]. However, PTQ methods can introduce additional runtime overhead due to online rotation for activations, and their gains often diminish for fine-grained formats such as NVFP4 where quantization error is already relatively small [18, 24]. This suggests that improving NVFP4 reasoning quality may require going beyond quantization-error minimization and considering the decoding process itself.

2.2 Token Entropy and Temperature under Quantized Reasoning Decoding

Token entropy in reasoning. Token-level uncertainty in LRMs is heterogeneous and is quantified by token entropy: $H_t = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v)$, where $p_t(v)$ denotes the probability of token v at step t [9]. Quantization perturbs this distribution [28, 29]: the resulting entropy shifts can encourage exploration during RL [29] but distort sampling at inference [28]. *Where* these shifts land — and how they should be controlled — is the question Sec. 3 takes up.

Decoding temperature in LRMs. Decoding temperature controls the sharpness of the token distribution through $\mathrm{softmax}(z_t/T)$. LRMs typically use non-zero temperatures (e.g., 0.6) because greedy decoding degrades reasoning accuracy [3, 2]. While temperature scaling has been studied for test-time reasoning diversity [30], its interaction with quantized inference remains underexplored.

2.3 Latency-Critical Large Reasoning Model Serving

The throughput–latency gap. NVFP4's headline 4× peak throughput over BF16 [8] is realized at compute-bound batch sizes, but production LRM serving operates an order of magnitude below that point. LRM workloads produce long outputs — R1 [2] generates on average ∼12K tokens per AIME [31] problem, up to 64K [2, 1] — so per-token decode latency, not aggregate throughput, sets the SLO [20, 21].

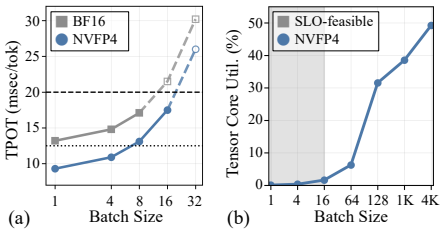

Figure 1: Limited batch-size scaling in Qwen3-32B on a single B200. (a) TPOT at 32K context; (b) Tensor Core utilization of NVFP4 decode GEMMs.

The small-M utilization collapse. Fig. 1(a) reports TPOT as a function of decode batch size M for Qwen3-32B at 32K context on a single B200; representative SLOs [32] are crossed at moderate batch sizes, confining feasible operation to $M \leq 8$. *In this region, NVFP4 Tensor-Core utilization stays below 1%* (Fig. 1(b)) — a near-total collapse of the peak-throughput advantage NVFP4 was deployed for. Prior FP4 quantization studies operate at the throughput-oriented batch sizes where this collapse is hidden; the latency-critical SLO-feasible regime, the focus of this paper, sits where NVFP4's headline benefit goes missing. Sec. 5 explains the mechanism — fixed Tensor-Core tile granularity — and engineers the matching kernel.

3 Observations

We probe NVFP4 reasoning failures along the granularity at which existing decoding methods reason about uncertainty: *token-level entropy* [9]. Under W4A4, mis-sampling concentrates at low token-level-entropy symbolic tokens (Sec. 3.1) and a token-level entropy threshold partially recovers accuracy (Sec. 3.2), but whether a token *appears* low- or high-entropy is dominated by the uncertainty of the surrounding reasoning step (Sec. 3.3). We therefore develop a *step-level entropy* analysis and use it as the primary control signal for quantized reasoning.

3.1 Low-Entropy Tokens as Quantization-Sensitive Decisions

We categorize 1.5M tokens generated from the R1-Qwen-14B [2] model by their entropy, following the entropy-based token categorization of [9] (detailed analysis setting is in Appendix B.1). As shown in Fig. 2(a-b), low-entropy tokens correspond to locally constrained *symbolic decisions* such as digits and operators, while high-entropy tokens appear at semantically flexible *branching points*. As described in Fig. 2(c-d), under NVFP4 quantization, errors at symbolic tokens (e.g., "four") can

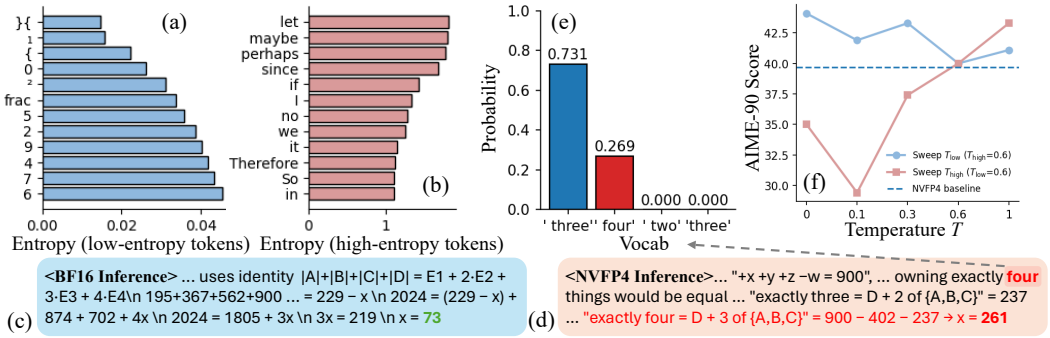

Figure 2: Example of (a) low- and (b) high-entropy tokens. Example reasoning under (c) BF16 and (d) NVFP4 on R1-Qwen-7B. (e) Next-token probability distribution at the symbolic position in (d). (f) Effect of entropy-conditioned temperature control accuracy (R1-Qwen-7B, AIME 2022-2024).

propagate through subsequent reasoning and lead to incorrect final outcomes. Notably, these errors are not driven by top-1 flips. As illustrated in Fig. 2(e), the correct token ("three") remains the most likely, but a non-top-1 alternative receives enough probability mass to be occasionally sampled, breaking the symbolic continuation. This pattern is systematic: over 46,375 low-entropy positions on Qwen3-32B AIME-120 matched prefixes, the BF16 top-1 token stays top-1 under NVFP4 in 99.91% of cases while the non-top-1 sampling probability rises by $1.54\times$ (Appendix B.3): quantization rarely reorders the distribution but leaks mass to alternatives, which is precisely what temperature corrects.

3.2 A Naive Fix: Entropy-Thresholded Temperature Sharpening

This suggests a direct intervention: sharpening the distribution at low-entropy positions to suppress mis-sampling. We test it with a fixed-threshold rule that, at each decoding position, computes the token entropy H_t, partitions tokens by an entropy threshold τ_0, and applies a different temperature to each group:

$$T_t = \begin{cases} T_{\text{low}}, & H_t < \tau_0, \\ T_{\text{high}}, & H_t \geq \tau_0. \end{cases} \tag{1}$$

Here, τ_0 is set to approximately 0.6, corresponding to the top 20% of token entropies observed in model responses. As shown in Fig. 2(f), this asymmetric sharpening improves NVFP4 reasoning accuracy: lowering the temperature T_{low} for low-entropy tokens yields consistent gains, while changes to the high-entropy temperature provide little benefit beyond moderate increases in T_{high}. Repeating this sweep over eight seeds with 95% confidence intervals, and under BF16, confirms the asymmetry (Table 9 in Appendix B.2): sharpening T_{low} helps only under NVFP4, whereas lowering T_{high} hurts at both precisions — low-entropy sharpening corrects an NVFP4-specific error, while preserving high-entropy exploration is a general property of reasoning decoding.

3.3 Token Entropy is Dominated by Step-Level Uncertainty

The result in Sec. 3.2 is positive but not yet operational: it shows that *some* notion of "low-entropy token" is the right intervention target, but the fixed-threshold rule's incomplete recovery indicates that token-level entropy is not the right signal. This limitation stems from a granularity mismatch: prior entropy-aware decoding methods reason about uncertainty at the *token* level, but the uncertainty that actually fluctuates during reasoning lives at the *step* level. We find that token-level entropy is not determined solely by the local token identity but is strongly influenced by the uncertainty of the surrounding reasoning step. Even typically deterministic tokens (e.g., digits) can exhibit elevated entropy when they appear in uncertain reasoning steps, which can in turn increase the likelihood of incorrect symbolic decisions. To analyze this effect, we group reasoning traces into step-level segments and examine how the low- and high-entropy token groups of Sec. 3.1 (Fig. 2(a-b)) behave within each step.

Definition of a reasoning step. We define a *reasoning step* as a contiguous segment of generated tokens delimited by a double newline (`\n\n`), the paragraph-level boundary that LRMs emit between intermediate reasoning units [10, 33]. Let $\{1, \ldots, T\}$ denote the token positions in a sequence,

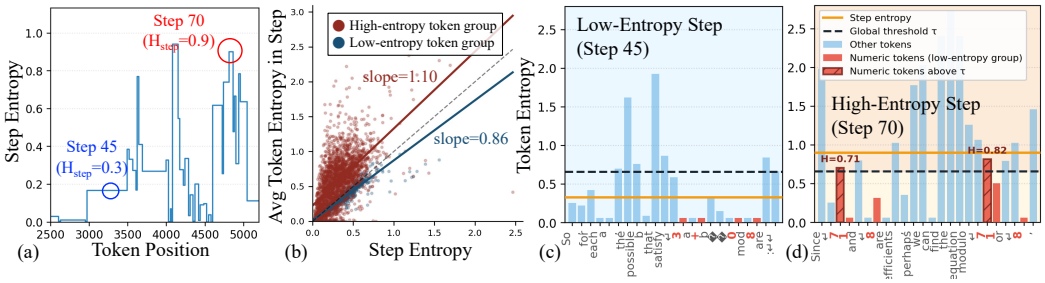

Figure 3: Step-level entropy dynamics. (a) Step-wise entropy trajectory. Relationship between step entropy and token entropy for (b) low- and high-entropy groups. Token entropy distributions in representative (c) low- and (d) high-entropy steps.

partitioned into segments $\{\mathcal{S}_k\}_{k=1}^{K}$. For a token position $t \in \mathcal{S}_k$, we define the *step entropy* as the average token entropy within the current reasoning step: $H_{\text{step}}(t) = \frac{1}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} H_i$.

Observations. Fig. 3(a) shows that step entropy varies substantially over the course of a single reasoning trace. For example, Step 45 has low step entropy ($H_{\text{step}} = 0.3$), while Step 70 has high step entropy ($H_{\text{step}} = 0.9$), indicating that the model alternates between confident and uncertain reasoning states during generation — variation that token-level entropy alone does not capture. Fig. 3(b) makes the token–step coupling concrete: the mean token-level entropy of both the low- and high-entropy token groups closely tracks the step entropy of the segment they fall in. In other words, the same token category occupies different absolute positions on the token-level entropy axis depending on the surrounding step's uncertainty, and even deterministic symbolic tokens can exhibit elevated token-level entropy inside an uncertain reasoning step.

The practical consequence is illustrated in Fig. 3(c-d). In low-entropy steps (Fig. 3(c)), symbolic tokens fall below the global threshold τ_0, and a fixed-threshold rule correctly identifies them as positions to sharpen. However, in high-entropy steps (Fig. 3(d)), symbolic tokens can exceed the same threshold due to elevated step-level uncertainty. These tokens remain locally deterministic, but a token-level rule misclassifies them as high-entropy positions, causing the decoding rule to skip necessary sharpening. The diagnosis is therefore not that the symbolic-token signal is wrong, but that it must be read *relative to the current step* — motivating a shift from token-level to step-level entropy analysis. We operationalize this step-relative formulation, which combines step-level uncertainty with token-level deviation within a step, in Sec. 4.

4 ReSET: Reasoning Step Entropy-based Temperature Scaling

We realize the token-to-step shift of Sec. 3 as **ReSET**, which replaces the fixed token-level threshold of Eq. 1 with one that adapts to step-level uncertainty. ReSET has two components: a *step-aware threshold* that is global in confident steps and step-relative in uncertain ones, and an *online step-entropy estimator* that supplies the per-token statistic this threshold needs under the causality of autoregressive decoding.

Step-aware threshold (SAT). The fixed-threshold rule of Sec. 3.2 fails in exactly one regime: high-uncertainty steps, where step-level entropy lifts symbolic tokens above the global cutoff τ_0 (Fig. 3(d)). We target this regime by comparing the online step-entropy estimate $\hat{H}_{\text{step}}(t)$ (defined below) against the running global mean $\bar{H} = \frac{1}{t+1} \sum_{i=0}^{t} H_i$:

$$\tau_t = \begin{cases} \tau_0, & \hat{H}_{\text{step}}(t) \le \bar{H} \quad \text{(confident step)}, \\ \hat{H}_{\text{step}}(t), & \hat{H}_{\text{step}}(t) > \bar{H} \quad \text{(uncertain step)}, \end{cases} \tag{2}$$

where τ_0 is the global threshold of Sec. 3.2; substituting Eq. 2 into Eq. 1 gives the full ReSET policy.

The novelty lies in the uncertain branch: setting $\tau_t = \hat{H}_{\text{step}}(t)$ makes the rule *step-relative* — a token is sharpened exactly when its token-level entropy falls below that of the typical token in its own step, regardless of where the global cutoff sits. This recovers the symbolic-vs-branching distinction inside a step that Sec. 3.3 showed token-level rules collapse: in Fig. 4(a), the symbol "7" crosses τ_0 but stays below $\hat{H}_{\text{step}}(t)$, and SAT correctly sharpens it. The choice introduces no additional sweepable

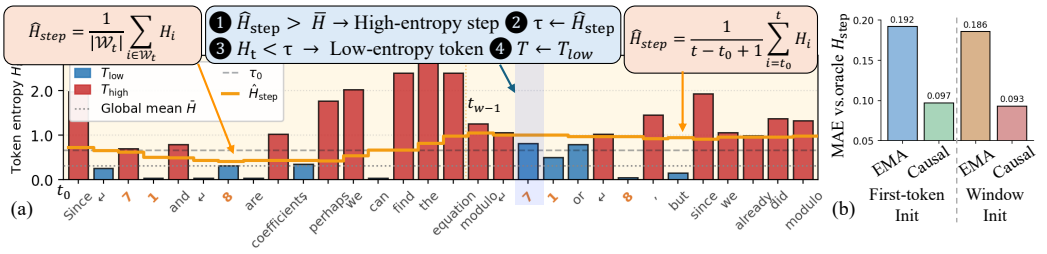

Figure 4: (a) ReSET temperature assignment on an R1-Qwen-14B AIME-120 trace. (b) MAE of online step-entropy estimators against the offline oracle of Sec. 3.3: two step-transition initializations (first-token: restart from the step's first token; window: sliding window over the first w positions) $\times$ two within-step aggregations (EMA vs. causal running average). Window + causal is HSE (Eq. 3).

hyperparameter; τ_t is fully determined by quantities already tracked. In confident steps $\hat{H}_{\text{step}}(t)$ becomes degenerately small, and a step-relative cutoff would misfire on symbolic tokens (Fig. 7); we therefore fall back to τ_0, which Sec. 3.2 validated for this regime.

Online step-entropy estimation. SAT needs $\hat{H}_{\text{step}}(t)$ at every decoding position, but the offline definition of Sec. 3.3 averages over not-yet-generated tokens. An online estimator must therefore handle two within-step regimes with opposite bias–variance trade-offs.

Late regime: within-step aggregation. Once enough samples within the step are available, the lowest-bias estimate is the causal running average over the current step, $\hat{H}_{\text{causal}}(t) = \frac{1}{t-t_0+1}\sum_{i=t_0}^{t} H_i$, where t_0 is the step's start (boundaries are detected online from newline tokens at negligible cost; Appendix B.10). As a baseline, we use an exponential moving average (EMA; Appendix Eq. 4), which weights recent tokens but ignores the step boundary. Fig. 4(b) shows the causal average wins under both initializations: resetting at the step boundary matters more than emphasizing recent tokens.

Early regime: step-transition initialization. In the first w positions of a step, the within-step average is high-variance: few samples, and early samples are often atypical (step-opening discourse tokens). A sliding-window initialization over the w most recent tokens (spanning the boundary) trades a small step-boundary bias for lower variance until the step is self-supporting (Fig. 4(b), window vs. first-token init; the gain is small in aggregate because it affects only the first w positions of each step).

The hybrid estimator HSE switches between the two regimes:

$$\hat{H}_{\text{step}}(t) = \begin{cases} \frac{1}{|\mathcal{W}_t|}\sum_{i\in\mathcal{W}_t} H_i, & t - t_0 < w \quad \text{(initialization)}, \\ \frac{1}{t-t_0+1}\sum_{i=t_0}^{t} H_i, & t - t_0 \geq w \quad \text{(within-step)}, \end{cases} \tag{3}$$

with $\mathcal{W}_t$ the size-w window ending at t. HSE adds only lightweight per-token bookkeeping—one entropy reduction, two scalar updates ($\hat{H}_{\text{step}}$, $\bar{H}$), and one branch—which is negligible compared with a full decode step and is included in the end-to-end measurements in Sec. 6.3.

Decoding temperatures and calibration. We use $T_{\text{low}} < T_{\text{base}} < T_{\text{high}}$ around the model's recommended base (T_{base}=0.6). T_{high} is method-relevant: NVFP4 *compresses* the entropy of high-entropy tokens in uncertain steps (e.g., "Alternatively"; Fig. 8), so a temperature above the default is needed to restore the diversity quantization removes; empirically, T_{high}=1.0 consistently improves accuracy (Table 16). T_{low} is selected per (model, task) on a held-out calibration split disjoint from evaluation (Table 19); τ_0 is the 80th-percentile token entropy on the same split. We use w=32 (Table 15).

5 NVFP4 Kernel Design for Small-M Decoding

ReSET recovers reasoning quality, but realizing NVFP4's latency benefit hinges on a decode-phase kernel matched to the small-M regime motivated in Sec. 2.3. We therefore propose a CUDA-core NVFP4 kernel for the linear projection $Y = XW$ ($X \in \mathbb{R}^{M\times K}$, $W \in \mathbb{R}^{K\times N}$) at small M, and dispatch to a Tensor-Core path for prefill and large-M decode (Appendix C.3).

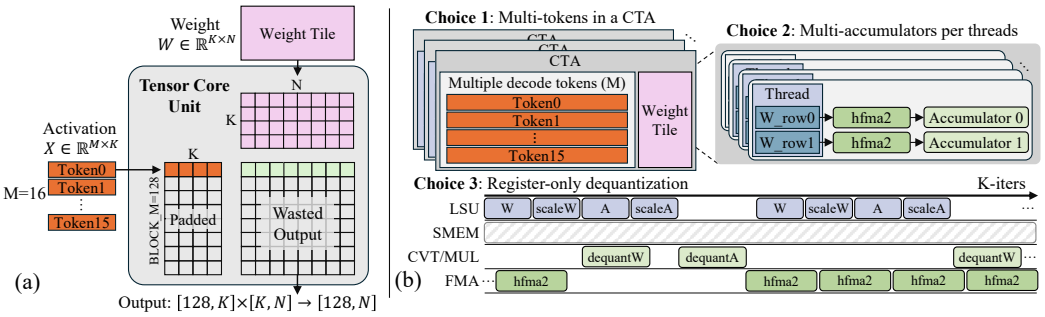

Figure 5: CUDA-core NVFP4 kernel design for small-M decoding. (a) Tensor-Core GEMM underutilization from the required 128-row M tile. (b) CUDA-core design choices.

5.1 Tensor-Core Inefficiency at Small M

Tile under-occupancy explains the Sec. 2.3 collapse. The sub-1% utilization of Sec. 2.3 traces to a fixed-tile constraint of Blackwell's NVFP4 GEMM path. Existing NVFP4 implementations (vLLM, CUTLASS, and MR-GPTQ's kernels [18]) dispatch linear layers to the block-scaled `tcgen05.mma` path (`.kind::mxf4nvf4`), whose M tile is fixed at 128 along the token dimension (Fig. 5(a)), so the logical $M \times K$ activation is padded into $128 \times K$ and only M of 128 output rows are useful: 6.25% tile occupancy at $M{=}8$, 3.13% at $M{=}4$. Throughput-oriented stacks hide this with batch sizes in the thousands; in the SLO-feasible region, the advantage is lost.

Missing NVFP4 CUDA-core kernel. A CUDA-core path is well matched to small-M decoding because it exposes M flexibly at the thread level rather than through a fixed tile. However, current NVFP4 inference frameworks continue to rely on Tensor-Core GEMM paths for decode projections, leaving this CUDA-core small-M regime unsupported. We therefore design, to our knowledge, the first CUDA-core NVFP4 GEMV kernel tailored to latency-critical small-M decoding, with support for NVFP4's E4M3 block scales and FP4 unpacking pipeline.

Challenges. A CUDA-core NVFP4 path must replace work that Tensor-Core MMA hides implicitly: (C1) reusing streamed weight tiles across active decode tokens, since every token shares W; (C2) launching enough parallel threads with independent accumulator chains, since a naive mapping serializes `half2` FMAs along K and under-occupies CUDA cores at small M; and (C3) performing FP4 unpacking and shared-scale dequantization without materializing intermediate buffers or inserting synchronization in the inner K loop.

5.2 CUDA-Core Decode Kernel

Our CUDA-core kernel addresses C1–C3 through the three design choices in Fig. 5(b).

Multi-token CTA fusion (C1). We group multiple active decode tokens within a single cooperative thread array (CTA), i.e., a CUDA thread block, so each HBM-streamed weight tile is loaded once and reused across token-wise dot products without inflating M to the 128-row Tensor-Core tile.

Multi-accumulator threading (C2). We assign multiple output rows per thread, creating independent accumulator chains that the warp scheduler interleaves to expose instruction-level parallelism even when M is small.

Register-only dequantization (C3). We hold unpacked FP16 values exclusively in registers and feed them directly into `half2` FMAs, eliminating shared-memory materialization and the synchronization barriers it would require. The resulting pipeline (Fig. 5(b)) overlaps load, conversion, scale application, and accumulation across successive K tiles.

Together, these choices avoid the small-M inefficiency of Tensor-Core GEMM by matching the actual decode shape. Although the CUDA-core path dequantizes NVFP4 operands and performs half-precision FMA rather than native FP4 arithmetic, this decode-specialized implementation achieves up to $2.5\times$ lower projection latency than the NVFP4 Tensor-Core baseline (Table 4). Implementation details and ablations are provided in Appendix C.

Table 1: Comparison with NVFP4 PTQ methods across reasoning benchmarks.

Task	Method	R1-Qwen 7B	R1-Qwen 14B	Qwen3 8B	Qwen3 14B	Qwen3 32B	Avg
AIME-120	BF16 Baseline	45.7	57.4	70.4	76.1	75.8	65.1
	RTN	39.6	52.4	62.5	70.4	74.4	59.9
	BRQ	41.4	49.8	53.8	66.9	73.0	57.0
	4/6	41.1	53.1	64.0	70.1	74.8	60.6
	MR-GPTQ	39.6	50.6	65.2	71.0	73.3	60.0
	ReSET	43.8	54.0	64.9	72.1	77.5	**62.5**
GPQA-Diamond	BF16 Baseline	48.3	58.4	54.4	61.3	64.1	57.3
	RTN	47.1	53.7	50.7	57.4	62.8	54.3
	BRQ	45.5	56.7	53.5	58.2	60.8	55.0
	4/6	44.6	55.4	51.9	57.6	64.6	54.8
	MR-GPTQ	43.4	57.4	52.4	60.9	60.6	54.9
	ReSET	46.0	57.6	53.2	58.5	62.9	**55.6**
LiveCodeBench	BF16 Baseline	28.2	38.5	43.2	46.5	49.9	41.3
	RTN	29.5	37.1	36.4	46.7	46.5	39.2
	BRQ	27.6	37.7	36.2	39.0	45.1	37.1
	4/6	27.0	36.4	35.8	45.3	45.7	38.0
	MR-GPTQ	27.2	34.9	42.9	47.2	45.1	39.5
	ReSET	28.4	37.9	42.1	46.1	46.7	**40.2**

Table 2: Impact of temperature scaling methods.

AIME-120 (↑)	R1-Qwen		Qwen3			Avg
	7B	14B	8B	14B	32B	
BF16	45.7	57.4	70.4	76.1	75.8	65.1
NVFP4	39.6	52.4	62.5	70.4	74.4	59.9
Fixed Threshold	41.8	53.2	63.3	70.3	75.8	60.9
Sliding Window	42.6	51.3	63.3	70.6	76.4	60.8
ReSET	**43.8**	**54.0**	**64.9**	**72.1**	**77.5**	**62.5**

Table 3: Per-projection pipeline latency at $M{=}1$ on Qwen3-32B (μs, ↓). (Q: Quantize; R: Rotate.)

Layer	vLLM		MR-GPTQ		**Ours**	
Type	Q	GEMM	Q+R	GEMM	Q	GEMV
QKV	1.81	13.41	4.65	11.84	**1.65**	**6.62**
Out	2.04	12.40	4.87	11.90	**1.81**	**7.73**
Gate-Up	1.83	25.89	4.55	25.90	**1.67**	**13.25**
Down	2.80	33.26	5.08	33.87	**1.79**	**15.00**

Table 4: Kernel latency at $M{=}1$ across Qwen3 projections (μs, ↓). Rows follow QKV, Out, Gate/Up, and Down.

Qwen3	N	K	vLLM	**Ours**	Speedup
8B	6144	4096	8.69	4.28	2.03×
	4096	4096	7.69	3.75	2.05×
	12288	4096	11.45	6.40	1.79×
	4096	12288	16.61	6.66	2.49×
14B	7168	5120	10.79	5.53	1.95×
	5120	5120	9.08	4.64	1.96×
	17408	5120	14.93	9.49	1.57×
	5120	17408	23.77	10.35	2.30×
32B	10240	5120	13.41	6.62	2.03×
	5120	8192	12.40	7.73	1.60×
	25600	5120	25.89	13.25	1.96×
	5120	25600	33.26	15.00	2.22×

6 Experiments

We evaluate the two axes of Sec. 2.1–2.3 in turn: Sec. 6.1–6.2 measure reasoning accuracy against PTQ baselines, Sec. 6.3 measures decode-phase latency from kernel up to end-to-end, and Sec. 6.4 ablates the policy components.

6.1 Experimental Settings

We evaluate ReSET on five LRMs from two families: R1-Distill-Qwen-7B/14B [2] and Qwen3-8B/14B/32B [3]. All models are quantized from public BF16 checkpoints to *real* NVFP4 W4A4 with NVIDIA ModelOpt (E2M1 elements, E4M3 block scales, group size 16); the KV cache remains in BF16. All baselines share this weight format and KV path on identical hardware, ensuring a like-for-like comparison. We report averages over 8 seeds on AIME-120 (AIME 2022–2025 combined) [31], GPQA-Diamond [34], and LiveCodeBench [35], using top-$p = 0.95$ and `max_tokens`=32k. Baselines (RTN, BRQ [17], 4/6 [16], MR-GPTQ [18]) decode at the model-default T=0.6; calibration protocols are in Appendix B.6. ReSET is applied on top of RTN.

6.2 Results on Reasoning Benchmarks

ReSET dominates the NVFP4 PTQ frontier. ReSET attains the best average accuracy on every benchmark in Table 1, surpassing all PTQ baselines, with the largest gain on AIME-120 (+2.6 over the NVFP4 baseline). The improvement traces to the step-aware threshold of Sec. 4: the global threshold τ_0 continues to identify symbolic tokens in low-uncertainty steps, while the step-relative threshold $\hat{H}_{\text{step}}(t)$ rescues symbolic tokens whose absolute entropy is elevated by surrounding context. The task-wise pattern mirrors how strongly each task exhibits the failure mode of Sec. 3.1: at low-entropy

Table 5: Five-model mean AIME-120 accuracy: fixed temperatures vs. ReSET. "Best fixed" picks the best $T \in \{0.1, 0.6, 0.8, 1.0\}$ per model (per-model results in Table 20).

Precision	Default (T=0.6)	Best fixed T	**ReSET**	Δ vs. default	Δ vs. best fixed
NVFP4	59.9	60.4	**62.5**	$+2.6$	$+2.0$
BF16	65.1	66.3	65.3	$+0.2$	-1.0

positions, NVFP4 inflates the non-top-1 sampling probability by $1.54\times$ on AIME-120 but only $1.08\times$ and $1.10\times$ on GPQA-Diamond and LiveCodeBench (Table 10), so AIME has the most to recover.

Step-aware control beats token-level heuristics. Table 2 compares different entropy-aware temperature policies on AIME-120 under the same decoding setup, where the primary difference lies in how entropy thresholds are determined. Fixed-threshold and sliding-window baselines improve over NVFP4 inconsistently across models, whereas ReSET is best on every model, confirming that step-relative entropy is the operative signal.

The gain is quantization-specific, not generic temperature tuning. Table 5 compares ReSET against fixed-temperature decoding at both precisions on AIME-120. Under NVFP4, ReSET improves over the default temperature by $+2.6$ points on average ($+1.6-4.2$ per model)and over the best fixed temperature selected separately for each model by $+2.0$. Applying the identical policy to BF16 changes accuracy by only $+0.2$ on average, so ReSET specifically mitigates NVFP4-induced sampling distortion rather than acting as a better global temperature (full sweep in Appendix B.9).

ReSET transfers beyond Qwen and beyond English, and not by reasoning longer. With the same `\n\n` boundary and no architecture-specific change, ReSET improves NVFP4 AIME-120 accuracy by $+5.3$, $+4.2$, and $+0.6$ points on R1-Llama-8B, Magistral-Small-24B, and Gemma-4-31B, and by $+5.2$ points on average on Japanese, French, and Spanish MT-AIME-2024 (Appendix B.11). The gain does not come from longer reasoning: ReSET leaves output length unchanged on Qwen3-32B (mean $+0.2\%$, median -0.7%), and when the same generations are truncated to matched token budgets the gain persists ($+2.7$ points at 24k, $+3.1$ at 32k; Appendix B.12).

6.3 System Evaluation

Kernel-level latency. Table 4 compares our CUDA-core GEMV against vLLM's CUTLASS GEMM at decode batch M=1 on B200, the regime Sec. 5 targets. Across 12 projection shapes (QKV/O/Gate-Up/Down for Qwen3-8B/14B/32B), our kernel is **1.57–2.49**$\times$ faster, with the largest gain on the Qwen3-8B down-projection (N=4096, K=12288). Larger-M and prefill Tensor-Core results, with per-choice ablations, are in Appendix C.

Quantization overhead is reduced, not added. A common concern with PTQ-time interventions is that they impose a computational burden on the decoding step. Rotation-fused activation quantization (Table 3) costs 4.55–5.08 μs per layer at M=1 on Qwen3-32B, averaging about $2.3\times$ the standard activation-quantization cost (1.81–2.80 μs). Our pipeline further reduces this overhead by consuming a non-swizzled scale layout for the CUDA-core kernel, keeping activation quantization at 1.65–1.81 μs. ReSET is a sampling-side intervention with no rotation, so the system inherits this lower pre-GEMM cost by construction.

End-to-end (E2E) latency reduction. Fig. 6 reports E2E speedup over BF16 on B200 (sm_100a, vLLM + FlashInfer/CUTLASS, CUDA 13.1, PyTorch 2.10.0; details in Appendix B) at `input_len`=512, $B \in \{1, 8\}$, and outputs up to 32k tokens. Combining the NVFP4 kernels (Sec. 5) with the ReSET sampler (Sec. 4), **Ours** reaches **1.69**$\times$ over BF16 and **1.22**$\times$ over the strongest NVFP4 baseline on Qwen3-8B at B=1, output 8k; on Qwen3-32B, **1.97**$\times$ (B=1) and **1.85**$\times$ (B=8). The ReSET sampler adds $\sim$100 μs per generated token on Qwen3-32B—$\sim$1.5% of a decode step—so the reported speedups already include the full sampling cost.

6.4 Ablation Studies

We ablate the two policy choices below; sweeps for w, T_{high}, T_{low}, applying ReSET on top of GPTQ and Wikitext-2 perplexity across PTQ methods are in Appendix B.8.

Truncation-based sampling cannot substitute for step-aware control. With $T = 0.6$ fixed, sweeping top-p and min-p thresholds (Table 6) yields limited or negative gains under NVFP4: top-p

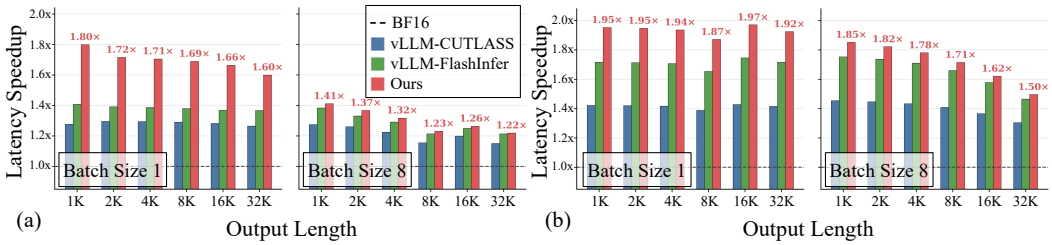

Figure 6: E2E speedup over BF16 for (a) Qwen3-8B and (b) Qwen3-32B with 512-token inputs.

Table 6: Comparison with truncation-based sampling methods on AIME-120.

Method	T	Top-p	Min-p	R1-Qwen		Qwen3			Avg
				7B	14B	8B	14B	32B	
NVFP4 Baseline	0.6	0.95	0.0	39.6	52.4	62.5	70.4	74.4	59.9
NVFP4 + Top-p Sweep	0.6	0.80	0.0	41.0	49.5	61.0	70.1	74.3	59.2
		0.85		39.7	50.6	63.3	70.2	74.5	59.7
		0.90		40.2	52.4	63.0	71.2	74.5	60.3
		0.99		38.4	53.9	63.6	69.6	75.2	60.1
		1.00		40.1	52.6	63.9	70.7	73.6	60.2
NVFP4 + Min-p Sweep	0.6	0.95	0.01	39.8	52.4	61.2	68.7	74.8	59.4
			0.03	40.3	49.7	60.6	70.0	74.5	59.0
			0.05	38.7	50.2	58.5	70.2	73.3	58.2
			0.10	41.0	51.2	57.6	69.8	73.3	58.6
ReSET	ReSET	0.95	0.0	**43.8**	**54.0**	**64.9**	**72.1**	**77.5**	**62.5**

improves the average only marginally, and min-p consistently underperforms the NVFP4 baseline. ReSET performs best across all models, confirming that step-aware entropy control — rather than distribution truncation — suppresses NVFP4's sampling errors.

Selective sharpening within low-uncertainty steps. Applying T_{low} uniformly to every token in low-uncertainty steps degrades accuracy (60.6 vs. 62.5 on AIME-120; Table 7): not every such token is strictly deterministic, and uniform sharpening suppresses valid alternatives. ReSET sharpens only the tokens below τ_0 within those steps, leaving the rest unchanged.

Table 7: Low-entropy step policy effect (AIME-120).

Policy	R1-Qwen		Qwen3			Avg
	7B	14B	8B	14B	32B	
All T_{low}	40.6	51.2	65.1	72.0	74.1	60.6
ReSET	43.8	54.0	64.9	72.1	77.5	**62.5**

Hyperparameters are robust. Across reasonable ranges, ReSET's accuracy varies little: the window size $w \in \{16, 32, 64, 128\}$ moves the AIME-120 average by 1.1 points (62.3 / 62.5 / 61.9 / 61.4; Table 15), and $T_{\text{low}} \in \{0.1, 0.2, 0.3, 0.4\}$ by 0.4 points (61.2–61.6; Table 19); T_{high} improves accuracy on every model as it grows from 0.6 toward 1.0, where the average peaks (Table 16). ReSET is also insensitive to the step-boundary rule (single \n, "Wait", or sentence endings all improve over NVFP4; Table 21). The calibration itself is not math-specific: thresholds calibrated on GPQA (science) or LiveCodeBench (code) problems yield the same AIME-120 gains as NuminaMath, and re-drawing the five calibration problems shifts τ_0 by only ± 0.04 (Appendix B.7). Combined with the per-model entropy-percentile calibration of τ_0 on five NuminaMath [36] problems (Table 12), ReSET deploys without task-specific tuning.

7 Conclusion

We presented ReSET, a step-aware temperature scaling method for improving NVFP4 LRMs. By analyzing uncertainty at the reasoning-step level, ReSET adaptively controls decoding temperature to better preserve symbolic reasoning accuracy under NVFP4. We further introduced a CUDA-core small-M NVFP4 kernel optimized for latency-critical decoding. Experimental results show that ReSET consistently improves reasoning accuracy while achieving substantial decoding speedups. The gain is specific to quantization—the same policy leaves BF16 accuracy essentially unchanged—and it holds across model families and languages, calibration domains, and matched token budgets, without retraining or task-specific tuning. Together, the two components make NVFP4 practical for latency-critical reasoning serving: the decoding policy restores the accuracy that quantization takes away, and the kernel delivers the speedup that Tensor-Core paths leave unrealized at small M.

Acknowledgments

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025-00561961); by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2020-II201373, Artificial Intelligence Graduate School Program (Hanyang University); No. RS-2025-02214497, Development of low-level optimization program API technology for AI semiconductors; No. RS-2026-25507427, Development of efficient architecture and training technology for high-performance lightweight AI models; and No. RS-2026-25536190, Development of SW Performance Profiler and Performance Enhancement Verification Technology for Heterogeneous AI Semiconductors); by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea government (MOTIE) (No. RS-2026-25534585, HRD Program for Industrial Innovation); and by the Artificial Intelligence Industrial Convergence Cluster Development Project funded by the Ministry of Science and ICT (MSIT, Korea) and Gwangju Metropolitan City.

References

[1] OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen,

Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

[2] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081): 633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

[3] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[4] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

[5] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

[6] Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 196–209, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/5edb57c05c81d04beb716ef1d542fe9e-Paper-Conference.pdf.

[7] Nvidia. nvidia/deepseek-r1-nvfp4. 2025. URL https://huggingface.co/nvidia/DeepSeek-R1-NVFP4.

[8] Nvidia. Nvidia blackwell architecture technical brief. 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture.

[9] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=yfcpdY4gMP.

[10] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

[11] Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng YU, Chun Yuan, and Lu Hou. Quantization hurts reasoning? an empirical study on quantized reasoning models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=BM192Ps5Nv.

[12] Keyu Lv, Manyi Zhang, Xiaobo Xia, Jingchen Ni, Shannan Yan, Xianzhi Yu, Lu Hou, Chun Yuan, and Haoli Bai. What makes low-bit quantization-aware training work for reasoning llms? a systematic study, 2026. URL https://arxiv.org/abs/2601.14888.

[13] NVIDIA. Quantization-aware distillation for nvfp4 inference accuracy recovery. 2026.

[14] Janghwan Lee, Sihwa Lee, Jinseok Kim, Yongjik Kim, Jieun Lim, Jinwook Oh, and Jungwook Choi. ReQAT: Achieving full-precision reasoning accuracy with 4-bit floating-point quantization-aware training. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=aKY52fnzgc.

[15] Haokun Lin, Xinle Jia, Haobo Xu, Bingchen Yao, Xianglong Guo, Yichen Wu, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. Duquant++: Fine-grained rotation enhances microscaling fp4 quantization, 2026. URL https://arxiv.org/abs/2604.17789.

[16] Jack Cook, Junxian Guo, Guangxuan Xiao, Yujun Lin, and Song Han. Four over six: More accurate nvfp4 quantization with adaptive block scaling, 2025. URL https://arxiv.org/abs/2512.02010.

[17] Yuantian Shao, Peisong Wang, Yuanteng Chen, Chang Xu, Zhihui Wei, and Jian Cheng. Block rotation is all you need for mxfp4 quantization, 2025. URL https://arxiv.org/abs/2511.04214.

[18] Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Noll Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling FP4 quantization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zCBGe9AqJZ.

[19] Janghwan Lee, Jiwoong Park, Jinseok Kim, Yongjik Kim, Jungju Oh, Jinwook Oh, and Jungwook Choi. AMXFP4: Taming activation outliers with asymmetric microscaling floating-point for 4-bit LLM inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics:*

ACL 2025, pages 14993–15013, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.776. URL https://aclanthology.org/2025.findings-acl.776/.

[20] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation*, OSDI'24, USA, 2024. USENIX Association. ISBN 978-1-939133-40-3.

[21] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 193–210, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

[23] Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Stosic Dusan, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verrilli, Ralph Wittig, Doug Burger, and Eric Chung. Microscaling data formats for deep learning, 2023. URL https://arxiv.org/abs/2310.10537.

[24] NVIDIA. Introducing nvfp4 for efficient and accurate low-precision inference. https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/, 2025. Accessed: 2026-05-05.

[25] NVIDIA. Nvidia model-optimizer. 2025. URL https://github.com/NVIDIA/Model-Optimizer.

[26] OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

[27] NVIDIA. nvidia/nvidia-nemotron-nano-9b-v2-nvfp4. 2025. URL https://huggingface.co/nvidia/NVIDIA-Nemotron-Nano-9B-v2-NVFP4.

[28] Janghwan Lee, Seongmin Park, Sukjin Hong, Minsoo Kim, Du-Seong Chang, and Jungwook Choi. Improving conversational abilities of quantized large language models via direct preference alignment. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11346–11364, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.612. URL https://aclanthology.org/2024.acl-long.612/.

[29] Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, Yao Lu, Xiaojuan Qi, Song Han, and Yukang Chen. QeRL: Beyond efficiency - quantization-enhanced reinforcement learning for LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zw8zxMJJlm.

[30] Yuheng Wu, Azalia Mirhoseini, and Thierry Tambe. On the role of temperature sampling in test-time scaling, 2025. URL https://arxiv.org/abs/2510.02611.

[31] Maxwell-Jia. Aime dataset. 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

[32] Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, Yi-Hsiang Lai, Xinhao Cheng, Xupeng Miao, and Zhihao Jia. Adaserve: Accelerating multi-slo llm serving with slo-customized speculative decoding. In *Proceedings of the 21st European Conference on Computer Systems*, EUROSYS '26, page 36–54, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722127. doi: 10.1145/3767295.3769315. URL https://doi.org/10.1145/3767295.3769315.

[33] Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. SEAL: Steerable reasoning calibration of large language models for free. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=klPszYDIRT.

[34] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

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

[36] Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/datasets/AI-MO/NuminaMath-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[38] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

A Microscaling Formats

Table 8 lists the key configuration parameters of two representative microscaled FP4 formats, MXFP4 and NVFP4. Both formats encode individual elements using FP4 with an E2M1 layout, while relying on additional scaling factors to recover dynamic range. In this family of formats, values are

Format	Element Type	Block Scale	Global Scale	Block Size
MXFP4	FP4 (E2M1)	FP8 (E8M0)	–	32
NVFP4	FP4 (E2M1)	FP8 (E4M3)	FP32	16

Table 8: Configuration details of MXFP4 and NVFP4 formats.

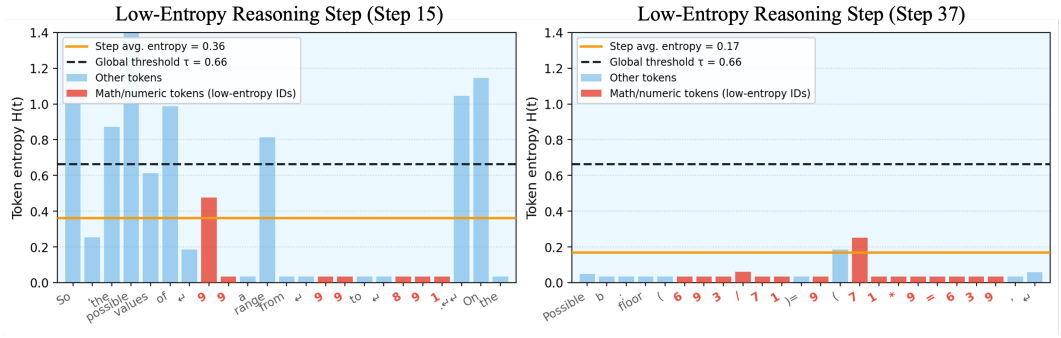

Figure 7: Using H_{step} only can misclassify low-entropy tokens.

reconstructed through a block-level scale shared across a group of elements, optionally combined with a global scaling factor.

MXFP4. MXFP4 adopts a block size of 32 and uses an FP8 (E8M0) representation for the block-wise scale. This exponent-only scale enables a broad dynamic range while keeping metadata compact, and the larger block size further reduces scaling overhead.

NVFP4. NVFP4 employs a smaller block size of 16 and represents the block scale in FP8 (E4M3), supplemented by an additional FP32 global scale. The combination of finer-grained blocks and an explicit global factor improves stability of magnitude calibration, albeit with slightly increased overhead. This configuration is particularly suited for low-precision pipelines (e.g., W4A4), where both activations and weights are quantized and more precise scaling control is required during long-context decoding.

B Additional Experimental Results and Details

B.1 Analysis Details

We analyzed the entropy characteristics of tokens produced during extended reasoning processes. Our data was derived from the R1-Qwen-14B model's outputs across 90 mathematical challenges from the 2022–2024 AIME datasets [31]. The resulting corpus consists of over 1.5 million individual tokens, each associated with its respective entropy metric. For the final visualization, we applied a frequency filter, selecting only those tokens that appeared 1,000 times or more. We then partitioned these frequent tokens into low-entropy and high-entropy categories to illustrate the contrast between deterministic and high-variance token generation.

Definition of a Reasoning Step. We define a *reasoning step* as a contiguous segment of tokens corresponding to a coherent intermediate reasoning unit. Following [10, 33], we approximate step boundaries using a delimiter rule, where segments are separated by double newline tokens ("\n\n"). The same rule is applied online during decoding (Sec. 4); Appendix B.10 evaluates alternative boundary rules.

B.2 Quantization Distorts Entropy Differently Across Reasoning Steps

Sec. 3 shows that quantization increases sampling errors at low-entropy symbolic tokens by increasing the probability of non-top-1 alternatives. However, we additionally observe that the direction of entropy change depends strongly on both token entropy and step-level uncertainty.

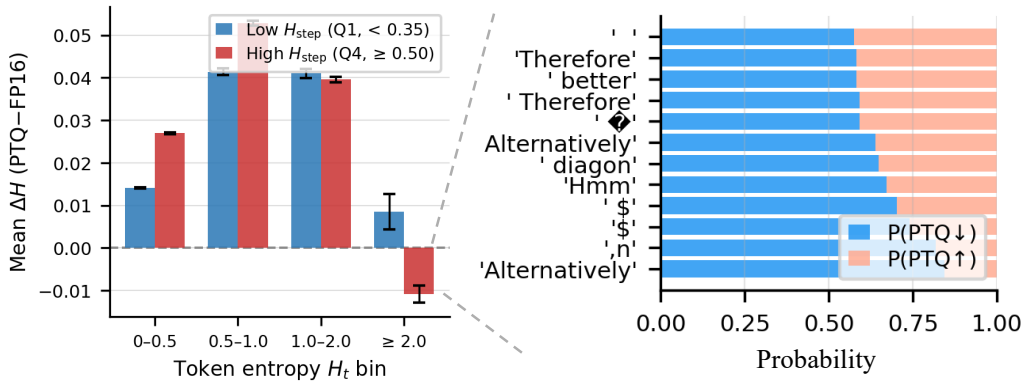

Figure 8: Entropy shift under quantization. (left) Mean entropy shift $\Delta H = H_{\text{PTQ}} - H_{\text{BF16}}$ across token entropy bins under low- and high-uncertainty steps. (right) Probability of entropy increase or decrease for representative high-entropy tokens under PTQ.

Fig. 8 (left) shows the mean entropy shift $\Delta H_t = H_t^{\text{PTQ}} - H_t^{\text{BF16}}$ across token entropy bins under low- and high-uncertainty reasoning steps. For low-entropy tokens, quantization consistently increases entropy, with larger shifts observed in high-uncertainty steps. In contrast, for very high-entropy tokens ($H_t \geq 2.0$), the entropy shift becomes negative in high-uncertainty steps, indicating that quantization instead sharpens the token distribution.

At the token level (Fig. 8 right), representative high-entropy discourse tokens such as "Alternatively" more frequently exhibit entropy reduction under PTQ. This suggests that quantization can artificially concentrate probability mass on a smaller subset of continuations in uncertain reasoning states.

Overall, quantization introduces two distinct effects: entropy inflation for low-entropy symbolic tokens and entropy collapse for high-entropy tokens in uncertain reasoning steps. These observations motivate step-aware decoding policies that adapt entropy thresholds according to the uncertainty of the current reasoning step.

Table 9: Branch-wise temperature sweep of Fig. 2(f) under NVFP4 and BF16 on R1-Qwen-7B, AIME 2022–2024 (mean accuracy ± 95% CI over eight seeds). One branch temperature is varied to T' while the other is fixed at 0.6.

Sweep	Precision	T'=0.0	T'=0.1	T'=0.3	T'=0.6	T'=1.0
Vary T_{low}=T' (T_{high}=0.6)	NVFP4	44.0 ± 3.4	41.9 ± 1.7	43.3 ± 2.1	39.6 ± 1.9	41.1 ± 3.6
	BF16	45.1 ± 3.0	45.5 ± 2.7	46.5 ± 3.2	45.7 ± 2.5	45.8 ± 2.7
Vary T_{high}=T' (T_{low}=0.6)	NVFP4	35.0 ± 2.9	29.4 ± 2.0	37.4 ± 2.1	39.6 ± 1.9	43.3 ± 3.5
	BF16	40.6 ± 2.2	41.5 ± 1.9	43.5 ± 2.8	45.7 ± 2.5	47.8 ± 2.2

Branch-wise temperature sweep with confidence intervals. Table 9 repeats the branch-wise sweep of Fig. 2(f) over eight seeds with 95% confidence intervals and adds the corresponding BF16 sweep. Lowering T_{low} raises NVFP4 accuracy to 41.9–44.0 from the 39.6 ± 1.9 default, whereas BF16 stays flat across the same sweep (45.1–46.5), showing no systematic benefit from sharpening at full precision: low-entropy sharpening corrects an NVFP4-specific error. In contrast, lowering T_{high} below 0.6 reduces accuracy at both precisions (NVFP4: 29.4–37.4; BF16: 40.6–43.5), while T'=1.0 reaches 43.3 and 47.8, respectively. Preserving exploration at high-entropy positions is thus a general property of reasoning decoding; under NVFP4, the entropy compression identified above further strengthens this effect.

B.3 Matched-Prefix Diagnostics: Rank Stability and Non-Top-1 Sampling Probability

Following the matched-prefix analysis of ReQAT [14], we feed identical prefixes to the BF16 and NVFP4 models (Qwen3-32B) and compare their next-token distributions. Positions are grouped by

Table 10: Ratio of mean non-top-1 sampling probability, $1 - p(\text{top-1})$, under NVFP4 to BF16 at matched prefixes (Qwen3-32B). Positions are grouped by task-specific BF16 entropy percentiles: low ($H < \text{P70}$) and high ($H \geq \text{P70}$).

Task	Low-entropy	High-entropy
AIME-120	1.54×	0.93×
GPQA-Diamond	1.08×	0.91×
LiveCodeBench	1.10×	0.78×

Table 11: Rank of the BF16 top-1 token under the NVFP4 next-token distribution at low-entropy positions ($H < \text{P70}$) on Qwen3-32B AIME-120 matched prefixes.

Diagnostic (BF16 top-1 token)	Result
Evaluated positions	46,375
Remains rank 1 under NVFP4	99.91%
Falls to rank 2	0.071%
Falls to rank ≥ 3	0.017%
Non-top-1 prob. ratio (NVFP4/BF16)	1.54×

each task's BF16 token-entropy distribution (bottom 70%: low entropy; top 30%: high entropy), and we measure the mean probability of sampling any token other than the top-1 token, $1 - p(\text{top-1})$.

Task-wise sensitivity to the low-entropy failure mode. Table 10 shows that, at low-entropy positions, NVFP4 inflates the non-top-1 sampling probability by 54% on AIME-120, compared with only 8% and 10% on GPQA-Diamond and LiveCodeBench. AIME therefore exhibits a stronger instance of the failure mode targeted by T_{low}, which explains why ReSET's gains are largest there (Table 1). At high-entropy positions, NVFP4 *decreases* the non-top-1 sampling probability on all three tasks (0.78–$0.93\times$), i.e., the distribution becomes more concentrated at uncertain positions, consistent with the entropy compression in Fig. 8 and with preserving exploration through T_{high}.

Deep token reordering is negligible. Table 11 measures the rank of the BF16 top-1 token under the NVFP4 distribution at the same low-entropy positions. Across 46,375 positions, the BF16 top-1 remains rank 1 under NVFP4 in 99.91% of cases; displacement to rank 3 or lower occurs in only 0.017%. NVFP4 thus primarily increases the probability of sampling non-top-1 alternatives ($1.54\times$) while preserving the top-1 ordering — precisely the failure mode that temperature sharpening corrects, and the operating assumption of ReSET holds overwhelmingly in practice.

B.4 Step entropy estimators

$$\begin{aligned}
\hat{H}^{\text{causal}}_{\text{step}}(t) &= \frac{1}{t - t_0 + 1} \sum_{i=t_0}^{t} H_i, \\
\hat{H}^{\text{window}}_{\text{step}}(t) &= \frac{1}{|\mathcal{W}_t|} \sum_{i \in \mathcal{W}_t} H_i, \\
\hat{H}^{\text{EMA}}_{\text{step}}(t) &= \alpha H_t + (1 - \alpha) \hat{H}^{\text{EMA}}_{\text{step}}(t - 1).
\end{aligned} \tag{4}$$

We provide the formal definitions of the step-entropy estimators used in Sec. 4. The causal estimator computes the running average over tokens within the current step, relying only on past observations. The sliding-window estimator restricts this average to a fixed-size window $\mathcal{W}_t$ to emphasize recent tokens. The exponential moving average (EMA) instead applies temporal smoothing with decay factor α=0.1.

B.5 Hardware and Software Settings

All experiments are performed on a single NVIDIA B200 GPU (sm_100a, 148 SMs, 180 GB HBM3e, 8 TB/s peak HBM bandwidth) with CUDA 13.1 and PyTorch 2.10.0. Inference is served through `vLLM` 0.19.1 with the `FlashInfer-CUTLASS` backend (FlashInfer 0.6.6); the vLLM-CUTLASS kernel baseline is `cutlass_scaled_fp4_mm` from the same release. Our NVFP4 runtime integrates two custom kernel variants: a CUDA-core GEMV variant for latency-critical small-M decode and a Tensor-Core GEMM variant for larger-M decode and prefill. The GEMV and GEMM variants are registered as `torch.ops.nvfp4r.gemv` and `torch.ops.nvfp4r.gemm`, respectively. During serving, the runtime dispatches to the GEMV variant for small-M decode projections and to the GEMM variant when Tensor-Core tiles are sufficiently occupied.

B.6 Evaluation Details

We conduct our experiments using the vLLM framework and models including Qwen3 and DeepSeek-R1, which are licensed under Apache 2.0 and MIT licenses, respectively. We ensure full compliance with these licenses by providing proper citations and adhering to their terms of use.

NVFP4 Quantization. We use the NVIDIA ModelOpt [25] library for NVFP4 quantization. All Transformer decoder layers are quantized. Weights are calibrated using min–max quantization, and the global scale is calibrated on the C4 dataset [37] using 32 samples with sequence length 2048.

4/6 (Four-Over-Six). We follow the original 4/6 algorithm, which selects quantization parameters by minimizing activation MSE, while weights are quantized using ModelOpt. For a fair comparison, we use a fixed global scale obtained from ModelOpt calibration instead of dynamic global scaling. The global scale is calibrated on the C4 dataset with sequence length 2048 and 32 samples.

MR-GPTQ. We follow the official MR-GPTQ implementation. Weight quantization is performed by minimizing MSE using the FineWeb-Edu dataset (sequence length 2048, 128 samples). Activation ranges are calibrated using MinMax. We disable the randomized Hadamard transform (block size 16), as it degrades accuracy under NVFP4 for Qwen3 models [18].

BRQ. We implement BRQ with a randomized Hadamard transform using block size 16.

Sampling setup. Unless otherwise stated, all results are obtained using stochastic decoding with temperature 0.6 and top-p sampling with $p = 0.95$.

Table 12: Global entropy threshold τ (80th percentile) per model.

Model	τ
R1-7B	0.6446
R1-14B	0.6488
Qwen3-8B	0.5505
Qwen3-14B	0.4863
Qwen3-32B	0.5363

Calibration for global threshold τ We calibrate the global threshold τ using a small subset of the NuminaMath-1.5 [36] dataset. Specifically, we randomly sample 5 problems, generate model responses, and compute token-level entropy for all generated tokens. We then set τ as the 80th percentile of the resulting entropy distribution. τ used in our experiments are in Table 12.

B.7 Calibration Robustness

Table 13: AIME-120 accuracy with τ_0 calibrated on different domains (no AIME-specific recalibration).

Calibration source (domain)	Qwen3-8B	Qwen3-14B
NVFP4 baseline	62.5	70.4
NuminaMath (math)	64.9	72.1
GPQA-main (science)	64.7	71.7
LiveCodeBench (code)	66.1	72.1

Table 14: Thresholds and AIME-120 accuracy for three five-problem NuminaMath calibration sets, each evaluated end to end.

Calibration set	Qwen3-8B		Qwen3-14B	
	τ_0	AIME-120	τ_0	AIME-120
NVFP4 baseline	–	62.5	–	70.4
Reported calibration set	0.551	64.7	0.486	72.0
Random draw 1	0.535	64.6	0.518	74.2
Random draw 2	0.582	63.8	0.530	72.8

Calibration domain. To test whether the gains depend on math-specific calibration, we calibrate τ_0 on GPQA-main (science) and LiveCodeBench (code) problems instead of NuminaMath, without

Table 15: Sensitivity to sliding window size w in step entropy estimation (AIME-120).

w	R1-Qwen-7B	R1-Qwen-14B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Avg
16	44.9	53.9	65.7	70.3	76.7	62.3
32	43.8	54.0	64.9	72.1	77.5	**62.5**
64	44.5	54.4	63.5	72.0	75.1	61.9
128	43.2	52.0	64.7	71.8	75.3	61.4

Table 16: AIME-120 score under NVFP4 ReSET with $T_{\text{low}} = 0.1$ across different T_{high}.

T_{high}	R1-Qwen-7B	R1-Qwen-14B	Qwen3-8B	Qwen3-14B	Qwen3-32B
0.6 (Baseline)	39.6	50.9	61.9	71.4	74.9
0.7	39.8	51.6	64.3	70.6	74.1
0.8	40.4	54.8	64.5	73.7	74.9
0.9	43.8	54.5	64.6	71.1	75.3
1.0 (Default in ReSET)	42.0	52.9	64.9	70.9	77.5

any AIME-specific recalibration, and evaluate on AIME-120 (Table 13). Thresholds calibrated on science and code consistently outperform the NVFP4 baseline and match the NuminaMath-calibrated accuracy, so ReSET does not require math-specific calibration.

Calibration examples. We further repeat the complete calibration procedure with two additional randomly drawn five-problem NuminaMath subsets and evaluate the resulting AIME-120 accuracy end to end (Table 14). Qwen3-8B consistently achieves 63.8–64.7 versus the 62.5 baseline, and Qwen3-14B 72.0–74.2 versus 70.4. Although calibration uses only five problems, each problem yields roughly 10k–15k generated tokens, i.e., tens of thousands of token-level entropy observations; accordingly, repeating threshold estimation across ten independently sampled five-problem subsets gives a stable $\tau_0 = 0.543 \pm 0.044$ for Qwen3-8B, close to the reported value (0.551).

B.8 Ablation Studies

Effect of sliding window size in step entropy estimation. We evaluate the impact of the window size w used in the hybrid step-entropy estimator. As shown in Table 15, performance is relatively stable across different window sizes, indicating that ReSET is not sensitive to the precise choice of w. The best performance is achieved at $w = 32$, which we use in all experiments. Larger window sizes do not provide additional gains and can slightly degrade performance, suggesting that overly long windows blur step-local statistics and weaken the effectiveness of step-aware control.

Effect of high-temperature scaling in uncertain steps. To validate the design choice of using a higher temperature in high-uncertainty steps, we evaluate ReSET with fixed $T_{\text{low}} = 0.1$ while sweeping T_{high}. Table 16 shows that increasing T_{high} consistently improves performance over the NVFP4 baseline across models. In particular, moving from the base temperature ($T_{\text{high}} = 0.6$) to higher values (0.8–1.0) yields substantial gains, indicating that higher temperature is beneficial in uncertain reasoning regimes. This trend is consistent with the observation in Sec. B.2 that quantization reduces entropy and concentrates token distributions in high-uncertainty steps. Increasing T_{high} counteracts this effect by restoring uncertainty, allowing the model to maintain multiple plausible continuations during reasoning.

Table 17: Perplexity Comparison across Models

Bit-Precision	Method	R1-Qwen-7B	R1-Qwen-14B	Qwen3-8B	Qwen3-14B	Qwen3-32B
BF16	Baseline	25.07	8.91	9.72	8.65	7.61
NVFP4	RTN	26.23	9.50	10.10	8.91	8.01
	4/6	26.19	9.49	10.08	8.86	7.98
	BRQ	27.00	9.81	10.35	9.53	8.17
	MR-GPTQ	27.48	9.64	10.06	8.95	7.81

Table 19: Sensitivity to T_{low} with fixed $T_{\text{high}} = 1.0$ in ReSET (AIME-120).

T_{low}	R1-Qwen-7B	R1-Qwen-14B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Avg
0.1	42.0	52.9	64.9	70.9	77.5	**61.6**
0.2	42.7	52.9	64.3	70.7	75.7	61.2
0.3	41.4	54.0	64.1	72.1	75.6	61.4
0.4	43.8	52.2	64.8	71.2	75.4	61.5

Perplexity evaluation across PTQ methods. We evaluate Wikitext-2 [38] perplexity across PTQ methods in Table 17. Advanced PTQ methods such as rotation and GPTQ provide only marginal perplexity gains over RTN under NVFP4, with differences remaining minimal across models.

Table 18: Comparison with NVFP4 PTQ methods across tasks. †: ReSET applied on top of GPTQ.

Task	Method	R1-Qwen 7B	R1-Qwen 14B	Qwen3 8B	Qwen3 14B	Qwen3 32B	Avg
AIME-120	RTN	39.6	52.4	62.5	70.4	74.4	59.9
	ReSET	43.8	54.0	64.9	72.1	77.5	**62.5**
	ReSET†	41.9	53.5	65.0	72.4	76.1	61.6
GPQA-Diamond	RTN	47.1	53.7	50.7	57.4	62.8	54.3
	ReSET	46.0	57.6	53.2	58.5	62.9	**55.6**
	ReSET†	44.4	57.4	52.2	59.6	64.3	**55.6**
LiveCodeBench	RTN	29.5	37.1	36.4	46.7	46.5	39.2
	ReSET	28.4	37.9	42.1	46.1	46.7	40.2
	ReSET†	29.5	37.3	43.8	46.7	48.8	**41.2**

Employing ReSET with GPTQ. As shown in Table 18, while ReSET itself substantially improves over the vanilla RTN baseline across most tasks and model scales, additionally combining it with GPTQ provides little to no further benefit. In many settings, ReSET† achieves performance comparable to, or only marginally different from, ReSET alone. We hypothesize that this behavior arises because conventional PTQ methods such as GPTQ mainly contribute by slightly reducing NVFP4 quantization errors, which indirectly helps control the increase of non-top1 token probabilities during low-entropy token selection, thereby improving reasoning accuracy. However, ReSET already performs a similar corrective role through its temperature scaling mechanism, effectively stabilizing the token distribution without requiring additional reconstruction-based PTQ refinement.

B.9 Fixed-Temperature Baselines under NVFP4 and BF16

Table 20: AIME-120 accuracy under fixed-temperature decoding and ReSET for all five models at both precisions. Δ is ReSET minus the default temperature (T=0.6).

Precision	Model	T=0.1	T=0.6 (Default)	T=0.8	T=1.0	**ReSET**	Δ
NVFP4	R1-Qwen-7B	35.2	39.6	40.8	41.2	**43.8**	+4.2
	R1-Qwen-14B	48.0	52.4	51.6	52.1	**54.0**	+1.6
	Qwen3-8B	54.2	62.5	60.3	62.9	**64.9**	+2.4
	Qwen3-14B	68.4	70.4	70.1	70.1	**72.1**	+1.7
	Qwen3-32B	72.9	74.4	75.2	73.6	**77.5**	+3.1
BF16	R1-Qwen-7B	39.2	45.7	46.2	46.8	45.7	+0.0
	R1-Qwen-14B	52.5	57.4	58.2	58.3	56.8	−0.6
	Qwen3-8B	68.9	70.4	72.1	73.5	71.5	+1.1
	Qwen3-14B	72.4	76.1	75.3	76.2	75.5	−0.6
	Qwen3-32B	74.0	75.8	76.5	76.1	77.0	+1.2

Table 20 reports the full temperature sweep summarized in Table 5: constant-temperature decoding at $T \in \{0.1, 0.6, 0.8, 1.0\}$ versus ReSET, for all five models at both precisions on AIME-120. Under NVFP4, ReSET outperforms both the default temperature (by +1.6–4.2 points, +2.6 on average) and the best fixed temperature selected separately for each model (by +1.6–2.6 points, +2.0 on

average). Under BF16, the same policy changes accuracy by only -0.6 to $+1.2$ points ($+0.2$ on average), confirming that ReSET targets NVFP4-induced degradation rather than acting as a generic temperature-tuning strategy.

B.10 Online Step-Boundary Detection and Robustness

Table 21: AIME-120 accuracy under alternative online reasoning-step boundary rules.

Boundary rule	Qwen3-8B	Qwen3-14B	Qwen3-32B
NVFP4 baseline (T=0.6)	62.5	70.4	74.4
Single \n	64.9	71.8	75.5
"Wait" marker	64.7	71.6	75.4
Sentence endings (., !, ?)	64.4	70.5	75.2
\n\n (ReSET default)	**64.9**	**72.1**	**77.5**

Online boundary detection. ReSET uses \n\n as the reasoning-step boundary (Appendix B.1). During decoding, a boundary is identified when the generated token is either a composite token containing \n\n or the second of two consecutive newline tokens. This requires only the cached newline-related token IDs and one Boolean state, and adds no measurable cost to the per-token bookkeeping of Sec. 4.

Robustness to the boundary rule. Table 21 evaluates alternative step-boundary rules adopted by prior reasoning methods [33, 10]: a single \n, the "Wait" reflection marker, and sentence-ending punctuation. ReSET outperforms the NVFP4 baseline under every rule, and the default \n\n achieves the highest or tied-highest accuracy on all models, so ReSET is not sensitive to the exact segmentation.

B.11 Additional Model Families and Languages

Table 22: AIME-120 accuracy on additional model families using the unchanged \n\n reasoning-step boundary.

Method	R1-Llama (8B)	Magistral-Small 2506 (24B)	Gemma-4 31B-IT
BF16	38.9	60.2	84.4
NVFP4	28.1	52.4	82.8
NVFP4 + ReSET	**33.4**	**56.6**	**83.4**

Table 23: MT-AIME-2024 accuracy across languages with the same \n\n boundary (Qwen3-8B).

Method	ja	fr	es	Avg.
BF16 (T=0.6)	65.8	75.8	73.3	71.6
NVFP4 (T=0.6)	60.0	61.7	62.1	61.3
NVFP4 + ReSET	**65.4**	**64.2**	**70.0**	**66.5**

Model families. To test whether the entropy dynamics ReSET exploits transfer beyond Qwen, we apply ReSET with the unchanged \n\n boundary and no architecture-specific modification to DeepSeek-R1-Distill-Llama-8B, Magistral-Small-2506 (24B), and Gemma-4-31B-IT (Table 22). ReSET improves NVFP4 AIME-120 accuracy by $+5.3$, $+4.2$, and $+0.6$ points, respectively.

Languages. Table 23 evaluates Qwen3-8B on the multilingual MT-AIME-2024 benchmark in Japanese, French, and Spanish with the same boundary. ReSET improves NVFP4 accuracy in every language and raises the average from 61.3% to 66.5% ($+5.2$ points), so the \n\n segmentation is robust across languages.

B.12 Output Length and Matched Token Budgets

Table 24: Qwen3-32B output-length statistics (tokens) over 960 AIME-120 generations per policy.

Policy	Mean	Median (P25–P75)
NVFP4 (T=0.6)	15,496	13,844 (8,435–20,883)
NVFP4 + ReSET	15,531 ($+0.2\%$)	13,746 (8,358–21,359)

Table 25: AIME-120 accuracy (Qwen3-32B) under different maximum generation-token budgets, obtained by truncating and re-scoring the same generations.

Max. token budget	8k	16k	24k	32k
NVFP4 (T=0.6)	30.4	59.0	70.9	74.4
NVFP4 + ReSET	29.7	59.2	**73.6**	**77.5**

Because ReSET raises the temperature in uncertain steps, one might expect it to lengthen reasoning traces. Table 24 compares output-length distributions over 960 Qwen3-32B generations per policy: the mean changes by only $+0.2\%$, the median decreases, and the central 50% ranges are comparable, so ReSET does not materially lengthen reasoning. Table 25 re-scores the same generations after truncating them to shorter maximum-token budgets. The two policies are similar at 8k and 16k, where many traces cannot finish; at 24k and 32k, ReSET gains $+2.7$ and $+3.1$ points without receiving a larger budget. The accuracy gain therefore persists under matched token budgets and is not explained by longer generations.

C NVFP4 Kernel Implementation Details

This appendix provides implementation details for the NVFP4 kernels used in Section 5. We first describe the CUDA-core small-M decode kernel, then describe the Tensor-Core GEMM path used for larger M and prefill. All kernel latencies are measured on a single NVIDIA B200 with CUDA-graph replay and are reported in microseconds.

C.1 CUDA-Core Small-M Decode Kernel

Multi-token CTA fusion. In small M scenario, naive CUDA-core mapping assigns different decode tokens to independent CTAs. This mapping is simple, but it repeatedly streams the same weight tile for each active token because all decode tokens share the same weight matrix. For small M, this repeated weight traffic is expensive: the amount of reuse available across tokens is limited, and launching independent CTAs prevents the reuse that does exist from being exploited within the CTA.

Our kernel instead fuses multiple active decode tokens within a CTA. Each CTA is assigned a tile of output rows and a small group of active decode tokens. It then traverses the K dimension once, loads a tile of the shared weight matrix, and reuses the loaded weight values across token-wise dot products for the active tokens in the CTA. This reduces redundant weight loads compared with processing each token independently, while avoiding padding the token dimension to the fixed Tensor-Core tile size. The approach is especially useful for small but non-unit decode batches, where M is still too small for efficient Tensor-Core GEMM but large enough to benefit from intra-CTA weight reuse.

Multi-accumulator threading. After moving from Tensor Cores to CUDA cores, the projection is implemented as many `half2` multiply-accumulate operations. If each thread maintains only one output accumulator, the inner loop forms a single dependent FMA chain along the K dimension: each new accumulation depends on the result of the previous `hfma2`. This limits instruction-level parallelism and can leave CUDA cores underutilized, especially when the loop also includes FP4 unpacking and scale application.

To reduce this dependency bottleneck, each thread maintains multiple output accumulators. In our evaluated configuration, each thread owns two weight rows and accumulates two independent output values. The two accumulators create independent `hfma2` streams, allowing the warp scheduler to interleave instructions from different dependency chains. This hides part of the latency from conversion, scaling, and FMA instructions without increasing shared-memory traffic. The number of accumulators is chosen to balance instruction-level parallelism against register pressure: using too few accumulators exposes insufficient parallelism, while using too many reduces occupancy by increasing per-thread register usage.

Register-only dequantization. CUDA-core NVFP4 execution must explicitly perform the unpacking and scaling operations that are normally handled inside Tensor-Core MMA. Packed FP4 values are loaded into registers, converted to FP16 pairs using `cvt.rn.f16x2.e2m1x2`, multiplied by the

corresponding shared NVFP4 scale, and then consumed by `hfma2` accumulation. The activation values and their scales are handled similarly, so the inner loop performs dequantization and accumulation together.

A straightforward implementation could first materialize the dequantized FP16 values in shared memory and then load them again for accumulation. However, this introduces additional shared-memory traffic, increases synchronization requirements, and lengthens the inner loop. These costs are particularly harmful in the small-M regime, where the compute work per CTA is limited and fixed overheads are difficult to amortize.

Our kernel instead keeps the converted FP16 values, scales, and partial accumulators in registers. Conversion and scale application are scheduled close to the consuming `hfma2` instructions, so the dequantized values are produced and consumed without being written to shared memory. Across successive K tiles, the kernel overlaps weight loading, FP4 conversion, scale multiplication, and accumulation through software pipelining. This register-only design reduces memory traffic and synchronization overhead while preserving the low memory footprint of NVFP4 weights.

Runtime dispatch. The CUDA-core kernel is used only in the small-M decode regime, where Tensor-Core GEMM suffers from tile under-occupancy. For larger M and prefill, we dispatch to the Tensor-Core NVFP4 GEMM path, where the fixed Tensor-Core tile is sufficiently occupied and the higher peak throughput becomes beneficial. The dispatch decision is made per shape and decode-batch regime, so the serving runtime can use the CUDA-core path for latency-critical decoding without affecting the high-throughput prefill path.

C.2 CUDA-Core Kernel Ablation

Table 26: CUDA-core small-M kernel ablation on Qwen3-32B `down_proj` (N=5120, K=25600). Latency is reported in microseconds, measured on B200 with CUDA-graph replay.

	$M=1$	$M=4$	$M=8$
Ours: full kernel (C1+C2+C3)	15.0	27.5	58.3
without C1 (no CTA fusion)	15.3	35.9	70.9
without C2 (single-row threading)	16.8	32.4	64.6
without C1 and C2	18.7	39.4	76.5

Table 26 shows that CTA fusion and multi-accumulator threading address complementary bottlenecks. CTA fusion becomes more important as M grows, reducing latency from 70.9µs to 58.3µs at $M = 8$ by reusing each streamed weight tile across multiple decode tokens. Multi-accumulator threading improves all regimes, including $M = 1$, by increasing per-thread instruction-level parallelism. Removing both choices compounds the slowdown, increasing latency by $1.31\times$ at $M=8$.

We do not isolate register-only dequantization as a separate ablation because replacing it with shared-memory materialization changes the kernel pipeline itself, including shared-memory allocation, synchronization placement, and pipeline depth. Instead, the ablation focuses on mapping choices that can be removed while preserving the same CUDA-core NVFP4 execution structure. Register-only dequantization remains part of the full kernel because it avoids extra shared-memory traffic and enables overlap between conversion, scale application, and accumulation in the inner K loop.

C.3 Tensor-Core GEMM Kernel and Runtime Dispatch

In addition to the CUDA-core small-M kernel, we implement an NVFP4 Tensor-Core GEMM kernel for larger M and prefill. This Tensor-Core kernel is separate from vLLM's `cutlass_scaled_fp4_mm`, which we use as the baseline in this comparison. It targets the NVFP4 layout and the Qwen3 linear-layer shapes evaluated in this work, allowing us to tune tiling, scheduling, and scale handling directly for the target workloads.

Tables 27 and 28 compare our kernels with vLLM's `cutlass_scaled_fp4_mm` (vLLM-CUTLASS). Table 27 reports latency at small decode sizes (M=4, 8). Here, Ours denotes the faster of our CUDA-core small-M kernel and our Tensor-Core GEMM kernel for each layer shape, and speedup is measured over vLLM-CUTLASS. Table 28 reports latency at M=128, where our Tensor-Core GEMM kernel is used; our kernel is up to $1.34\times$ faster than vLLM-CUTLASS.

Table 27: Per-layer latency at small decode sizes (M=4, 8). Latency is reported in microseconds.

				M=4			M=8		
Model	**Layer**	N	K	**vLLM**	**Ours**	**Speedup**	**vLLM**	**Ours**	**Speedup**
Qwen3-8B	qkv_proj	6144	4096	8.32	6.18	1.35×	8.05	6.01	1.34×
Qwen3-8B	o_proj	4096	4096	7.47	6.00	1.25×	7.42	5.81	1.28×
Qwen3-8B	gate/up_proj	12288	4096	10.34	8.67	1.19×	10.35	8.26	1.25×
Qwen3-8B	down_proj	4096	12288	16.05	11.57	1.39×	15.49	11.62	1.33×
Qwen3-14B	qkv_proj	7168	5120	10.92	7.40	1.48×	10.09	7.42	1.36×
Qwen3-14B	o_proj	5120	5120	9.09	6.88	1.32×	8.66	6.81	1.27×
Qwen3-14B	gate/up_proj	17408	5120	14.88	11.21	1.33×	13.59	11.19	1.21×
Qwen3-14B	down_proj	5120	17408	23.58	15.40	1.53×	22.22	15.02	1.48×
Qwen3-32B	qkv_proj	10240	5120	13.50	8.43	1.60×	11.95	8.45	1.41×
Qwen3-32B	o_proj	5120	8192	12.31	9.31	1.32×	11.92	9.46	1.26×
Qwen3-32B	gate/up_proj	25600	5120	26.08	17.51	1.49×	23.14	17.33	1.34×
Qwen3-32B	down_proj	5120	25600	34.14	20.54	1.66×	31.94	20.56	1.55×

Table 28: Per-layer latency for NVFP4 Tensor-Core GEMM at M=128. Latency is reported in microseconds.

Model	**Layer**	N	K	**vLLM** (μs)	**Ours** (μs)	**Speedup**
Qwen3-8B	qkv_proj	6144	4096	13.76	11.36	1.21×
Qwen3-8B	o_proj	4096	4096	13.79	11.23	1.23×
Qwen3-8B	gate/up_proj	12288	4096	18.85	14.78	1.27×
Qwen3-8B	down_proj	4096	12288	18.91	14.82	1.28×
Qwen3-14B	qkv_proj	7168	5120	13.89	11.30	1.23×
Qwen3-14B	o_proj	5120	5120	13.79	11.52	1.20×
Qwen3-14B	gate/up_proj	17408	5120	20.93	18.69	1.12×
Qwen3-14B	down_proj	5120	17408	25.06	18.88	1.33×
Qwen3-32B	qkv_proj	10240	5120	18.91	14.94	1.27×
Qwen3-32B	o_proj	5120	8192	16.86	12.74	1.32×
Qwen3-32B	gate/up_proj	25600	5120	25.06	24.99	1.00×
Qwen3-32B	down_proj	5120	25600	33.25	24.80	1.34×

The runtime integrates the CUDA-core small-M kernel and the Tensor-Core GEMM kernel through a shape-aware dispatch policy. For M=1, 2, the CUDA-core small-M kernel is always used because it is faster in this regime. For $M \geq 128$, the Tensor-Core GEMM kernel is used because the fixed Tensor-Core tile is sufficiently occupied. For intermediate sizes, the faster kernel depends on the projection shape, so the runtime selects between the CUDA-core and Tensor-Core kernels based on measured latency for each (M, shape) pair. Because vLLM captures a separate CUDA graph for each batch size, this selection is fixed at graph-capture time and incurs no branching overhead during replay.

D Limitations

Our analysis and evaluation focus primarily on reasoning-oriented autoregressive decoding workloads under NVFP4 quantization. Although ReSET consistently improves reasoning accuracy across the evaluated model families and benchmarks, the relationship between token entropy and quantization-induced reasoning degradation may vary across other architectures, domains, or task types that were not explored in this work. In particular, these entropy dynamics may depend on how the underlying reasoning model is trained, including factors such as post-training reinforcement learning, supervised fine-tuning objectives, or reasoning-trace distillation strategies. As a result, the optimal entropy thresholds and temperature hyperparameters may vary depending on model-specific entropy characteristics and decoding behavior.

In addition, our analysis attributes many reasoning failures to incorrect sampling at locally deterministic low-entropy tokens. While we observe that such symbolic-token errors frequently correlate with downstream reasoning failures, reasoning degradation in quantized models is likely influenced by multiple interacting factors beyond isolated symbolic mistakes alone. For example, quantization

may also affect long-range reasoning consistency, discourse planning, or intermediate semantic representations in ways not fully captured by token-level entropy analysis.

Our experiments also focus on NVFP4 inference on NVIDIA Blackwell GPUs and latency-critical small-batch decoding settings. Although this setting is practically important for modern reasoning-model serving, the effectiveness of the proposed decoding strategy and kernel design under other hardware platforms, larger-batch throughput-oriented serving regimes, or alternative low-precision formats remains underexplored. In particular, more aggressive microscaling formats such as MXFP4 may introduce larger quantization errors and different entropy dynamics, which could provide an important direction for future investigation.

E Broader Impacts

This work studies low-precision inference for large reasoning models. Improving quantized inference efficiency may help reduce deployment cost and resource usage for large language models. However, low-precision quantization can also introduce unintended reasoning errors or incorrect generations, particularly in applications requiring reliable multi-step reasoning. Our work aims to mitigate accuracy degradations in reasoning benchmarks under NVFP4 inference, but quantized reasoning models may still exhibit unexpected behaviors depending on the deployment setting and quantization configuration.

F LLM Usage

We employed large language models (LLMs) exclusively for the purpose of code refinement and grammatical editing of the manuscript. All scientific contributions, experimental results, and interpretations were generated and verified by the human authors.

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: The abstract and introduction summarize the paper's main contributions, including entropy-aware temperature scaling, and decode-specialized kernel design.

   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: Limitations regarding evaluation scope, model dependence, and alternative quantization formats are discussed in Sec. D in Appendix.

   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 does not present formal theoretical results requiring theorem statements or proofs.

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: The paper describes the evaluated models, datasets, quantization settings, and decoding settings needed to reproduce the main experimental results.

   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: [Yes]

Justification: The ReSET sampler and the NVFP4 CUDA-core/Tensor-Core kernels are released at `https://github.com/aiha-lab/ReSET`, together with the evaluation scripts; the experimental setup and implementation details are described in Sec. 6 and Appendix B–C.

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: The paper specifies the relevant evaluation details, including models, benchmarks, decoding configurations, quantization settings, and calibration/evaluation setup. The work does not involve model training, so optimizer and training split details are not applicable.

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: Accuracy results are averaged over 8 random seeds (Sec. 6.1), and the branch-wise temperature sweep supporting the main analysis is reported with 95% confidence intervals over eight seeds (Table 9). Latency measurements are repeated with CUDA-graph replay and summarized by their central value.

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: The paper reports the GPU platform used for the main latency experiments, including B200 hardware, and describes the relevant CUDA/NVFP4 kernel execution setting. It also specifies the model sizes and serving configurations needed to estimate compute and memory requirements.

   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: The research is focused on quantization, decoding, and inference-kernel efficiency. It uses standard models and benchmarks and does not involve deception, privacy-invasive data collection, or human-subject experiments.

   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: [Yes]

Justification: The paper briefly discusses potential societal impacts and risks of quantization-induced reasoning errors and unintended model behavior in low-precision large language model deployment.

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: The paper does not release a new high-risk pretrained model, image generator, or scraped dataset. The work proposes inference and kernel techniques evaluated using existing models and benchmarks.

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: [Yes]

Justification: The paper cites all models, datasets, and frameworks used. Specifically, our experiments rely on vLLM (Apache 2.0), Qwen3 (Apache 2.0), and DeepSeek-R1 (MIT). These are publicly available assets, and we have complied with their respective licenses, with all sources properly credited.

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: The paper does not introduce a new dataset or pretrained model asset. The main contributions are algorithmic and systems-level methods for decoding and inference kernels.

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: The paper does not involve crowdsourcing or human-subject experiments.

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: The paper does not involve human-subject research, so IRB approval or equivalent review is not applicable.

    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: LLMs were not used as an important, original, or non-standard component of the core research methodology. Any use of LLM tools for writing, editing, or formatting does not affect the scientific methodology, originality, or experimental results.

    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.