QUANTIFYING GPU ENERGY SAVINGS FROM LATENT COMMUNICATION IN MULTI-AGENT LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent-space communication replaces decoded text between agents with hidden states and KV-cache transfer. Although prior work reports speed and token reductions, its energy implications have not been measured directly. We compare single-agent, text-based multi-agent (TextMAS), and latent multi-agent (LatentMAS) inference on identical NVIDIA A100 hardware for Qwen3-8B and LLaMA 3.1-8B-Instruct on GSM8K. Relative to TextMAS, latent communication reduces net GPU energy by 70.7% for Qwen3-8B and 80.6% for LLaMA. Text and latent communication differ by only 0.3 and 2.5 percentage points, respectively; question-clustered confidence intervals include zero and do not establish equivalence. The saving does not come from a cheaper individual latent step: latent steps and decoded tokens take similar time per unit (38.5 vs. 37.1 ms). Instead, the reduction is primarily associated with replacing hundreds of decoded intermediate tokens with 40 bounded latent steps. The single-agent comparison is model-dependent: for Qwen3-8B, latent communication is slightly more accurate, uses 16.6% less energy, and runs 1.18× faster than the matched single-agent baseline, whereas for LLaMA the single-agent baseline is both more accurate and lower-energy than either multi-agent mode. A second Qwen3 benchmark shows the same qualitative energy and accuracy pattern. These results quantify a substantial GPU-energy reduction from latent communication in the single-stream setting, while highlighting reduced transparency as an important practical limitation.

1 INTRODUCTION

The deployment of multi-agent systems (MAS) built on large language models has grown rapidly, with frameworks such as chain-of-agents pipelines demonstrating improved accuracy on complex reasoning tasks through sequential collaboration among specialized agents (Zou et al., 2026; Yang et al., 2026). In the standard text-based approach, each agent decodes its output to natural language, which the next agent re-tokenizes and processes from scratch. This design is simple and interpretable but incurs substantial computational overhead: every agent boundary requires full vocabulary-sized softmax decoding followed by re-encoding.

Recent work has shown that agents can instead communicate through *latent representations*—passing last-layer hidden states or KV caches between agents without decoding to text (Zou et al., 2026; Du et al., 2026; Yang et al., 2026). These methods report large inference speedups and token reductions. However, all published results use proxy metrics (wall-clock time, token counts, bandwidth) rather than direct energy measurements, and none measures single-agent inference under the same protocol, so the natural question—how much energy does multi-agent collaboration cost in the first place, and how does the communication channel change that bill—has no answer in the literature.

Contributions. Prior latent-communication work reports only proxy metrics—speedups, token counts, bandwidth—and reports them as unqualified wins: no study measures energy, explains *why* the mechanism is cheaper, or checks whether its own advantage survives outside the exact regime it was measured in. We make five contributions:

1. **A controlled direct GPU-board-energy comparison** of single-agent, text-, and latent-communication multi-agent LLM inference on GSM8K for two model families. Latent

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

communication reduces net GPU energy by 70.7% (Qwen3-8B) and 80.6% (LLaMA 3.1-8B) relative to TextMAS; question-clustered accuracy intervals include zero for both models (Section 5).

2. **A mechanistic explanation for why latent communication is cheaper.** We show that a latent step takes similar time to a decoded text token per unit (38.5 vs. 37.1 ms) and argue that the main reduction comes from replacing long intermediate text generations with a bounded $m{=}40$ latent-step budget. For Qwen3, the solver remains the dominant decoded stage, while the intermediate agents account for most of the measured savings (Section 7).
3. **A prompt-matched single-agent comparison.** Using the solver's exact prompt for the single-agent baseline, we find that for Qwen3-8B, LatentMAS is slightly more accurate, 16.6% lower-energy, and 1.18$\times$ faster than the matched single-agent baseline, while TextMAS achieves a similar accuracy at much higher energy; for LLaMA the same protocol reverses this pattern, with single-agent both more accurate and lower-energy than either multi-agent mode (Section 6).
4. **Scope conditions for the efficiency result.** The measurements concern single-stream inference on one sequential topology; published batched-serving numbers from Zou et al. show that the relative speed ordering can change with serving configuration (Section 8). We also document an exploratory, protocol-specific harder-benchmark result as a limitation rather than treating the GSM8K result as universal (Appendix A).
5. **A direct analysis of the transparency cost.** Probed latent hidden states decode to near-degenerate tokens (99% top-1 probability); free-form KV-cache probing recovers only partial, role-specific summaries; and an analytically estimated text-first hybrid retains some transparency while recovering only part of latent mode's efficiency advantage (Section 8).

A secondary contribution is methodological: we control for scorer/truncation asymmetry and prompt wording, two confounds that can distort latent-versus-text comparisons (Section 4.2).

2 Related Work

Latent communication between LLM agents. Zou et al. (2026) introduced LatentMAS, a training-free framework where agents communicate via KV-cache transfer and latent thought generation, reporting large speedups and token reductions across 9 benchmarks (Qwen3, LLaMA 3). Yang et al. (2026) proposed RecursiveMAS with lightweight RecursiveLink modules (0.31% trainable parameters) for recursive latent loops across heterogeneous models. Du et al. (2026) introduced Interlat, a trainable adapter with latent-space compression achieving large inference speedups. Wenzel (2026) measured latent-channel *information* directly (bandwidth, probe accuracy), finding SAE-sparse channels retain most probe accuracy at large compression while text serialization is comparatively lossy. Wu & Zhu (2026) showed large payload compression via post-hoc sparse autoencoders with minimal accuracy impact.

These works do not measure energy in the comparison studied here or compare the communication modes with a single-agent baseline under the same protocol; direct LLM power measurement is otherwise limited to single-agent edge deployments (Paramanayakam et al., 2025). We add the energy axis for single-agent, text-, and latent-MAS communication on GSM8K; a mechanistic account of where the savings come from; and a focused analysis of the serving regime in which the comparison is made. We also report LLaMA 3.1-8B GSM8K results under LatentMAS-style latent communication, a benchmark not evaluated for LLaMA in Zou et al. (2026).

Latent reasoning in LLMs. Several works apply the same hidden-state-feedback primitive within a single model's own reasoning rather than between agents. Hao et al. (2024) (Coconut) *trains* a model to reason in continuous latent space—unlike LatentMAS, this requires fine-tuning rather than a training-free realignment operator. Lu et al. (2025) study a model *pretrained* for recurrent-depth latent computation and find scaling recurrent steps from 4 to 32 yields only modest gains, with explicit chain-of-thought still clearly ahead—evidence that more continuous latent steps do not straightforwardly substitute for text, even under more favorable conditions than a training-free retrofit like ours. Zhang et al. (2025) probe Coconut's latent tokens directly and find them structurally incoherent from the model's own perspective, corroborating our own near-degenerate probe-decoding result

(Section 8) despite a different mechanism (trained vs. training-free). Goyal et al. (2024) introduced "pause tokens" for extra computation without latent-space reasoning per se, and Pham et al. (2024) proposed multiagent debate through embeddings. None measure energy.

Energy efficiency in machine learning. Henderson et al. (2020) called for systematic energy/carbon reporting; tools like CarbonTracker (Anthony et al., 2020), CodeCarbon (Schmidt et al., 2021), and ML CO_2 Impact (Lacoste et al., 2019) estimate training energy, and Desislavov et al. (2023) analyzed inference energy trends, while Geissler & Lukowicz (2024) noted current tools capture only partial system energy. Paramanayakam et al. (2025) demonstrated edge-device power measurements for function calling—the closest precedent for direct LLM energy measurement, though at single-agent scale. Chen et al. (2024) proposed cost-reduction strategies for LLM APIs without measuring energy.

Multi-agent LLM systems. The MAS literature shows multi-agent collaboration improves accuracy through role specialization (Zou et al., 2026; Yang et al., 2026), sequential chains (Zhang et al., 2024), hierarchical topologies (Hong et al., 2023), mixture-of-agents (Wang et al., 2025), and frameworks like AutoGen (Wu et al., 2024). We adopt Zou et al. (2026)'s sequential 4-agent pipeline, additionally measuring the single-agent condition it is compared against and varying only the communication channel while measuring energy.

3 Background: Latent Communication in Sequential MAS

We briefly recap LatentMAS (Zou et al., 2026), the training-free framework our experiments instantiate (full formal treatment in Appendix B); readers already familiar can skip to Section 4.

Latent thought generation (Zou et al., 2026; Hao et al., 2024) replaces a transformer's decode-embed cycle—predict a token, re-embed it, feed it back in—with direct reuse of the last-layer hidden state as the next input embedding, skipping the softmax and embedding lookup entirely. Repeating this for m steps yields the agent's *latent thoughts* without ever materializing text; since a raw hidden state is not drawn from the token-embedding distribution, Zou et al. (2026) apply a training-free *linear alignment operator* before feeding it back in (identity plus norm rescaling, in our configuration).

Agents pass information as *KV cache*—accumulated attention key/value projections at every layer, for every token and latent step processed—rather than text. A receiving agent prepends this to its own cache, conditioning on everything upstream without re-running that computation or requiring the upstream agent to decode a token.

We adopt LatentMAS's *sequential* topology: planner $\rightarrow$ critic $\rightarrow$ refiner $\rightarrow$ solver, each output feeding the next. The planner proposes a plan without answering; the critic reviews it; the refiner incorporates feedback; the solver produces the final answer inside `\boxed{}`. In *text mode*, every agent decodes real text the next agent re-tokenizes; in *latent mode*, the first three agents generate m latent thoughts and pass KV caches forward, and only the solver decodes text. Cache transfer preserves the sender's computed latent context without decoding or re-encoding it; it does not imply that latent and text communication expose identical representations. We also evaluate *single-agent* (one model, one generation, no pipeline)—the baseline both MAS modes are measured against (Section 6), LatentMAS's "Single" column but without an energy measurement in the source paper. Zou et al. (2026) also define a *hierarchical* setting we do not evaluate.

4 Method

4.1 Experimental Setup

Communication modes. We compare the three conditions defined in Section 3—single agent, TextMAS, and LatentMAS—holding the underlying model fixed and varying only the communication channel. LatentMAS uses $m = 40$ latent steps, following Zou et al. (2026)'s ablation showing accuracy plateaus between 20–40 steps, with their recommended identity alignment and norm rescaling (no learned realignment matrix).

Models. Two 8B-parameter instruction-tuned models from different families: Qwen3-8B (Yang et al., 2025) and LLaMA 3.1-8B-Instruct (Grattafiori et al., 2024), both in bfloat16. Qwen3 uses its native thinking mode (`<think>` tag); LLaMA runs without, as it lacks native support.

Benchmark. We evaluate on GSM8K (Cobbe et al., 2021), scored by exact-match on the final numeric answer. Qwen3-8B uses the first 100 problems of the `test` split; LLaMA uses the first 200, reflecting how each model's runs were originally sized (Section 8)—the subset is identical across modes and seeds *within* a model, which is what the text-vs-latent comparison needs, but the two models' absolute accuracies are not on matched sample sizes. Each condition repeats over 3 seeds (42, 123, 456), varying only decoding stochasticity (temperature 0.6, top-p 0.95).

Hardware. All experiments run on Azure `Standard_NC24ads_A100_v4` VMs with a single NVIDIA A100 80GB PCIe GPU (300W TDP), 24 vCPUs, and 220GB RAM. Generations use `max_new_tokens` 2048.

4.2 Energy Measurement Protocol

We measure GPU board power via `nvidia-smi` at 1 Hz throughout each experiment:

1. **Idle baseline**: 60-second measurement with the model loaded but no inference, establishing baseline power draw ($\bar{P}_{\text{idle}} \approx 60$–$73$W depending on model and run).
2. **Continuous monitoring**: A background process polls `nvidia-smi` (power draw, utilization, memory, temperature) every second during the experiment.
3. **Phase markers**: Timestamps recorded at instance and agent boundaries for fine-grained attribution.
4. **Energy computation**: Trapezoidal integration of power over time: $E = \int P(t)\,dt$. Net energy subtracts the idle baseline: $E_{\text{net}} = E_{\text{total}} - \bar{P}_{\text{idle}} \cdot \Delta t$.

Total and per-instance energy are obtained by direct integration of the measured power trace. For the *per-agent* decomposition (Section 7) we attribute energy in proportion to wall-clock time within each agent's phase window, which is valid here because mean GPU power during active generation is nearly mode-invariant (Section 7). We use "time" and "energy" interchangeably only in this per-agent context.

Answer extraction. We score with the exact grader of the LatentMAS reference implementation (Zou et al., 2026): the predicted answer is the content of the last `\boxed{}` in the solver output (first number inside it, for GSM8K), normalized (trim, lowercase) and string-matched against gold. Generations without a `\boxed{}` answer are scored incorrect.

Answer-length asymmetry and scorer validation. Latent-mode solvers tend to produce longer final-answer generations than text-mode solvers, so truncation or answer-extraction errors could affect the two modes asymmetrically. We therefore retain complete, untruncated predictions and independently verify all reported accuracy counts against the saved outputs.

Statistical analysis. Text-vs-latent comparisons are paired by benchmark question. Because each question recurs across three seeds, we treat the question—not each question-seed outcome—as the independent sampling unit. We report 95% CIs on ΔAcc from a 20,000-resample cluster bootstrap that resamples question IDs and retains all three seed outcomes within each sampled cluster. These intervals do not establish equivalence. Energy reduction is 1 minus the ratio of the seed-mean net energies, with a t-based 95% CI ($\mathrm{df}=2$)—wide and indicative only with 3 seeds. We retained only aggregate per-seed scores for single-agent (Section 8), so those comparisons use seed-level means $\pm$ SD. Full derivations in Appendix D.

Measurement uncertainty. `nvidia-smi` reports GPU board power only, at roughly $\pm 5\%$ accuracy. We hold the GPU model and VM configuration fixed, but do not assume that calibration error cancels across cloud sessions; absolute Wh and reduction percentages should therefore be read with this instrumentation uncertainty in mind. Integration error from 1 Hz polling is negligible for run

Table 1: **Main results on GSM8K.** Accuracy (%) and net GPU energy (Wh) for single-agent, text-based, and latent multi-agent communication. Qwen3-8B: 3 seeds × 100 instances per condition; LLaMA 3.1-8B: 3 seeds × 200 instances per condition (Section 8). Both models' Single columns use the prompt-matched baseline that controls for a prompt-wording confound, detailed in Section 6 (†); this is the single-agent baseline used consistently throughout the paper. Energy reduction is computed from mean net energy, text→latent; question-clustered accuracy CIs are given below.

Model	**Accuracy (%)**			**Net Energy (Wh)**			**Energy**
	Single	Text	Latent	Single	Text	Latent	**Reduction**
Qwen3-8B	94.3±0.6†	95.7±2.5	96.0±2.0	69.2±0.7†	196.7±5.7	57.7±1.8	**–70.7%**
LLaMA-3.1-8B	81.2±2.0†	72.8±1.3	75.3±0.8	37.8±2.8†	262.5±13.4	50.9±1.3	**–80.6%**

totals; the per-agent decomposition instead uses sub-millisecond timestamps, unaffected by polling rate. The power profile (Figure 3) is a separate, higher-resolution diagnostic run used qualitatively only. Full detail in Appendix D.

5 Results

5.1 Main Results

Table 1 summarizes accuracy and energy for both models on GSM8K. In both conditions, latent communication reduces net GPU energy substantially relative to text mode (70.7% for Qwen3-8B, 80.6% for LLaMA 3.1-8B); the question-clustered accuracy intervals include zero.

Qwen3-8B. Text and latent accuracy is 95.7% vs. 96.0% (ΔAcc $+0.3$ pp, question-clustered 95% CI $[-2.3, +3.0]$), while both exceed the single-agent baseline descriptively (94.3%; prompt-matched, Section 6). The interval permits a small accuracy disadvantage for latent communication, so this result does not establish equivalence. Energy reduction text→latent is 70.7% (196.7→57.7 Wh; 95% CI $[68.9, 72.5]$), with low seed variance.

LLaMA 3.1-8B. Latent is numerically higher than text in every seed (75.3% vs. 72.8%; ΔAcc $+2.5$ pp, question-clustered 95% CI $[-2.0, +6.8]$). The interval is compatible with either a modest latent advantage or disadvantage, so we do not claim equivalence or maintained accuracy. Energy reduction is 80.6% (95% CI $[79.3, 81.8]$), the larger of the two conditions. Unlike Qwen3-8B, LLaMA's prompt-matched single-agent baseline (81.2±2.0%, 37.8±2.8 Wh) is both more accurate and lower-energy than either multi-agent mode here—a reversal discussed in Section 6. Zou et al. (2026) did not evaluate LLaMA on GSM8K, so these are, to our knowledge, the first reported LLaMA 3.1-8B GSM8K results under this framework.

Energy per correct answer. Latent communication is more energy-efficient per correct answer than text mode in both conditions, and than single-agent inference for Qwen3-8B (0.60 vs. 2.06 vs. 0.73 Wh/correct; full table, including per-instance timing and speedup, in Appendix E).

5.2 A Second Benchmark: ARC-Easy

GSM8K carries every result above, so we ran a second, structurally different benchmark (multiple-choice science QA) on Qwen3-8B: 100 instances, 3 seeds, identical protocol.

Table 2: **ARC-Easy**, Qwen3-8B, 3 seeds × 100 instances per condition.

	Single	Text	Latent
Accuracy (%)	95.7±1.2	99.0±1.0	98.0±1.0
Net Energy (Wh)	65.3±2.3	191.2±9.8	46.7±1.3

Latent is 1.0 pplower than text in the seed means (98.0% vs. 99.0%); only aggregate per-seed ARC-Easy scores were retained, so we make no inferential claim about this difference. Energy drops

Table 3: **Prompt-matched comparison on GSM8K** for Qwen3-8B. Single-agent uses the same solver prompt as TextMAS and LatentMAS, with no upstream context. Values are means over three seeds.

Mode	Acc. (%)	ΔAcc	Energy (Wh)	ΔEnergy	Wh / Δpp
Single	94.3	–	69.2	–	–
Text	95.7	+1.3 pp	196.7	+184.3%	95.2
Latent	96.0	+1.7 pp	57.7	**–16.6%**	**saves energy**

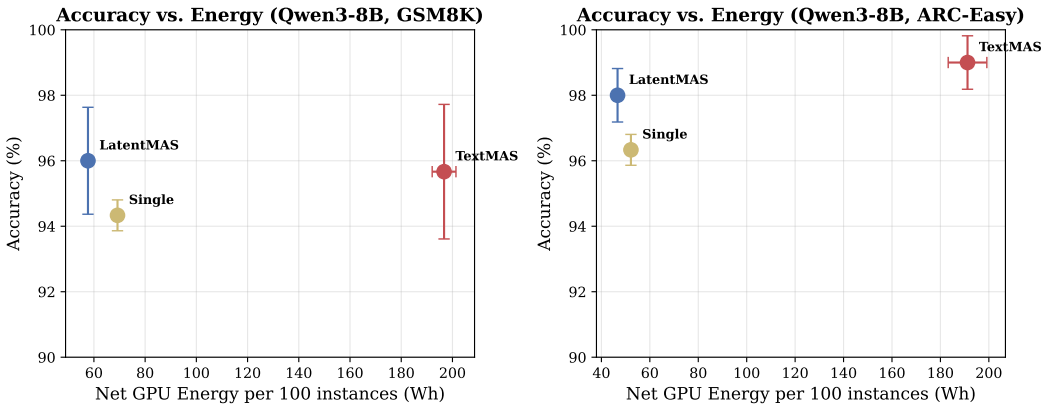

Figure 1: **Accuracy vs. net energy**, Qwen3-8B, 3 seeds each, on GSM8K (left) and ARC-Easy (right). Single-agent uses a prompt-matched baseline on both benchmarks (Section 6). In these experiments, LatentMAS is higher-accuracy and lower-energy than single-agent; TextMAS matches or exceeds LatentMAS's accuracy at roughly 3–4× the energy.

75.6% (95% CI $[74.0, 77.1]$)—close to GSM8K's 70.7%. The result provides a second Qwen3 benchmark with the same qualitative efficiency pattern. Both modes are close to Zou et al. (2026)'s published Qwen3-8B ARC-Easy numbers (95.6/99.1/98.8%), providing an independent cross-check of our pipeline.

6 Prompt-Matched Single-Agent Comparison

To compare against single-agent inference fairly, we measure both models with the solver's exact prompt and no upstream context. This prompt-matched baseline is used throughout the paper, including Table 1.

Table 3 and Figure 1 show that LatentMAS is descriptively higher-accuracy, lower-energy, and faster than the prompt-matched single-agent baseline: $+1.7$ pp accuracy, 16.6% less energy, and $1.18\times$ the speed (29.5 vs. 34.7 s/instance). TextMAS has a similar accuracy but uses substantially more energy. These single-agent comparisons are descriptive; the retained single-agent results are aggregate per-seed scores rather than paired per-instance outcomes. For LLaMA, the same matched-prompt, energy-measured baseline ($81.2_{\pm 2.0}$%, $37.8_{\pm 2.8}$ Wh) reverses this pattern: single-agent is both more accurate (-8.4 pp Text, -5.9 pp Latent relative to single) and lower-energy (Text $+594$%, Latent $+34.7$% relative to single) than either multi-agent mode. We return to a plausible explanation for this reversal—a likely system-prompt mismatch with Zou et al. (2026)'s own published LLaMA setup—in Appendix A.2.

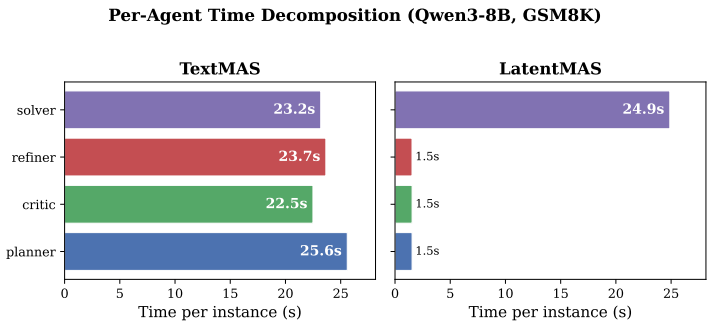

Figure 2: **Per-agent time decomposition** for Qwen3-8B on GSM8K (mean over 300 pooled instances, 3 seeds). Text mode: all four agents contribute 22–26s each. Latent mode: three intermediate agents take ∼1.5s each; only the solver matches text mode. Energy savings come almost entirely from the intermediate agents.

7 Analysis

7.1 Per-Agent Energy Decomposition

To understand *where* energy is saved, we decompose per-instance computation time by agent role (Figure 2), pooled across all 3 seeds. For Qwen3-8B, in text mode all four agents contribute comparable time (22–26 s each). In latent mode, the three intermediate agents (planner, critic, refiner) each consume only ∼1.5 s compared to ∼22–26 s in text mode—a ∼94% reduction—while the solver's time is essentially unchanged (23.2 s text vs. 24.9 s latent).

Why the intermediate agents are cheap: step count, not per-step cost. LatentMAS issues four agent invocations where a single agent issues one, so a cheaper *step* is not the obvious explanation for why more invocations cost less. With KV caching, every step—latent or textual—processes exactly one new position and attends over the accumulated cache, so the per-step cost should be similar either way; we verified this directly: text-mode decoding costs 37.1 ms/token and latent-step generation costs 38.5 ms/step, with no meaningful difference in this measurement. The large intermediate-agent speedup is therefore primarily a step-count effect: $m = 40$ fixed latent steps substitute for the 605–689 decoded tokens text mode needs, consistent with Section 3's expressiveness argument (Zou et al., 2026). The solver still decodes real text; its cost is nearly mode-invariant for Qwen3 but also decreases in latent mode for LLaMA, so step count is the main mechanism rather than an exhaustive explanation of every cross-model difference.

For LLaMA 3.1-8B the same pattern holds, with one difference worth noting rather than glossing over: intermediate-agent time drops just as sharply (planner 6.9→1.1 s, critic 21.0→1.1 s, refiner 10.1→1.1 s), but the solver is *also* faster in latent mode (9.6 vs. 6.5 s)—unlike Qwen3-8B, where the solver's cost is mode-invariant. We have no mechanistic explanation for this cross-model difference and flag it as an open question rather than force it into the framing that holds for Qwen3-8B. Mean GPU power is nearly identical between modes for both models (134.2 vs. 133.7 W for Qwen3-8B; 159.2 vs. 158.2 W for LLaMA), so energy reduction maps almost directly onto time reduction throughout.

7.2 Temporal Power Profiles

A qualitative, illustrative diagnostic (same GSM8K instance, Single/TextMAS/LatentMAS, deliberate inter-instance pauses so the trace shows a clean idle-active-idle cycle; not used for any reported number) confirms the same story visually: Single and TextMAS occupy a similar power range (∼140–160W) but differ enormously in duration, and LatentMAS's active period is barely visible at that scale against TextMAS's much longer one.

7.3 Why Energy Reduction Differs Between Models

The energy reduction is largely explained by the fraction of text-mode computation spent in the intermediate agents. If intermediate-agent time falls by 94% while solver time remains unchanged, the expected reduction is $R_{\text{energy}} \approx 0.94(1 - f_s)$, assuming mode-invariant GPU power. For Qwen3-8B, where the solver is nearly mode-invariant and f_s=0.244, this predicts 71.1%, close to the measured 70.7%. For LLaMA, f_s=0.201 gives a lower first-order prediction of 75.1% versus the measured 80.6%, because the solver is also faster in latent mode. Thus, the expression is a useful approximation for the Qwen3 result, not a complete cross-model model.

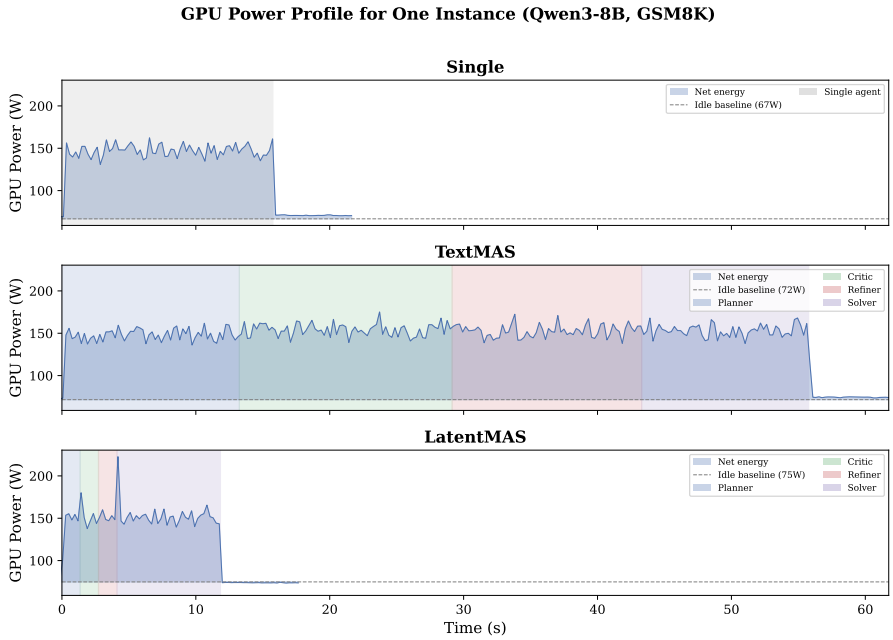

Figure 3: **GPU power draw over time** for the *same* GSM8K instance in all three panels, from a dedicated diagnostic run with a deliberate 15 s inter-instance pause and 5 Hz polling (Qwen3-8B; distinct from the 1 Hz back-to-back main measurements). All three panels share the same x-axis scale. Single (top) and TextMAS (middle) occupy a similar power range but very different durations; shaded bands in TextMAS mark agent boundaries. LatentMAS (bottom) is barely visible at this scale relative to TextMAS. The shaded area under each trace represents net energy (above idle baseline).

8 Discussion

Practical implications. A single Qwen3-8B GSM8K experiment (100 instances) consumes ∼197 Wh in text mode vs. ∼58 Wh in latent mode. These batch-size-1, single-GPU numbers are not a deployment projection; production serving uses continuous batching (e.g., vLLM (Kwon et al., 2023)), addressed in the next two paragraphs.

Memory overhead and batch-capacity trade-off. Latent communication accumulates KV cache across intermediate agents rather than discarding it at each text boundary: peak GPU memory for Qwen3-8B on GSM8K is 19.11 GiB (latent) vs. 16.74 GiB (text), a +14% per-request overhead—inconsequential at batch size 1, but under batched serving, KV-cache memory becomes the binding constraint on achievable concurrency. Since energy savings track time savings almost exactly in our single-stream data (Section 7), latent communication's advantage could narrow once text mode compensates through higher batch parallelism; we did not run batched experiments and cannot locate a crossover point. This is not a purely hypothetical concern, as the next paragraph shows.

Published batching results qualify the comparison. With Zou et al. (2026)'s default batch size 20, single-agent is faster than LatentMAS on GSM8K at Qwen3-8B (449 vs. 543s) and 14B (536 vs.

1952s), while LatentMAS wins at 4B (375 vs. 469s). Their stage-synchronized batched path pads items to the longest sequence, unlike our batch-size-1 path. Thus, our speed and energy advantage over single-agent is specific to the unbatched regime; we do not project it to serving workloads.

Energy–transparency trade-off. Latent communication eliminates text decoding at three of four agent boundaries in our pipeline, saving energy at the cost of an inspectable intermediate trace. In text mode, an operator can examine the planner's strategy, the critic's objections, and the refiner's corrections before the solver answers; in latent mode, these states remain in the KV cache and cannot be assigned to individually observable reasoning steps. This is distinct from embedding-space overlap: latent thoughts may occupy the same region as valid token embeddings (Zou et al., 2026), but that does not establish that individual states are interpretable. We evaluated two probing methods on Qwen3-8B (10 GSM8K and 5 GPQA-Diamond instances; details in Appendix C).

Per-step argmax decoding through the LM head is near-degenerate: the planner repeatedly selects the same paragraph-break token with mean top-1 probability 0.99. Free-form KV-cache probing recovers partial, role-specific summaries, but the text before the model's `</think>` tag is often nearly identical across probes, while role differences appear mainly in the post-think summary. A text-first hybrid would preserve more of the trace, but we did not implement it; our 43.6% time-reduction estimate is analytical, compared with 68.9% for full latent mode.

These probes therefore recover limited legibility rather than an inspectable record of intermediate reasoning. In settings that require auditing why a system reached an answer, practitioners must weigh the latent mode's efficiency against the loss of directly attributable intermediate evidence.

Scope of the result. Our results are positive, but not a general recommendation. The exploratory GPQA-Diamond result under our protocol underscores that task and evaluation conditions matter; transparency, not just energy, is often the binding constraint in regulated settings; and the speed/energy edge may change under batching (paragraph above). The method also requires direct access to hidden states and KV caches, which is incompatible with hosted inference APIs, mixed-vendor pipelines, and token-level tool-calling frameworks.

Limitations and future work. Our measurements capture GPU board power only; full-system energy would yield different absolute numbers but similar relative comparisons (Henderson et al., 2020; Anthony et al., 2020). All runs use HuggingFace Transformers at batch size 1, one 4-agent sequential pipeline (other topologies (Yang et al., 2026) may differ), and a fixed latent step count ($m = 40$) whose accuracy-vs-energy trade-off we did not verify empirically. Beyond GSM8K, our only other Qwen3-8B benchmark is ARC-Easy (Section 5.2), one confirmatory point; the GPQA-Diamond comparison is exploratory and protocol-specific (Appendix A). Our two models use different subset sizes (100 for Qwen3-8B, 200 for LLaMA), which does not bias the within-model comparison but leaves point estimates unequally precise. We retained only aggregate per-seed scores for both models' single-agent runs, hence Table 3's descriptive statistics rather than a paired test. The simultaneous accuracy, speed, and energy result relative to single-agent is measured on one model and one benchmark and does not generalize to LLaMA, where the comparison reverses (Section 6); whether Qwen3's advantage depends on its thinking mode, solution-length distribution, or $m = 40$ is open.

9 Conclusion

We directly measure GPU energy for single-agent, TextMAS, and LatentMAS inference. Relative to TextMAS on GSM8K, latent communication reduces energy by 70.7% for Qwen3-8B and 80.6% for LLaMA 3.1-8B; question-clustered accuracy intervals include zero. Similar per-step costs indicate that the savings primarily come from replacing hundreds of decoded tokens with 40 latent steps. ARC-Easy shows the same energy pattern with a descriptive 1.0 pplatent accuracy disadvantage. For Qwen3, LatentMAS is slightly more accurate, faster, and lower-energy than single-agent; for LLaMA, single-agent is more accurate and lower-energy than both MAS modes. These results are limited to single-stream inference with direct KV-cache access, and latent communication reduces transparency; broader tasks and serving regimes remain future work.

AI USE STATEMENT

In this work, we used generative AI coding and research assistants (Claude by Anthropic via Claude Code, and GitHub Copilot) to implement and debug the experimental pipeline; orchestrate experiments; diagnose a prediction-truncation and scorer bug; write statistical-analysis and figure-generation code; audit results; and draft and revise manuscript text. These assistants were not used to fabricate or alter model outputs, telemetry, or reported measurements: the results derive from evaluated model inference and `nvidia-smi` telemetry on GPU hardware. We reviewed all AI-assisted code, analyses, and text, directed the experimental design and scope, and corrected AI-proposed explanations that did not hold up under scrutiny. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 4 documents the models, benchmark subsets, decoding configuration, hardware, scorer, and energy protocol; Appendix D provides the statistical and measurement details. The remaining appendices report the exploratory negative result, formal mechanism, and transparency probes. The supplementary material includes anonymized experiment code and configuration, selected raw predictions and telemetry summaries, and a script that independently reproduces the reported aggregates with the current scorer. We intend to release these materials publicly after the review process.

REFERENCES

Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. CarbonTracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051*, 2020.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

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

Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27106–27129. Association for Computational Linguistics, 2026.

Daniel Geissler and Paul Lukowicz. Leveraging hybrid intelligence towards sustainable and energy-efficient machine learning. In *Companion Proceedings of the 2024 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2024.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, pp. 27896–27923, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, et al. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Peter Henderson, Jieru Hu, Joshua Romoff, et al. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

Sirui Hong, Mingchen Zhuge, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. 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*, pp. 611–626, 2023.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *arXiv preprint arXiv:2507.02199*, 2025.

Varatheepan Paramanayakam, Andreas Karatzas, Iraklis Anagnostopoulos, and Dimitrios Stamoulis. Less is more: Optimizing function calling for LLM execution on edge devices. In *Design, Automation and Test in Europe Conference*, 2025.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. In *International Conference on Learning Representations*, volume 2024, pp. 51899–51928, 2024.

Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler, and Sasha Luccioni. CodeCarbon: Estimate and track carbon emissions from machine learning computing. Zenodo, 2021. URL `https://doi.org/10.5281/zenodo.4658424`.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, 2025.

Markus Wenzel. Latent communication between language model agents: Channels, alignment, and the limits of text. *arXiv preprint arXiv:2607.14103*, 2026.

Di Wu and Xiaohui Zhu. Post-hoc sparse coding of latent communication between vision-language model agents. *arXiv preprint arXiv:2608.10198*, 2026.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Xiyuan Yang, Jiaru Zou, Rui Pan, et al. Recursive multi-agent systems. *arXiv preprint arXiv:2604.25917*, 2026.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, et al. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024.

Yuyi Zhang, Boyu Tang, Tianjie Ju, Sufeng Duan, and Gongshen Liu. Do latent tokens think? a causal and adversarial analysis of chain-of-continuous-thought. *arXiv preprint arXiv:2512.21711*, 2025.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026.

A CONDITIONS WHERE LATENT COMMUNICATION UNDERPERFORMS

We evaluated latent communication on one additional benchmark under a protocol that differs from the published LatentMAS GPQA-Diamond evaluation. We include the result as an exploratory, protocol-specific limitation rather than presenting it as a replication or a general claim about latent communication.

A.1 EXPLORATORY GPQA-DIAMOND RESULT UNDER OUR PROTOCOL

Table 4 shows our GPQA-Diamond results (100 instances, 3 seeds, `max_new_tokens`=4096), a harder graduate-level science benchmark. The published LatentMAS evaluation used 198 questions and an 8,192-token budget, so the results are not directly comparable.

Table 4: **Exploratory GPQA-Diamond accuracy (%).** Results are not directly comparable to the published LatentMAS evaluation.

Model	Text	Latent	ΔAcc
Qwen3-8B	52.3±1.5	39.7±0.6	**–12.7 pp**
LLaMA-3.1-8B	28.7±4.7	25.0±2.6	**–3.7 pp**

For Qwen3-8B, this is a large, consistent regression across all 3 seeds. We traced a likely mechanism: 31–38 of every 100 latent-mode solver generations never emit `\boxed{}` at all, apparently still reasoning when they exhaust the 4096-token budget, versus 5–6/100 for text mode. GPQA questions are longer and harder than GSM8K's; the latent solver, receiving compressed KV-cache context rather than a text summary it can lean on, appears to need *more* room to work than we gave it, not less—consistent with the answer-length asymmetry we describe in Section 4.2. We consider this a plausible but unconfirmed diagnosis (we did not rerun with a larger budget to verify it) rather than a settled finding, which is why we describe it as an open limitation in Section 8 rather than a result.

A.2 LLAMA: AN INVERSION ACROSS THREE BENCHMARKS, AND A LIKELY EXPLANATION

On two benchmarks where Zou et al. (2026) also report LLaMA-3.1-8B-Instruct results under their Sequential MAS setting, our Single and TextMAS accuracies are consistently higher than their published numbers, while our LatentMAS accuracy closely matches theirs (Table 5).

Table 5: **Our LLaMA-3.1-8B-Instruct results vs. Zou et al. (2026)'s own published numbers** (their Sequential MAS setting). Accuracy (%); our values are mean ± SD over 3 seeds, 100 (ARC-Easy) or 200 (MedQA) instances/seed, unmatched (aggregate) single-agent prompt, mirroring their own evaluation protocol.

	Zou et al. (2026) (published)			Ours		
Benchmark	Single	Text	Latent	Single	Text	Latent
ARC-Easy	85.7	89.6	91.2	93.0±1.7	95.3±1.2	90.3±2.1
MedQA	59.0	65.0	65.3	68.5±3.1	68.3±0.9	64.3±2.5

On both benchmarks, our Single and Text accuracies exceed Zou et al. (2026)'s published numbers by 3–10 pp, while our Latent accuracy is within 1–2 pp of theirs. We looked for a methodological explanation and found one candidate in Zou et al. (2026)'s own released prompt appendix: the system prompt hardcoded for every agent role, across all evaluated models, is "You are Qwen, created by Alibaba Cloud. You are a helpful assistant."—plausibly reused unmodified when they evaluated LLaMA, despite it being the wrong model identity. Our pipeline uses a model-aware system prompt (`"You are a helpful assistant."` for LLaMA, with no false identity claim). If an incorrect identity claim depresses LLaMA's instruction-following more in conditions that decode directly to free text (Single, Text) than in the latent condition—where the agent works from a fixed KV-cache budget rather than following an explicit textual system-prompt instruction at each step—this would

produce exactly the pattern we observe: elevated Single/Text, matched Latent. Our new prompt-matched, energy-measured single-agent GSM8K result (Section 6), where LLaMA's single-agent accuracy is unusually high relative to both multi-agent modes, is consistent with this story, though it uses our own solver prompt rather than Zou et al. (2026)'s and so is not a like-for-like replication of the comparison above.

This is a well-motivated hypothesis, not a confirmed fact: we did not have access to Zou et al. (2026)'s original LLaMA runs, our benchmarks and exact evaluation splits may differ in ways we have not fully controlled for, and other explanations (e.g. decoding hyperparameters, prompt formatting details, subset composition) remain possible. Testing it directly—rerunning our pipeline on LLaMA with Zou et al. (2026)'s exact system prompt—is future work we intend to pursue.

B Latent Communication Mechanism: Full Formal Treatment

This appendix gives the full derivation summarized in Section 3.

Auto-regressive latent thought generation. A standard transformer f_θ generates text auto-regressively: given input embeddings $E = [e_1, \ldots, e_t]$ (from the input token embedding layer W_{in}), it computes last-layer hidden states $H = [h_1, \ldots, h_t]$ and predicts the next token as $\mathrm{softmax}(h_t W_{\text{out}})$, where W_{out} is the LM head. The generated token is then re-embedded and appended to the input for the next step.

Latent thought generation (Zou et al., 2026; Hao et al., 2024) replaces this decode-embed cycle with direct reuse of the hidden state: instead of decoding h_t to a token and re-embedding it, the model feeds h_t itself back in as the next input, bypassing the vocabulary-sized softmax and the embedding lookup entirely. Repeating this for m steps yields a sequence of hidden states $H = [h_{t+1}, \ldots, h_{t+m}]$—the agent's *latent thoughts*—without ever materializing text.

Feeding a last-layer hidden state directly into an input slot risks distribution mismatch: h_t is drawn from the model's output space, not the token-embedding space the model was trained to consume. Zou et al. (2026) address this with a training-free *linear alignment operator*: a projection $W_a \in \mathbb{R}^{d_h \times d_h}$ such that $e = hW_a$ maps a hidden state back toward the valid input-embedding distribution, computed once per run and reused for every latent step at negligible cost. Our implementation follows their default configuration: the identity matrix for W_a, combined with norm rescaling of each aligned vector to the mean input-embedding norm (Section 4).

KV cache as inter-agent working memory. In text-based MAS, an agent's contribution to the pipeline is its decoded text, appended to the next agent's prompt and re-encoded from scratch. LatentMAS instead transfers the agent's full *KV cache*—the accumulated key/value projections at every transformer layer, for every token and latent step the agent has processed—directly to the next agent, which prepends it to its own cache before generating. The next agent can therefore condition directly on the sender's internal computation without re-running it or requiring decoded intermediate text. This is not "lossless" relative to text: the channels expose different representations, and our experiments do not establish informational equivalence between them.

The sequential MAS topology. We adopt LatentMAS's *sequential* MAS setting (Zou et al., 2026): four LLM agents in a fixed pipeline,

$$\text{planner} \rightarrow \text{critic} \rightarrow \text{refiner} \rightarrow \text{solver},$$

where each agent's output (text or KV cache, depending on mode) feeds into the next. The planner proposes a step-by-step approach to the question without answering it; the critic reviews that plan and produces feedback; the refiner incorporates the feedback into an improved plan; the solver receives everything upstream and produces the final answer inside `\boxed{}`. In *text mode*, every agent decodes up to `max_new_tokens` and the next agent re-tokenizes that text as a fresh prompt. In *latent mode*, the first three agents generate m latent thoughts each and pass KV caches forward; only the solver decodes text, so a full pipeline run produces exactly one text generation regardless of how many upstream agents contributed.

C TRANSPARENCY MITIGATION: ADDITIONAL DETAIL

This appendix gives the full detail summarized in Section 8.

Per-step argmax decoding, full numbers. Applying the LM head directly to each latent hidden state, across both GSM8K and GPQA-Diamond instances: the planner's argmax token has mean top-1 probability 0.99 and mean entropy ratio 0.003–0.005 (entropy relative to a uniform distribution over the vocabulary), and the single argmax token is the same paragraph-break token at essentially every one of the 40 steps. The critic and refiner are somewhat less degenerate (top-1 probability 0.74–0.83, 1.2–1.9 unique tokens across 40 steps), but still overwhelmingly emit structural delimiters (a paragraph break or the closing `</think>` tag) rather than content. This is consistent with thinking-mode instruction tuning teaching the model that the next *visible* token after a latent step should be a delimiter, not a word; we did not test a non-thinking-mode model with this probe, so we cannot isolate thinking-mode tuning from other differences between models.

Free-form KV-cache probing, methodology and examples. We insert a summarization prompt after each latent agent's accumulated KV cache and generate $\sim$200 tokens at temperature 0.6. The portion of the generated text before the model's own `</think>` closing tag is frequently near-identical across the planner, critic, and refiner probes of the *same* instance (byte-identical in 2 of 10 GSM8K instances despite temperature-0.6 sampling; a shared multi-sentence prefix in the rest). The text *after* `</think>` is genuinely role-differentiated: in a representative case, the critic probe continues "The reasoning provided is logically sound and mathematically accurate..." while the refiner probe continues "The solution is already accurate... but to further refine it, we can present the reasoning in a more formal... **Refined Solution:**".

Hybrid pipeline, arithmetic. A pipeline keeping the planner in text mode while running the critic, refiner, and solver in latent mode would cost, by summing the measured per-agent times in Figure 2: planner-in-text (25.6 s) + critic-in-latent (1.5 s) + refiner-in-latent (1.5 s) + solver (24.9 s) $\approx$ 53.6 s/instance, a 43.6% time reduction versus full text mode (94.9 s), compared to full latent mode's 68.9% time reduction. This projection assumes costs from separately measured all-text and all-latent runs compose additively; we did not test a mixed-mode pipeline, so cross-mode interactions may change the result.

D STATISTICAL METHODS AND MEASUREMENT UNCERTAINTY: FULL DETAIL

This appendix gives the full detail summarized in Section 4.2.

Statistical analysis. Text and latent modes use the same questions and seeds, but repeated outcomes for one question are not independent. We therefore use question ID as the sampling unit: each of 20,000 bootstrap draws resamples question IDs with replacement and retains all three paired seed outcomes for every sampled question. Percentile 95% intervals on latent-minus-text accuracy are $[-2.3, +3.0]$ ppfor Qwen3-8B and $[-2.0, +6.8]$ ppfor LLaMA. The reported energy reduction is 1 minus the ratio of the seed-mean net energies; its 95% CI is a t-based interval ($\mathrm{df} = 2$) over the three per-seed reductions $1 - E_{\text{net,latent}}/E_{\text{net,text}}$. With only 3 seeds these intervals are indicative only. We retained only aggregate per-seed scores for single-agent and ARC-Easy, so those accuracy comparisons are descriptive.

Measurement uncertainty. `nvidia-smi` reports GPU board power only—excluding CPU, DRAM, network, PSU efficiency losses, and cooling—and its `power.draw` reading is specified to roughly $\pm$5% of value. We use the same GPU model and VM configuration throughout, but cloud sessions may use different physical boards, so we do not assume that calibration offsets cancel; absolute Wh and reduction percentages retain instrumentation uncertainty. Integration error from 1 Hz polling is negligible for run totals: each run contributes thousands of samples, and trapezoidal energy matches mean-power $\times$ duration to within 0.1% in every run. The per-agent decomposition (Figure 2) is computed from sub-millisecond wall-clock timestamps recorded around each agent invocation, averaged over 100 instances—*not* by slicing the 1 Hz power trace—so it is unaffected by

the sampling rate; its only assumption is that mean GPU power is mode-invariant, which we verify directly (Section 7). The power profile (Figure 3) is from a separate, higher-resolution (5 Hz) diagnostic run with deliberate pauses between instances, purely for illustrative clarity (Section 7); it is a raw trace and we use it qualitatively, not for any reported energy or accuracy number. Restricting measurement to GPU board power is standard practice for GPU-centric ML workloads (Geissler & Lukowicz, 2024; Paramanayakam et al., 2025; Henderson et al., 2020) and is sufficient for relative comparisons between communication modes on identical hardware.

E Energy-Efficiency Table

Table 6, referenced from Section 5, gives the full net-energy-per-correct-answer and per-instance timing breakdown.

Table 6: **Energy efficiency.** Net energy per correct answer (Wh) and inference time (per-instance, seconds). Both models' Single columns use the prompt-matched baseline (Section 6), consistent with Table 1.

Model	**Wh / Correct**			**Time (s/instance)**			**Speedup**
	Single	Text	Latent	Single	Text	Latent	(text→latent)
Qwen3-8B	0.73	2.06	0.60	34.7	94.9	29.5	3.22×
LLaMA-3.1-8B	0.23	1.80	0.34	6.9	47.7	9.7	4.90×