EdgeReasoning: Characterizing Reasoning LLM Deployment on Edge GPUs

Benjamin Kubwimana
NVIDIA
bkubwimana@nvidia.com

Qijing Huang
NVIDIA
jennyhuang@nvidia.com

***Abstract*—Edge intelligence paradigm is increasingly demanded by the emerging autonomous systems, such as robotics. Beyond ensuring privacy-preserving operation and resilience in connectivity-limited environments, edge deployment offers significant energy and cost advantages over cloud-based solutions. However, deploying large language models (LLMs) for reasoning tasks on edge GPUs faces critical challenges from strict latency constraints and limited computational resources.**

To navigate these constraints, developers must balance multiple design factors—choosing reasoning versus non-reasoning architectures, selecting appropriate model sizes, allocating token budgets, and applying test-time scaling strategies—to meet target latency and optimize accuracy. Yet guidance on optimal combinations of these variables remains scarce.

In this work, we present EdgeReasoning, a comprehensive study characterizing the deployment of reasoning LLMs on edge GPUs. We systematically quantify latency-accuracy tradeoffs across various LLM architectures and model sizes. We systematically evaluate prompt-based and model-tuning-based techniques for reducing reasoning token length while maintaining performance quality. We further profile test-time scaling methods with varying degrees of parallelism to maximize accuracy under strict latency budgets. Through these analyses, EdgeReasoning maps the Pareto frontier of achievable accuracy-latency configurations, offering systematic guidance for optimal edge deployment of reasoning LLMs.

***Index Terms*—Large Language Models, Inference, Prompt Engineering, SoC, Hardware, Energy**

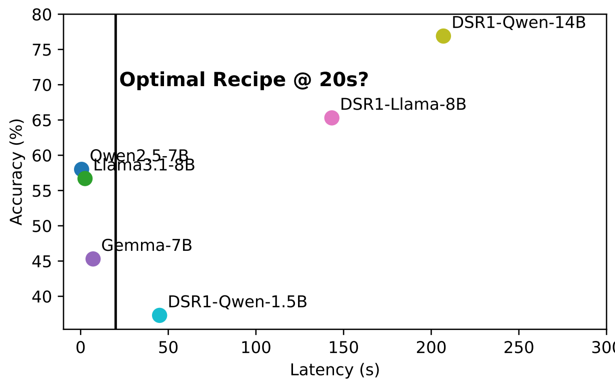


Fig. 1: Discrete accuracy-latency tradeoffs fail to capture continuous operational requirements of real-world systems like assistive robots.

I. Introduction

The rapid advancement of autonomous systems—from robotics and drones to self-driving vehicles—has created an unprecedented demand for intelligent decision-making and reasoning capabilities at the edge [44]. Consider personal assistive humanoid robots: when a user requests "Can you help me prepare dinner within 5 minutes?", the robot must perform real-time planning and execution under strict latency constraints. Such scenarios reveal a critical tension - tasks with generous latency budgets (e.g., "Plan my weekly schedule") benefit from larger models with longer reasoning chains for optimal planning, while latency-sensitive tasks (e.g., "Avoid that obstacle now!") demand smaller models that sacrifice optimality for speed.

This operational reality presents fundamental challenges for edge deployment of reasoning models. First, the autoregressive nature of LLMs creates highly variable token generation times, making latency hard to control—potentially resulting in missed deadlines or no responses. Second, deploying reasoning-capable models incurs substantial latency dominated by decoding processes, particularly problematic for real-time systems. Third, the discrete accuracy-latency tradeoffs shown in Fig. 1 fail to capture the continuous spectrum of real-world requirements. These challenges necessitate: (1) precise token length control to meet latency constraints, (2) hardware-aware functions mapping latency budgets to maximum decodable tokens, and (3) continuous optimization across the latency-accuracy frontier.

While cloud-based large language models (LLMs) [7], [27], [31] have demonstrated remarkable reasoning abilities, the edge intelligence paradigm offers compelling advantages that extend far beyond privacy preservation and connectivity resilience. Most notably, edge deployment presents transformative cost efficiencies that alter the economics of AI-powered autonomous systems. Recent developments in lightweight reasoning models [20], [21] have achieved comparable accuracy to larger commercial models at two orders of magnitude lower cost, as demonstrated in Section III-B.

However, realizing these cost advantages while maintaining reasoning quality presents significant technical challenges. Edge deployment imposes latency constraints and computational limitations that complicate the inference strategies of reasoning LLMs. The complex interplay between model architecture choices, token budget allocation, and test-time scaling strategies creates a vast design space where suboptimal decisions can undermine both performance and efficiency. As illustrated in Fig. 1, current approaches to edge LLM deployment exist as isolated solutions, leaving practitioners without systematic guidance for navigating the critical trade-offs between reasoning depth, inference speed, and accuracy. This gap is particularly problematic for autonomous systems where both real-time responsiveness and reliable decision-making are essential requirements.

In this work, we address these challenges through EdgeReasoning, a comprehensive study for characterizing LLM reasoning deployment on edge GPUs. Our contributions include:

1) Empirical characterization of latency, power, and energy tradeoffs across LLM architectures on edge hardware.
2) Performance models that analytically maps token counts to latency and energy performance for edge GPUs.
3) Systematic exploration of prompt-based optimization techniques to reduce reasoning token overhead.
4) Evaluation of test-time scaling methods to maximize accuracy under dynamic latency constraints.

Through this study, we demonstrate how edge reasoning can achieve cost efficiencies that make autonomous AI systems economically sustainable while providing deterministic latency control essential for real-time applications. Crucially, EdgeReasoning study enables autonomous systems to select optimal accuracy configurations within task-specific latency requirements, maximizing performance across diverse operational scenarios.

II. Background

A. Reasoning LLMs

Recent advances in large language models have enabled multistep logical reasoning and complex problem solving capabilities. Reasoning LLMs (e.g., OpenAI o1 [27], DeepSeek-R1 [7]) generate intermediate "chains of thought" (CoT) [36] that decompose complex problems into sequential inference steps before producing final answers. These models achieve superior accuracy on challenging tasks, including mathematics and coding, compared to traditional direct-generation counterparts. However, reasoning LLMs generate significantly longer output sequences than non-reasoning models, creating substantial computational overhead for edge deployment.

To address edge deployment constraints, *lightweight language models* have been developed [23], [32], [34]. Complementing these, knowledge distillation from large reasoning LLMs followed by task-specific fine-tuning has yielded compact reasoning models that retain high accuracy. For example, DeepSeek-R1 [7] is available in 1.5B, 7B, 8B, and 14B parameter variants optimized for edge devices. Additionally, DeepScaleR/DeepCodeR [20], [21], fine-tuned with reinforcement learning, attains parity with large models such as OpenAI's o1 on mathematical and coding tasks—demonstrating that sub 15B models can deliver near state-of-the-art reasoning performance within edge-scale compute and memory budgets.

TABLE I: NVIDIA Jetson Orin Series Compute Specifications

CUDA Cores	Tensor Cores	DLA	Memory
2048 (5.3TFLOPs)	64 (275TOPs)	2 (52.5TOPS)	64GB @ 204.8GB/s

B. Test-Time Scaling

Recent work has shifted focus from training-time scaling to test-time scaling, allowing LLMs to "think with more tokens" [27], [29]. Test-time scaling laws demonstrate predictable accuracy gains from increased inference computation through generating more or longer reasoning chains. Test-time scaling can be achieved through two main approaches: *sequential scaling* [24], which extends the length of individual reasoning chains, and *parallel scaling* [2], where multiple reasoning paths are generated simultaneously across processing units and aggregated via voting or consensus mechanisms. While both approaches multiply computational requirements, parallel scaling avoids linear latency increases through parallelization, making it particularly attractive when hardware resources are underutilized. More sophisticated inference strategies integrate both sequential and parallel scaling [8], [9], [14], [38].

C. Reasoning Token Optimization

Besides employing lightweight models and parallel test-time scaling techniques, optimizing reasoning length offers another approach to deploy reasoning models under latency constraints while preserving accuracy. [30] This can be achieved through *prompt-based methods*, which instruct models to use fixed token budgets [12] or disable CoT reasoning [22], trading off reasoning depth for reduced latency. However, these approaches are limited as not all models are trained with token budget awareness.

Alternatively, *fine-tuning techniques* like length-difference positional encoding [3] or explicit output length control (e.g., L1 [1]) achieve precise sequence length control. While effective at reducing output length for reasoning, these methods lack system-level integration to show practical latency-accuracy improvements in real deployments. See also latency-aware test-time scaling [35].

D. Edge GPUs

Deploying a reasoning LLM on an edge device, such as NVIDIA Jetson AGX Orin [25], imposes strict latency and memory constraints. Edge GPUs have limited compute throughput and memory bandwidth and capacity compared to server accelerators, making the lengthy decode phase of reasoning LLMs especially challenging.

NVIDIA's Jetson AGX Orin, which we use for all studies in this paper, is a representative edge AI GPU platform that integrates advanced compute capabilities in a low-power package. As shown in Table I, the Orin system-on-chip (SoC) features an NVIDIA Ampere-architecture GPU with 2048 CUDA cores, 64 Tensor Cores, and 2 NVDLAv2 Cores. The Tensor Cores accelerate mixed-precision matrix operations, enabling high-throughput FP16 and INT8 computations for deep learning acceleration. The Jetson Orin's GPU can deliver

TABLE II: Comparison of Lightweight Reasoning and Non-Reasoning Models for 150 MMLU-Redux Questions.

Model	Acc. (%)	Time (s)	TPS	Perf/W	Energy/Q (J)
gemma-7B [32]	33.9	7.1	7.2	0.3	210.3
llama3.1-8B [11]	58.3	2.5	6.6	0.3	77.9
qwen2.5-7B [33]	60.8	0.6	7.2	0.3	26.4
DSR1-Qwen-1.5B [7]	38.3	45.0	9.3	1.1	403.6
DSR1-LLama-8B [7]	61.7	143.3	7.8	0.3	4205.5
DSR1-Qwen-14B [7]	80.6	207.0	4.7	0.2	2599.2

TABLE III: Costs Comparison of Reasoning LLM Deployments

Metric	**OpenAI o1-preview**	**DeepScaleR-1.5B**	
Parameter Size	Unknown	1.5B in fp16	
Accuracy (AIME2024)	40.0%	**43.1%**	
Accuracy (Math500)	81.4%	**87.8%**	
Batch Size	Unknown	1	30
Throughput (User TPS)	89.7 [26]	44.0	21.2
Price (Input $/1M tokens)	$15 [28]	$0.302	**$0.027**
Price (Output $/1M tokens)	$60	$0.302	**$0.027**

up to roughly 5.3 TFLOPs of FP32 compute or up to 275 Sparse INT8 TOPS for deep learning workloads. The memory hierarchy includes 4MB of GPU L2 cache and 3MB of aggregate GPU L1 cache (192KB $\times$ 16 SMs). The platform features 64GB of LPDDR5 memory and operates within a configurable power envelope of 15–60W, making it well-suited for embedded applications in robotics and autonomous driving. The GPU is complemented by a 12-core ARM Cortex-A78AE CPU for control-heavy processing tasks.

III. Motivation

A. Comparison of Reasoning vs Non-Reasoning LLMs

Table II presents a comparison between reasoning and non-reasoning models across multiple performance metrics: MMLU-Redux accuracy [10], average decoding time, tokens per second (TPS), performance per watt, and total energy consumption per question. We evaluate distilled reasoning models from the DeepSeek-R1 [6] family and three popular lightweight non-reasoning alternatives (Gemma [32], Llama3 [11], Qwen2.5 [33]). Reasoning models demonstrate substantially higher accuracy than their non-reasoning counterparts on MMLU-Redux benchmarks [10]. Furthermore, accuracy scales positively with model size among the distilled reasoning models. When comparing models of similar scale (7-8B parameters), reasoning models achieve more than 7% higher accuracy than non-reasoning alternatives. However, this performance gain comes with significant computational overhead: ***reasoning models incur over 20× higher inference latency than non-reasoning models***. Consequently, both energy consumption and cost per token increase by a similar factor of at least 20× compared to non-reasoning counterparts. These efficiency gaps underscore the need for output token optimization strategies for practical deployment of reasoning models at the edge.

B. Comparison of Edge vs. Cloud Deployment

Edge deployment provides substantial energy and cost savings compared to cloud solutions, while also ensuring data privacy and operational resilience in connectivity-constrained environments. Table III demonstrates the significant cost efficiency of deploying DeepScaleR-1.5B on edge devices like the NVIDIA Jetson AGX Orin compared to cloud-based APIs. While OpenAI's reasoning models charge more than $4 per million output tokens ($4.4 on o4-mini and $60 on o1-preview), DeepScaleR achieves $0.302 per million tokens by running entirely on-device. DeepScaleR-1.5B further excels in accuracy, outperforming the commercial cloud model o1-preview on the AIME2024 and MATH500 benchmarks due to its RL fine-tuning for math and reasoning. This shows that ***edge deployment of reasoning models can deliver competitive accuracy at radically lower costs.***

Cost calculations derived from profiling the AIME2024 benchmark on the Orin platform reveal: In single-batch processing (FP32), the system handled 195,624 tokens in 4,358 seconds using 0.0317 kWh. At electricity rates of $0.15/kWh and hardware amortized at $0.045/hour, this yields $0.302 per million tokens ($0.024 energy + $0.278 hardware). Notably, batch processing (size 30) completed the same workload in 398 seconds using only 0.003 kWh, reducing costs to $0.027 per million tokens ($0.0023 energy + $0.025 hardware). These results demonstrate that ***edge deployment costs also benefit from batching and increased queries per second (QPS)***.

IV. Edge GPU Performance Characterization and Modeling

In this section, we characterize the latency, power, and energy consumption of lightweight reasoning models across different sizes (1.5B, 8B, 14B) with respect to various prefill and decode lengths when deployed on the Jetson Orin GPU using vLLM [15] as the inference engine.

A. Characterization of Latency

We first analyze end-to-end inference latency, decomposing it into prefill (initial prompt processing) and decode (token generation) components to reveal distinct computational behaviors. Beyond empirical measurements, we develop and validate accurate analytical performance models fitted to real-world Jetson Orin measurements, enabling rapid evaluation and navigation of latency-accuracy trade-offs during inference strategy selection.

Prefill Latency. Fig. 2 illustrates how measured prefill latency varies with input token counts for single-batch inference. We observe a distinctive stepped pattern where latency exhibits sub-quadratic scaling at token counts that are multiples of 128, with either linear increases or plateau regions within each 128-token segment.

This behavior stems from tensor quantization effects in the CUTLASS kernels when utilizing Tensor Cores, which process data in fixed-size blocks and require padding to align with hardware-optimized dimensions. The performance

TABLE IV: Fitted coefficients for prefill latency model.

Model	a	b	c
DSR1-Qwen-1.5B	1.56×10^{-7}	2.31×10^{-6}	0.046
DSR1-LLama-8B	6.65×10^{-7}	2.90×10^{-4}	0.104
DSR1-Qwen-14B	1.23×10^{-6}	5.3×10^{-4}	0.189

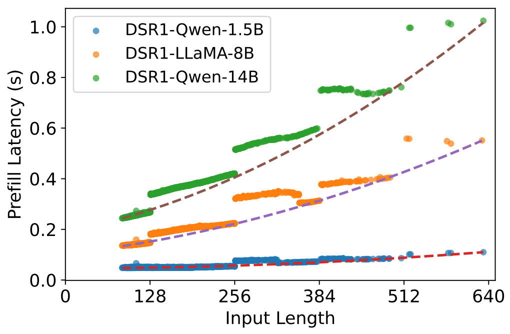
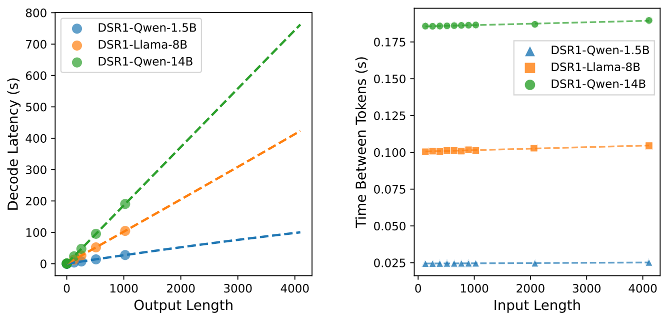


Fig. 2: Prefill latency vs. input sequence length.

TABLE V: Fitted coefficients for decode latency model

Model	m	n
DSR1-Qwen-1.5B	-1.50×10^{-7}	0.024
DSR1-LLama-8B	6.92×10^{-7}	0.010
DSR1-Qwen-14B	1.13×10^{-6}	0.187



(a) Input length=512. (b) Time between tokens

Fig. 3: Decode latency vs output and input sequence lengths.

characteristics within these 128-token segments depend on computational intensity. At lower token counts, the system operates in a memory-bandwidth-limited region where kernels are constrained by memory bandwidth rather than compute capacity, resulting in linear latency increases over input token counts within each segment. As prefill token counts grow larger, the system transitions to a compute-bound region where the kernels become limited by arithmetic throughput. In this regime, padding effects become more pronounced since workloads within the same token segment require identical FLOPS, leading to the observed plateau behavior. Additional performance variations that deviate from the primary trend are likely attributable to the selection of different CUTLASS kernel variants optimized for different GEMM shapes.

For the prefill phase of a given LLM, the theoretical compute and memory complexity scales linearly with input length I in the projection and feedforward layers, and quadratically in the attention layers. Based on this, we model the prefill latency as a quadratic function: $L_{\text{prefill}}(I) = aI^2 + bI + c$.

To account for Tensor Core padding effects, we restrict the model fitting to data points where the input length is a multiple of 64. In practice, input lengths are rounded up to the nearest multiple of 128 to form a padded length I_{pad}, defined as $I_{\text{pad}} = \left\lceil \frac{I}{128} \right\rceil \cdot 128$

Substituting I with I_{pad}, the fitted prefill latency functions can be expressed as:

$$L_{\text{prefill}}(I) = aI_{\text{pad}}^2 + bI_{\text{pad}} + c \tag{1}$$

Table IV lists the fitted coefficients for the prefill latency models of the 1.5B, 8B, and 14B DSR1 models. The fitted functions are also plotted in dashed lines in Fig. 2.

Decode Latency. Fig. 3a demonstrates how measured decode latency varies with output length O with fixed input length of 512 for three different models. Decode latency always grows near linearly with respect to the output length due to the autoregressive nature of the decoding process. Fig. 3b shows how the time between tokens varies with input length I for the DSR1-Llama-8B model. We observe a slight 3.1% TBT increase with input length increases from 1 to 4k.

At each decoding step i, the input context length increases by one, i.e., $I_i = I_{i-1} + 1$. Since the attention layer's latency grows linearly with the input context length during decode, we model the time between tokens (TBT) as $TBT_i = mI_i + n$. The total decoding latency, L_{decode}, is the sum of TBT across all O output steps: $L_{\text{decode}} = \sum_{i=0}^{O-1} TBT_i$. By simplifying the expression for L_{decode}, we obtain the following theoretical decode latency model:

$$L_{\text{decode}}(I, O) = nO + m\left(IO + \frac{O(O-1)}{2}\right) \tag{2}$$

where I is the initial input length.

To derive the decode latency model, we first fit the decode latency model in Eqn. 2 using 100 MMLU-Redux data points with various input and output lengths. The corresponding coefficients m and n for different LLM are listed on Table V.

Since the m is very small, the TBT is almost equal to n. The average time between tokens (TBT) for the 1.5B, 8B, and 14B models are 0.029s, 0.092s, and 0.187s, respectively. They are corresponding to the slopes of lines in Fig. 3b.

Given the negligible magnitude of the slope coefficient m, the average TBT can be effectively approximated by n. The corresponding TBT values for the 1.5B, 8B, and 14B models are 0.024s, 0.10s, and 0.186s, respectively. These values correspond to the slopes of the curves shown in Fig. 3b, confirming that TBT remains relatively constant across different context lengths for each model size.

Total latency. Combining Eqn. 1 and 2, we have the total inference latency on Jetson Orin GPU for the three models defined as:

$$L = L_{\text{prefill}} + L_{\text{decode}} \tag{3}$$

We validate our fitted analytical latency models on 50 held-out MMLU-Redux test questions. Table VI shows that the predicted latencies match the measured values closely, with total MAPE under 2% across all models. We use these fitted latency models throughout the remainder of this paper to accelerate latency evaluation and optimal inference strategy search, as real measurements on the complete dataset to produce one latency point would require weeks to finish. For instance, a full latency evaluation on all MMLU-Redux questions using DSR1-LLaMA-14B takes 8 days to complete, while the analytical model produces results within seconds.

TABLE VI: Mean Absolute Percentage Error (MAPE) of Latency Model

Model	Prefill	Decode	Total
DSR1-Qwen-1.5B	9.80%	0.42%	0.46%
DSR1-LLaMA-8B	13.39%	0.45%	0.49%
DSR1-Qwen-14B	7.59%	0.53%	0.56%

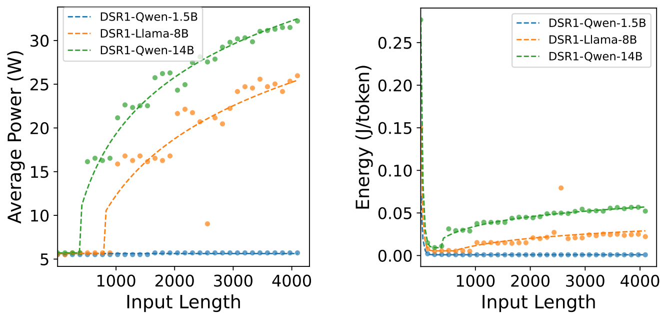
Takeaway #1: Edge inference latency of LLMs can be accurately fitted using polynomial functions.

Prefill-to-decode latency ratio for reasoning models. Table VII presents the prefill-to-decode token and latency ratios when running the complete MMLU-Redux dataset across our three reasoning models. The results reveal a striking disparity between token generation patterns and actual latency distribution. While the models generate 2.4-7.3× more decode tokens than prefill tokens, the latency imbalance is far more pronounced, with decode phase consuming 192-569× longer than prefill phase. This dramatic difference stems from the sequential nature of autoregressive generation during decode, where each token must be generated individually, compared to the parallel processing of all input tokens during prefill. The Qwen models exhibit higher token ratios (7.1-7.3×) due to their more verbose reasoning chains, yet all models show consistently extreme latency ratios, with decode dominating over 99.5% of total inference time. This analysis underscores the critical importance of decode optimization for reasoning workloads on edge devices.

TABLE VII: Prefill-to-decode Ratios for Full MMLU-Redux

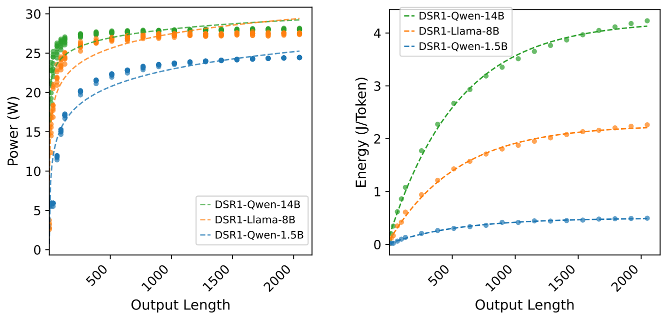
Model	P-to-D Tokens Ratio	P-to-D Latency Ratio
DSR1-Qwen-1.5B	1:7.3	1:521
DSR1-LLaMA-8B	1:2.4	1:192
DSR1-Qwen-14B	1:7.1	1:569

Takeaway #2: Edge inference latency of reasoning LLMs is dominated by decode.



(a) Prefill power. (b) Energy per token.

Fig. 4: Prefill power (left) and energy per token (right) as a function of input sequence length.



(a) Power. (b) Energy per token.

Fig. 5: Decode phase power (left) and energy per token (right) as a function of output sequence length

B. Characterization of Power and Energy

Beyond latency, understanding power consumption and energy efficiency is critical for edge deployment scenarios. The Jetson AGX Orin 64GB platform we used in the study supports four configurable power modes (15W, 30W, 50W, and MAXN) that set peak frequencies across GPU, CPU, DLA, and PVA units. All experiments are conducted in MAXN mode to capture peak performance characteristics. We analyze power consumption and energy usage as functions of input length, output length, and model size to establish fundamental scaling relationships for edge inference workloads.

Prefill Power and Energy. Fig. 4a shows that average power consumption increases with input sequence length for single-batch inference, measured on the Jetson AGX Orin with 5 repeated samples per data point. This trend occurs because longer input sequences increase the computational intensity of the workload, leading to higher GPU utilization. The larger 8B and 14B models reach over 20W at 4K input sequence length, while the smaller 1.5B model consumes only 6W—representing just 10% of the platform's 60W peak power capacity. This shows the significant differences in power consumption between model sizes.

Fig. 4b shows the energy consumption per input token across different input sequence lengths. The results demonstrate that smaller models consistently achieve superior energy efficiency compared to larger models due to their reduced FLOPs and memory requirements. Across all three models, we observe a characteristic trend where energy per token initially decreases from short input lengths until reaching a minimum around 300 tokens. We attribute this behavior to the dominance of projection and feed-forward (FFN) layers in this regime, where increased input length leads to better weight reuse and improved energy efficiency. Beyond this point, as the workload becomes attention-bound, further increases in input length provide diminishing returns from weight reuse. Consequently, we observe that energy per token plateaus for large input sequences, with oscillations around the steady-state value.

We also develop an analytical power model based on the observed data to accelerate evaluation. Since the prefill power consumption $P_{\text{prefill}}(I)$ exhibits two distinct regimes depending on input length I, we define our power model as:

$$P_{\text{prefill}}(I) = \begin{cases} u, & I \le v, \\ w \ln(I) + x, & I > v. \end{cases} \tag{4}$$

For shorter input sequences ($I \le v$), power remains constant at u watts, indicating low GPU utilization. For longer sequences ($I > v$), power consumption increases logarithmically, reflecting higher computational intensity and improved hardware utilization. The implied energy consumption model follows from $E_{\text{prefill}}(I) = \int_0^{L_{\text{prefill}}(I)} P_{prefill}(t)\,dt$, where energy is the time integral of instantaneous power. We additionally fit a direct piecewise energy model that captures the amortization of short-sequence overheads and energy increase at longer lengths:

$$E_{\text{prefill}}(I) = \begin{cases} A\,e^{-\lambda I} + C, & I \le v_e, \\ \alpha_e \ln I + \beta_e, & I > v_e, \end{cases} \tag{5}$$

where v and v_e are model-specific transition points. For the distilled models used here, typical transitions are v=800 (8B) and v=384 (14B), while the 1.5B case is effectively constant over the measured range. The fitted coefficients for each model are provided in appendix Table XX.

TABLE VIII: Mean Absolute Percentage Error (MAPE) of Energy Model

Model	Prefill	Decode	Total
DSR1-Qwen-1.5B	–	6.8%	6.0%
DSR1-Llama-8B	–	6.4%	5.7%
DSR1-Qwen-14B	–	6.6%	5.8%

Decode Power and Energy. Fig. 5a shows the average power consumption and energy for varying output sequence lengths with a fixed input sequence length of 512 tokens. The results demonstrate that power consumption increases logarithmically with output sequence length. While the computation of the projection and FFN layers remains constant during decoding, this increase is due to the growing computational and memory demands in the attention layer as the context window expands. The analysis also demonstrates significant efficiency gains from model size reduction: the 1.5B model achieves a 7× improvement in energy per token compared to the 14B model, highlighting the substantial energy benefits of deploying smaller models for resource-constrained edge environments. The decode power is fitted with the same functional form used for the prefill power in Eqn. 4 For the orin GPU, this yields Eqn. 6.

$$P_{\text{decode}}(O) = \begin{cases} 5.9\,\text{W}, & 0 < O < 64, \\ y \ln O + z, & O \ge 64\ , \end{cases} \tag{6}$$

where O is the output sequence length, and y and z are fitted parameters that capture the logarithmic scaling behavior observed in our measurements of different models. The corresponding energy consumption is given by $E_{\text{decode}}(I) = \int_0^{L_{\text{decode}}(I)} P_{\text{decode}}(t)\,dt$.

Total Energy. We model the total energy as $E = E_{\text{prefill}}(O) + E_{\text{decode}}(O)$. Since decode latency is significantly longer than prefill, decode energy consumption also dominates the total energy budget.

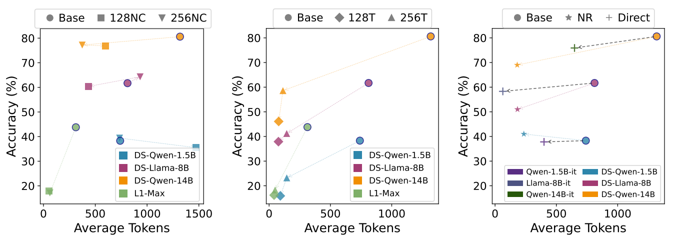
Takeaway #3: Average power and total energy consumption increase logarithmically with sequence length on NVIDIA Jetson AGX Orin platform.

V. Evaluation of Inference Strategies

This section compares different inference strategies for showing the accuracy-latency/energy tradeoffs and guiding optimal inference strategies on edge GPUs. We evaluate the tradeoffs for reasoning vs non-reasoning models and reasoning models in different sizes. We also evaluate different prompt-based and fine-tuning-based methods for reducing output sequence length while maintaining accuracy.

This section systematically evaluates inference strategies to quantify accuracy-latency and cost tradeoffs and guide optimal configuration selection for reasoning models on edge GPUs. We analyze three critical dimensions: (1) reasoning capability vs. model size tradeoffs, (2) output sequence length reduction techniques, and (3) energy and cost efficiency implications. Our evaluation encompasses three model categories:

- **Standard Models (Non-reasoning):** Baseline architectures generating direct responses without explicit reasoning chains: *Qwen2.5-1.5B-it* [33], *Llama3.1-8B-it* [11], and *Qwen2.5-7B-it* [33].
- **Reasoning Models:** Lightweight reasoning-optimized models from the Deepseek-R1 (DSR1) family: *DSR1-Qwen-1.5B*, *DSR1-Llama-8B*, and *DSR1-Qwen-14B*. These distilled models perform standard autoregressive inference without token constraints.
- **Budget-Aware Reasoning Model:** *L1* [1], a *DSR1-Qwen-1.5B* variant fine-tuned via reinforcement learning to maximize accuracy under user-specified token budgets.



(a) Soft limit (b) Hard limit (c) No reasoning

Fig. 6: Accuracy versus average output length across budgeting techniques.

For output length optimization, we evaluate three prompt-based approaches applied to reasoning models:

- **Hard-Length Control ([n]T):** Explicit length instructions (e.g., "Answer in [n] words") with strict token enforcement. Configurations: *128T*, *256T*.
- **Soft-Length Control ([n]-NC):** Identical instructions without token enforcement. Configurations: *128-NC*, *256-NC*.
- **No Reasoning (NR):** Bypasses explicit reasoning by injecting predefined thinking blocks between delimiters [22]:

```
<|beginning of thinking|>
Okay, I think I have finished thinking.
<|end of thinking|>
```

All configurations are evaluated on the MMLU-Redux benchmark [10], comprising 3,000 multiple-choice questions spanning humanities, social sciences, STEM, and professional domains. The benchmark tests both factual knowledge and reasoning capabilities across difficulty levels from elementary to graduate. For each configuration, we report four key performance metrics: (1) accuracy on the MMLU-Redux benchmark, (2) average decoded tokens per question, (3) average inference latency per question, and (4) average cost per million tokens derived from energy measurements.

Our results reveal fundamental tradeoffs between critical metrics:

- **Accuracy vs. Output Length:** Fig. 6 demonstrates how accuracy varies with generated sequence length across model classes and length-control methods, revealing the compression-performance frontier.
- **Accuracy vs. Latency:** Fig. 7 quantifies the accuracy-latency tradeoff on edge hardware.
- **Accuracy vs. Cost Efficiency:** Fig. 8 correlates reasoning quality with operational economics through cost per million tokens.

A. Impact of Model Selection

Model selection significantly impacts the achievable accuracy-latency tradeoff, making it crucial to understand how to choose between reasoning and non-reasoning architectures, as well as among reasoning models of different sizes.

Impact of model size. Analysis of reasoning models across different sizes—*DSR1-Qwen-1.5B* (blue), *DSR1-Llama-8B* (plum), and *DSR1-Qwen-14B* (orange)—shows that larger models typically achieve higher accuracy at the cost of increased inference latency. Fig. 6 demonstrates that larger models in their Base configuration (○ markers) naturally generate more reasoning tokens when unconstrained.

Fig. 6 reveals an intriguing trade-off space where smaller models with higher token budgets can be competitive with larger models operating under smaller token budgets. Notably, in Fig. 6b, *DSR1-Llama-8B Base*(○) (generating 811 tokens on average) achieves higher accuracy than *DSR1-Qwen-14B 128T*(◊) (generating only 91.5 tokens), suggesting that reasoning depth can compensate for reduced model scale. Conversely, in Fig. 6a, *DSR1-Llama-8B Base*(○) (generating 811 tokens on average) underperforms *DSR1-Qwen-14B 256T-NC*(▽)(generating only 374 tokens), suggesting that model scale can also compensate for reduced reasoning depth.

The crossover analysis provides practical deployment insights. For example, in Fig. 7b, DSR1-Qwen-14B *256T* (△) achieves comparable accuracy to DSR1-Llama-8B *Base* at 4× lower latency (21s vs 87s) by operating within a 113-token budget. This indicates that for latency budgets exceeding 21s, DSR1-Qwen-14B with >113 token allocation becomes preferable.

The Pareto-optimal frontier in Fig. 7 reveals three distinct operational regimes:

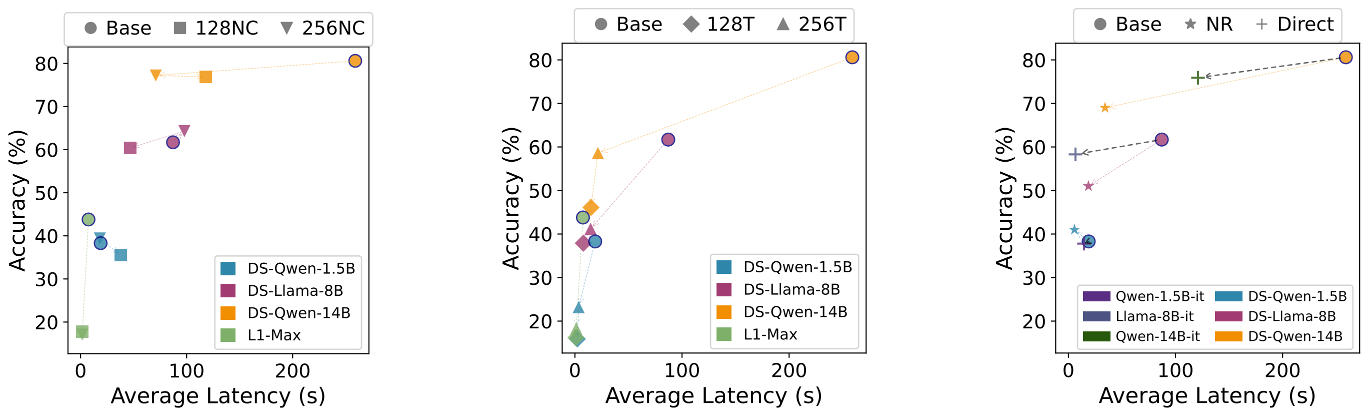
- Sub-5s latency: Exclusively served by 1.5B models.
- 15-30s latency: Non-reasoning 8B models are preferred.
- >30s latency: DSR1-Qwen-14B emerges as optimal.

Takeaway #4: Only ultra-lightweight models (1.5B) can achieve real-time inference (<1s) on resource-constrained edge platforms.

B. Impact of Reasoning Token Control Methods

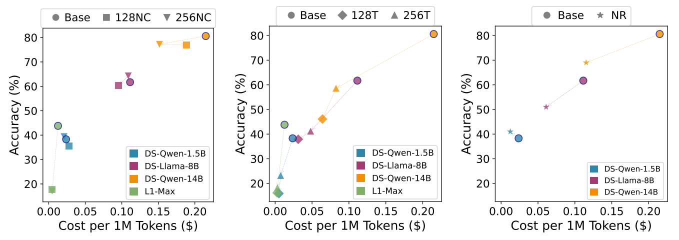
Output token length control is essential for latency management and meeting real-time constraints. While token-to-latency scaling factors are model-dependent, they exhibit predictable relationships that can be characterized and fitted to accurate performance models for the Orin GPU, as demonstrated in Section IV-A.

Prompt-based. Fig. 6 shows that in-prompt length control can significantly reduce output token length, but often at the cost of lower accuracy because it limits test-time scaling. However, in-prompt length control rarely adheres to the user's specification. For example, under the *128-NC* setting (soft limit, □) on *DSR1-Qwen-14B*, the model emits four times as many tokens as the *128T* (◊) with hard cutoff. Even so, *128-NC* still generates roughly half as many tokens as the uncontrolled *Base* run, while maintaining comparable accuracy, demonstrating that the prompt makes the model modestly token-aware.



(a) Soft limit (b) Hard limit (c) No reasoning

Fig. 7: Accuracy versus latency across budgeting techniques.



(a) Soft limit (b) Hard limit (c) No reasoning

Fig. 8: Accuracy versus cost across budgeting techniques.

No-thinking *NR* (marked as $\star$) provides another way to shorten outputs by skipping the explicit reasoning. Compared to *Base* (marked as $\circ$), *NR* reduces sequence length for all *DSR1-Qwen-1.5B* (blue), *DSR1-Llama-8B* (plum), and *DSR1-Qwen-14B* (orange) models. When comparing *DSR1-Llama-8B NR* ($\star$) with the *Direct* (marked as $+$) non-reasoning baselines, *Direct* achieves slightly higher accuracy with fewer tokens, indicating that using a small non-reasoning model can outperform disabling reasoning in a larger one. Interestingly, on the 1.5B model *NR* attains the best accuracy overall, suggesting that suppressing the reasoning phase in very small models could be beneficial.

Takeaway #5: Prompt-based approaches are effective in reducing reasoning tokens.

Budget-aware models. As shown in Fig. 6, standard DSR1 models lack precise output length control. To address this limitation, we evaluate *L1*—a model specifically fine-tuned to enhance instruction-following capabilities for token budget adherence. Using its *L1-max* variant (green), which strictly enforces output lengths within specified token budgets, we observe: Without token constraints (*Base*, $\circ$), *L1-max* achieves higher accuracy than *DSR1-Qwen-1.5B* (blue) while generating over 2$\times$ fewer tokens; 2) When constrained by in-prompt length specifications (e.g., *128T-NC*($\square$), *256T-NC*($\triangledown$) in Fig. 6a), *L1-max* consistently adheres to token budgets, demonstrating the efficacy of RL fine-tuning for output control. However, we observe excessive conservatism: For a 256-token budget in the *256T*($\triangle$) configuration, *L1-max* generates fewer than 50 tokens—significantly underutilizing allocated capacity. By leveraging the token length control capabilities of the *L1* model and the analytical latency model from Eqn. 3 (Sec. IV-A), we can systematically determine output token length constraints that satisfy specified latency targets L_C.

Takeaway #6: Fine-tuned token-budget-aware models combined with latency performance modeling enable adherence to latency constraints.

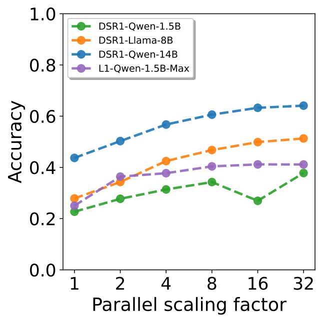
C. Impact of Sequential Test Time Scaling

Fig. 6 reveals a consistent trend across all base models: accuracy generally increases with output sequence length, regardless of the prompting approach. However, this relationship exhibits diminishing returns beyond certain token thresholds, specifically, $\sim$300 tokens for *DSR1-Qwen-1.5B* (using L1's budget-aware tuning) and $\sim$400 tokens for both *DSR1-Llama-8B* and *DSR1-Qwen-14B*. These inflection points suggest where parallel scaling may surpass sequential scaling for accuracy gains, as shown in [24].

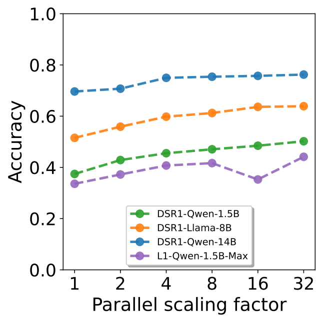
The near-linear relationship between output length and inference latency (Section IV-A, Fig. 7) enables effective accuracy-latency co-optimization on edge GPUs. By strategically constraining token budgets at these optimal lengths, we can maximize accuracy while minimizing latency penalties.

Takeaway #7: Sequential scaling holds even when reasoning token control is applied.

Reasoning vs. non-reasoning models. Fig. 7c reveals distinct performance profiles between reasoning and non-reasoning approaches. The non-reasoning models direct generations ($+$ markers: Qwen2.5-1.5B-it, Llama3.1-8B-it,

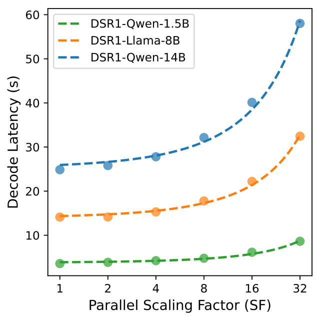


(a) Output length = 128

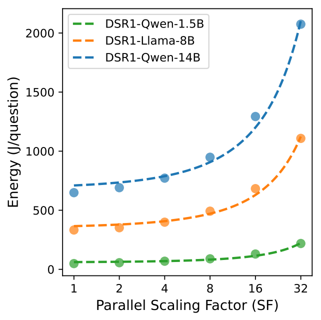


(b) Output length = 512

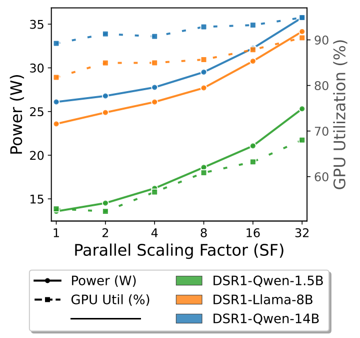
Fig. 9: Accuracy vs. parallel scaling factor at output lengths 128 (a) and 512(b) on Full MMLU-Redux.



(a) Decode latency



(b) Energy per question



(c) Power and GPU utilization

Fig. 10: Parallel-scaling on Orin: (a) decode latency, (b) energy per question, (c) power and GPU utilization

Qwen2.5-14B-it) demonstrate competitive accuracy under low latency compared to the reasoning model counterparts. DS-Llama-8B's *Base* configuration (○ markers plum) without token control achieves 5.7% higher accuracy than the non-reasoning counterpart, *Llama3.1-8B-it*, but at the cost of 13× longer runtime (87.2s vs 6.60s) as shown in Fig. 7. When *DSR1-Llama-8B* is constrained to 128 tokens (*128T*) to achieve sub-10s inference time, accuracy drops by 34% compared to direct *Llama3.1-8B-it*. Additionally, the direct *Llama3.1-8B-it* consistently outperforms all 1.5B reasoning models configurations, establishing it as the preferred choice for latency budgets below 20 seconds.

Takeaway #8: Non-reasoning models offer a competitive latency-accuracy trade-off compared to reasoning models on a low token and latency budget.

D. Cost Analysis.

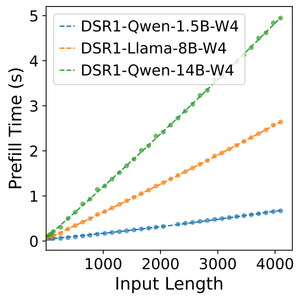
Fig. 8 illustrates the accuracy-cost trade-offs inherent in different inference strategies. The results confirm that superior accuracy typically incurs higher computational costs due to the deployment of larger reasoning models and extended output sequences. The analysis provides clear guidance for model selection based on token pricing constraints. For budgets below $0.01 per million tokens, ultra-lightweight models such as *DSR1-Qwen-1.5B* and *L1* represent the only viable options. Within the $0.01-$0.1 per million token range, non-reasoning models deliver optimal accuracy-to-cost ratios. Beyond $0.1 per million tokens, both *DSR1-Llama-8B* and *DSR1-Qwen-14B* emerge as compelling alternatives, though users must carefully balance model size against token length budget constraints.

E. Parallel Test Time Scaling

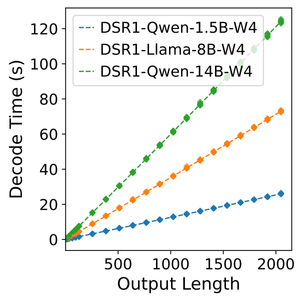
The preceding studies examined single-batch inference without parallel scaling. As discussed in Section II, parallel scaling represents another test-time scaling approach that can increase accuracy with minimal latency overhead. We now quantify how parallel scaling impacts latency, power, and energy efficiency across our target models. For consistency, all experiments use a fixed 128 output token budget. The prefill phase is executed once with a batch size of 1; during the decode phase we increase the batch size to match the target parallelization factor. Results from the parallel decoders are combined with a lightweight majority-voting scheme to produce the final answer.

Impact on Accuracy. First, we study how parallel scaling impacts the accuracy by evaluating on MMLU-Redux. Fig.9a shows that scaling from 1× to 32× yields accuracy improvements of approximately 1.5× to 1.8× across both model sizes under a 128-token output budget. Conversely, Fig.9b reveals different behavior when the output budget increases to 512 tokens: accuracy gains plateau after only 4× scaling for larger models, with even more limited improvements observed in smaller models. This plateau effect indicates that under higher token budgets, sequential scaling becomes the dominant factor driving accuracy improvements, while additional parallel samples produce diminishing returns. Models fine-tuned for length control exhibit distinct behavior. The L1-Qwen-1.5B-Max variants show negligible benefits from parallel scaling beyond 2× (128-token budget) and 8× (512-token budget). Furthermore, smaller models experience accuracy degradation at the 16× scaling factor.

Impact on Decode Latency. Fig. 10a presents an ablation study of decoding latency versus parallel scaling factors

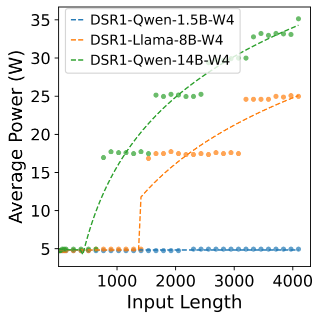


(a) Prefill Time

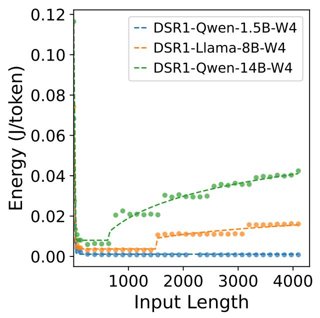


(b) Decode Time

Fig. 11: Prefill (left) and decode phase latency (right) as a function of sequence length for the quantized models.



(a) Power



(b) Energy/token

Fig. 12: Prefill phase power (left) and energy/token (right) as a function of sequence length for the quantized models.

(SF) on the NVIDIA Jetson Orin platform. Since both the compute and memory complexity of decoding scales with batch size, in theory larger batch size should lead to higher decode latency. However, Tensor Core on GPUs introduces a potential optimization: the initial scaling steps may incur minimal overhead due to batch dimension padding in 128-size blocks. Fig. 10a reveals a slight latency increase for SF < 128, with latency rising approximately 2× from SF=1 to SF=64 across all models. While this modest increase partially validates the Tensor Core hypothesis, the non-flat latency profile demonstrates that scaling isn't completely free.

Impact on Power and Energy. Fig.10c illustrates how average GPU power consumption varies with parallel scaling factors and overall GPU utilization. Power consumption increases substantially with parallel scaling, rising from 14W to 25W for the 1.5B model and from approximately 25W to 35W for the larger 8B and 14B models. These discrete power trends correspond to distinct GPU power states triggered by different utilization levels, as shown on the secondary axis. This scaling behavior aligns with the increased computational and memory complexity introduced by parallel batching.

Fig.10b demonstrates how energy per question varies with parallel scaling factor across the three models. The energy consumption follows a similar trend to decode latency, as longer inference times naturally result in higher energy consumption, particularly when power draw is simultaneously increasing. For the 14B model, energy per question increases modestly by less than 1.5× from SF=1 to SF=4, indicating efficient resource utilization in this range. However, at SF=16, energy consumption doubles, reflecting the transition to higher energy overhead with parallel scaling.

Takeaway #9: Parallel scaling improves accuracy with minimal latency and energy overhead at small scaling factors (≤ 8).

Impact on Utilization. Fig.10c shows that GPU utilization rises linearly with the parallel scale factor. DRAM read bandwidth dominates—rising above $\sim$20% on the 1.5B model and above $\sim$60% on the 14B model—since decode kernels continually fetch weights, and activation tiles from DRAM. Write bandwidth stays below 10%, reflecting KV-cache write back and output logits commits.

CPU utilization holds steady $\leq$ 20% regardless of scale factor, revealing a large pool of idle host-side compute. In other words, generation batching trades extra latency and energy for proportional gains in on-chip compute efficiency, and further latency reductions can be unlocked by offloading lightweight graph kernels—tokenization, layer-norm, softmax, embedding lookups—to the host CPU and overlapping them with GPU matmuls. Due to the shared memory nature of Orin's SoC this would present minimal communication overheads. Moreover, on Jetson Orin the dedicated deep-learning (DLA) and programmable vision (PVA) accelerators sit unused during transformer inference; exploring how to map parts of the attention/FFN workload onto these engines could yield additional throughput and energy-efficiency wins.

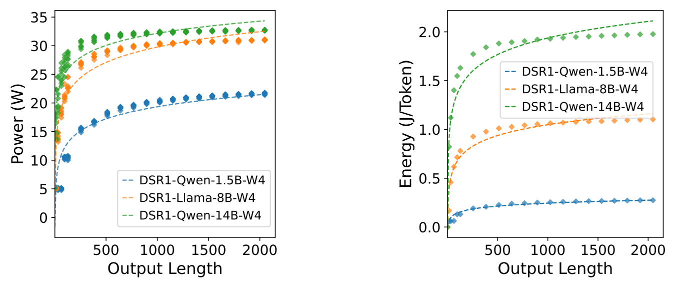
Takeaway #10: Parallel scaling utilizes hardware resources effectively and improves the overall GPU utilization.

F. Impact of Quantization

We evaluate the effect of quantization on reasoning models by applying W4A16 (4-bit weights, 16-bit activations) using the LLM Compressor AWQ configuration in vLLM. On the Jetson Orin GPU, however, computation falls back to INT8 since its Ampere architecture does not support INT4.

Fig. 11 presents prefill and decode latency for the quantized models, while Fig. 12 and 13 show power and energy per token during prefill and decode. They have a shorter prefill and decode time at lower energy/token compared to their non-quantized models shown in Fig. 2 and 3.

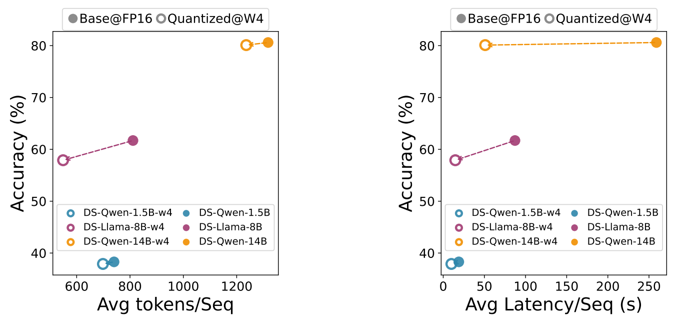
Fig. 14 demonstrates that AWQ quantization reduces accuracy relative to FP16 by (i) DSR1-Qwen-1.5B: -1.04%, (ii) DSR1-Llama-8B: -6.16% and DSR1-Qwen-14B: -0.62%



(a) Power

(b) Energy/token

Fig. 13: Decode phase power (left) and energy/token (right) as a function of sequence length at 512 input length.



(a) Average Ouput Tokens

(b) Average Latency(s)

Fig. 14: Comparison of quantized and non-quantized models on accuracy, average output token length, and latency.

of relative loss. Finally, Figure 14a indicates that quantized models generate fewer decoding tokens than their FP16 counterparts. As a result, quantization improves latency by roughly 2–5×, with larger models benefiting more than smaller ones, as shown in Fig.14.

Takeaway #11: AWQ-based W4 quantization improves latency and reduces energy per token with minor accuracy loss. Gains increase with larger model size.

G. Impact of Inference Frameworks

In this section, we present a latency comparison across popular inference frameworks, including the Hugging Face Transformers library (HFT) [37], vLLM [15] and TRT-LLM. We evaluate end-to-end inference time using three input–output sequence length combinations on DSR1-Llama-8B model, and observe that vLLM(v0.86) achieves a speedup of 1.11× to 1.13× over HFT(v4.46.2) and a similar performance when compared to TRT-LLM (v0.12).

VI. Discussion and Future Work

Our study reveals significant opportunities for co-optimizing GPU architecture and software to enhance edge inference performance for reasoning LLMs. The bandwidth-bound nature

TABLE IX: Inference Engine Performance Comparison on DeepSeek-R1-Distill-Llama-8B

Input Length	Output Length	Latency (s) HF → vLLM → TRT-LLM (Speedup)
16	128	14.23 → 12.73 (1.12×) → 12.79 (1.00×)
64	128	14.29 → 12.75 (1.12×) → 12.46 (1.05×)
128	128	14.41 → 12.78 (1.13×) → 12.88 (0.99×)

of reasoning LLM inference becomes evident when examining the operational characteristics of the Jetson AGX Orin platform. With a FLOPs-to-bytes ratio of approximately 1375 for fp16 tensor operations—significantly higher than the operational intensity of batch size 1 GEMV operations—the system is constrained by memory bandwidth rather than computational throughput. This bottleneck is particularly pronounced in reasoning LLMs where decoding operations dominate 99% of the inference time, creating a critical need for enhanced memory bandwidth to achieve optimal performance. Beyond GPU utilization, our analysis reveals that other computational resources within the Orin SoC remain underutilized during inference. Both ARM CPU cores and DLA units present opportunities for performance optimization through heterogeneous computing approaches.

Several optimization strategies warrant investigation to address these performance limitations. **Quantization** [19], [39], [40] can reduce model precision to 4-bit or lower while maintaining accuracy. **Kernel fusion** [5], [17], [43] can minimize memory traffic by combining not only attention operations but also normalization, activation functions, and other tensor operations into unified kernels. **Prefetching** [41] can overlap memory transfers with computation to hide latency. **Speculative decoding** [4], [16], [18] can increase computational intensity by predicting multiple tokens in parallel. These optimizations, combined with inference-time scaling strategies, offer a comprehensive approach to maximizing reasoning LLM performance on edge devices.

VII. Conclusion

This work presents a comprehensive characterization of Large Language Model (LLM) reasoning workloads on edge GPU platforms. We systematically quantify the impact of model scale, input/output sequence lengths, and inference-time scaling techniques on latency, power consumption, and energy efficiency. By deriving analytical models that map these parameters to performance metrics, we enable rapid evaluation of optimal deployment strategies without exhaustive hardware testing. Furthermore, we assess token control methodologies for multi-step reasoning tasks, characterizing their fundamental latency-accuracy tradeoffs. Our analysis demonstrates the superior cost-effectiveness of edge deployment for LLM reasoning and provides concrete configuration guidelines for maximizing accuracy under diverse latency constraints. These findings deliver both practical deployment frameworks and fundamental insights for efficient edge AI systems.

REFERENCES

[1] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[2] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[3] Bradley Butcher, Michael O'Keefe, and James Titchener. Precise length control for large language models. *Natural Language Processing Journal*, 11:100143, 2025.

[4] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[5] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[6] DeepSeek. Deepseek-r1 (0528) via openrouter. https://openrouter.ai/deepseek/deepseek-r1-0528, 2024. Accessed: 2025-06-23.

[7] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[8] Edward Beeching, Lewis Tunstall, Sasha Rush. Scaling test-time compute efficiently. https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute, 2024. Accessed: 2025-06-24.

[9] Zitian Gao, Boye Niu, Xuzheng He, Haotian Xu, Hongzhang Liu, Aiwei Liu, Xuming Hu, and Lijie Wen. Interpretable contrastive monte carlo tree search reasoning. *arXiv preprint arXiv:2410.01707*, 2024.

[10] Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are we done with mmlu? In *arXiv preprint arXiv:2406.04127*, 2024.

[11] Meta GenAI. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[12] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

[13] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

[14] Coleman Hooper, Sehoon Kim, Suhong Moon, Kerem Dilmen, Monishwaran Maheswaran, Nicholas Lee, Michael W Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. Ets: Efficient tree search for inference-time scaling. *arXiv preprint arXiv:2502.13575*, 2025.

[15] 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 ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[16] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. *arXiv preprint arXiv:2211.17192*, 2023.

[17] Ao Li, Bojian Zheng, Gennady Pekhimenko, and Fan Long. Automatic horizontal fusion for gpu kernels. In *2022 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 14–27. IEEE, 2022.

[18] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[19] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[20] Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025.

[21] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. 2025.

[22] Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025.

[23] Microsoft. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[24] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[25] NVIDIA Corporation. Jetson AGX Orin series – technical specifications. https://developer.nvidia.com/embedded/jetson-agx-orin, 2022. Available online at NVIDIA Developer (Accessed June 2025).

[26] OpenAI. Openai o1-preview (2024-09-12) via openrouter. https://openrouter.ai/openai/o1-preview-2024-09-12, 2024.

[27] OpenAI. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[28] OpenAI. Openai api pricing. https://platform.openai.com/docs/pricing, 2025.

[29] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[30] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models, 2025. *URL https://arxiv. org/abs/2503.16419*.

[31] Gemini Team. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[32] Gemma Google Team. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[33] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[34] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[35] Zili Wang, Tianyu Zhang, Haoli Bai, Lu Hou, Xianzhi Yu, Wulong Liu, Shiming Xiang, and Lei Zhu. Faster and better llms via latency-aware test-time scaling. *arXiv preprint arXiv:2505.19634*, 2025.

[36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[38] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

[39] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *International Conference on Machine Learning*, pages 38087–38099, 2023.

[40] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation. *arXiv preprint arXiv:2303.08302*, 2023.

[41] Ahmet Caner Yüzügüler, Jiawei Zhuang, and Lukas Cavigelli. Preserve: Prefetching model weights and kv-cache in distributed llm serving. *arXiv preprint arXiv:2501.08192*, 2025.

[42] Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, et al. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024.

[43] Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Wu, Cody Hao Yu, Ameer Haj-Ali, Yida Wang, Jun Yang, Danyang Zhuo, Koushik Sen, et al. Ansor: Generating high-performance tensor programs for deep learning. *14th USENIX Symposium on Operating Systems Design and Implementation*, pages 863–879, 2020.

[44] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

Appendix A
Artifact Appendix: EdgeReasoning

A. Abstract

This section describes how to obtain the EdgeReasoning artifact and reproduce the key results in the paper. The artifact is validated on NVIDIA Jetson Orin AGX (ARM64 + JetPack 6.2/CUDA 12.8) and `x86_64` servers with NVIDIA GPUs; other systems may work but were not evaluated.

B. Artifact check-list (meta-information)

- **Algorithm:** Evaluation of LLM inference on edge systems
- **Program:** Python framework for LLM energy/latency modeling.
- **Data set:** JSON validation files, YAML configurations.
- **Run-time environment:** Ubuntu 22.04, NVIDIA Jetpack 6.2, CUDA 12.8, Docker, VLLM, PyYAML, NumPy.
- **Hardware:** NVIDIA Jetson Orin AGX 64GB, H100, RTX A6000.
- **Disk space required:** ~64GB
- **How much time is needed to prepare workflow?** ~30 mins.
- **How much time is needed to complete experiments?** ~24 hours.
- **Publicly available?:** Yes.
- **Code licenses:** BSD-3-Clause license.
- **Workflow automation:** GNU Make + Bash scripts, Jupyter Notebook.
- **Archived:** https://doi.org/10.5281/zenodo.17168238

C. Installation

To set up the artifact, clone the repository and use the provided Make commands to setup for tegra or a server system.

```
git clone \
https://github.com/edge-inference \
edgereasoning.git
cd edgereasoning
make venv
source .venv/bin/activate
make setup
```

D. Experiment workflow

The evaluation code is under `edgereasoning/eval/` and splits between server and Tegra hosts.

Tegra: On Tegra systems one can call the following benchmarks to produce prefill and decode data used in figures 1-5

```
#enter container environment
cd eval/tegra && ./open.sh 1
./launch.sh prefill
./launch.sh decode
```

The framework consists of a benchmarking suite and analytical models for latency, power, and energy.

Server: Server systems can run MMLU-Redux [10] benchmarks faster to produce accuracy results presented in figures 6-8:

```
#in edgereasoning directory
make server-mmlu       # MMLU-Redux evaluation
make planner          # Planner benchmarks
```

Configurations: Each evaluation has configuration files such as `eval/tegra/mmlu/configs/` that define test runs such as

```
decode.yaml
prefill.yaml
base.yaml
scale.yaml
budget.yaml
noreasoning.yaml
```

Such configurations can be edited to produce desired test configuration of token budget, prompt style and more.

E. Evaluation and expected results

Post-processing: After running the benchmarks, process the raw logs with the `token2metrics` module located at `edgereasoning/third_party/token2metrics`. This step aggregates per-token latency and power measurements. The following steps will produce figures 1-5 in `edgereasoning/outputs/`

```
#inside edgereasoning/
python postprocess.py --sub-config prefill
python postprocess.py --sub-config decode
```

Plotting: Figures 1-5 can be produced using token2metrics by running the following

```
cd third_party/token2metrics/prefillenergy/
./run.sh
cd third_party/token2metrics/decodeenergy/
./run.sh
```

Analytical Models: Fitting coefficients are produced along the figures files. These coefficients can be used to update `edgereasoning/models/analytic.yaml` Execute the following commands to test the analytical latency and energy prediction models for the Tegra device.

```
python latency_model.py -i 128 -o 128
python energy_model.py -i 128 -o 128
python energy_model.py --help
```

A successful run creates a summary table of latency, power, and energy metrics. Passing `--verbose` (`-v`) prints the predicted–empirical differences using the raw validation data under `edgereasoning/validation`.

Notebook: For convenience, `edgereasoning/notebook.ipynb` mirrors the full workflow and can be executed end-to-end to reproduce the evaluation and analytical estimates.

F. Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- https://cTuning.org/ae

Appendix B
Evaluations Results Appendix

Table XXI and XI details model performance (accuracy, latency, and cost) on the 3000-question MMLU-Redux benchmark [10] shown in Figs.6,7, and 8.

TABLE X: MMLU-Redux — Base, Quantized (LLMC-AWQ-W4), and Direct (3k samples/row).

Family	**Model**	**Config**	**Acc. (%)**	**Avg toks/question**	**Avg Latency (s)**	**Cost ($/1M toks)**
Base	DSR1-Qwen-1.5B	Distilled	38.3	740.2	18.92	0.024
Base	DSR1-Llama-8B	Distilled	61.7	811.1	87.16	0.111
Base	DSR1-Qwen-14B	Distilled	80.6	1,317.8	259.02	0.215
Base	L1-Max	Distilled	43.8	312.6	7.50	0.013
Quantized	DSR1-Qwen-1.5B	LLMC-AWQ-W4	37.9	698.5	9.93	0.015
Quantized	DSR1-Llama-8B	LLMC-AWQ-W4	57.9	549.1	14.69	0.053
Quantized	DSR1-Qwen-14B	LLMC-AWQ-W4	80.1	1,235.8	—	—
Direct	Qwen2.5-7B-it	Direct	60.9	40.2	4.26	0.019
Direct	Gemma-7B-it	Direct	33.9	44.7	4.71	0.020
Direct	Llama3.1-8B-it	Direct	58.3	63.5	6.60	0.027

TABLE XI: MMLU-Redux — Budgeted decoding (Hard/Soft/NR). *T*=hard limit; *NC*=soft limit (natural completion).

Model	**BudgetType**	**ConfigLabel**	**Acc. (%)**	**Avg toks/question**	**Avg Latency (s)**	**Cost ($/1M toks)**
DSR1-Llama-8B	Soft	128 (NC)	60.4	437.0	46.939	0.096
DSR1-Llama-8B	Soft	256 (NC)	64.3	933.0	97.908	0.109
DSR1-Llama-8B	NR	NR	51.0	182.9	18.661	0.061
DSR1-Llama-8B	Hard	128T	37.9	76.3	7.888	0.031
DSR1-Llama-8B	Hard	256T	41.2	143.6	14.661	0.048
DSR1-Qwen-1.5B	Soft	128 (NC)	35.5	1,474.0	38.001	0.028
DSR1-Qwen-1.5B	Soft	256 (NC)	39.4	734.8	18.175	0.021
DSR1-Qwen-1.5B	NR	NR	41.0	234.9	5.644	0.012
DSR1-Qwen-1.5B	Hard	128T	15.9	91.5	2.221	0.005
DSR1-Qwen-1.5B	Hard	256T	23.2	144.1	3.468	0.007
DSR1-Qwen-14B	Soft	128 (NC)	76.9	599.0	118.091	0.189
DSR1-Qwen-14B	Soft	256 (NC)	77.2	374.2	70.917	0.152
DSR1-Qwen-14B	NR	NR	69.0	180.7	34.201	0.115
DSR1-Qwen-14B	Hard	128T	46.1	78.2	15.013	0.064
DSR1-Qwen-14B	Hard	256T	58.6	112.9	21.485	0.082
L1-Max	Soft	128 (NC)	17.8	54.3	1.353	0.004
L1-Max	Soft	256 (NC)	17.1	62.3	1.552	0.005
L1-Max	Hard	128T	16.2	40.7	1.019	0.003
L1-Max	Hard	256T	18.3	48.9	1.213	0.003

Table XII lists the additional model performance evaluation (accuracy, latency, and cost) on the MMLU benchmark [13] with 15k questions.

TABLE XII: MMLU [13] accuracy (15k questions) for base, quantized, and budgeted DSR1 models.

Model	Configuration	Accuracy (%)	Avg toks/q
DSR1-Qwen-1.5B			
	Base	41.67	1141.6
	Budget 128T	24.60	88.7
	Budget 256T	29.60	113.7
	LLMC-AWQ-W4	37.73	984.4
	Budget 128T	24.60	86.9
	Budget 256T	29.10	120.4
DSR1-Llama-8B			
	Base	60.38	345.6
	Budget 128T	31.03	101.5
	Budget 256T	41.80	169.3
	LLMC-AWQ-W4	60.44	455.4
	Budget 128T	32.10	97.7
	Budget 256T	43.50	157.1
DSR1-Qwen-14B			
	Base	86.59	1145.4
	Budget 128T	28.30	193.4
	Budget 256T	37.70	185.7
	LLMC-AWQ-W4	86.69	1148.4
	Budget 128T	27.10	109.6
	Budget 256T	37.10	162.0

Tables XIII, XIV, and XV show the model performance evaluation (accuracy and latency) on the Natural-Plan benchmark [42].

TABLE XIII: Baseline (reasoning models)

Task	Model	Acc. (%)	Avg out toks/Q	Lat. (s)
calendar	1.5B	0.60	2792	8.90
meeting	1.5B	1.00	3880	19.90
trip	1.5B	1.25	2490	7.88
calendar	8B	9.00	2798	21.10
meeting	8B	10.00	2866	24.50
trip	8B	7.88	2251	17.10
calendar	14B	11.70	2297	30.00
meeting	14B	19.30	1494	22.10
trip	14B	13.88	2340	30.40

TABLE XIV: Budgeting (NR + Hard limit at 512 tokens)

Task	Model	Acc. (%)	Avg out toks/Q	Lat. (s)
calendar	1.5B	2.00	511	2.840
meeting	1.5B	1.90	425	1.350
trip	1.5B	0.00	507	1.420
calendar	8B	8.10	67	0.552
meeting	8B	11.90	284	2.510
trip	8B	3.90	398	3.094
calendar	14B	12.60	40	0.615
meeting	14B	19.00	341	5.223
trip	14B	10.90	380	4.984

TABLE XV: Direct models (Qwen2.5)

Task	Model sz.	Acc. (%)	Avg out toks/Q	Lat. (s)
calendar	1.5B	5.30	22	0.087
meeting	1.5B	9.40	271	1.369
trip	1.5B	2.50	242	0.804
calendar	14B	31.90	28	0.464
meeting	14B	27.20	283	4.408
trip	14B	6.44	259	3.440

APPENDIX C
EDGE CPU EVALUATION

This section presents characterization results for a 12-core Arm Cortex-A78AE CPU, evaluated as an alternative inference platform.

TABLE XVI: Prefill Latency: CPU vs. GPU

Len	1.5B		8B		14B	
	CPU (s)	GPU (s)	CPU (s)	GPU (s)	CPU (s)	GPU (s)
128	8.44	0.051	46.5	0.148	79.29	0.270
256	17.0	0.054	89.7	0.223	167.0	0.421
512	37.1	0.095	157	0.554	344.2	0.764
1024	75.6	0.158	384	0.801	734.2	1.521

TABLE XVII: Decode Latency: CPU vs. GPU

Output Length	8B		14B	
	CPU (s)	GPU (s)	CPU (s)	GPU (s)
64	259.9	52.1	461.7	95.3
128	63.8	12.9	113.5	23.7
256	128.8	26.1	228.8	47.5
1024	521.5	104.5	926.5	190.5

APPENDIX D
QUANTIZED MODELS EVALUATION

Tables XVIII and XIX present a performance comparison between the base FP16 models and their W4A16-quantized counterparts.

TABLE XVIII: Prefill Performance: Base vs Quantized. Averaged across input length sweep range [128, 4096]

Model	Time (s)	Tok/s	Power (W)
Base			
DSR1-Qwen-1.5B	0.33	5.3	5.6
DSR1-Llama-8B	2.60	1.2	17.0
DSR1-Qwen-14B	3.63	0.7	23.5
Quantized (AWQ W4)			
DSR1-1.5B–AWQ-W4	0.15	9.8	4.8
DSR1-8B–AWQ-W4	0.55	5.1	13.6
DSR1-14B–AWQ-W4	2.21	1.8	20.5

TABLE XIX: Decode Performance: Base vs Quantized Decode. Input length 512: and output length sweep range: [128, 2048]

Model	Time (s)	Tok/s	Power (W)
Base (distilled)			
DeepSeek-R1-Distill-Qwen-1.5B	20.86	38.2	19.6
DeepSeek-R1-Distill-Llama-8B	86.42	9.0	24.4
DeepSeek-R1-Distill-Qwen-14B	158.18	5.0	26.5
Quantized (AWQ W4)			
DSR1-1.5B-llmc-awq-w4	10.64	73.6	16.2
DSR1-8B-llmc-awq-w4	29.94	25.9	25.4
DSR1-14B-llmc-awq-w4	51.06	15.1	28.5

APPENDIX E
APPENDIX: EDGEREASONING

A. *Fitted Coefficients for Energy and Power Modeling*

TABLE XX: Fitted parameters for *prefill* power and energy models (DeepSeek R1 distilled). I: input length in tokens.

Model	**Power Function**	**Energy Function**	**Key Parameters**
1.5B	Constant: $P = 5.636$	Exp. decay: $E = Ae^{-\lambda I} + C$	$A = 0.07308$, $\lambda = 0.03195$, $C = 0.000923$
8B	Const. ($I \leq 800$), Log ($I > 800$)	Piecewise: Exp. decay ($I \leq 640$), Log ($I > 640$)	Exp: $A = 0.15871$, $\lambda = 0.03240$, $C = 0.00553$; Log: $\alpha = 0.01233$, $\beta = -0.07349$
14B	Const. ($I \leq 384$), Log ($I > 384$)	Piecewise: Exp. decay ($I \leq 384$), Log ($I > 384$)	Exp: $A = 0.29327$, $\lambda = 0.03058$, $C = 0.009234$; Log: $\alpha = 0.01605$, $\beta = -0.07643$

TABLE XXI: Fitted parameters for *decode* power and energy models (distilled models). O output length in tokens.

Model	**Power Function**	**Energy Function**	**Key Parameters**
1.5B	Log: $P = \alpha \ln O + \beta$	Log: $E = \alpha \ln O + \beta$	Power: $\alpha = 0.756538$, $\beta = 3.213711$; Energy: $\alpha = -0.059992$, $\beta = 0.091465$
8B	Log: $P = \alpha \ln O + \beta$	Log: $E = \alpha \ln O + \beta$	Power: $\alpha = 8.806744$, $\beta = 2.701709$; Energy: $\alpha = 0.555184$, $\beta = 0.324112$
14B	Log: $P = \alpha \ln O + \beta$	Log: $E = \alpha \ln O + \beta$	Power: $\alpha = 16.886830$, $\beta = 1.619387$; Energy: $\alpha = 1.764896$, $\beta = 0.515518$

TABLE XXII: Fitted parameters for prefill power and energy models for quantized models. I: input length in tokens.

Model	Power Function	Energy Function	Key Parameters
1.5B	Constant: $P = 4.83$	Exp. decay: $E = 0.093e^{-0.109N} + 0.0011$	$A = 0.093$, $\lambda = 0.109$, $C = 0.0011$
14B	Const. ($I \leq 384$), Log ($I > 384$)	Piecewise: Exp. decay ($I \leq 640$), Log ($I > 640$)	Exp: $A = 0.160$, $\lambda = 0.129$, $C = 0.008$; Log: $\alpha = 0.0157$, $\beta = -0.089$
8B	Const. ($I \leq 1400$), Log ($I > 1400$)	Piecewise: Exp. decay ($I \leq 1500$), Log ($I > 1500$)	Exp: $A = 0.101$, $\lambda = 0.121$, $C = 0.0037$; Log: $\alpha = 0.0066$, $\beta = -0.040$

TABLE XXIII: Fitted parameters for *decode* power and energy models (quantized W4). O: output length in tokens

Model	**Power Function**	**Energy Function**	**Key Parameters**
DSR1-Qwen-1.5B-W4	Log: $P = \alpha \ln O + \beta$	Log: $E = \alpha \ln O + \beta$	Power: $\alpha = 3.0401$, $\beta = -1.6672$; Energy: $\alpha = 0.04338$, $\beta = -0.05468$
DSR1-Llama-8B-W4	Log: $P = \alpha \ln O + \beta$	Log: $E = \alpha \ln O + \beta$	Power: $\alpha = 3.8723$, $\beta = 3.0186$; Energy: $\alpha = 0.15962$, $\beta = -0.05413$
DSR1-Qwen-14B-W4	Log: $P = \alpha \ln O + \beta$	Log: $E = \alpha \ln O + \beta$	Power: $\alpha = 3.0515$, $\beta = 11.0898$; Energy: $\alpha = 0.24460$, $\beta = 0.24737$