UNDERSTANDING MOE INFERENCE ENERGY CONSUMPTION VIA EXPERT ACTIVATION TRAJECTORY ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model inference incurs substantial operational cost because deployed models must serve requests continuously over long periods. Existing work often analyzes request-level inference energy as an aggregate GPU-level measurement. This black-box view is limited for Mixture-of-Experts (MoE) models, where different inputs dynamically activate different experts and form different internal computation paths. This paper studies the inference energy of a Mixtral-style MoE model through a mechanism-oriented lens. We decompose MoE inference into an input-routing-output-energy chain: input prompts induce prefill expert activation trajectories, these trajectories predict later output behavior, output behavior directly drives decode energy, deployment configuration changes the hardware energy coefficient, and correctness determines whether the consumed energy is useful.

Across request-level energy traces, reconstructed expert routing features, deployment comparisons, full benchmark correctness results, and expert intervention experiments, we find that output behavior is the strongest direct driver of decode energy; prefill expert activation substantially improves output prediction; expert importance cannot be captured by activation frequency alone; deployment configuration changes energy consumption even when output behavior and correctness are nearly unchanged; and a small number of experts have disproportionate effects on output quality under intervention. These results suggest that MoE inference energy is not merely a single aggregate energy number, but an interpretable and potentially optimizable process shaped by routing, output behavior, deployment, and correctness.

1 INTRODUCTION

How should large language models be evaluated? Currently, an increasing number of studies evaluate them based on their performance on benchmark tests. Standard benchmarks typically measure whether a model can correctly answer questions, follow instructions, solve reasoning tasks, or outperform existing models on task-level metrics (Liang et al., 2023; Biderman et al., 2024). This benchmark-centric evaluation approach is crucial for comparing model capabilities. However, current research has not given sufficient attention to a model's energy consumption and its specific causes. As large models transition from offline evaluation to continuous online service, the energy consumption associated with inference has become a recurring operational cost (Luccioni et al., 2023; Faiz et al., 2024; Chung et al., 2025). A model is not evaluated just once in a benchmark; it may need to process millions of requests under different task workloads, hardware environments, and deployment configurations. Therefore, benchmarks that report only accuracy rates or task scores do not fully reflect a model's practicality in real-world deployments (Reddi et al., 2020; Chung et al., 2025).

This issue is particularly important for Mixture-of-Experts (MoE) language models. MoE models enhance model capacity by activating only a subset of experts for each token (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Jiang et al., 2024). This sparse activation mechanism creates a fundamental difference between MoE inference and dense model inference: two requests with similar input lengths—or even similar output lengths—may activate different experts, resulting in different computational paths and, consequently, varying energy consumption costs across multi-GPU

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.

hardware. Existing research on inference energy consumption and system benchmarks often reports metrics such as aggregate energy, average power, latency, throughput, or joules per generated token; consequently, inference energy consumption is typically treated as a black-box hardware figure (Reddi et al., 2020; Chung et al., 2025; Faiz et al., 2024). This research approach overlooks an important feature of the MoE model—the "expert trajectory"—namely, how the expert trajectory fully reflects the energy-consuming generation process of a request.

This paper studies the inference energy of a Mixtral-style MoE model through this benchmark-and-energy perspective. We use `mistralai/Mixtral-8x7B-Instruct-v0.1` (Jiang et al., 2024) and collect request-level inference records on an 8-GPU server under multiple HuggingFace deployment mappings (Wolf et al., 2020). Our data include GPU power traces, integrated request-level decode energy, output length, decode time, hit-max behavior, correctness labels, deployment identifiers, per-GPU energy decomposition, reconstructed expert-routing features, and expert-intervention results. The full benchmark experiments cover `gsm8k`, `mmlu`, `arc_challenge`, and `hellaswag` (Cobbe et al., 2021; Hendrycks et al., 2021; Clark et al., 2018; Zellers et al., 2019), while the expert-intervention analysis further includes a broader benchmark mixture. This produces a multi-dimensional empirical basis for analyzing MoE inference energy rather than only reporting a single aggregate energy value.

Our current research indicates that MoE inference energy should be understood as a structured formation process. The input prompt first triggers prefill routing and expert activation. These early routing signals influence output behavior, including output length, hit-max behavior, and decode time. Subsequently, output behavior directly drives the majority of decode-time energy, while the deployment configuration determines the execution cost of similar generation behavior on the physical GPU. Finally, we conducted benchmarking of model output correctness to verify whether the energy consumed resulted in useful answers. From this perspective, energy is not merely a hardware-side metric; it is the result of the input-routing-output energy chain.

Based on this formulation, we have raised several more specific questions. First, does output behavior explain request-level decode energy? Second, can prefill expert activation predict future output behavior before decoding is complete? Third, how should we understand "expert importance" and attempt to quantify it? Can expert importance be understood beyond simple activation frequency? Fourth, when output behavior is controlled, how does the deployment configuration affect the energy consumption parameter? Fifth, once we take into account the correctness of the output behavior, what does this mean for inference energy? These questions connect benchmark evaluation, MoE routing analysis, multi-GPU energy measurement, and correctness-aware efficiency into a single empirical framework.

After analyzing the experimental data, we made the following findings: 1. Output behavior is the strongest directly observable driver of decode energy and accounts for most of the variation in energy. We found that using prefill expert activations significantly improves the prediction of future output behavior, indicating that early routing trajectories already contain energy-relevant information before generation is complete. When recording data related to expert activation frequency, we found that some frequently activated experts had a relatively weak impact on output and energy consumption, while others with lower activation frequencies exhibited greater sensitivity to output, energy, or correctness. Therefore, activation frequency alone is far from sufficient to reflect the "value" of an expert. Deployment configuration can alter the energy coefficient of inference even when output behavior is identical. Finally, our research on correctness-constrained evaluation indicates that raw energy and joules per token are insufficient metrics. If energy is consumed on incorrect answers, this energy consumption is considered a waste, and we do not regard it as "effective energy consumption."

Main contributions. Our contributions are summarized as follows:

- We collect and analyze request-level MoE inference energy data under multi-GPU deployment, including power traces, decode energy, output behavior, correctness, deployment identifiers, and expert-related information.
- We formulate MoE inference energy as an input-routing-output-energy formation process rather than a single aggregate GPU measurement.
- We show that output behavior directly explains decode energy, while prefill expert activation provides an upstream signal for predicting future output behavior.
- We provide a multi-dimensional analysis of expert importance, showing that activation frequency alone is insufficient for identifying experts that affect output behavior, energy, or correctness.

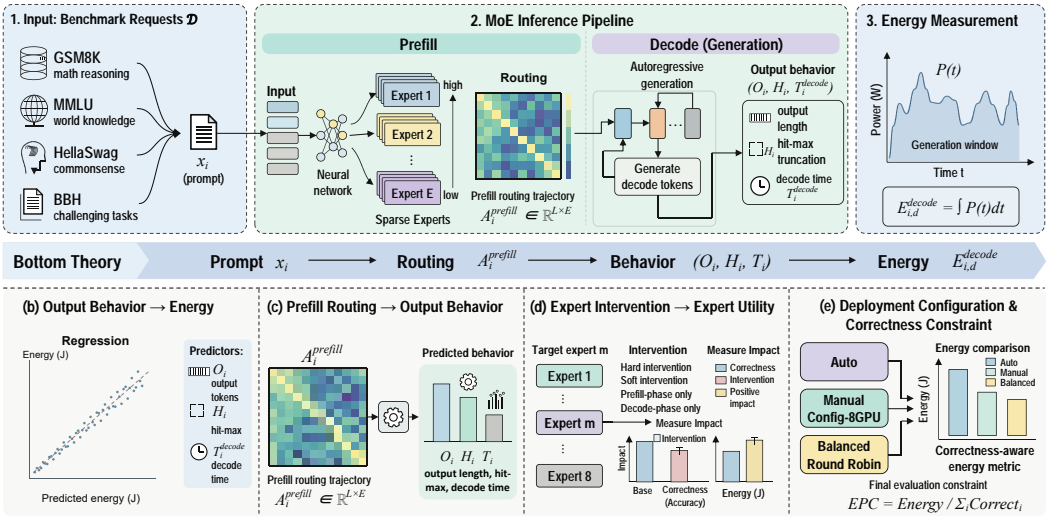

Figure 1: Overview of the experimental pipeline and analysis framework for MoE inference energy. The reported results are obtained from complementary experimental runs. Full-benchmark deployment experiments provide request-level output behavior, correctness, and GPU energy measurements, while separate controlled modeling and expert-intervention runs provide routing and expert-level information. Together, these records support four complementary analyses: decode-energy modeling from output behavior, output-behavior prediction from prefill routing, expert-utility analysis through intervention, and deployment comparison under correctness constraints.

- We demonstrate that deployment configuration changes the hardware energy coefficient under similar inference behavior, and we evaluate energy under correctness constraints using energy-per-correct, paired quality-energy deltas, and correctness-critical expert analysis.

2 Problem Formulation and Overview

2.1 From Aggregate Energy to Routed Energy Formation

A conventional request-level energy analysis starts from the hardware side. It measures power over time, integrates the power trace, and relates the resulting energy to input length, output length, latency, or average power (Reddi et al., 2020; Chung et al., 2025; Luccioni et al., 2023). This approach is appropriate for a first-order description of inference cost. However, it compresses the internal computation path into a small number of external variables. For MoE inference, this compression hides a key mechanism: different requests can take different routed computation paths (Shazeer et al., 2017; Lepikhin et al., 2021; Jiang et al., 2024). The so-called routing computation path refers to the process from the moment a model receives an input through a series of circuit behaviors until an output is generated. From the broader perspective of the MoE model, this computational process is carried out through a relay of parallel computations performed by several experts across multiple layers. In this paper, we view MoE inference energy as a formation process rather than a single number. The term *formation process* means that request-level energy is not produced by one factor alone.

2.2 Input-Routing-Output-Energy Chain

For request i, let x_i denote the input prompt, A_i^{prefill} denote the prefill expert activation trajectory, O_i denote output length, H_i denote whether generation reaches the maximum token budget, T_i^{decode} denote decode time, $E_{i,d}^{\text{decode}}$ denote decode energy under deployment d. The central chain is:

$$x_i \longrightarrow A_i^{\text{prefill}} \longrightarrow (O_i, H_i, T_i^{\text{decode}}) \longrightarrow E_{i,d}^{\text{decode}}. \tag{1}$$

This chain defines the empirical questions in this paper. The input behavior triggers the corresponding expert activation trajectory, thereby generating the corresponding output behavior; a mapping

relationship is then established between the output behavior and energy consumption. It is worth noting that we found that both the model's input and output behaviors can be modeled using just a few feature parameters. A small number of feature parameters can account for the majority of the variation.

3 Experimental Setup and Data Collection

We organize the experiments into three complementary suites. A controlled modeling suite provides request-level routing, output-behavior, and energy records for routing-to-output and output-to-energy modeling; the full benchmark suite provides large-scale deployment, correctness, and energy comparisons; and a separate expert-intervention suite provides screening, validation, and mechanism-level evidence about expert utility. The suites share the same model and HuggingFace execution path, but not every variable is jointly recorded for every request. Full configuration and data-alignment details are provided in Appendix C.

3.1 Model, Backend, and Hardware

All experiments use `mistralai/Mixtral-8x7B-Instruct-v0.1`, a decoder-only MoE model with eight experts per MoE layer (Jiang et al., 2024). The model is loaded with the HuggingFace `transformers` backend using FP16 inference (Wolf et al., 2020). Quantization and speculative decoding are not used. Generation is deterministic: sampling is disabled, temperature is set to 0.0, and top-p is set to 1.0. Deterministic generation is important because paired deployment comparison and expert intervention require the same prompt to produce comparable outputs under controlled conditions.

The full benchmark experiments are conducted on an 8-GPU server equipped with NVIDIA RTX 4090 GPUs. We compare three HuggingFace device-mapping configurations: automatic mapping, manual contiguous layer placement, and balanced round-robin layer placement. These configurations provide controlled deployment variants for testing how similar generation behaviors incur different energy costs (Rajbhandari et al., 2022; Hwang et al., 2023; He et al., 2022).

3.2 Benchmarks, Energy Measurement, and Interventions

The full benchmark uses `gsm8k`, `mmlu`, `arc_challenge`, and `hellaswag` (Cobbe et al., 2021; Hendrycks et al., 2021; Clark et al., 2018; Zellers et al., 2019). These benchmarks cover mathematical reasoning, broad knowledge assessment via multiple-choice questions, scientific reasoning, and common-sense sentence completion. For GSM8K, correctness depends on matching the extracted numerical answers. For multiple-choice benchmarks, correctness is based on the extraction of answer options. Different task types use different generation limits. GSM8K allows for longer reasoning-style outputs, while multiple-choice benchmarks use shorter answer-letter generations. This makes the benchmark mixture more valuable for energy analysis: it includes both short-output classification-style requests and longer-output reasoning requests. After filtering out incomplete or invalid records, the full benchmark generates 26,574–26,575 valid evaluated requests per deployment.

3.3 Request-Level Energy Measurement

For each benchmark request, we measure GPU power within the generation window. GPU power is sampled from all visible GPUs via `nvidia-smi` at 0.10-second intervals. Each power sample records the GPU index, GPU name, instantaneous power draw, utilization, memory usage, and timestamp. For each request, the benchmark runner records the monotonic start and end times of the generation and calculates the request-level decode energy by integrating the sampled GPU power within the measured window (Chung et al., 2025). In addition to the total decode energy, the runner also stores the output length, decode time, hit-max behavior, correctness label, deployment identifier, and per-GPU energy decomposition. Per-GPU decomposition is important because MoE inference on multiple GPUs is not merely a matter of total energy. Different deployment mappings may distribute layers, experts, and computations across devices in different ways. Therefore, request-level and per-GPU records enable us to further analyze whether energy differences stem from output behavior, deployment mapping, or hardware-side imbalances.

Table 1: Compact experimental setup. Full configuration details are provided in Appendix C.

Item	Setting
Model	Mixtral-8x7B-Instruct-v0.1
Backend	HuggingFace `transformers`, FP16, deterministic decoding
Hardware	8 NVIDIA RTX 4090 GPUs
Full benchmark	`gsm8k`, `mmlu`, `arc_challenge`, `hellaswag`
Deployments	Auto-8GPU, ManualContig-8GPU, Balanced-8GPU
Energy measurement	`nvidia-smi` power traces, 0.10s sampling interval
Expert intervention	hard ban, soft downweighting, prefill-only, decode-only, all-phase

The data collected in this section constitute the core output of this study; they support matched deployment comparisons, correctness-constrained energy evaluation, residual analysis, and subsequent diagnosis of anomalous energy cases. We can try to answer some more complex and specific questions: To what extent do different deployments affect the energy consumption parameters of matched prompts? Do high-energy requests also imply correct answers? Do certain tasks systematically generate higher energy per correct answer?

3.4 Routing Feature Construction and Expert Intervention

For the full benchmark deployment comparison, we disabled routing capture to reduce runtime and storage overhead. Expert-routing analysis was conducted in separate modeling and intervention runs. In these runs, we reconstructed or captured router logits and converted them into expert assignment features (Lepikhin et al., 2021; Jiang et al., 2024). We used input length to distinguish between the prefill and decode phases (Agrawal et al., 2024), and constructed phase-specific expert counts, expert shares, routing entropy, concentration features, and routing-confidence summaries. These routing-derived records enable us to test whether prefill expert activations can predict future output behavior before decoding is complete. This is important because output length and decode time can only be fully observed after energy has been consumed. Second, routing features provide a foundation for studying the expert-level structure of MoE inference, including whether activation frequency is sufficient to identify important experts.

Expert intervention experiments are conducted separately from the full benchmark deployment runs. The validation stage evaluates 32 selected experts under hard all-phase intervention. The mechanism stage evaluates 16 selected experts under hard intervention, soft downweighting, prefill-only intervention, decode-only intervention, and all-phase intervention. The expert-intervention validation further includes a broader benchmark mixture, including `gsm8k`, `arc_challenge`, `hellaswag`, `mmlu`, `mmlu_pro`, `bbh`, and `gpqa` (Wang et al., 2024; Suzgun et al., 2023; Rein et al., 2024), in order to evaluate whether expert correctness effects are benchmark-dependent.

4 Empirical Results

4.1 Output Behavior Directly Predicts Decode Energy

Autoregressive decoding generates output tokens sequentially (Yu et al., 2022; Agrawal et al., 2024). We therefore begin with an interpretable post-hoc output-behavior model that relates request-level decode energy to three directly observed generation variables: output length, whether generation reaches the maximum token budget, and decode time. Here, ϵ_i denotes the residual decode-energy variation not explained by these observed output variables:

$$E_i^{\text{decode}} = \alpha_0 + \alpha_1 O_i + \alpha_2 H_i + \alpha_3 T_i^{\text{decode}} + \epsilon_i. \tag{2}$$

We refit this model using a prompt-grouped train/test split and 23,992 valid requests. The three-variable output-behavior model reaches $R^2 = 0.890$ on the held-out set, while five-fold grouped cross-validation gives $R^2 = 0.893 \pm 0.004$. Thus, observed output behavior explains approximately nine-tenths of the predictable request-level decode-energy variation in this controlled modeling suite. Full error metrics and ablations are reported in Appendix D.1.

Output behavior, however, is not the complete explanation. To quantify the remaining hardware-side structure, we further fit a deployment-aware linear model that adds input length, task fixed effects, deployment fixed effects, and a deployment-specific decode-time slope:

$$E_{i,d}^{\text{decode}} = \gamma_0 + \gamma_1 O_i + \gamma_2 H_i + \gamma_3 I_i + \lambda_{\text{task}(i)} + \delta_d + \kappa_d T_i^{\text{decode}} + \epsilon_{i,d}, \tag{3}$$

where I_i is input length, $\lambda_{\text{task}(i)}$ is a task fixed effect, δ_d is a deployment fixed effect, and κ_d is the deployment-specific coefficient on decode time. The interpretable deployment-aware model reaches held-out $R^2 = 0.982$; a nonlinear gradient-boosted tree using the same observable information reaches $R^2 = 0.989$. The small gap indicates that output behavior and deployment already capture most of the predictable decode-energy structure. These models are intentionally post-hoc, whereas prefill routing in Section 4.2 provides an earlier signal. Full metrics and cross-validation results are in Appendix D.1.

These results support a layered interpretation of energy formation. Prefill routing is an upstream internal representation associated with later output behavior, while output behavior is the proximal observable layer most closely associated with decode energy. We summarize this relation as

$$x_i \rightarrow A_i^{\text{prefill}} \rightarrow Y_i \rightarrow E_{i,d}^{\text{decode}}, \tag{4}$$

where $Y_i = (O_i, H_i, T_i^{\text{decode}})$. A nested-model decomposition can further test whether routing contributes mainly through output behavior or retains residual predictive value after output behavior is controlled; the full formulation is moved to Appendix D.3.

4.2 Prefill Expert Activation Predicts Output Behavior

Output behavior is the direct driver of decode energy, but output behavior itself must be explained. We test whether prefill expert activation predicts future output length. Let I_i denote input length, $Task_i$ denote task information, R_i^{prefill} denote prefill routing summaries, and C_i^{prefill} denote prefill expert count features. The output prediction model is:

$$\log(O_i + 1) = \beta_0 + \beta_1 \log(I_i + 1) + \beta_2^\top Task_i + \beta_3^\top R_i^{\text{prefill}} + \beta_4^\top C_i^{\text{prefill}} + \epsilon_i. \tag{5}$$

Using only input length and task type, the baseline ridge model achieves $R^2 = 0.351$. Adding prefill expert and routing features improves the ridge model to $R^2 = 0.714$. Tree-based models further strengthen the result: expert-prefill tree features reach $R^2 = 0.803$, and routing-confidence features reach $R^2 = 0.830$.

These results indicate that prefill expert activation is a measurable intermediate representation that contains information about later generation behavior before decoding starts. We interpret prefill activation as an internal representation induced by the input and task demand. Since output behavior directly drives energy, this representation becomes an upstream signal for energy formation.

4.3 Expert Utility is Conditional and Multi-dimensional

Activation frequency describes how often an expert is used, but it does not directly measure how much the model depends on that expert for useful outputs. Likewise, a large behavioral change after intervention is not automatically a positive utility signal, because an intervention can either damage or improve correctness. We therefore use correctness retention as the main externally observable utility target, while activation frequency, routing confidence, output/energy sensitivity, task dependence, and intervention phase are kept as complementary expert profiles rather than being added together as interchangeable dimensions. Detailed estimation and validation procedures are provided in Appendix E.

For request i, let $q_i^{(0)} \in \{0, 1\}$ denote the correctness of the original model, and let $q_i^{(e,a)}$ denote correctness after expert e is intervened on under protocol a. We define the paired correctness loss as

$$D_{i,e,a}^{Q} = q_i^{(0)} - q_i^{(e,a)}. \tag{6}$$

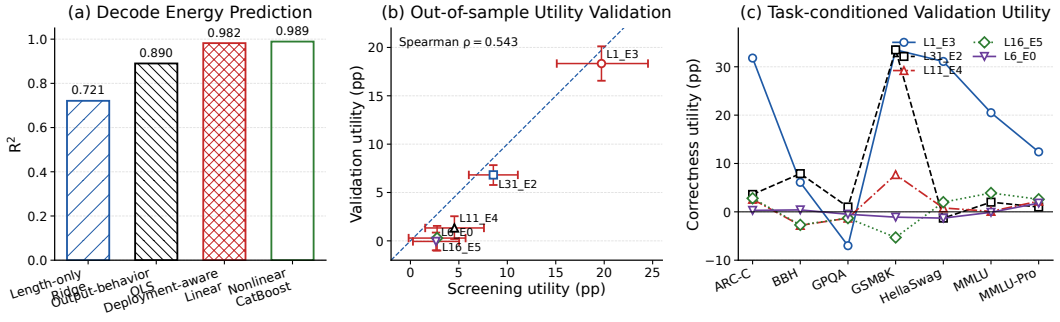

Figure 2: Decode-energy prediction and expert-utility results. Left: held-out decode-energy prediction performance, showing the progression from a length-only baseline to output-behavior, deployment-aware, and nonlinear models. Right: expert-utility analysis. The screening utility is compared with utility measured on disjoint validation requests, and task-conditioned results show that expert utility can vary substantially across benchmarks. The results indicate that a small number of experts have stable correctness effects, while fine-grained rankings for many other experts remain uncertain.

A positive value means that preserving the expert helps retain a correct answer under the tested protocol, while a negative value means that the intervention improves correctness for that request. For benchmark b, the task-conditional retention utility and the workload-level utility score are

$$\widehat{\theta}^{Q}_{e,b,a} = \frac{1}{n_b} \sum_{i \in \mathcal{D}_b} D^{Q}_{i,e,a}, \qquad \widehat{\mathrm{Score}}^{Q}_{e}(\pi, a) = 100 \sum_{b} \pi_b \widehat{\theta}^{Q}_{e,b,a}, \tag{7}$$

where π_b specifies the target benchmark mixture. The score is measured in accuracy percentage points and therefore has a direct interpretation: it estimates how much correctness is lost, on average, when the expert is intervened on under the specified workload and protocol. In the main analysis we use equal benchmark weights. Correct-to-wrong and wrong-to-correct rates are treated as components of the same paired effect rather than independent dimensions, because their difference is exactly the accuracy change. Estimation stability is reported through confidence intervals and rank uncertainty instead of being added as an extra score.

We score all 256 experts in the screening stage and evaluate the frozen ranking on disjoint validation requests. The task-balanced utility score has a Spearman correlation of 0.543 with validation utility, compared with 0.202 for the previous Stage2 composite proxy and -0.037 for activation frequency. On the disjoint validation set, `L1_E3` and `L31_E2` show the strongest supported retention utilities, 18.335 and 6.820 percentage points, respectively. At the same time, the full 256-expert ordering is not stable enough to justify an exact universal ranking. Overlap removal, confidence intervals, rank uncertainty, and comparator details are reported in Appendix E.

Utility is also task- and phase-dependent. For `L1_E3`, hard prefill intervention produces a 22.898-point task-balanced correctness loss, whereas hard decode intervention gives -1.267 points; `L31_E2` shows positive effects in both phases. We therefore treat expert utility as a conditional profile rather than a single unconditional rank. Correctness utility and Stage2 energy sensitivity are reported separately because they were not jointly measured under the same intervention protocol.

4.4 Deployment Changes the Energy Coefficient

Deployment configuration determines how MoE computation is mapped onto physical GPUs (Rajbhandari et al., 2022; Hwang et al., 2023; He et al., 2022). Equation 3 makes this hardware effect explicit through δ_d and the deployment-specific decode-time coefficient κ_d. Because energy divided by time has units of power, κ_d can be interpreted as a regression-derived effective aggregate power slope after controlling for output length, hit-max behavior, input length, and task:

$$\kappa_d = \frac{\partial E^{\mathrm{decode}}_{i,d}}{\partial T^{\mathrm{decode}}_{i}}, \qquad [\kappa_d] = \mathrm{J/s} = \mathrm{W}. \tag{8}$$

Across the controlled 6- and 8-GPU mappings, the estimated effective power slope ranges from 562.6 W to 700.5 W. For the three controlled 8-GPU mappings, it is 672.6 W for Auto-8GPU, 688.9 W for Balanced-8GPU, and 700.5 W for ManualContig-8GPU. Thus, similar decode duration is associated with different aggregate energy increments under different model-to-GPU mappings after observable request behavior is controlled; confidence intervals and full deployment results are reported in Appendix F.

In the full benchmark, the three 8-GPU deployments have nearly identical accuracy, around 0.672, and nearly identical output lengths. However, their energy-per-correct differs: Auto-8GPU reaches 1461.07 J per correct answer, ManualContig-8GPU reaches 1483.01 J, and Balanced-8GPU reaches 1499.20 J. Paired deployment analysis further shows nonzero energy differences under matched requests with unchanged output-token counts and correctness, reinforcing the conclusion that deployment changes the hardware cost of otherwise similar inference behavior.

In addition to the three full-benchmark 8-GPU deployments, we also analyze additional deployment variants from earlier controlled deployment experiments, including alternative 8-GPU and 6-GPU mappings. These additional results support the same conclusion. Full deployment tables are provided in Appendix F.

4.5 Correctness-Constrained Efficiency

Unlike previous research, which focused on "raw energy consumed per token," we link energy consumption to the accuracy of the model's output. A system may appear efficient by generating short but incorrect outputs, or it may consume a significant amount of energy by generating long outputs—yet the high energy consumption is justified because the results are correct. We believe it is necessary to use accuracy to constrain energy consumption metrics, specifically "energy consumed per correct answer":

$$EPC_{d,b} = \frac{\sum_{i \in \mathcal{D}_{d,b}} E_i^{\text{decode}}}{\sum_{i \in \mathcal{D}_{d,b}} q_i + \delta}. \tag{9}$$

Task-level EPC varies strongly across benchmarks: short multiple-choice outputs have relatively low EPC, while `gsm8k` combines longer outputs and lower accuracy with much higher EPC. Useful efficiency therefore depends jointly on generation behavior and correctness. Where δ is a small numerical stabilizer.

Expert intervention further shows that correctness is concentrated in a small number of experts. We summarize the two strongest validation effects below; the full flip definitions and validation table are provided in Appendix G.2.

Table 2: Key correctness-critical experts under validation hard-ban intervention. Full validation results are provided in Appendix G.

Expert	Base Acc.	Int. Acc.	ΔAcc.	C2W	NFI
L1_E3	0.586	0.389	-0.197	0.283	0.197
L31_E2	0.586	0.539	-0.046	0.059	0.046

Hard-banning `L1_E3` reduces validation accuracy by 19.7 percentage points, while `L31_E2` produces a smaller 4.6-point drop. Their effects are task- and phase-dependent: `L1_E3` is mainly prefill-sensitive, whereas `L31_E2` shows both prefill and decode sensitivity. Full benchmark- and mechanism-level results are in Appendix G.

5 Discussion and Limitations

The results support a structured view of MoE inference energy: output behavior explains about 89% of held-out decode-energy variance, deployment-aware structure raises the interpretable fit to about 98%, and prefill routing provides an earlier signal about later generation behavior. Expert

utility is task- and phase-dependent rather than reducible to activation frequency. Energy-aware MoE evaluation should therefore consider routing, output behavior, deployment, and correctness jointly.

The study is limited to one Mixtral-style model, the HuggingFace backend, and a specific multi-GPU system, so exact coefficients and expert effects should not be treated as universal. The deployment analysis is not a complete expert-placement or scheduling optimization study, correctness-aware metrics remain benchmark-dependent, and routing reconstruction and power measurement introduce system noise. In addition, expert-utility validation is strongest for a small set of candidates rather than a stable total ordering of all 256 experts.

6 Conclusion

This paper studies MoE inference energy as an input–routing–output–energy process. Prefill expert activations provide an upstream representation of later generation behavior; observable output behavior explains about 89% of held-out decode-energy variance; and deployment-aware structure raises the interpretable fit to about 98%. Expert-intervention results further show that correctness utility is concentrated in a small number of experts and depends on task and phase. Together, these results turn request-level MoE energy from a black-box measurement into a decomposable process and motivate future routing-aware and deployment-aware optimization that reduces avoidable energy cost while protecting correctness-critical experts.

7 AI Use Statement

Generative AI tools were used in the process of writing this paper, primarily to assist with the following two types of tasks. First, to assist with the paper's language expression and stylistic refinement, including checking spelling and grammar, improving sentence clarity and readability, and optimizing academic expression without altering the original technical meaning, experimental results, or research conclusions. Second, to assist with the retrieval and organization of relevant literature, including identifying and summarizing related research. The author assumes full responsibility for the final content of this paper.

8 Reproducibility Statement

The experiments use a fixed model, deterministic decoding configuration, documented deployment settings, saved power traces, and saved request-level correctness and energy records. The analysis package includes benchmark summaries, expert-intervention summaries, paired deployment comparisons, and scripts used for result aggregation. The appendix reports the full configuration, key formulas, and complete result tables used to support the main claims. These files are sufficient to reproduce the main numerical results and empirical tables reported in the paper.

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. In *18th USENIX Symposium on Operating Systems Design and Implementation*, 2024. URL https://www.usenix.org/conference/osdi24/presentation/agrawal.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, et al. Lessons from the trenches on reproducible evaluation of language models, 2024. URL https://arxiv.org/abs/2405.14782.

Jae-Won Chung, Jiachen Liu, Jeff J. Ma, Ruofan Wu, 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.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

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.

Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Prateek Sharma, Fan Chen, and Lei Jiang. LLMCarbon: Modeling the end-to-end carbon footprint of large language models, 2024. URL https://arxiv.org/abs/2309.14393.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems*, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/5a54f79333768effe7e8927bcccffe40-Abstract-mlsys2023.html.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. FastMoE: A fast mixture-of-expert training system, 2021. URL https://arxiv.org/abs/2103.13262.

Jiaao He et al. FasterMoE: Modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2022. URL https://doi.org/10.1145/3503221.3508418.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proc. of ICLR*, 2021.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. In *Proceedings of Machine Learning and Systems*, 2023. URL https://arxiv.org/abs/2206.03382.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

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*, pp. 611–626, 2023.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.16668.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.09110.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of BLOOM, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253): 1–15, 2023. URL http://jmlr.org/papers/v24/23-0069.html.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, et al. MLPerf inference benchmark. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, pp. 446–459, 2020.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31094–31116. PMLR, 2023. URL https://proceedings.mlr.press/v202/sheng23a.html.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation*, pp. 521–538, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Appendix

List of Contents

A: Related Work
B: Detailed Definitions and Measurement Foundations
B.1 Request-Level Energy
B.2 Output Behavior Variables
B.3 MoE Routing and Expert Activation
B.4 Routing Summary Features
B.5 Prediction Metrics
C: Full Experimental Setup
C.1 Model and Decoding Configuration
C.2 Deployment Configurations
C.3 Benchmarks and Generation Settings
C.4 Energy Monitoring Protocol
C.5 Routing Reconstruction and Expert Intervention
D: Full Prediction Results
D.1 Decode Energy Prediction
D.2 Output Length Prediction
D.3 Feature-Group Interpretation
E: Expert Utility and Importance Details
E.1 Utility Definition and Paired Response Model
E.2 Data Separation and Out-of-Sample Validation
E.3 Frequency versus Functional Utility
E.4 Conditional Utility and Complementary Intervention Evidence
F: Deployment Analysis Details
F.1 Controlled Deployment Energy Summary
F.2 Additional Paired Deployment Deltas
F.3 Full Benchmark Deployment-Level Correctness-Energy Summary
F.4 Paired Quality-Energy Deltas
G: Correctness-Constrained Results
G.1 Energy per Correct Answer
G.2 Correctness-Critical Experts
G.3 Benchmark-Level Expert Effects
G.4 Mode-Phase Intervention Results
H: Additional Reproducibility and Data-Joining Details
H.1 Request Matching for Deployment Analysis
H.2 Request Matching for Expert Intervention Analysis
H.3 How the Appendix Supports the Main Claims

A Related Work

Inference energy and carbon measurement. A growing body of work studies the energy cost and environmental impact of large language models. Prior studies have measured or estimated the carbon footprint of large-scale language models across training, deployment, and model life cycles (Luccioni et al., 2023; Faiz et al., 2024). System-level benchmark efforts such as MLPerf Inference and ML.ENERGY further emphasize the importance of measuring inference performance, power, and energy under reproducible settings (Reddi et al., 2020; Chung et al., 2025). These works establish that inference energy is an important deployment metric. However, they usually treat request-level energy as a system-level or aggregate measurement. In contrast, our work asks how energy is formed

inside a sparse MoE inference process. We decompose request-level energy through input prompts, prefill expert activation trajectories, output behavior, deployment configuration, and correctness.

LLM serving and inference systems. Many recent systems improve the efficiency of large language model inference by optimizing batching, scheduling, memory management, offloading, or prefill-decode execution. Orca introduces iteration-level scheduling for transformer generation (Yu et al., 2022). vLLM uses PagedAttention to reduce KV-cache memory waste and improve serving throughput (Kwon et al., 2023). FlexGen studies high-throughput generation under limited GPU memory through offloading and compression (Sheng et al., 2023). Sarathi-style scheduling further emphasizes the different roles of prefill and decode in LLM inference (Agrawal et al., 2024). These systems mainly optimize throughput, latency, or memory efficiency. Our study is complementary: rather than proposing a new serving engine, we analyze request-level energy formation and show that output behavior and deployment configuration jointly shape decode energy.

Mixture-of-Experts models and routing systems. Mixture-of-Experts models increase model capacity through sparse conditional computation (Shazeer et al., 2017). Early and large-scale MoE systems such as GShard and Switch Transformers demonstrate that routing tokens to a subset of experts can scale model capacity while controlling active computation (Lepikhin et al., 2021; Fedus et al., 2022). Mixtral further shows that sparse MoE language models can achieve strong open-model performance by selecting experts dynamically for each token (Jiang et al., 2024). System work on MoE, including DeepSpeed-MoE, Tutel, FastMoE, FasterMoE, and MegaBlocks, focuses on efficient training or inference through expert parallelism, adaptive execution, communication optimization, and block-sparse computation (Rajbhandari et al., 2022; Hwang et al., 2023; He et al., 2021; 2022; Gale et al., 2023). These works treat routing primarily as a mechanism for sparse computation and system efficiency. Our work treats expert activation trajectories as empirical signals: we use them to predict output behavior, analyze energy-relevant variation, and identify experts with functional importance beyond activation frequency.

Correctness-aware evaluation. Language model evaluation increasingly emphasizes that raw output metrics are not sufficient for understanding model usefulness. Benchmarking frameworks such as HELM and lm-evaluation-harness encourage standardized, reproducible evaluation across tasks and models (Liang et al., 2023; Biderman et al., 2024). Reasoning benchmarks such as GSM8K further show the importance of correctness-oriented evaluation and verifier-style analysis for generated answers (Cobbe et al., 2021). In system benchmarking, accuracy constraints are also important because faster or cheaper inference is only meaningful when task quality is preserved (Reddi et al., 2020). Our work brings this idea into MoE energy analysis. We argue that raw energy and joules per token are insufficient because they do not reveal whether the consumed energy produces correct answers. We therefore evaluate energy under correctness constraints using energy-per-correct, paired quality-energy deltas, correctness flips, and correctness-critical expert analysis.

Position of this work. Overall, prior work provides strong foundations for measuring inference energy, optimizing LLM serving, scaling MoE computation, and evaluating model correctness. However, these directions are usually studied separately. This paper connects them through a mechanism-oriented empirical study of MoE inference energy. The central distinction is that we do not only ask how much energy a request consumes; we ask how that energy is formed through routing, output behavior, deployment, and correctness.

B Detailed Definitions and Measurement Foundations

This appendix expands the notation and measurement definitions used in the main text. The main paper deliberately keeps the mechanism chain compact. Here we provide the full definitions needed to reproduce the request-level energy measurements, routing features, output-behavior variables, prediction metrics, deployment comparisons, and correctness-constrained quantities. The purpose of this appendix is to make the empirical claims in the main text auditable.

B.1 Request-Level Energy

For request i, let s_i and e_i denote the monotonic start and end time of the measured generation window. Let $\mathcal{G}$ be the set of visible GPUs and let $P_g(t)$ denote the instantaneous power draw of GPU g at time t (Chung et al., 2025). The ideal continuous request-level energy is

$$E_i = \sum_{g \in \mathcal{G}} \int_{s_i}^{e_i} P_g(t)\, dt. \tag{10}$$

In practice, GPU power is sampled discretely. Let t_k denote a sampled timestamp and let Δt_k denote the time interval represented by that sample. For the request-specific sample set $\mathcal{K}_i$, the discrete approximation is

$$E_i \approx \sum_{g \in \mathcal{G}} \sum_{k \in \mathcal{K}_i} P_g(t_k) \Delta t_k. \tag{11}$$

The full benchmark runner reports this request-window energy as `decode_energy_j`. Since the measured window corresponds to generation, this quantity is used consistently for output-to-energy modeling, deployment comparison, and energy-per-correct analysis. The analysis focuses on relative patterns and matched comparisons rather than claiming that a single measured energy coefficient is universal across hardware, drivers, frameworks, or workloads.

Why request-level integration is necessary. Average power or average GPU utilization can hide substantial request-level heterogeneity. Two requests can have similar average utilization but different generation lengths, different per-GPU distributions, or different deployment-induced energy costs. Integrating the power trace over the request window provides the unit of analysis needed for paired comparisons and correctness-constrained evaluation.

B.2 Output Behavior Variables

For each request, we record the generated text, output token count O_i, generation time T_i, and a hit-max indicator H_i. The hit-max indicator is defined as

$$H_i = \mathbb{I}[O_i = O_{\max}]. \tag{12}$$

This variable identifies generations that stop because they reach the maximum token budget rather than because the model naturally terminates. It is useful because capped generations may have different energy behavior from naturally terminated generations. In the main analysis, output behavior is treated as the nearest observable layer before energy consumption: autoregressive decoding consumes energy as output tokens are produced (Yu et al., 2022; Agrawal et al., 2024), and longer or slower generations generally lead to higher measured generation-window energy.

B.3 MoE Routing and Expert Activation

For request i, token t, layer l, and expert e, let $a_{i,t,l,e} = 1$ if token t activates expert e at layer l, and 0 otherwise. The expert activation trajectory is

$$A_i = \{a_{i,t,l,e} \mid t \in \mathcal{T}_i,\ l \in \mathcal{L},\ e \in \mathcal{E}\}. \tag{13}$$

The request-level activation count is

$$C_{i,l,e} = \sum_{t \in \mathcal{T}_i} a_{i,t,l,e}. \tag{14}$$

Because the paper emphasizes early prediction before decode-time generation has unfolded, we separately define the prefill expert count:

$$C^{\text{prefill}}_{i,l,e} = \sum_{t \in \mathcal{P}_i} a_{i,t,l,e}, \tag{15}$$

where $\mathcal{P}_i$ denotes the set of prefill tokens. The core modeling question is whether features derived from $C^{\text{prefill}}_{i,l,e}$ and routing summaries can predict later output behavior. If they can, then prefill routing is not only an internal log; it is an upstream representation associated with later energy-relevant generation behavior.

B.4 Routing Summary Features

The modeling experiments use several families of routing features. First, expert-count features describe how often each expert is selected in each layer and phase. Second, expert-share features normalize counts within a request or layer, making comparisons less sensitive to input length. Third, entropy and concentration features summarize whether routing is diffuse across experts or concentrated in a small number of experts. Fourth, routing-confidence features summarize the sharpness of router probabilities. These features are aggregated at request level and used as predictors for output length and energy-relevant behavior.

This feature design separates two roles of routing. At the mechanistic level, routing determines which experts participate in sparse computation (Shazeer et al., 2017; Lepikhin et al., 2021; Jiang et al., 2024). At the empirical modeling level, reconstructed routing features are treated as observable internal signals that may explain request-to-request variation. This paper mainly makes the second claim: routing features have predictive and explanatory value for output behavior and energy analysis.

B.5 Prediction Metrics

For target y_i and prediction $\hat{y}_i$, we report MAE, RMSE, MAPE, and R^2:

$$\begin{aligned} \mathrm{MAE} &= \frac{1}{N}\sum_{i=1}^{N}|y_i-\hat{y}_i|, \\ \mathrm{RMSE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}. \end{aligned} \tag{16}$$

$$\begin{aligned} \mathrm{MAPE} &= \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i-\hat{y}_i}{y_i+\delta}\right|, \\ R^2 &= 1-\frac{\sum_i(y_i-\hat{y}_i)^2}{\sum_i(y_i-\bar{y})^2}. \end{aligned} \tag{17}$$

The metrics are complementary. MAE reports average absolute error in the target unit; RMSE penalizes large errors more strongly; MAPE reports relative error; and R^2 measures explained variance. In this paper, R^2 is the most concise indicator for comparing feature groups, while MAE and RMSE are used to assess the absolute magnitude of prediction error. Where $\delta > 0$ is a small numerical constant used to avoid division by zero.

C Full Experimental Setup

This appendix provides the full configuration supporting the compact setup description in the main text. The key design principle is controlled comparison: we use deterministic decoding, fixed deployment identifiers, saved request identifiers, and saved request-level outputs so that energy, output behavior, deployment, and correctness can be joined without ambiguity.

Organization of the experimental suites. The paper uses three complementary datasets rather than one request table containing every variable. The controlled modeling suite records request-level routing-derived features together with output behavior and measured energy for routing-to-output and output-to-energy modeling. The full benchmark suite emphasizes large-scale deployment, correctness, and energy comparison. The expert-intervention suite separately records screening, validation, and mechanism runs for expert-level functional analysis. These suites use the same Mixtral model family and HuggingFace execution path, but variables from different intervention protocols are not treated as if they had been jointly measured on the same requests. This separation is important for interpreting the routing, correctness, and energy results in Sections 4.1–4.5.

C.1 Model and Decoding Configuration

All experiments use `mistralai/Mixtral-8x7B-Instruct-v0.1` (Jiang et al., 2024). The model is loaded using the HuggingFace `transformers` backend with FP16 inference (Wolf et al., 2020).

Quantization, speculative decoding, and stochastic sampling are disabled. The tokenizer chat template is used when available. Deterministic decoding makes paired comparisons more interpretable because output differences are less likely to be caused by random sampling.

The experiments are not designed to compare serving frameworks. They use a fixed HuggingFace-based execution path so that the analysis can focus on request-level energy formation, routing-derived features, deployment differences, and correctness. Results should therefore be interpreted as system-specific empirical evidence for this model-backend-hardware setting, not as universal constants for all MoE serving systems.

C.2 Deployment Configurations

Table 3: Deployment configurations used in the full benchmark experiments. The three configurations provide controlled variants of HuggingFace device mapping for testing whether similar output behavior can incur different energy costs.

Short name	Deployment ID	Device mapping strategy
Auto-8GPU	HF_AUTO_8GPU_A2000	HuggingFace automatic device map
ManualContig-8GPU	HF_8GPU_MANUAL_CONTIG_A2000	contiguous layer-to-GPU assignment
Balanced-8GPU	HF_8GPU_BALANCED_A2000	round-robin balanced layer assignment

For manual mappings, the embedding layer is placed on the first GPU and the final normalization and language-modeling head are placed on the last GPU. The manual-contiguous strategy assigns consecutive transformer layers to each GPU block. The balanced strategy assigns layers across GPUs in a round-robin pattern. These mappings are not claimed to be globally optimal expert-placement strategies. They are controlled deployment variants used to test whether deployment affects the hardware energy coefficient under similar generation behavior (Rajbhandari et al., 2022; Hwang et al., 2023; He et al., 2022).

C.3 Benchmarks and Generation Settings

Table 4: Benchmark datasets and generation settings used in the full benchmark correctness-energy evaluation.

Benchmark	Dataset	Split	Metric	Max new tokens
gsm8k	openai/gsm8k	test	exact numeric match	256
mmlu	cais/mmlu	test	multiple-choice accuracy	16
arc_challenge	allenai/ai2_arc	test	multiple-choice accuracy	16
hellaswag	Rowan/hellaswag	validation	multiple-choice accuracy	16

For `gsm8k`, the model is prompted to solve the problem and produce a final numeric answer. Correctness is evaluated by extracting the last numeric answer from the response and comparing it with the reference answer. For multiple-choice benchmarks, the model is prompted to answer with an option letter, and correctness is computed by comparing the extracted choice with the reference label. The generation limits intentionally differ across task types: `gsm8k` requires longer reasoning-style outputs, while the multiple-choice benchmarks are evaluated with short answer-letter generations (Cobbe et al., 2021; Hendrycks et al., 2021; Clark et al., 2018; Zellers et al., 2019).

C.4 Energy Monitoring Protocol

Power is sampled from all visible GPUs using `nvidia-smi`. For each sample, the monitor records GPU index, GPU name, instantaneous power draw, utilization, memory usage, and timestamp. The sampling interval is 0.10 seconds. For each request, the runner records the monotonic start and end time of generation. Energy is computed by summing the GPU power trace over the request window (Chung et al., 2025).

The runner performs warm-up requests before measurement. A short cooldown interval is used before monitored generation to reduce initialization and scheduling artifacts. The analysis excludes incomplete or invalid request records. The final benchmark summaries contain 26,574–26,575 evaluated requests per full-benchmark deployment. The slight difference in request count across deployments comes from completed and valid records after loading, generation, and filtering.

C.5 Routing Reconstruction and Expert Intervention

For full benchmark deployment comparison, routing capture is disabled to reduce runtime and storage overhead. Expert activation analysis is performed in separate modeling and intervention runs. In these runs, router logits are reconstructed or captured and converted into expert assignment features (Lepikhin et al., 2021; Jiang et al., 2024). The prefill/decode split is determined using input length (Agrawal et al., 2024). Phase-specific expert counts, expert shares, layer-phase entropy, and concentration features are then used for modeling and expert analysis.

Expert intervention experiments include hard banning and soft downweighting. Hard intervention removes or masks an expert under the tested phase setting. Soft intervention reduces the expert's routing influence without completely removing it. Phase settings include prefill-only, decode-only, and all-phase intervention. The validation stage identifies correctness-critical experts; the mechanism stage compares intervention mode and phase.

Why intervention is separated from full benchmark deployment comparison. The full benchmark is designed for deployment-level correctness-energy comparison at large request scale. Expert intervention is designed to test functional sensitivity of selected experts. The paper treats them as complementary: full benchmark runs provide large-scale correctness-energy evidence, while intervention runs identify expert-level functional effects.

D Full Prediction Results

This appendix supports the first two empirical questions in the main text. The first question asks whether output behavior explains decode energy. The second asks whether prefill expert activation predicts future output behavior before decoding starts. Together, the two prediction analyses support the chain from prefill routing to output behavior and from output behavior to energy.

D.1 Decode Energy Prediction

Table 5: Decode energy prediction results. The progression from baseline features to output-length and deployment-aware features shows that output behavior is the strongest direct observable driver of decode energy, while deployment-aware information explains additional residual variation.

Model	Main features	MAE	RMSE	MAPE	R^2
Baseline Ridge	input/task/basic	2189.656 J	2976.022 J	0.242	0.243
Expert-prefill Ridge	baseline + prefill expert/routing	1702.375 J	2449.521 J	0.171	0.487
Hardware Ridge	hardware/deployment features	1459.462 J	2249.836 J	0.147	0.568
Length-only Ridge	output length	1025.238 J	1805.762 J	0.091	0.721
Hardware-aware length Ridge	output length + deployment	746.483 J	1538.230 J	0.065	0.798
Output-behavior OLS	output tokens + hit-max + decode time	927.05 J	1103.55 J	0.085	0.890
Deployment-aware Linear	output behavior + input/task + deployment interaction	251.08 J	443.55 J	0.023	0.982
Nonlinear CatBoost	output behavior + input/task + deployment	**154.88 J**	**343.36 J**	**0.014**	**0.989**

Table 5 reports the original compact ablation from request metadata, routing, output length, and deployment features. To test the output-to-energy relationship more directly, we additionally refit the three-variable output-behavior model in Equation 2 using a prompt-grouped split. Eight degenerate decode records with no aligned nonzero decode-energy measurement are excluded, leaving 23,992

valid requests. The held-out results are $R^2 = 0.890$, MAE $= 927.05$ J, RMSE $= 1103.55$ J, and MAPE $= 8.50\%$. Five-fold grouped cross-validation gives $R^2 = 0.893 \pm 0.004$ (range 0.889–0.898), confirming that the strong output-behavior fit is not specific to a single random split.

We then add input length, task fixed effects, deployment fixed effects, and a deployment-specific decode-time interaction as in Equation 3. This interpretable model reaches $R^2 = 0.982$ on the held-out test set (MAE $= 251.08$ J; RMSE $= 443.55$ J; MAPE $= 2.26\%$), and $R^2 = 0.9833 \pm 0.0004$ under five-fold grouped cross-validation. A nonlinear gradient-boosted tree model using the same observable output, input, task, and deployment information reaches $R^2 = 0.989$ on the held-out set and $R^2 = 0.9891 \pm 0.0019$ under grouped cross-validation.

Feature-attribution analysis of the nonlinear model assigns the largest average absolute contribution to decode time (38.8%), followed by deployment (25.8%), output tokens (24.8%), hit-max behavior (9.2%), input length (1.0%), and task type (0.4%). We interpret these values as predictive importance within this fitted model rather than causal effects. The close performance of the interpretable deployment-aware model and the nonlinear model indicates that output behavior and deployment jointly account for nearly all of the predictable decode-energy structure in this dataset.

Interpretation. These results clarify the role of output behavior in the energy-formation chain. Output length and decode time are the closest directly observable variables to request-level decode energy, while deployment changes the hardware cost associated with realizing that behavior. This does not mean expert routing is unimportant: routing remains an upstream signal associated with later generation behavior and is therefore useful before generation finishes. By contrast, the output-energy models in this subsection are post-hoc explanatory models because final output length and decode time are only fully observed after the corresponding energy has been consumed.

D.2 Output Length Prediction

Table 6: Output length prediction results. Prefill expert and routing features substantially improve output prediction beyond input length and task type.

Group	Target	Model	Features	MAE	RMSE	MAPE	R^2
Baseline	output tokens	Ridge	input length + task	31.603	44.298	0.538	0.351
Expert-prefill	output tokens	Ridge	input+task+prefill expert/routing	20.914	29.424	0.430	0.714
Hardware	output tokens	Ridge	input+task+deployment/hardware	20.896	29.405	0.430	0.714
Baseline	output tokens	Tree	input length + task	29.199	42.806	0.509	0.394
Expert-prefill	output tokens	Tree	input+task+prefill expert/routing	14.493	24.434	0.390	0.803
Confidence	output tokens	Tree	routing confidence features	13.158	22.664	0.377	0.830

The improvement from the baseline models to expert-prefill models indicates that prefill routing contains information about future generation behavior. The ridge model improves from $R^2 = 0.351$ to $R^2 = 0.714$ when prefill expert and routing features are added. Tree-based models further improve performance, with expert-prefill features reaching $R^2 = 0.803$ and confidence features reaching $R^2 = 0.830$.

Interpretation. This result supports the use of prefill expert activation as an upstream energy-relevant representation. It does not require claiming that prefill routing causally determines output length. The more careful interpretation is that both the prompt and task demand shape early routing behavior, and this early routing behavior is predictive of later generation behavior. Since generation behavior directly drives energy, prefill routing becomes useful for early energy-aware request characterization.

D.3 Feature-Group Interpretation

The prediction results suggest a layered interpretation of the inference process. Input length and task type provide a weak but meaningful prior about output behavior. Prefill routing features add internal information about how the model processes the prompt. Output behavior then becomes the

strongest direct predictor of decode energy. Deployment-aware information modifies the mapping from output behavior to measured energy. This layered interpretation motivates the main paper's input-routing-output-energy chain.

Routing-mediated versus routing-residual energy contribution. To distinguish whether routing contributes to energy mainly through subsequent output behavior or through an additional residual pathway, let X_i denote external request features such as input length and task type, A_i^{prefill} prefill routing features, and Y_i output behavior. We compare the nested models

$$M_1 : E_i \sim X_i, \tag{18}$$

$$M_2 : E_i \sim X_i + A_i^{\text{prefill}}, \tag{19}$$

$$M_3 : E_i \sim X_i + Y_i, \tag{20}$$

$$M_4 : E_i \sim X_i + Y_i + A_i^{\text{prefill}}. \tag{21}$$

The corresponding predictive gains are

$$\Delta_{\text{routing,total}} = R^2(M_2) - R^2(M_1), \tag{22}$$

$$\Delta_{\text{routing,direct}} = R^2(M_4) - R^2(M_3). \tag{23}$$

If $\Delta_{\text{routing,total}}$ is large but $\Delta_{\text{routing,direct}}$ is small, routing is primarily useful as an upstream predictor of output-mediated energy demand. If $\Delta_{\text{routing,direct}}$ remains large after conditioning on output behavior, this points to residual path-dependent effects such as expert placement, cross-GPU communication, or load imbalance (Rajbhandari et al., 2022; Hwang et al., 2023; He et al., 2022). This decomposition is interpretive and predictive rather than a causal mediation estimate.

E Expert Utility and Importance Details

This appendix gives the full definition and validation procedure for the expert-utility analysis in Section 4.3. The main distinction is between usage statistics and practical utility. Activation frequency and routing confidence describe how an expert is used; intervention outcomes describe what changes when the expert is perturbed; correctness-retention utility measures whether preserving that expert helps maintain correct outputs under a specified workload and intervention protocol.

E.1 Utility Definition and Paired Response Model

For a fixed expert e, benchmark b, and intervention protocol a, each paired request belongs to one of three correctness-transition categories: correct-to-wrong, wrong-to-correct, or unchanged correctness. Let the corresponding counts be $N^+_{e,b,a}$, $N^-_{e,b,a}$, and $N^0_{e,b,a}$. We model these paired outcomes as

$$\left(N^+_{e,b,a}, N^-_{e,b,a}, N^0_{e,b,a}\right) \sim \text{Multinomial}\left(n_b; p^+_{e,b,a}, p^-_{e,b,a}, p^0_{e,b,a}\right), \tag{24}$$

with task-conditional correctness utility

$$\theta^Q_{e,b,a} = p^+_{e,b,a} - p^-_{e,b,a}. \tag{25}$$

The empirical estimator is the paired mean used in Equation 7. This makes the relationship to the correctness-flip statistics explicit:

$$\widehat{\theta}^Q_{e,b,a} = \widehat{\text{C2W}}_{e,b,a} - \widehat{\text{W2C}}_{e,b,a} = \widehat{\text{Acc}}_b^{(0)} - \widehat{\text{Acc}}_b^{(e,a)}. \tag{26}$$

Therefore, C2W, W2C, net flip impact, and accuracy loss are not treated as four independent score dimensions. As a weak-regularization sensitivity check, we also consider a symmetric $\text{Dirichlet}(1/2, 1/2, 1/2)$ prior, for which the posterior mean of the net correctness utility is

$$\widetilde{\theta}^Q_{e,b,a} = \frac{N^+_{e,b,a} - N^-_{e,b,a}}{n_b + 3/2}. \tag{27}$$

The main results use the direct paired estimate rather than the prior-smoothed value. For a workload mixture π, we aggregate task effects by $100 \sum_b \pi_b \widehat{\theta}^{Q}_{e,b,a}$. The main analysis uses equal weights across the seven benchmark families in the screening and validation experiments. This weighting defines the evaluation workload; it is not tuned after observing the ranking. In a deployment with a known request distribution, π can instead be set to the observed workload proportions.

Uncertainty is kept separate from utility. We use task-stratified shared-request bootstrap resampling so that, within each benchmark, all experts are evaluated on the same resampled request indices. This preserves the paired comparison structure between experts. Confidence intervals, rank intervals, and Top-k selection frequencies are reported as evidence about estimation reliability rather than added to the utility score itself.

E.2 Data Separation and Out-of-Sample Validation

The screening stage contains paired hard-intervention results for all 256 experts on 608 requests. The original validation experiment evaluates 32 selected experts on a larger benchmark mixture. Because some underlying questions appear in both stages, we match by the original sample identity rather than by prompt text alone: option shuffling can change the prompt while leaving the underlying question unchanged. We identify 166 overlapping questions and remove them from validation, leaving 4,372 requests that are disjoint from the screening questions.

The screening ranking is frozen before evaluating this disjoint set. Among the 32 original validation candidates, the task-balanced utility score has a Spearman correlation of 0.543 with validation utility, with a request-bootstrap interval of approximately $[0.290, 0.654]$. The previous Stage2 composite proxy reaches 0.202, and the Stage2 activation-frequency proxy reaches -0.037. Because these Stage2 features come from a different controlled workload, they are treated only as cross-workload descriptive proxies. A C2W-only score computed on the same screening data reaches a Spearman correlation of 0.303, so the present results do not justify claiming that the net-utility score dominates every simpler same-condition comparator.

Selecting the top eight experts by the task-balanced utility score gives a mean validation loss of 3.509 percentage points when those experts are intervened on one at a time; the 95% bootstrap interval is $[2.773, 4.291]$. This is not the accuracy change from simultaneously removing eight experts. Relative to the C2W-only top-eight selection, the mean difference is 0.150 percentage points with interval $[-0.103, 0.428]$, which does not establish a clear advantage. We therefore use the score because it has a direct signed interpretation and an auditable validation procedure, rather than claiming universal superiority over all possible ranking rules.

Table 7: Expert utility estimated from screening interventions and evaluated on disjoint validation requests. Utility and confidence intervals are reported in accuracy percentage points (pp).

Expert	Screen U.	Scr. CI-L	Scr. CI-H	Val. U.	Val. CI-L	Val. CI-H	Rank-L	Rank-H	$P(\text{Top8})$
L1_E3	19.717	15.104	24.520	18.335	16.556	20.115	1	1	1.000
L31_E2	8.557	6.027	11.086	6.820	5.777	7.830	2	2	1.000
L11_E4	4.539	1.524	7.589	1.333	0.193	2.544	3	31	0.888
L16_E5	2.753	-0.186	5.693	0.287	-1.007	1.545	3	149	0.447
L6_E0	2.641	0.260	5.022	-0.059	-0.980	0.856	3	107	0.354

On the disjoint validation set, `L1_E3` has a task-balanced utility of 18.335 percentage points with 95% interval $[16.556, 20.115]$, and `L31_E2` has 6.820 points with interval $[5.777, 7.830]$. `L11_E4` is much smaller at 1.333 points with interval $[0.193, 2.544]$, while several other screened experts have intervals that include zero. Using approximate simultaneous bootstrap bands over the evaluated experts, `L1_E3` and `L31_E2` are the two experts whose positive utility is most consistently supported across screening and validation.

A broader repeated grouped-validation check over all 256 experts gives a different message: fine-grained global ordering is unstable. Across repeated request-grouped folds, the task-balanced net-utility ranking has a mean Spearman correlation of approximately -0.030 between the training-derived ranking and held-out expert effects. The selected top experts can still exhibit substantially

larger held-out effects than random selection, but this result prevents us from interpreting the method as an exact total ordering of all 256 experts. The practical conclusion is narrower: current data support reliable identification of a small set of high-impact experts, while the remaining fine ranking has substantial uncertainty.

E.3 Frequency versus Functional Utility

Activation frequency remains useful as a usage statistic, but it is not the target quantity in the utility score. Frequency answers how often an expert is selected; paired utility answers whether preserving that expert helps retain correct outputs under a specified intervention. The two can disagree because frequently used experts may be replaceable for a task, while less frequent experts can have large functional effects on a smaller subset of requests.

Table 8: Contrast between activation frequency and intervention sensitivity. High-frequency experts are not always high-impact experts, and lower-frequency experts can have stronger intervention effects.

Expert	Type	Freq. imp.	Interv. sens.	$\lvert\Delta O\rvert$	$\lvert\Delta E\rvert$
L12_E5	high-frequency, low-intervention	0.973	0.362	5.156	388.325 J
L13_E1	high-frequency, low-intervention	0.954	0.378	7.094	467.751 J
L24_E3	high-frequency, low-intervention	0.748	0.320	3.953	397.955 J
L10_E4	low-frequency, high-intervention	0.211	0.817	11.563	1162.762 J
L31_E2	low-frequency, high-intervention	0.272	0.802	12.094	1001.156 J
L16_E3	low-frequency, high-intervention	0.223	0.708	10.969	906.898 J

Table 8 provides a descriptive example of this difference in the controlled intervention workload. It should not be read as the definition of the new correctness-retention score. In particular, output-length and energy changes are useful for characterizing what an expert changes, while the signed paired correctness effect determines whether that change is useful under the benchmark objective.

E.4 Conditional Utility and Complementary Intervention Evidence

The utility of an expert is not constant across tasks or phases. On the disjoint validation requests, `L1_E3` has large positive correctness-retention effects on `arc_challenge` (31.79 percentage points), `gsm8k` (33.45), and `hellaswag` (31.12), but the estimated effect on `gpqa` is negative (-7.03). `L31_E2` has a particularly large effect on `gsm8k` (33.45) but much smaller effects on several other tasks. These results motivate reporting a task-conditioned utility profile rather than interpreting the workload-average score as a universal property of the expert.

The mechanism experiment provides an additional phase-conditioned view. After excluding questions already used in screening or validation, 419 requests remain across six benchmarks. For `L1_E3`, hard all-phase, hard prefill, and hard decode intervention give task-balanced correctness losses of 23.424, 22.898, and -1.267 percentage points, respectively. The prefill-minus-decode contrast is 24.165 points with a 95% bootstrap interval of $[16.229, 32.055]$. For `L31_E2`, the corresponding hard prefill and hard decode effects are 2.791 and 4.699 points. This supports a mainly prefill-stage correctness role for `L1_E3`, while `L31_E2` shows evidence of dependence across both phases. Phase effects are not additive, and a near-zero decode-only effect is not by itself evidence that an expert can be safely removed from decode.

The Stage2 output/energy intervention experiment supplies a complementary behavioral view. These measurements were collected under a different `mask/decode` protocol and are therefore not numerically merged with the correctness-retention score. In particular, correctness and energy were not jointly measured for the same intervention requests, so a single empirical quality–energy utility score would not be supported by the present data.

Table 9: Expert intervention effects on output and decode energy. This table focuses on behavioral and energy changes rather than correctness. Correctness-oriented intervention results are reported in Appendix G.

Expert	Intervention	Phase	n	Mean ΔO	Mean $\lvert\Delta O\rvert$	Mean ΔE	Similarity
L15_E0	mask	decode	64	3.344	8.719	1199.277 J	0.526
L15_E2	mask	decode	64	-3.281	7.219	957.877 J	0.551
L7_E2	mask	decode	64	-8.500	11.688	298.705 J	0.590
L2_E3	mask	decode	64	-0.547	7.641	851.258 J	0.579
L13_E4	mask	decode	64	4.453	6.359	985.003 J	0.530

These intervention results show that perturbing selected experts can alter output length, energy, and text similarity, but the direction and meaning of the change depend on the target quantity. For practical expert management, the current evidence therefore separates three questions: how often an expert is used, how strongly its intervention changes model behavior, and how much correctness is retained by preserving it. This separation avoids assigning a high practical value merely because an expert is frequently activated or because its intervention produces a large but potentially beneficial change.

A joint energy–correctness optimization would require both quantities to be measured under the same request and intervention protocol. One possible application-level utility is $U(q, E) = q - \lambda E/E_{\text{ref}}$, but λ represents a deployment preference rather than a data-determined constant. We therefore leave such a combined score as a downstream decision rule rather than treating it as an experimental result of the present study.

F Deployment Analysis Details

This appendix supports the deployment-related claim in the main text. The deployment analysis separates two effects: output behavior determines the main amount of generation work, while deployment configuration changes how expensive similar work is on the hardware.

F.1 Controlled Deployment Energy Summary

Table 10: Deployment-level decode energy summary from controlled deployment experiments. The deployments have similar mean output lengths but different decode energy and joules per token.

Deployment	Mean output tokens	Mean decode energy	J/token
Auto-8GPU-A2000	219.08	11869.60 J	54.10
Auto-8GPU-B10000	219.70	11957.22 J	54.35
Balanced-8GPU-A2000	219.08	12262.08 J	55.86
ManualContig-8GPU-A2000	219.08	12407.84 J	56.59

The controlled deployment results show that similar output behavior does not imply identical energy cost. The mean output token counts are almost the same across several deployments, but the mean decode energy and joules per token differ. This motivates the deployment-aware energy model in the main text.

F.2 Additional Paired Deployment Deltas

Table 11: Paired deployment energy delta relative to the 8-GPU auto baseline in additional controlled deployment experiments. Negative values indicate lower mean decode energy than the baseline under matched requests. The numerical suffix denotes the physical GPU subset used in each 6-GPU run.

Compared deployment	Mean ΔE^{decode}
Auto-6GPU-012345	-2354.28 J
ManualContig-6GPU	-1759.69 J
Auto-6GPU-014567	-1685.77 J
Auto-6GPU-012367	-1086.62 J

These additional paired deltas are not the main full-benchmark result, but they reinforce the same point: deployment is part of the energy formation chain. The same model and similar request behavior can be assigned to hardware differently, changing the measured energy cost.

F.3 Full Benchmark Deployment-Level Correctness-Energy Summary

Table 12: Full benchmark deployment-level correctness-energy summary. Accuracy and output length are nearly identical, but energy per correct answer differs across deployments.

Deployment	N	Acc.	Out.	Time	Energy	J/tok.	Corr.	EPC
Auto-8GPU	26574	0.672	20.45	1.76	981.36 J	41.79	17849	1461.07 J
ManualContig-8GPU	26575	0.672	20.44	1.76	996.11 J	42.01	17850	1483.01 J
Balanced-8GPU	26574	0.672	20.45	1.78	1006.97 J	42.63	17849	1499.20 J

Table 12 is central to the correctness-energy argument. The three deployments have essentially identical accuracy and mean output length, yet their energy per correct answer differs. This means the difference is not explained by one deployment producing much longer or much more correct outputs. Instead, the deployment changes the energy cost of similar benchmark behavior.

F.4 Paired Quality-Energy Deltas

Table 13: Paired deployment quality-energy deltas. Requests are matched before computing deltas, so energy differences are compared under aligned prompt and output records.

Deployment A	Deployment B	Pairs	ΔE	ΔT	ΔO	ΔCorrect	ΔScore
Bal-8GPU	Man-8GPU	26574	10.82 J	0.02	0.00	0.00	0.000
Bal-8GPU	Auto-8GPU	26573	25.61 J	0.02	0.00	0.00	0.000
Man-8GPU	Auto-8GPU	26574	14.79 J	-0.00	0.00	0.00	0.000

The paired comparison is stricter than comparing deployment-level averages. By matching request identifiers, it reduces the risk that energy differences come from different prompt mixtures. The observed nonzero energy deltas under zero output-token and correctness deltas provide direct support for the claim that deployment changes the hardware energy coefficient.

G Correctness-Constrained Results

This appendix expands the correctness-constrained analysis. The main motivation is that low energy is not automatically useful. An inference system can consume less energy by producing short, incomplete, or wrong answers. Conversely, a longer generation can be expensive but useful if it solves the task correctly. We therefore analyze energy jointly with correctness.

G.1 Energy per Correct Answer

For deployment d and benchmark b, energy per correct answer is defined as

$$\mathrm{EPC}_{d,b} = \frac{\sum_{i \in \mathcal{D}_{d,b}} E_i^{\mathrm{decode}}}{\sum_{i \in \mathcal{D}_{d,b}} q_i + \delta}, \tag{28}$$

where $q_i \in \{0, 1\}$ is the correctness label. EPC is not a replacement for accuracy. It is a complementary metric that asks how much measured generation-window energy is spent per correct answer. It is most useful when interpreted together with accuracy, output length, and task type.

The task-level results show why correctness-constrained efficiency is necessary. Multiple-choice tasks have short outputs and relatively low EPC. In contrast, `gsm8k` combines long outputs with lower accuracy, producing much higher EPC. This does not mean that multiple-choice tasks are always easier in an absolute sense; it means that under this evaluation format they require less measured decode energy per correct answer.

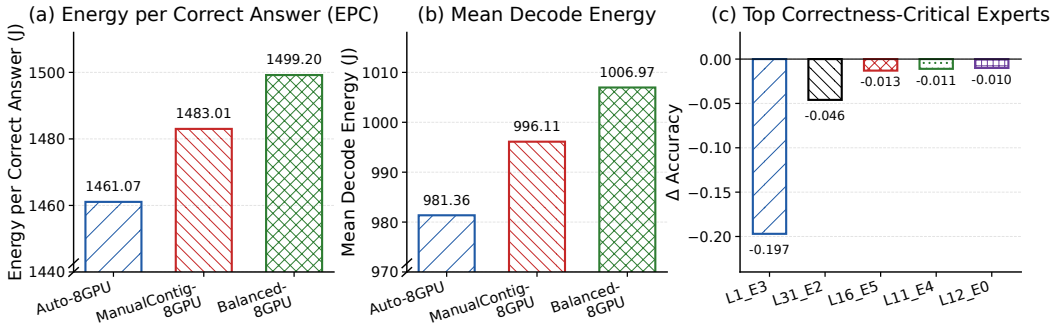

Figure 3: Deployment and correctness-constrained efficiency results. The figure combines deployment-level energy per correct answer, mean decode energy, and the largest correctness drops under expert intervention.

G.2 Correctness-Critical Experts

For expert e, we define accuracy change, correct-to-wrong rate, wrong-to-correct rate, and net flip impact as

$$\begin{aligned} \Delta Acc_e &= Acc_e^{\text{intervention}} - Acc^{\text{baseline}}, \\ C2W_e &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[q_i^{\text{base}} = 1 \wedge q_i^{\text{int}} = 0], \\ W2C_e &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[q_i^{\text{base}} = 0 \wedge q_i^{\text{int}} = 1], \\ NFI_e &= C2W_e - W2C_e. \end{aligned} \tag{29}$$

A large positive NFI_e indicates that intervention converts correct answers into wrong answers more often than the reverse. This is the main signal used to identify correctness-critical experts.

Table 14: Top correctness-critical experts under validation hard-ban intervention. Each expert is evaluated on 4,538 aligned baseline–intervention request pairs. The table separates accuracy changes, correctness flips, and behavioral changes.

Expert	Accuracy			Correctness flips			Behavior		
	Base	Int.	ΔAcc.	C2W	W2C	NFI	Ans.	ΔO	ΔT
L1_E3	0.586	0.389	-0.197	0.283	0.086	0.197	0.678	-7.10	-0.72
L31_E2	0.586	0.539	-0.046	0.059	0.013	0.046	0.180	-11.32	-1.07
L16_E5	0.586	0.573	-0.013	0.062	0.049	0.013	0.285	0.96	-0.05
L11_E4	0.586	0.575	-0.011	0.054	0.043	0.011	0.254	-0.01	-0.13
L12_E0	0.586	0.576	-0.010	0.042	0.032	0.010	0.212	0.12	-0.12
L10_E1	0.586	0.579	-0.007	0.040	0.033	0.007	0.210	0.08	-0.12
L14_E0	0.586	0.579	-0.007	0.046	0.039	0.007	0.241	0.03	-0.13
L13_E1	0.586	0.580	-0.006	0.044	0.038	0.006	0.228	0.24	-0.11

The strongest result is L1_E3. Hard-banning it produces the largest accuracy drop and the largest positive net flip impact, meaning that the intervention mainly turns correct answers into wrong answers. L31_E2 has a smaller overall accuracy drop but remains notable because it has a positive net flip impact and strong task-specific effects. The remaining experts have smaller average drops, showing that correctness-criticality is concentrated rather than uniformly distributed across all selected experts.

G.3 Benchmark-Level Expert Effects

The benchmark-level results show that correctness-criticality is not uniform across tasks. L1_E3 is strongly damaging to remove on several benchmarks, but the effect is not negative for every dataset. L31_E2 has a smaller average validation drop but a large drop on `gsm8k`. This supports a task-dependent view of expert importance and suggests that expert protection or scheduling decisions should consider benchmark or workload composition.

G.4 Mode-Phase Intervention Results

Table 15: Average effects of intervention mode and phase in the mechanism subset. Hard prefill intervention reveals stronger correctness effects than decode-only intervention on average.

Mode	Phase	Exp.	Mean ΔA	Med. ΔA	Min ΔA	C2W	W2C	NFI	Ans.	ΔO
hard	all	16	-0.014	-0.004	-0.178	0.050	0.036	0.014	0.261	-2.03
hard	prefill	16	-0.013	-0.002	-0.176	0.045	0.032	0.013	0.225	-1.67
hard	decode	16	0.000	0.002	-0.031	0.012	0.012	0.000	0.128	-1.34
soft-2	all	16	-0.003	0.000	-0.015	0.032	0.029	0.003	0.211	-1.04
soft-4	all	16	-0.001	-0.002	-0.031	0.038	0.036	0.001	0.239	-1.20

Table 16: Mode-phase intervention effects for key correctness-critical experts. The results separate prefill-stage and decode-stage sensitivity.

Expert	Mode	Phase	Base Acc.	Int. Acc.	ΔAcc.	NFI
L1_E3	hard	all	0.548	0.370	-0.178	0.178
L1_E3	hard	prefill	0.548	0.372	-0.176	0.176
L1_E3	hard	decode	0.548	0.554	0.007	-0.007
L1_E3	soft-2	all	0.548	0.549	0.002	-0.002
L1_E3	soft-4	all	0.548	0.561	0.013	-0.013
L31_E2	hard	all	0.548	0.508	-0.039	0.039
L31_E2	hard	prefill	0.548	0.525	-0.023	0.023
L31_E2	hard	decode	0.548	0.516	-0.031	0.031
L31_E2	soft-2	all	0.548	0.535	-0.013	0.013
L31_E2	soft-4	all	0.548	0.516	-0.031	0.031

The mode-phase results explain where correctness-critical effects arise. For L1_E3, hard prefill-only intervention has almost the same effect as hard all-phase intervention, while hard decode-only intervention does not reduce accuracy. This suggests that the correctness-critical role of L1_E3 is mainly formed during prefill. For L31_E2, both prefill and decode interventions reduce accuracy, indicating a more cross-stage role.

H Additional Reproducibility and Data-Joining Details

The analysis uses deterministic decoding, fixed deployment identifiers, saved request identifiers, saved request-level outputs, saved power traces, and saved correctness labels. Each table in the paper is generated from saved benchmark summary files or expert-intervention summary files. The main reproducibility requirement is to preserve the mapping between request identifiers, deployment identifiers, generated outputs, energy records, and correctness labels.

H.1 Request Matching for Deployment Analysis

For deployment analysis, paired comparisons should be computed only on matched request identifiers. This prevents differences in prompt mixture, benchmark composition, or request count from being mistaken for deployment effects. The paired deployment tables therefore report the number of matched pairs and then compute differences in energy, time, output tokens, correctness, and score on those matched pairs.

H.2 Request Matching for Expert Intervention Analysis

For expert intervention analysis, baseline and intervened records should be joined by request identifier, benchmark, expert, intervention mode, and intervention phase. This prevents mixing unmatched samples and ensures that output changes, energy changes, and correctness flips are computed on aligned request pairs. Correct-to-wrong and wrong-to-correct flips are meaningful only when the baseline and intervention records refer to the same original request.

H.3 How the Appendix Supports the Main Claims

The appendices support the main paper as follows. Appendix B defines the energy, routing, output, and evaluation metrics. Appendix C documents the model, decoding, benchmark, deployment, monitoring, and intervention settings. Appendix D supports the claim that output behavior drives energy and that prefill routing predicts output behavior. Appendix E supports the claim that expert importance is multi-dimensional. Appendix F supports the claim that deployment changes the energy coefficient under similar behavior. Appendix G supports the claim that useful inference efficiency must be evaluated under correctness constraints.