PACE: RETHINKING ON-DEVICE LLM EFFICIENCY FROM TOKEN THROUGHPUT TO TIMELY CORRECTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

On-device LLM systems are often optimized for token throughput or energy per token, but these metrics do not capture whether a request is answered correctly and on time. We present PACE, a task-aware framework for evaluating on-device LLM efficiency through timely accuracy (i.e., the fraction of requests answered correctly before a deadline) and total request energy. PACE studies how deployment, execution mode, and generation budget jointly affect task completion. Across three LLMs and three deployment paths on two real mobile phones, we find that similar token-level efficiency can hide large differences in task-level cost: two deployments with only a 1.01% difference in energy per token differ by 2.14× in energy per timely correct answer at a 10-second deadline. We further find that faster execution matters only when it helps correct responses meet the deadline, while larger generation budgets matter only when the extra tokens improve answer correctness. We then introduce PACE-SELECT, a simple calibration-based rule that chooses the lowest-energy configuration satisfying a target timely accuracy. Our results show that the most efficient configuration depends on the task requirement and deadline, and that on-device LLM efficiency should be evaluated by timely task completion rather than token efficiency alone.

1 INTRODUCTION

On-device LLM inference is becoming increasingly efficient through quantization, mobile NPUs, and specialized runtimes (Frantar et al., 2022; Xu et al., 2024; Wei et al., 2025). Yet these systems are still commonly evaluated using execution-level metrics such as token throughput, latency, and energy per token (Laskaridis et al., 2024; Li et al., 2025; Zou et al., 2026). Such metrics describe how efficiently a model executes, but not whether a user receives a useful result. For many applications, a request is useful only if the model produces a correct answer within a required time and at an acceptable energy cost. This raises a more practical question: when does better token-level efficiency actually translate into better task completion?

This question is particularly important for on-device scenarios. A request must run within the compute and energy budget of a single device, while mobile runtimes expose different execution paths and resource settings (Xu et al., 2024; Xue et al., 2024; Zou et al., 2026). At the same time, generation length can directly affect both answer quality and inference cost, especially for reasoning workloads (Snell et al., 2024; Han et al., 2024; Chen et al., 2024). These choices therefore affect not only how fast the model runs, but also whether it can complete the task within the application's time and energy constraints. For a local assistant, the relevant unit of efficiency is therefore not a generated token, but a completed request.

Figure 1 illustrates this mismatch. On OnePlus/GSM8K (Cobbe et al., 2021), two Qwen2.5-3B (Qwen et al., 2025) deployments differ by only 1.01% in mean energy per token. At a 10-second deadline, however, their timely accuracies are 31.2% and 63.0%, and the former requires $2.14\times$ more energy per timely correct answer. Thus, nearly identical token costs can correspond to very different costs of delivering a correct result on time.

We present PACE, a task-aware framework for evaluating on-device LLM efficiency at the request level. PACE measures timely accuracy—the fraction of requests answered correctly before a

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.

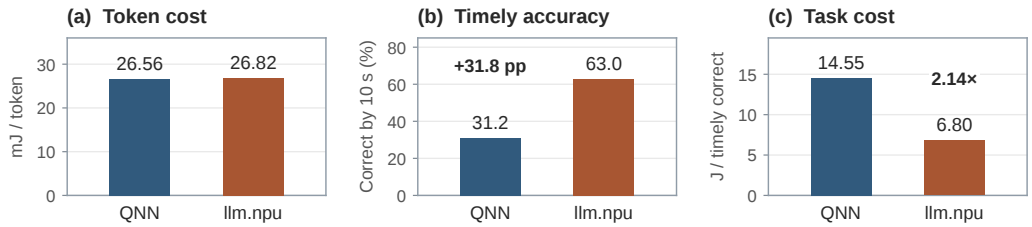

Figure 1: Token and task costs for ExecuTorch/QNN (QNN) and llm.npu on OnePlus/GSM8K. Balanced modes, budget 512. (a) Mean per-request energy/token. (b) Timely accuracy at 10 seconds. (c) Energy per timely correct answer, including energy from incorrect and late requests.

deadline—together with complete-request energy. It studies three configuration choices that jointly shape task completion: deployment, execution mode, and generation budget. Instead of asking which configuration is fastest or has the lowest energy per token, PACE asks which configurations can satisfy a required level of timely accuracy and how much energy they require to do so.

Our measurements reveal three effects that make this task-level view necessary. First, faster execution matters only when it helps correct responses meet the deadline; once those responses already finish in time, further speedup does not improve timely accuracy and may only increase energy cost. Second, larger generation budgets matter only when the extra tokens improve answer correctness; otherwise they increase latency and energy without improving task completion. Third, final accuracy limits what acceleration can achieve: if a deployment does not produce enough correct answers, making it faster cannot satisfy a higher timely-accuracy target. These observations connect two lines of work that are often evaluated separately: efficient on-device LLM execution (Xu et al., 2024; Wei et al., 2025; Zou et al., 2026) and adaptive test-time computation (Snell et al., 2024; Han et al., 2024).

These observations lead to a simple principle for configuration selection method PACE-SELECT: *establish feasibility before optimizing efficiency.* Given a deadline D and a required timely accuracy q, PACE-SELECT first uses calibration requests to identify configurations that satisfy the task requirement, and then selects the one with the lowest mean request energy. If no measured configuration is feasible, it makes no selection. Unlike prior energy-aware approaches centered on execution-level constraints and configuration costs (Chung et al., 2025; Zou et al., 2026), PACE-SELECT explicitly uses timely task correctness as the feasibility condition.

Across three LLM families, three deployment paths, and two Snapdragon 8 Elite phones, our measurements show that the value of execution speed and generation budget depends strongly on the task requirement and deadline. For example, on llm.npu/GSM8K with a 20-second deadline and a 60% timely-accuracy target, PACE-SELECT reduces mean request energy by 38.6% relative to selecting the fastest execution mode, while both policies meet the target across all evaluated splits. More broadly, our results show that on-device LLM efficiency should be evaluated by the cost of timely task completion rather than token efficiency alone.

2 Task-oriented evaluation framework

Evaluating task completion. PACE asks which on-device configuration uses the least energy while delivering enough correct answers within a given deadline. This requires evaluating the full request, from input processing (prefill) through response generation (decoding), including runtime overhead. We describe a configuration as $c = (r, m, b)$, where r identifies the deployed model and inference framework, m is the execution setting, and b is the output-token budget. We apply one configuration to a batch of requests and measure the correctness, completion time, and energy of each request. These choices determine the execution path, resources, and generation allowance whose task outcomes we compare. Figure 2 summarizes how PACE connects configuration measurements and task-level analysis to configuration selection under a deadline and accuracy requirement.

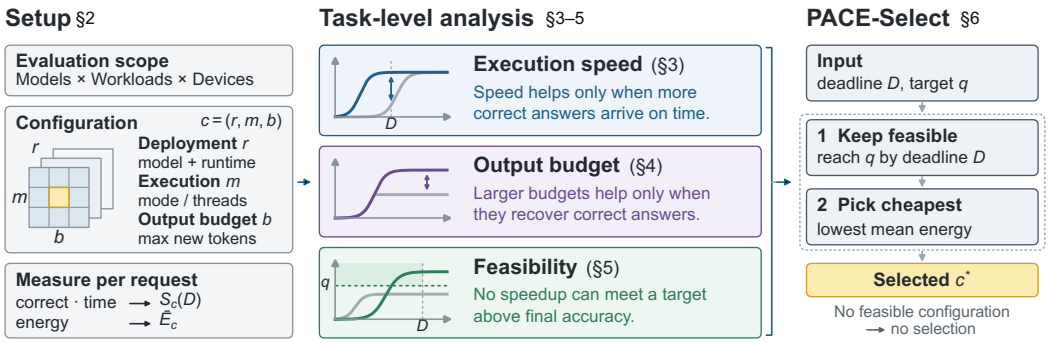

Figure 2: Overview of PACE.

Task requirements and energy cost. A task requirement (D, q) specifies a deadline D and the minimum fraction $q \in [0, 1]$ of requests that must be answered correctly by that deadline. For n included requests on a fixed workload, let $Y_{ic} \in \{0, 1\}$ indicate whether configuration c answers request i correctly, and let T_{ic} be its complete-request time in seconds. Timely accuracy is the fraction of requests that meet both conditions:

$$S_c(D) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{Y_{ic} = 1,\ T_{ic} \le D\}. \tag{1}$$

Configuration c meets the task requirement when $S_c(D) \ge q$. Final accuracy, $A_c = n^{-1} \sum_i Y_{ic}$, counts correct answers regardless of when they arrive. Since $S_c(D) \le A_c$, a configuration with $A_c < q$ cannot meet the requirement at any deadline (Section 5).

Among configurations that meet the requirement, we minimize mean request energy $\bar{E}_c = n^{-1} \sum_i E_{ic}$, where E_{ic} is request energy in joules. Section 6 uses calibration data to make this choice. Time and energy cover the full inference window after model loading. We apply deadlines to completed runs; generation is not stopped at D. Incorrect and late answers therefore still contribute their full energy cost. We also report energy per timely correct answer (EPC). This divides the energy spent on all requests by the number answered correctly on time:

$$\mathrm{EPC}_c(D) = \frac{\bar{E}_c}{S_c(D)}. \tag{2}$$

EPC is defined when $S_c(D) > 0$.

Experimental setup. We evaluate configurations $c = (r, m, b)$ across the following tasks and devices.

Datasets. GSM8K contains 1,319 grade-school mathematics questions, and MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023) (MATH below) contains 500 competition-level mathematics problems. We use them to study answer correctness, deadline requirements, and configuration selection. HammerBench single-turn contains 654 function-calling requests and is used to analyze the distribution of request time across execution stages.

Models and deployments. Our primary model, Qwen2.5-3B (Qwen et al., 2025), runs through three inference paths: ExecuTorch/QNN (QNN) and llm.npu on the NPU, and T-MAC on the CPU. The deployed model and inference framework together specify r. BitNet-2B (Ma et al., 2025) extends the execution-mode and output-budget analyses on QNN, while Llama 3.1-8B Grattafiori et al. (2024) supports the cross-model comparison on QNN.

Devices and execution settings. Experiments primarily run on OnePlus smartphones, with supplementary measurements on Redmi; both use Snapdragon 8 Elite processors. The setting m varies NPU operating modes or CPU thread counts, while b specifies the output-token budget.

Evaluation metrics. As defined above, final accuracy A_c measures answer correctness, and timely accuracy $S_c(D)$ measures correctness by deadline D. We use mean request energy $\bar{E}_c$ to com-

pare configurations meeting (D, q), while $\mathrm{EPC}_c(D)$ reports energy per timely correct answer. Appendix A gives the detailed configurations and measurement protocol.

3 Execution Speed

Execution settings change when an answer arrives and how much energy the request consumes, but not the answer itself. On OnePlus/GSM8K at budget 512, six execution-setting comparisons across QNN, llm.npu, and T-MAC produce identical generated text on all 1,319 paired requests (Appendix A.3). We can therefore study how execution affects timely accuracy and request energy while task outcomes stay fixed.

Deadline-sensitive value of speed. Faster execution improves timely accuracy only when it moves correct answers across the deadline. For Qwen2.5-3B under llm.npu, burst delivers 55.09% more timely correct answers than balanced at a 5-second deadline. By 20 seconds, both modes have delivered the same 915 correct answers, and balanced uses 38.5% less mean request energy (Figure 3a,b). Past this point, extra speed adds energy without adding timely accuracy.

Energy savings under relaxed deadlines. When the deadline is sufficiently loose, a slower mode can meet the same timely-accuracy target with less energy. BitNet-2B shows the same pattern from the low-power side: at 10 seconds, low-power delivers 589 timely correct answers versus 858 for balanced, but both deliver 923 by 30 seconds. Low-power then uses 9.77% less mean request energy over the 1,318 requests with energy records (Figure 3d; Appendix E.2). What matters is not whether a mode is faster overall, but whether it is fast enough for the deadline.

Power–energy mismatch. Lower instantaneous power does not necessarily reduce the energy consumed by a complete request. For Qwen2.5-3B under QNN, low-power nearly halves mean power but more than doubles completion time, so mean request energy is 9.1% higher than balanced (Figure 3c; Appendix A). Energy savings must therefore be measured over the complete request, not inferred from power or a mode label.

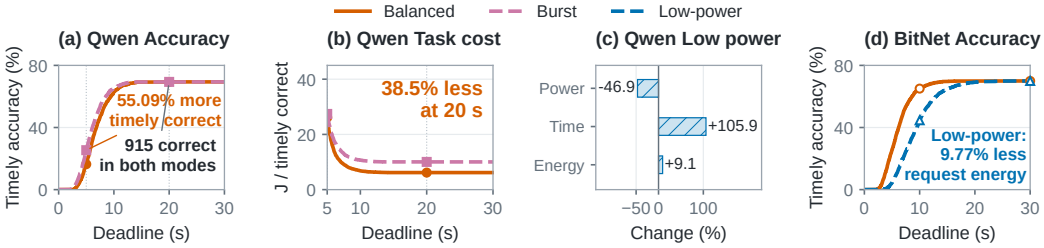

Figure 3: Execution-mode trade-offs on OnePlus/GSM8K, budget 512. (a,b) Qwen2.5-3B under llm.npu: timely accuracy and energy per timely correct answer. (c) Qwen2.5-3B under QNN: low-power relative to balanced. (d) BitNet-2B (W4A16): timely accuracy.

Limits of stage-level speedup. The end-to-end benefit of accelerating one stage depends on how much of the request time that stage occupies. For six-thread T-MAC, TTFT takes 92% of aggregate request time on HammerBench single-turn but only 36% on GSM8K (Figure 4a). With TTFT unchanged, halving the remaining time would cut mean request time by 4% on HammerBench but by 32% on GSM8K (Appendix B.1). The same stage optimization can thus matter greatly for one workload and little for another.

TTFT versus request latency. A configuration with the lowest TTFT does not necessarily finish the complete request fastest. In a controlled T-MAC experiment on 12 GSM8K questions with three repeats each, $(8, 8)$ prefill/decode threads give the lowest mean TTFT, yet $(8, 6)$ finishes requests 21.0% faster than $(8, 8)$ on average (Figure 4b). The gap lies after the first token: with prefill fixed at six threads, moving decode from six to eight threads slows decoding in 35 of 36 paired runs (Appendix B.1). Deadline attainment should therefore be judged by complete-request time, not by TTFT alone.

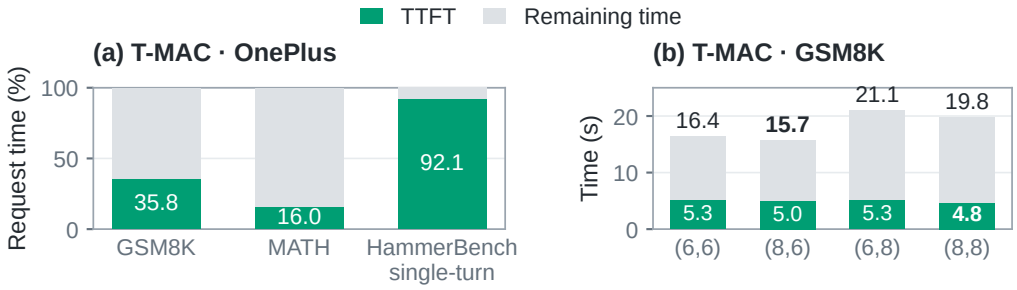

Figure 4: Stage timing for T-MAC on OnePlus; remaining time is $T - \mathrm{TTFT}$. (a) Stage shares of aggregate request time with six threads; GSM8K and MATH use budget 512, and HammerBench single-turn uses *No cap*. (b) Mean stage times for four (prefill, decode) thread settings on 12 GSM8K questions; bar labels give mean complete-request time.

> **Takeaway.** Faster execution is useful only when it helps more correct answers arrive before the deadline. Once multiple settings meet the required timely accuracy, extra speed brings no further task-level benefit, so the lower-energy setting is preferable. We should evaluate execution modes using complete-request time and energy.

4 Output Budget

Increasing the output budget can recover correct answers, but the useful range differs by task. In the measured Qwen2.5-3B deployments, GSM8K answer recovery saturates at 512 tokens, whereas MATH continues to gain correct answers when this cap is removed. We examine which requests benefit from longer outputs and how much additional generation accompanies those gains, connecting work on reasoning budgets and test-time computation to on-device configuration choice (Wei et al., 2022; Snell et al., 2024; Han et al., 2024; Chen et al., 2024). Comparisons pair requests within each deployment and device in reference modes: balanced for the NPU paths and six threads for T-MAC. *No cap* removes the additional output limit while retaining EOS termination and the 1,024-token total context limit.

Task-dependent budget saturation. On OnePlus/GSM8K, increasing T-MAC's budget from 256 to 512 recovered 69 correct answers and lost two, a net gain of 67. All 69 recovered answers had been length-terminated at 256. Relaxing 512 to *No cap* left the set of correct answers unchanged (Figure 5). QNN and llm.npu on OnePlus likewise recovered answers up to 512 and preserved the same correct-answer sets beyond it (Table 1); the paired llm.npu/Redmi results agree (Appendix B.4).

The same 512-token cap still restricted answer recovery on OnePlus/MATH. Relaxing it to *No cap* recovered 62, 46, and 32 answers for T-MAC, QNN, and llm.npu, respectively, among valid generation pairs, with no reverse transitions. Every recovered answer had been length-terminated at 512. Thus, the GSM8K plateau does not identify a sufficient budget for MATH: the paired records locate further answer recovery in requests cut short by the smaller cap.

Where additional tokens go. Table 1 groups per-request output-length increases by correctness at both budgets. From 256 to 512 on GSM8K, roughly half of T-MAC's added tokens went to requests whose answers became correct, while over 80% of QNN's and llm.npu's went to requests that remained incorrect. Beyond 512, all added tokens in the GSM8K pairs came from requests incorrect at both budgets. On MATH, continued answer recovery coexisted with 77.1%–90.3% of added tokens going to requests that remained incorrect across the three deployments. These paired measurements separate generation associated with answer recovery from generation on requests that remain unresolved.

Cross-model budget behavior. BitNet-2B provides a further instance of the GSM8K plateau on OnePlus in balanced mode (Figure 5d). Increasing 256 to 512 tokens recovered nine answers and lost one, raising the correct count from 915 to 923. Relaxing 512 to *No cap* preserved all 923 correct answers; its 4,833 additional tokens came entirely from requests that remained incorrect

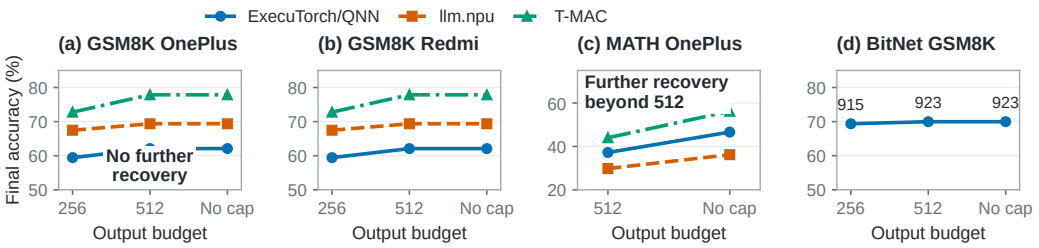

Figure 5: Task-dependent final-accuracy returns across output budgets. (a–c) Qwen2.5-3B in reference modes on OnePlus/GSM8K, Redmi/GSM8K, and OnePlus/MATH. (d) BitNet-2B, OnePlus/GSM8K, balanced; labels give correct counts.

Table 1: Paired answer transitions and added tokens for Qwen2.5-3B on OnePlus in reference modes. W/C denote incorrect/correct; "Still wrong" is the added-token share from W→W pairs.

Deployment	Budget	Pairs	W→C / C→W	Added tokens	Added tokens on W→W pairs (%)
GSM8K / OnePlus					
T-MAC	256 → 512	1319	69/2	9,767	47.60%
T-MAC	512 → *No cap*	1319	0/0	1,353	100.00%
QNN	256 → 512	1319	35/0	11,608	80.64%
QNN	512 → *No cap*	1319	0/0	3,486	100.00%
llm.npu	256 → 512	1319	25/0	5,794	81.29%
llm.npu	512 → *No cap*	1319	0/0	434	100.00%
MATH / OnePlus					
T-MAC	512 → *No cap*	496	62/0	43,738	77.14%
QNN	512 → *No cap*	498	46/0	52,339	86.78%
llm.npu	512 → *No cap*	500	32/0	51,427	90.29%

(Appendix E.2). Across these comparisons, added tokens measure generation volume; the budget decision in Section 6 evaluates the configurations using measured complete-request time and energy.

> **Takeaway.** A larger output budget is useful only when the extra tokens recover correct answers. If the additional generation does not improve correctness, it only adds time and energy. Because the useful budget differs across tasks, it should be chosen based on timely accuracy and request energy under the target deadline.

5 Task Feasibility

Feasibility before efficiency. Sections 3 and 4 show that execution speed and generation budget affect two complementary parts of timely task completion: when correct answers arrive and which answers can be recovered. For a task requirement (D, q), a configuration is useful only if it answers at least a fraction q of requests correctly by deadline D. We summarize this condition with the task-feasibility frontier, the earliest observed deadline at which configuration c reaches target q:

$$D_c^\star(q) = \inf\{D \geq 0 : S_c(D) \geq q\}. \tag{3}$$

For $0 < q \leq A_c$, $D_c^\star(q)$ is the $\lceil nq \rceil$-th smallest completion time among correct requests. If $A_c < q$, then $D_c^\star(q) = \infty$: no deadline can make the measured configuration feasible. We take $D_c^\star(0) = 0$. Figure 6 shows these frontiers across deployment paths and models on OnePlus.

Deadline-limited feasibility. On GSM8K at budget 512, balanced llm.npu reaches a 60% timely-accuracy target at 9.407 seconds, ahead of QNN at 18.894 seconds and six-thread T-MAC at 20.151 seconds. T-MAC's timely correct count continues to rise after 20 seconds: from 788 at 20 seconds to 941 at 25 seconds and 994/1,319 (75.36%) at 30 seconds. It is the only one of these three deployment

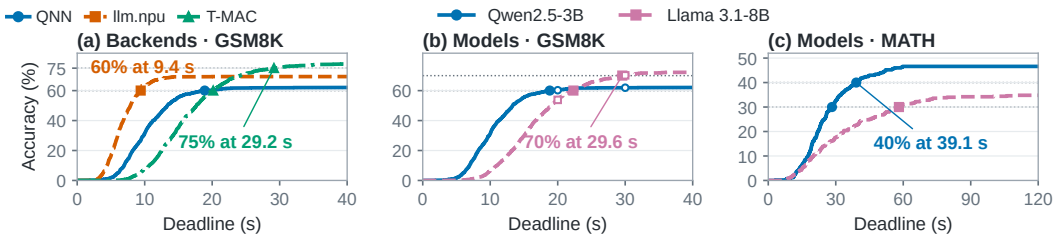

Figure 6: **Task-feasibility frontiers on OnePlus.** Filled markers show the earliest observed deadline $D_c^\star(q)$ at which each configuration reaches the indicated timely-accuracy target. (a) Qwen2.5-3B on GSM8K, budget 512, with balanced NPU modes and six CPU threads. (b–c) QNN, balanced, *No cap*; hollow markers in (b) mark 20 and 30 seconds.

paths to reach a 75% target, with a frontier of 29.153 seconds (Figure 6a). Figure 6b shows the same deadline dependence across models under QNN in balanced mode with *No cap*. Qwen2.5-3B has higher timely accuracy at 20 seconds (60.3% versus 53.8%), while Llama 3.1-8B is higher at 30 seconds (70.2% versus 61.9%). A configuration's correct outputs matter only once they arrive within the application deadline.

Accuracy-limited feasibility. A longer deadline can make late correct answers count, but it cannot increase a measured configuration's final accuracy. On GSM8K at budget 512, QNN and llm.npu reach final accuracies of 62.09% and 69.37%, so neither can satisfy a 75% target at any deadline. T-MAC reaches 77.86% and therefore has a finite frontier for that target. Under QNN with *No cap*, Llama has higher final accuracy than Qwen on GSM8K (72.63% versus 62.09%), whereas Qwen is higher on MATH (46.60% versus 34.80%). Accordingly, Qwen reaches the 40% MATH target at 39.110 seconds, while Llama cannot reach it (Figure 6c). Thus, final accuracy determines whether a finite frontier exists, while completion times determine where that frontier lies.

Energy comparison within the feasible set. On GSM8K under QNN with *No cap*, both Qwen and Llama satisfy a 30-second / 60% requirement. Their mean request energies are 4.35 J and 8.75 J, respectively, over the same 1,319 requests, including incorrect and late answers. When the target is raised to 70%, only Llama remains feasible at 30 seconds; Qwen's lower energy is no longer a valid option for that requirement. Appendix E reports the full cost and deadline tables and outcome breakdown by deployment. PACE-SELECT applies this separation in Section 6: it first screens configurations by timely accuracy and then minimizes mean request energy among the feasible candidates.

> **Takeaway.** The deadline determines whether correct answers arrive in time, while final accuracy determines whether a configuration can ever reach the required accuracy. If its final accuracy is below the target, a longer deadline cannot make it feasible. Energy should be compared only among configurations that can meet the same task requirement.

6 PACE-SELECT: FEASIBILITY-FIRST CONFIGURATION SELECTION

Feasibility-first selection. Sections 3–5 show that execution speed, generation budget, and deployment choice matter insofar as they help satisfy the required timely accuracy. PACE-SELECT turns this principle into a configuration decision. Given a deadline D and minimum timely accuracy q, calibration first identifies the measured configurations $c = (r, m, b)$ that satisfy the requirement, then selects the one with the lowest mean request energy:

$$\widehat{\mathcal{C}}_{\mathrm{ok}}(D, q) = \{c : \widehat{S}_c(D) \geq q\}, \qquad c^\star = \operatorname*{arg\,min}_{c \in \widehat{\mathcal{C}}_{\mathrm{ok}}(D,q)} \widehat{\bar{E}}_c. \tag{4}$$

If no measured candidate satisfies the calibration target, PACE-SELECT returns no selection. Otherwise, the selected configuration is fixed for the evaluation batch.

We replay 23 measured Qwen2.5-3B configurations on OnePlus over 20 overlapping splits, using 20% of each comparison's common requests for calibration and the remainder for evaluation. Candi-

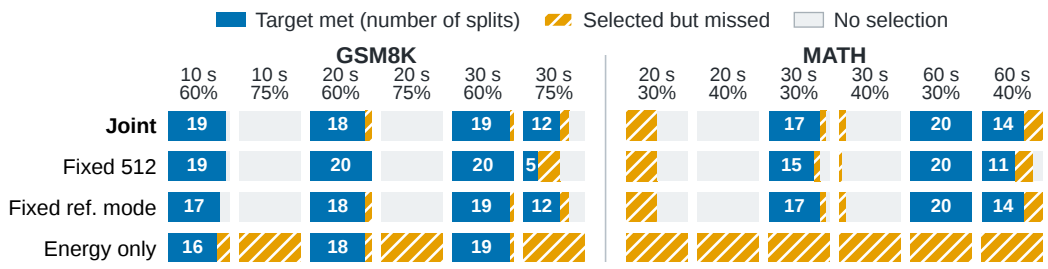

Figure 7: Task requirements determine which configuration flexibility is useful. Each bar divides 20 evaluation splits into target met, selected but missed, and no selection; numbers give splits meeting the target. Fixed 512 and fixed reference mode retain Joint's task screen and energy objective; Energy only omits the task screen. Table 7 gives exact counts.

dates vary deployment path, execution mode, and output budget where measurements are available; Appendix C gives candidate coverage, split construction, and accounting rules.

Task-aware screening. At budget 512 in reference modes, PACE-SELECT screens deployments by timely accuracy before comparing energy. On GSM8K at 10 s / 60%, it meets the evaluation target in 19 of 20 splits, compared with 5 of 20 when selecting the deployment with minimum mean per-request J/token. On MATH at 60 s / 40%, it meets the target in 10 of 20 splits, whereas selecting the minimum-request-energy deployment meets it in none. Low execution-level cost is therefore not a reliable proxy for satisfying the task requirement.

Task-dependent configuration flexibility. Joint selection varies deployment, execution mode, and output budget within the measured candidate set. To isolate the value of each dimension, fixed-reference selection restricts the execution mode, while fixed-512 selection restricts the output budget; both retain the same task screen and energy objective as Joint.

At GSM8K 10 s / 60%, mode flexibility raises the number of splits meeting the target from 17/20 with fixed-reference selection to 19/20 with Joint. Budget flexibility raises this count from 5/20 with fixed 512 to 12/20 at GSM8K 30 s / 75%, and from 11/20 to 14/20 at MATH 60 s / 40% through access to the measured larger-budget candidates. Figure 7 reports the complete grid. These results connect the deadline-dependent value of execution speed in Section 3 and the task-dependent returns to generation in Section 4 to configuration selection.

Energy savings from mode selection. Within each deployment at budget 512, mode selection is compared with the fastest measured mode, defined by calibration mean complete-request time. On llm.npu/GSM8K at 20 s / 60%, both policies meet the target in all 20 evaluation splits. PACE-SELECT reduces mean request energy from 6.965 to 4.280 J, a **38.6% saving**. Once the target is satisfied, the additional speed of the fastest mode provides no further benefit for meeting that requirement, allowing PACE-SELECT to choose the lower-energy mode.

Energy savings from budget selection. To isolate this effect, we compare Joint with fixed-512 selection, which uses the same task screen and energy objective but cannot change the output budget. On MATH at 60 s / 40%, the two policies both meet the target in 11 shared splits. Over these matched successful splits, Joint reduces mean request energy from 22.287 to 13.160 J, a **41.0% saving**. Table 2 combines these paired energy comparisons with target attainment over all 20 splits.

Lower-cost calibration through correctness reuse. Across six reference-to-target mode pairs spanning QNN, llm.npu, and T-MAC, each pair preserves the recorded complete generated text and task outcome on all 1,319 requests. This allows reference-mode correctness to be reused while target-mode execution time and energy are measured on a smaller paired sample. In a separate nine-configuration OnePlus/GSM8K replay at budget 512, paired calibration with $k = 16$ measured requests per target mode yields 63 of 120 decisions that meet the evaluation target, compared with 57 for direct per-mode lookup at the same total calibration request count. It also reduces recorded

Table 2: Representative PACE-SELECT results under matched task requirements. Mode selection compares with the fastest measured mode; Joint compares with fixed-512 selection. PACE denotes PACE-SELECT. Splits meeting target uses all 20 splits; Both gives n_{pair}, the splits where both policies selected a configuration and met the evaluation target. Energy averages evaluation mean request energy over these paired splits; savings compare the unrounded paired means.

Scope	Task requirement	Attainment		n_{pair}	Energy (J)		Saving
		PACE	Baseline		PACE	Baseline	
Mode · llm.npu	GSM8K · 20 s / 60%	20/20	20/20	20	4.280	6.965	**38.6%**
Mode · T-MAC	GSM8K · 30 s / 60%	20/20	20/20	20	11.210	11.450	2.1%
Joint	GSM8K · 30 s / 75%	12/20	5/20	5	11.116	11.472	3.1%
Joint	MATH · 60 s / 40%	14/20	11/20	11	13.160	22.287	**41.0%**

calibration inference time from 4.14 to 3.12 hours and recorded energy from 7.42 to 6.13 kJ. Appendix C.1 reports the full calibration-efficiency curves.

> **Takeaway.** PACE-SELECT first identifies the measured configurations that meet the target on calibration requests, and then chooses the lowest-energy one among them. Which configuration choice matters most—deployment, execution mode, or output budget—depends on the task requirement. When different execution modes produce the same outputs, their correctness can be reused to reduce calibration cost.

7 RELATED WORK

On-device execution and energy efficiency. Quantization and specialized runtimes improve LLM deployment efficiency (Frantar et al., 2022; Lin et al., 2023; Xu et al., 2024; Wei et al., 2025). Serving systems such as DistServe and Sarathi-Serve optimize throughput–latency tradeoffs (Zhong et al., 2024; Agrawal et al., 2024). MELT and PalmBench evaluate mobile model quality and resource costs (Laskaridis et al., 2024; Li et al., 2025), while ML.ENERGY and EnerInfer study energy-efficient configurations and runtime control (Chung et al., 2025; Zou et al., 2026). PACE complements these efforts by evaluating complete deployment paths through timely accuracy and complete-request energy, without attributing cross-artifact differences to individual quantizers or kernels.

Reasoning budgets and configuration selection. Test-time computation and reasoning-budget control adjust inference effort to improve answer quality (Snell et al., 2024; Muennighoff et al., 2025; Han et al., 2024), while overthinking studies identify extended computation with limited additional benefit (Chen et al., 2024). PACE studies output caps on existing phone deployments, using paired records to distinguish truncation-related answer recovery from additional generation on requests that remain incorrect. FrugalGPT and RouteLLM make query-dependent model choices to balance quality and cost (Chen et al., 2023; Ong et al., 2024). Unlike per-query routing, PACE-SELECT uses calibration data to choose one measured deployment, execution mode, and generation budget for a batch, minimizing mean request energy under an empirical timely-accuracy constraint.

8 CONCLUSION

PACE examines how on-device execution efficiency translates into timely correct delivery under task requirements. The evidence links three decisions: evaluate modes at the required deadline, assess budgets through task-dependent answer returns and costs, and screen deployments for feasibility before comparing energy. PACE-SELECT implements these decisions as one empirical configuration choice per batch, minimizing mean request energy among candidates meeting the calibration timely accuracy target. In retrospective replay, task-aware screening meets GSM8K 10 s / 60% in 19 of 20 splits versus 5 for per-token energy selection, and mode selection saves 38.6% of mean request energy over the fastest mode while both meet the target. These results support evaluating execution and generation jointly through timely accuracy and total request energy.

AI USE STATEMENT

Generative AI tools were mainly used to improve writing. Generative AI was not used to generate experimental data or conduct device measurements; all quantitative results were derived from pre-existing request-level measurement records. The authors reviewed the AI-assisted text, verified the reported values against the source records and replay outputs, and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Section 2 defines the evaluation metrics and configuration-selection rule. Appendix A documents the devices, deployments, workloads, scoring procedures, energy-measurement scope, and handling of missing records, while Appendix C specifies the selection replay protocol. These descriptions support reproduction of the reported analysis and configuration-selection results from the recorded measurements.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve, 2024. URL https://arxiv.org/abs/2403.02310.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance, 2023. URL https://arxiv.org/abs/2305.05176.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMs, 2024. URL https://arxiv.org/abs/2412.21187.

Jae-Won Chung, Jeff J. Ma, Ruofan Wu, Jiachen Liu, Oh Jun Kweon, Yuxuan Xia, Zhiyu Wu, and Mosharaf Chowdhury. The ML.ENERGY Benchmark: Toward Automated Inference Energy Measurement and Optimization, 2025. URL https://arxiv.org/abs/2505.06371.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers, 2022. URL https://arxiv.org/abs/2210.17323.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-Budget-Aware LLM Reasoning, 2024. URL https://arxiv.org/abs/2412.18547.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. Melting point: Mobile evaluation of language transformers, 2024. URL https://arxiv.org/abs/2403.12844.

Yilong Li, Jingyu Liu, Hao Zhang, M Badri Narayanan, Utkarsh Sharma, Shuai Zhang, Pan Hu, Yijing Zeng, Jayaram Raghuram, and Suman Banerjee. Palmbench: A comprehensive benchmark of compressed large language models on mobile platforms, 2025. URL https://arxiv.org/abs/2410.05315.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step, 2023. URL https://arxiv.org/abs/2305.20050.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration, 2023. URL https://arxiv.org/abs/2306.00978.

Shuming Ma, Hongyu Wang, Shaohan Huang, Xingxing Zhang, Ying Hu, Ting Song, Yan Xia, and Furu Wei. BitNet b1.58 2B4T Technical Report, 2025. URL https://arxiv.org/abs/2504.12285.

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, 2025. URL https://arxiv.org/abs/2501.19393.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs with Preference Data, 2024. URL https://arxiv.org/abs/2406.18665.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Jun Wang, Jiamu Zhou, Muning Wen, Xiaoyun Mo, Haoyu Zhang, Qiqiang Lin, Cheng Jin, Xihuai Wang, Weinan Zhang, Qiuying Peng, and Jun Wang. HammerBench: Fine-Grained Function-Calling Evaluation in Real Mobile Device Scenarios, 2024. URL https://arxiv.org/abs/2412.16516.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2022. URL https://arxiv.org/abs/2201.11903.

Jianyu Wei, Shijie Cao, Ting Cao, Lingxiao Ma, Lei Wang, Yanyong Zhang, and Mao Yang. T-mac: Cpu renaissance via table lookup for low-bit llm deployment on edge, 2025. URL https://arxiv.org/abs/2407.00088.

Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. Fast on-device llm inference with npus, 2024. URL https://arxiv.org/abs/2407.05858.

Zhenliang Xue, Yixin Song, Zeyu Mi, Xinrui Zheng, Yubin Xia, and Haibo Chen. PowerInfer-2: Fast Large Language Model Inference on a Smartphone, 2024. URL https://arxiv.org/abs/2406.06282.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving, 2024. URL https://arxiv.org/abs/2401.09670.

Bohua Zou, Nian Liu, Binqi Sun, Matteo Mascherin, Debayan Roy, Yutao Liu, Yu Peng, Ning Jia, and Haibo Chen. EnerInfer: Energy-Aware On-Device LLM Inference, 2026. URL https://arxiv.org/abs/2606.23001.

A MEASUREMENT AND DATA HANDLING

A.1 DEVICES, WORKLOADS, AND DEPLOYMENTS

Section 2 names the models, devices, and workloads; this subsection records the measurement protocol behind them. GSM8K uses the full 1,319-question split on each complete configuration (Cobbe et al., 2021), MATH-500 uses the subset documented by PRM800K (Lightman et al., 2023), and HammerBench single-turn covers 654 requests (Wang et al., 2024); all three run on OnePlus, with GSM8K repeated on Redmi. Llama 3.1-8B supports the cross-model comparison on QNN.

Table 3 summarizes the deployed Qwen2.5-3B artifacts. The Llama 3.1-8B comparison in Section 5 uses OnePlus/QNN in balanced mode with *No cap*; Appendix E specifies its coverage and paired costs. The BitNet-2B mode and budget comparisons use a W4A16 artifact with QNN 2.28 and its native prompt template (Appendix E.2). The experiment label llm.npu denotes a mllm/QNN ahead-of-time (AOT) bundle invoked through the PACE benchmark and measurement harness; it identifies the measured deployment rather than a reproduction of the original llm.npu paper's system.

Table 3: Deployment properties recorded for the measured artifacts. All paths use a 1,024-token total context limit. G32 and G128 denote quantization group sizes.

Property	QNN	llm.npu	T-MAC
Execution path	ExecuTorch hybrid PTE / QNN HTP	mllm / QNN AOT	CPU table lookup
Runner	`qnn_llama_runner_htp_mode`	`mllm-qwen2-aot-runner`	`llama-cli`
Software / target	QNN 2.37	QAIRT 2.37 / HTP v79	T-MAC / llama.cpp
Quantization	W4A16	LPBQ W4A16 G32; W8A8 KV cache	GPTQ INT4 G128 (Frantar et al., 2022)
Prefill / decode	128-token prefill graph	32-token prefill; single-token decode graph	Same thread count for prefill (`-tb`) and decode (`-t`)

The two Qwen NPU runners use the same rendered mathematics prompt template, with actual token counts recorded separately. Artifacts, conversion, tokenization, and numerical representations can differ across deployments; cross-deployment comparisons evaluate these choices together as the complete serving path. The QNN modes are `balanced`, `burst`, and `low_power`; the llm.npu modes are `balanced`, `burst`, and `low_power_saver`. T-MAC uses four, six, or eight threads. Output budgets are 256 tokens, 512 tokens, and *No cap*; the latter retains EOS termination and the total context limit, as specified in the measurement protocol below. Reference mathematics configurations use balanced NPU modes or six T-MAC threads, with a 512-token output budget. Table 4 lists the observed combinations used in the analyses. Each row specifies its own mode–budget combinations; unmeasured combinations are not imputed.

Table 4: Measured Qwen2.5-3B configuration coverage. For GSM8K, each entry applies to both OnePlus and Redmi. Reference settings are balanced for the NPU paths and six threads for T-MAC. Llama 3.1-8B coverage is specified in Appendix E.

Workload / device	Deployment	Reference-setting budgets	Other settings and budgets
GSM8K / both	QNN	256, 512, *No cap*	Burst, low-power: 512
	llm.npu	256, 512, *No cap*	Burst, low-power-saver: 512
	T-MAC	256, 512, *No cap*	Four, eight threads: 512
MATH / OnePlus	QNN	256, 512, *No cap*	Burst: 512
	llm.npu	512, *No cap*	—
	T-MAC	512, *No cap*	—
HammerBench / OnePlus	T-MAC	*No cap*	—

A.2 Measurement protocol

Complete-request time T_{ic} and energy E_{ic} cover the inference window after model loading has finished. This window excludes model loading, configuration switching, calibration acquisition, and selection overhead. Deadlines classify completed runs retrospectively: generation does not stop at D. Mean request energy therefore includes the full recorded energy of incorrect and late answers and is independent of the analysis deadline. *No cap* removes the additional output limit while retaining end-of-sequence (EOS) termination and the 1,024-token total context limit; available output capacity depends on input length.

Time is divided into recorded time to first token (TTFT) and the remainder $T - \mathrm{TTFT}$, including subsequent generation and runtime overhead. The T-MAC and llm.npu runners locate the first-token boundary at the end of prompt evaluation. Unless an analysis explicitly retains the repeats for paired controls (Appendix D.1), each reported request-level timing and energy value is the mean of three repeated runs under the same configuration. Mean TTFT and complete-request time then average these request-level values over the valid requests for that metric. Remaining time and stage shares use requests with both timings available: remaining time is $T - \mathrm{TTFT}$ per request, and stage shares divide summed stage times by summed complete-request times over this common set. The stage-composition calculation in Appendix B.1 accelerates the measured remainder while holding TTFT fixed.

Request energy is measured with an external USB-connected power measurement device at the phone power input. The instrument records device-level input power independently of the inference runtime, and request energy is obtained by integrating the measured power over the complete inference window. The same external measurement setup, inference-window definition, and three-repeat averaging procedure are used across configurations. Energy-based analyses use requests with complete, valid power measurements; paired comparisons retain only requests with valid measurements for both configurations.

A.3 Data handling and interpretation

Mean J/token is the arithmetic mean of per-request energy/token ratios, rather than the ratio of batch totals. EPC is undefined when $S_c(D) = 0$. All mathematics comparisons use a common scoring protocol. GSM8K uses relaxed answer extraction and numeric equivalence; MATH uses answer extraction and PRM800K/SymPy-based scoring. The MATH-500 subset and HammerBench single-turn source are documented at `https://github.com/openai/prm800k` and `https://github.com/MadeAgents/HammerBench`, respectively.

Resource-setting pairs were checked for agreement in correctness, parsed answers, input/output lengths, and available termination labels. These checks establish agreement in recorded output summaries. For the six reference-to-target comparisons used in Section 3, the saved full generated text was also byte-identical for all 1,319 paired requests in each comparison. This shared-output check supports reusing task outcomes when comparing execution time and energy; calibration reuse is

evaluated separately in Section 6. For QNN, length termination is identified by the runner's recorded stopping indicator, which is more appropriate than a threshold of $N_{ic} \geq b$ when the generated length can fall one below the nominal budget.

The Qwen mechanism analysis covers 12 resource-setting pairs and 11 budget pairs; Appendix E.2 adds two mode pairs and two budget pairs for BitNet.

The three OnePlus/GSM8K reference configurations each contained 1,319 requests with quality and cost records. The selection inventory contained 23 candidates and 23,785 records; 18 candidates had complete costs for every record. Each comparison used its own common complete requests. Joint selection used 1,317 GSM8K and 491 MATH requests, with 15 and eight candidates, respectively.

Budget pairs match question identifiers within a deployment, device, and reference mode, retaining valid scores and generated lengths at both budgets. Added tokens sum the per-request output-length increases over these pairs; none of the tabulated pairs has a negative length change. The "Added tokens on W→W pairs" share divides the added tokens from requests incorrect at both budgets by all added tokens in the comparison. Figure 5 instead reports full-batch accuracy, retaining acquisition failures as unsuccessful deliveries, separately from completed incorrect answers.

A small number of requests lacked usable records after acquisition or synchronization failures. Four of the 500 T-MAC/MATH requests under *No cap* lacked complete generation, timing, scoring, or power-measurement records. The resulting denominators are 496 for valid generation pairs, 500 for full-batch quality, and at most 494 for complete time-and-energy pairs. For QNN/MATH, acquisition failures and one incomplete inference-window measurement left 498 valid generation pairs at 512-to-*No cap*. Table 5 reports the affected counts; cost analyses use complete records and leave missing values unimputed.

Table 5: Analysis denominators for T-MAC/MATH, 512 tokens to *No cap*. Generation pairs require valid scores and lengths; full-batch quality retains acquisition failures as unsuccessful deliveries. Cost pairs additionally require complete timing and energy records. Of the four acquisition failures, one had a correct answer at 512 tokens, so the paired recoveries yield a full-batch net gain of 61 correct answers.

Analysis	Requests	Correct at 512 → *No cap*
Scored generation pairs	496	219 → 281
Full-batch recorded quality	500	220 → 281
Complete time and energy pairs	≤ 494	Compute only on common complete records

A.4 Repeated measurements and comparison scope

The full-batch measurements compare complete end-to-end configurations using the common protocol above: timing and energy are measured three times under the same configuration and reported as request-level averages. This repeated-measurement design reduces run-to-run variation while preserving the behavior of each deployed serving path. We use the full-batch measurements for configuration-level comparisons; mechanism-specific claims are supported separately by the controlled stage experiment in Appendix B.1 and the output-budget controls in Appendix D.1. For the latter, short- and long-budget runs are additionally interleaved so that incremental budget costs are measured under closely matched run order and device conditions, with unchanged-EOS outputs serving as a control.

B Additional stage and failure-cost evidence

B.1 Stage composition and resource-setting details

For six-thread T-MAC on OnePlus, mean input/output lengths were 145.03/161.55 tokens on GSM8K at budget 512 and 499.89/26.47 on HammerBench under *No cap*. Let f be the fraction of total time before the first token, and suppose only the remaining stage is accelerated by a factor α. If TTFT is unchanged, the fractional reduction in total time is $(1 - f)(1 - 1/\alpha)$. Halving the

remainder would reduce average request time by 32.13% on GSM8K but only 3.93% on HammerBench.

The controlled T-MAC stage-thread experiment used 12 GSM8K questions, three repeats per question and configuration, and 36 valid paired records for each of $(6, 6)$, $(8, 6)$, $(6, 8)$, and $(8, 8)$ (prefill, decode threads). Mean $(\mathrm{TTFT}, T - \mathrm{TTFT}, T)$ in seconds were $(5.259, 11.142, 16.401)$, $(4.996, 10.686, 15.682)$, $(5.290, 15.760, 21.050)$, and $(4.775, 15.070, 19.845)$, respectively. With prefill fixed at six threads, increasing decode threads from six to eight slowed decoding in 35 of 36 paired runs.

For llm.npu on OnePlus/GSM8K at budget 512, burst used 6.963 J per request, compared with 4.281 J for balanced. For QNN on the same device and workload at budget 512, low-power increased completion time by 105.93% and mean request energy from 4.544 to 4.960 J relative to balanced.

B.2 Incorrect-request costs

At budget 512 on OnePlus/MATH, incorrect requests accounted for a disproportionate share of complete-request energy (Figure 8). For llm.npu, 70.2% of questions were answered incorrectly and accounted for 78.97% of recorded energy; the corresponding T-MAC shares were 56.0% and 65.26%. The figure also relates accuracy and output length to task difficulty. Difficulty groups use the benchmark's existing annotations for retrospective analysis, rather than as inputs to the selector. The five groups contain 43, 90, 105, 128, and 134 questions; statistics are descriptive and use completed run records.

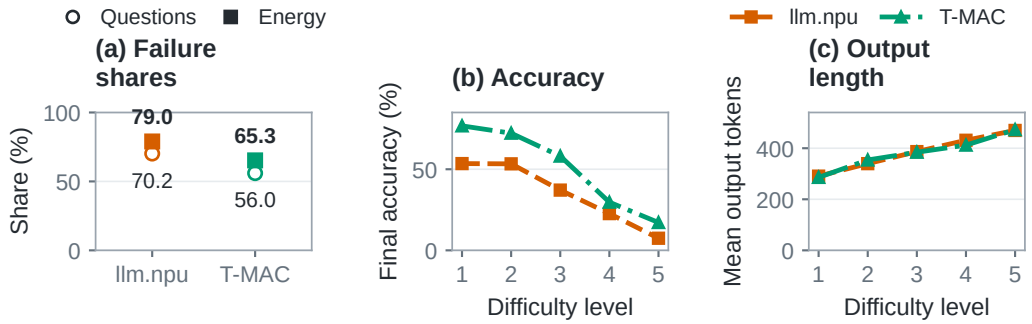

Figure 8: Incorrect-request costs on OnePlus/MATH, budget 512, reference modes, 500 questions per deployment. (a) Shares of incorrect questions and their recorded energy. (b,c) Final accuracy and mean output length by benchmark difficulty group.

B.3 Token costs and EPC comparisons

Generation behavior helps explain the token-efficiency mismatch introduced in Figure 1. QNN and llm.npu generated 171.58 and 159.62 tokens per GSM8K request, with mean completion times of 11.758 and 7.334 seconds. They produced 819 and 915 correct answers overall, of which 412 and 831 arrived within 10 seconds. Mean J/token was 0.02656 for QNN and 0.02682 for llm.npu; at 10 seconds, their EPC values were 14.55 and 6.80 J. Only llm.npu met a 60% timely accuracy target at that deadline.

For positive mean energies and timely accuracies, Equation 2 gives the condition under which a new configuration improves EPC:

$$\frac{S_{\mathrm{new}}(D)}{S_{\mathrm{old}}(D)} > \frac{\bar{E}_{\mathrm{new}}}{\bar{E}_{\mathrm{old}}}. \tag{5}$$

This ratio condition describes EPC changes; Section 2 distinguishes this descriptive cost from the constrained mean-energy objective.

B.4 SUPPLEMENTARY BUDGET RESULTS

Table 6 reports the paired llm.npu/GSM8K results on Redmi. At each budget, its 1,319 requests match OnePlus in correctness, parsed answers, input/output lengths, and termination labels. The device-specific records contain different timing and energy measurements; the matching answer-transition and added-token aggregates are reported for both phones.

Table 6: Paired budget results for llm.npu on Redmi/GSM8K in balanced mode, using the definitions in Table 1.

Deployment	Budget	Pairs	W→C / C→W	Added tokens	Added tokens on W→W pairs (%)
GSM8K / Redmi					
llm.npu	256 → 512	1319	25/0	5,794	81.29%
llm.npu	512 → *No cap*	1319	0/0	434	100.00%

C SELECTION REPLAY PROTOCOL AND FULL RESULTS

Candidate sets and splits. All candidates were measured Qwen2.5-3B deployments on OnePlus with budgets 256, 512, or *No cap* where available. For each fixed candidate set, retain common request identifiers with complete scores, inference time, and energy, then sort identifiers before splitting. NumPy `default_rng(seed)` permutes them independently for seeds 0–19. The first $\lfloor 0.2n \rfloor$ identifiers calibrate selection and all remaining identifiers evaluate the frozen decision. For joint GSM8K selection, $n = 1{,}317$ yields 263 calibration and 1,054 evaluation requests; for MATH, $n = 491$ yields 98 and 393. Comparisons varying one configuration dimension use their own common complete sets. The GSM8K mode comparisons in Table 9 use 1,318 common requests for llm.npu (263 calibration and 1,055 evaluation) and 1,319 for T-MAC and QNN (263 calibration and 1,056 evaluation). The splits reuse a fixed pool of measured requests. We therefore interpret variation across splits as replay robustness to calibration composition, rather than as independent device trials or prospective generalization.

Task grids and baselines. GSM8K uses deadlines $\{10, 20, 30\}$ seconds with targets $\{60\%, 75\%\}$; MATH uses $\{20, 30, 60\}$ seconds with $\{30\%, 40\%\}$. Mode selection fixes deployment and budget 512, allowing mode/thread selection; its baseline minimizes calibration mean complete-request time. Budget selection fixes deployment and reference mode; baselines fix 512 or the largest measured budget. Deployment selection fixes budget 512 and reference modes; proxy baselines maximize decode throughput or minimize mean per-request J/token or mean request energy. Reference modes are balanced on the NPU paths and six threads on T-MAC. Joint selection allows all three dimensions. Its fixed-512 baseline restricts budget and its fixed-reference baseline restricts mode, both applying the same empirical task screen and mean-energy objective. The energy-only baseline minimizes calibration mean request energy without a task constraint. No unmeasured configuration is imputed.

Decision and outcome accounting. Equation 4 screens candidates by empirical timely accuracy and minimizes mean request energy, resolving ties by configuration identifier. It returns no selection when the feasible set is empty. Evaluation outcomes distinguish no selection, selection followed by a missed target, and selection followed by target attainment. All attainment counts in Table 7 use the full 20-split denominator. No-selection cases have no selected configuration and contribute no energy or accuracy observation. Thus 0/20 with 20 no-selections differs from 0/20 with none.

Paired energy comparisons retain only splits where both policies selected a configuration and both met the evaluation target. The reported savings are 38.6% for llm.npu mode selection at GSM8K 20 s / 60%, 2.1% for T-MAC modes at 30 s / 60%, and 3.1% and 41.0% for joint versus fixed-512 selection at the two representative targets. The corresponding paired subsets contain 20, 20, five, and eleven splits, respectively. For paired subset P, the mean for policy p is $|P|^{-1} \sum_{s \in P} \bar{E}^{\mathrm{eval}}_{p,s}$; savings are one minus the ratio of the selected-policy mean to the baseline mean, expressed as a percentage. These percentages concern the matched successful subset, while the attainment table

retains every split. Tables 8 and 9 report the paired energies and subset sizes for joint and mode selection, respectively. Table 2 collects the representative comparisons in the main text. Request energy covers the complete inference window, including incorrect and late requests, and excludes calibration acquisition, model loading, switching, and selection overhead. Calibration acquisition costs are reported separately in Appendix C.1.

Table 7: Exact outcomes for Figure 7. Entries give target attainment out of 20 splits, followed by no-selection counts in parentheses; remaining splits selected but missed the target. Fixed ref. denotes fixed reference mode.

Task	Requirement	Joint	Fixed 512	Fixed ref.	Energy only
GSM8K	10 s / 60%	19/20 (1)	19/20 (1)	17/20 (3)	16/20 (0)
GSM8K	10 s / 75%	0/20 (20)	0/20 (20)	0/20 (20)	0/20 (0)
GSM8K	20 s / 60%	18/20 (0)	20/20 (0)	18/20 (0)	18/20 (0)
GSM8K	20 s / 75%	0/20 (20)	0/20 (20)	0/20 (20)	0/20 (0)
GSM8K	30 s / 60%	19/20 (0)	20/20 (0)	19/20 (0)	19/20 (0)
GSM8K	30 s / 75%	12/20 (5)	5/20 (8)	12/20 (5)	0/20 (0)
MATH	20 s / 30%	0/20 (10)	0/20 (10)	0/20 (10)	0/20 (0)
MATH	20 s / 40%	0/20 (20)	0/20 (20)	0/20 (20)	0/20 (0)
MATH	30 s / 30%	17/20 (1)	15/20 (3)	17/20 (1)	0/20 (0)
MATH	30 s / 40%	0/20 (18)	0/20 (19)	0/20 (18)	0/20 (0)
MATH	60 s / 30%	20/20 (0)	20/20 (0)	20/20 (0)	0/20 (0)
MATH	60 s / 40%	14/20 (0)	11/20 (3)	14/20 (0)	0/20 (0)

Restricted candidates and single-dimension results. Joint and fixed-reference selection make identical decisions in all 20 splits at GSM8K 30 s / 75% and MATH 60 s / 40%, so budget freedom suffices to reproduce those gains. At GSM8K 20 s / 60% and 30 s / 60%, fixed 512 meets the target in every split, while joint selection attains it in 18 and 19 splits, respectively. The complete grid distinguishes these calibration-dependent outcomes from empty feasible sets.

With deployment and reference mode fixed, budget selection on T-MAC/GSM8K at 20 s / 60% attains the target in 17/20 splits, versus 9/20 for fixed 512. On QNN/MATH at 60 s / 40%, the corresponding counts are 14/20 for budget selection, 0/20 for fixed 512, and 20/20 for the largest measured budget. In the 14 jointly successful splits, budget selection and the largest-budget baseline choose the same configuration, with equal mean request energy of 13.131 J.

Table 8: Joint-selection energy on baseline-specific n_{pair} splits where both policies selected and met the evaluation target. Fixed ref. denotes fixed reference mode.

Task	Requirement	Baseline	n_{pair}	Joint (J)	Baseline (J)	Saving
GSM8K	30 s / 75%	Fixed 512	5	11.116	11.472	3.1%
GSM8K	30 s / 75%	Fixed ref.	12	11.087	11.087	0.0%
MATH	60 s / 40%	Fixed 512	11	13.160	22.287	41.0%
MATH	60 s / 40%	Fixed ref.	14	14.084	14.084	0.0%

For the two QNN targets in Table 9, the policies select the same configuration on jointly successful splits and have equal paired energy, with different attainment counts over all splits. The 38.6% llm.npu saving averages evaluation means across splits; the 38.52% mode comparison in Section 3 uses the complete recorded batches.

Table 9: Mode selection on OnePlus/GSM8K at budget 512. Outcomes give target-met / target-missed / no-selection counts over all 20 splits. PACE denotes PACE-SELECT; Fastest minimizes calibration mean complete-request time. Energy uses the n_{pair} splits where both policies selected and met the target.

Deployment	Requirement	Outcomes (met/missed/none)		n_{pair}	Energy (J)	Saving
		PACE	Fastest		PACE / Fastest	
llm.npu	20 s / 60%	20/0/0	20/0/0	20	4.280 / 6.965	38.6%
T-MAC	30 s / 60%	20/0/0	20/0/0	20	11.210 / 11.450	2.1%
QNN	20 s / 60%	6/3/11	17/3/0	6	4.541 / 4.541	0.0%
QNN	30 s / 60%	16/0/4	20/0/0	16	4.539 / 4.539	0.0%

C.1 CALIBRATION REUSE ACROSS EXECUTION MODES

Shared task outcomes and measurement scope. This separate calibration experiment uses nine OnePlus/GSM8K configurations at output budget 512: three modes each for QNN, llm.npu, and T-MAC. Balanced is the reference mode for QNN and llm.npu; six threads is the reference for T-MAC. The six reference-to-target pairs are balanced to burst and low-power for QNN, balanced to burst and low-power-saver for llm.npu, and six to four and eight threads for T-MAC. Each pair agrees on correctness, parsed answers, input/output token counts, termination reason, and saved complete generated text for all 1,319 requests. The text comparison is byte-for-byte; original generated token IDs were not retained.

Calibration allocation and evaluation. The experiment compares $k \in \{16, 32, 64, 128\}$ measured requests per target mode over 20 random splits and six task targets, giving 120 configuration decisions at each k. At each measurement budget, direct per-mode lookup and paired calibration use the same total number of calibration requests (Table 10). Paired calibration concentrates measurements in the reference modes, reuses their correctness observations, and maps their execution costs to the target modes using paired measurements. The shared-correctness alternative reuses the task outcomes without the paired cost mapping. Figure 9 shows the attained decisions and the reported historical inference-window time and recorded energy of the calibration requests. These acquisition costs are separate from the evaluation request energies in Table 2; model loading, switching, and fitting overhead are outside the inference windows.

Table 10: Calibration efficiency on OnePlus/GSM8K at budget 512. Counts report attained decisions out of 120 per k; total requests are the matched calibration request budget. Time (hours) and recorded energy (kJ) concern historical calibration inference windows. D denotes direct lookup and P paired mapping.

k	Total requests	Direct	Shared	Paired	Time: D / P	Energy: D / P
16	864	57/120	63/120	63/120	4.14 / 3.12	7.42 / 6.13
32	960	59/120	63/120	64/120	4.61 / 3.64	8.24 / 7.02
64	1,152	56/120	64/120	63/120	5.53 / 4.69	9.90 / 8.82
128	1,536	63/120	63/120	64/120	7.31 / 6.81	13.07 / 12.47

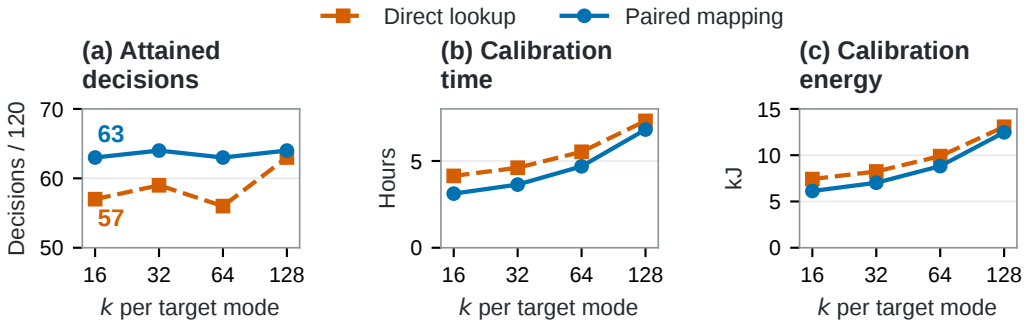

Figure 9: Reusing task outcomes improves calibration efficiency in the nine-configuration replay. (a) Attained decisions out of 120. (b) Historical calibration inference-window time. (c) Recorded calibration energy. Each point uses the matched total request budget in Table 10; k is the measured request count per target mode. Lines connect the four reported measurement budgets.

D Mechanism Controls

D.1 Repeated Output-Budget Controls

Protocol and scope. We evaluate GSM8K and MATH-500 on OnePlus PLQ110, selecting eight questions per task from each of three observed outcome categories: EOS-terminated, truncated-and-recovered, and truncated-and-still-incorrect. Consistent with the three-repeat measurement protocol in Appendix A, each short- and long-budget condition is measured three times; in this control, the two budget conditions are additionally interleaved to reduce run-order effects. The budgets are 256 and 512 on GSM8K, and 512 and *No cap* on MATH-500. All 288 request records specify six prefill threads, six decode threads, and a 1,024-token total context limit. Timing and externally measured USB power cover the same complete inference window after model loading as in Appendix A. The three repeats are retained individually for paired analysis, giving 24 short–long pairs per task and category and 144 pairs overall. The categories are selected using observed outcomes; their correctness rates describe these selected questions, rather than estimating full-batch accuracy or the prevalence of recovery. These controls are used only for within-run short-versus-long budget comparisons; the control export records the execution settings used here but does not retain a deployment identifier, so no cross-deployment conclusion is drawn from this experiment.

Generated text and termination. Table 11 reports all six groups. We pair records by task, observed category, question identifier, and repeat, then compare the stored completion strings without normalization. Across both tasks, the 48 EOS pairs have identical full generated text and no additional output tokens. All 96 truncated pairs have the shorter generated text as an exact text prefix of the longer output. These comparisons directly establish text preservation and continued generation in the measured runs. All recovered runs terminate with EOS at the larger budget. Among the still-incorrect runs, GSM8K has 18 EOS terminations and six length terminations; MATH has 12 of each. Continued generation can therefore leave an answer incorrect either after natural termination or when another length limit is reached.

Complete-request costs. Table 12 gives mean long-minus-short differences over the 24 pairs in each row, retaining the reported precision. GSM8K recovery adds 63.3 tokens, 4.178 seconds, and 2.889 J per request; continued incorrect generation adds 103.1 tokens, 8.381 seconds, and 6.093 J. On MATH, recovery adds 129.4 tokens, 9.393 seconds, and 5.914 J; continued incorrect generation adds 233.4 tokens, 19.327 seconds, and 11.221 J. The EOS controls add no tokens but retain small timing and energy differences across physical runs. The repeated measurements thus connect output extension to complete-request costs while distinguishing the selected recovery and still-incorrect categories. These category means complement the full-batch token accounting in Table 1; they are not weighted to estimate the full batch's incremental time or energy.

Table 11: Repeated budget controls: correctness, generated-text correspondence, and long-budget termination. Each row contains eight questions with three paired repeats (24 pairs). Accuracy is reported for short / long budgets. EOS / length gives long-budget termination counts. "Prefix" denotes the shorter output as an exact text prefix of the longer output.

Task	Selected category	Accuracy (%) short / long	Text relation	EOS / length
GSM8K	EOS-terminated	100 / 100	Identical 24/24	24 / 0
GSM8K	Truncated, recovered	0 / 100	Prefix 24/24	24 / 0
GSM8K	Truncated, still incorrect	0 / 0	Prefix 24/24	18 / 6
MATH	EOS-terminated	87.5 / 87.5	Identical 24/24	24 / 0
MATH	Truncated, recovered	0 / 100	Prefix 24/24	24 / 0
MATH	Truncated, still incorrect	0 / 0	Prefix 24/24	12 / 12

Table 12: Mean cost increments for all six repeated budget-control groups. Differences are long budget minus short budget; percentages divide each mean difference by the short-budget mean. Each row contains 24 pairs from eight questions.

Task	Selected category	Δ tokens	Δ time (s)	Δ energy (J)
GSM8K	EOS-terminated	0	-0.343	+0.117 (1.1%)
GSM8K	Truncated, recovered	+63.3 (24.7%)	+4.178 (17.4%)	+2.889 (17.5%)
GSM8K	Truncated, still incorrect	+103.1 (40.3%)	+8.381 (33.1%)	+6.093 (35.3%)
MATH	EOS-terminated	0	-0.069	+0.056 (0.3%)
MATH	Truncated, recovered	+129.4 (25.3%)	+9.393 (22.5%)	+5.914 (20.4%)
MATH	Truncated, still incorrect	+233.4 (45.6%)	+19.327 (44.7%)	+11.221 (37.2%)

E SUPPLEMENTARY MODEL AND DEPLOYMENT COMPARISONS

Configurations and coverage. We compare Llama 3.1-8B with Qwen2.5-3B on OnePlus using QNN 2.37, balanced mode, and *No cap*, retaining the 1,024-token context limit. Both use the common mathematics scorer in Appendix A.3 and matching benchmark question identifiers. Llama has 1,319 complete GSM8K records and 500 MATH records, including one execution failure. Accuracy uses the full batch, treating the failure as unsuccessful delivery. Request costs use the common complete records: 1,319 GSM8K and 498 MATH pairs, since the two models lack costs for different MATH requests.

Table 13: Model comparison on OnePlus/QNN in balanced mode with *No cap*. Accuracy is the percentage correct, followed by correct / full-batch counts. Decode averages valid per-request rates; time and energy average the common complete cost pairs.

Model	Accuracy (%)	Decode (tok/s)	Time (s)	Energy (J)
GSM8K (1,319 questions; 1,319 cost pairs)				
Qwen2.5-3B	62.09 (819/1,319)	14.92	11.93	4.35
Llama 3.1-8B	72.63 (958/1,319)	7.71	19.71	8.75
MATH (500 questions; 498 cost pairs)				
Qwen2.5-3B	46.60 (233/500)	14.89	35.14	13.14
Llama 3.1-8B	34.80 (174/500)	7.65	60.46	26.54

Model comparison tables. Tables 13 and 14 report the full-batch quality, paired request costs, and selected deadline points underlying Section 5. Llama's final accuracy is 10.54 percentage points higher on GSM8K, whereas Qwen's is 11.80 points higher on MATH.

Table 14: Timely accuracy (%) for the configurations in Table 13, followed by timely correct / full-batch counts. Deadlines classify complete requests retrospectively.

Task	Deadline	Qwen2.5-3B	Llama 3.1-8B
GSM8K	20 s	60.27 (795/1,319)	53.83 (710/1,319)
GSM8K	30 s	61.94 (817/1,319)	70.20 (926/1,319)
MATH	30 s	32.60 (163/500)	17.60 (88/500)
MATH	60 s	46.60 (233/500)	31.00 (155/500)

Task cost and comparison scope. Llama's higher timely accuracy at 30 seconds still comes with higher EPC: 12.47 versus 7.03 J. This comparison evaluates the Qwen2.5-3B and Llama 3.1-8B artifacts together with their recorded run conditions under QNN, balanced mode, and *No cap*. The configuration-selection replay in Section 6 uses the Qwen2.5-3B candidate set.

E.1 DEADLINE CONSTRAINTS ACROSS QWEN DEPLOYMENT PATHS

Grouping outcomes as timely correct, late correct, or incorrect separates timing constraints from final-answer quality across deployment paths. At a 20-second deadline on OnePlus/GSM8K with budget 512 in reference modes, QNN had 795 timely correct, 24 late correct, and 500 incorrect answers. The corresponding counts were 915, zero, and 404 for llm.npu, and 788, 239, and 292 for six-thread T-MAC. T-MAC's 239 late correct answers identify potential gains from acceleration that preserves correctness; llm.npu had already delivered every correct answer by this deadline.

T-MAC's timely correct count reached 941 at 25 seconds, exceeding llm.npu's 915, and 994/1,319 (75.36%) at 30 seconds. QNN and llm.npu had final accuracies of 62.09% and 69.37%, placing a 75% target beyond reach on their recorded outputs. T-MAC's final accuracy of 77.86% made that target attainable with sufficient time. Using Equation 3, the observed 60% boundaries were 18.894, 9.407, and 20.151 seconds for QNN, llm.npu, and T-MAC, respectively. Only T-MAC reached 75%, with a boundary of 29.153 seconds (Figure 10).

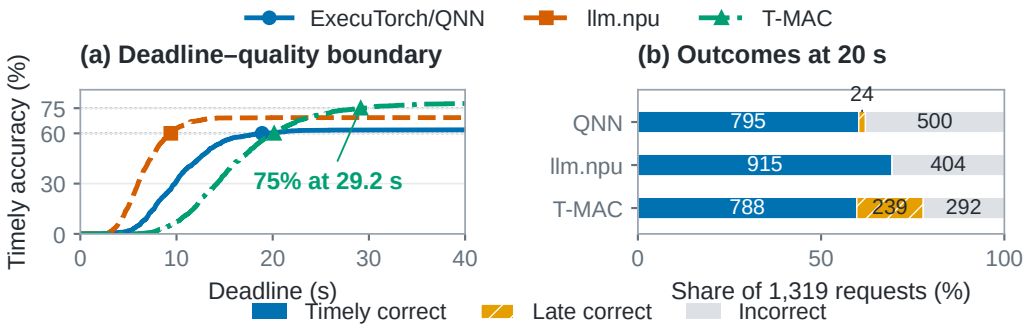

Figure 10: Deadline and accuracy constraints for Qwen2.5-3B deployments on OnePlus/GSM8K, budget 512, reference modes. (a) Timely accuracy; markers locate minimum observed deadlines for 60% and 75% where attained. (b) Outcomes at 20 seconds, with request counts. Each configuration contains 1,319 requests.

At budget 512 in reference modes on OnePlus/MATH, QNN, llm.npu, and T-MAC answered 186, 149, and 220 of 500 questions correctly. The ordering of the two NPU paths reversed relative to GSM8K, reinforcing the need for workload-specific accuracy when screening deployments.

E.2 BitNet-2B mode and budget comparisons

The BitNet-2B experiments use a BitDistiller W4A16 deployment on OnePlus with QNN 2.28, its native chat template, and a 1,024-token total context limit. The five GSM8K configurations in Table 15 each contain 1,319 successful requests scored with the common relaxed extraction and numeric-equivalence protocol. Balanced mode covers nominal output budgets 256, 512, and *No cap*; burst and low-power cover 512. These configurations extend the mode and budget evidence in Sections 3–4. The selection replay in Section 6 continues to use the 23 Qwen configurations.

Table 15: BitNet-2B on OnePlus/GSM8K. Correct counts and timely accuracies use all 1,319 questions; $S(10)$ and $S(30)$ are percentages at 10 and 30 seconds. Time averages all requests, while mean request energy uses the n_E available valid records.

Mode	Budget	Correct	$S(10)$	$S(30)$	Time (s)	Energy (J)	n_E
Balanced	256	915	64.59	69.37	6.758	2.493	1,318
Balanced	512	923	65.05	69.98	6.908	2.629	1,319
Balanced	*No cap*	923	65.20	69.98	6.974	2.729	1,319
Burst	512	923	58.15	69.98	7.970	6.077	1,319
Low-power	512	923	44.66	69.98	10.475	2.373	1,318

At budget 512, the three modes match in per-question correctness, parsed answers, recorded input/output lengths, and prompt template. At 10 seconds, balanced and low-power deliver 858 and 589 correct answers; at 30 seconds both deliver 923. One low-power request lacks energy, leaving 1,318 common cost pairs. On these pairs, mean energies are 2.630 and 2.373 J, respectively, giving the 9.77% reduction reported in Section 3; each request-level energy value follows the three-repeat averaging protocol in Appendix A. Burst and balanced have 1,319 common cost records: burst averages 6.077 J versus balanced's 2.629 J, with mean completion times of 7.970 and 6.908 seconds. The mode labels therefore require evaluation through their measured request outcomes and costs.

All 1,319 questions pair across the balanced-mode budgets with matching recorded input lengths and gold answers. From 256 to 512, nine incorrect answers become correct and one correct answer becomes incorrect; 914 remain correct and 395 remain incorrect. This change adds 6,244 tokens, of which 5,719 (91.59%) come from questions incorrect at both budgets. From 512 to *No cap*, all correctness labels remain unchanged, and all 4,833 added tokens come from the 396 questions incorrect at both budgets. The 256-token batch has one missing energy record; this does not affect its full-batch quality or token accounting.