Characterizing LLM Inference Energy-Performance Tradeoffs across Workloads and GPU Scaling

Paul Joe Maliakel
HPC Group
TU Wien, Vienna, Austria
paul.maliakel@tuwien.ac.at

Shashikant Ilager
Multiscale Networked Systems Group
University of Amsterdam, Netherlands
s.s.ilager@uva.nl

Ivona Brandic
HPC Group
TU Wien, Vienna, Austria
ivona.brandic@tuwien.ac.at

***Abstract*—LLM inference exhibits substantial variability across queries and execution phases, yet inference configurations are often applied uniformly. We present a measurement-driven characterization of workload heterogeneity and energy–performance behavior of LLM inference under GPU dynamic voltage and frequency scaling (DVFS). We evaluate five decoder-only LLMs (1B–32B parameters) across four NLP benchmarks using a controlled offline setup. We show that lightweight semantic features predict inference difficulty better than input length, with 44.5% of queries achieving comparable quality across model sizes. At the hardware level, the decode phase dominates inference time (77–91%) and is largely insensitive to GPU frequency. Consequently, reducing GPU frequency from 2842 MHz to 180 MHz achieves an average of 42% energy savings with only a 1–6% latency increase. We further provide a use case with an upper-bound analysis of the potential benefits of combining workload-aware model selection with phase-aware DVFS, motivating future energy-efficient LLM inference systems.**

***Keywords*-Energy efficiency, LLM inference, benchmarking, DVFS**

I. Introduction

Large Language Models (LLMs) have become a dominant interface for general-purpose AI, powering applications such as interactive assistants, search, and enterprise automation [1]–[4]. Despite rapid algorithmic progress, the cost of LLM inference in terms of energy, latency, and infrastructure provisioning has emerged as a first-order concern for sustainable AI systems [5], [6].

In production, LLM serving systems operate on heterogeneous hardware and must handle diverse query streams. These range from short factual questions to long-context reasoning prompts, with outputs varying from a few tokens to long-form generation [4], [7], [8]. However, existing inference systems typically apply uniform configurations across all queries, implicitly assuming homogeneous workloads and uniform hardware sensitivity. This motivates a systematic analysis of how workload characteristics and hardware configurations jointly affect inference performance and energy cost.

Prior work on efficient LLM inference has focused on system-level optimizations such as batching, KV-cache management, operator fusion, and scheduling [9], [10]. While effective for throughput and latency, these approaches largely rely on coarse-grained workload descriptors, such as input length or token count. Consequently, the interaction between semantic query characteristics, execution phases, and low-level hardware controls such as GPU frequency scaling remains insufficiently understood from an energy-efficiency perspective.

At the workload level, inference cost is often assumed to scale with input length [7]. In practice, this assumption frequently breaks down. Short queries may be semantically difficult due to dense entity usage or implicit reasoning requirements [11]–[13], while long queries may involve comparatively simple retrieval or summarization [14]–[16]. These observations motivate workload characterization based on linguistic and semantic properties beyond surface-level token counts [17], [18].

At the hardware level, LLM inference energy and latency are strongly influenced by accelerator configuration choices, particularly GPU Dynamic Voltage and Frequency Scaling (DVFS). Existing systems typically apply a single frequency across the entire inference process [4], [19]. However, decoder-only inference exhibits a structural asymmetry: a compute-bound *prefill* phase that processes input tokens in parallel which is followed by a memory-bound autoregressive *decode* phase that repeatedly accesses model weights and the growing KV cache [9], [19], [20]. This asymmetry suggests that applying a uniform GPU frequency across the entire inference process may be inherently inefficient.

Accordingly, this paper presents a measurement-driven benchmarking study that jointly characterizes workload-level query heterogeneity and hardware-level energy–performance behavior under GPU DVFS. Using a controlled offline replay-based setup, we evaluate five decoder-only LLMs (1B–32B parameters) across four representative NLP benchmarks spanning classification and generation tasks: BoolQ, HellaSwag, TruthfulQA, and NarrativeQA [16], [21]–[23]. We perform systematic GPU frequency sweeps and measure latency and energy at both end-to-end and phase-level granularity.

Our goal is not to propose a new serving system or scheduling algorithm, but to provide empirical insights into where and why energy is consumed during LLM inference, and which workload and hardware characteristics create opportunities for energy savings. These characterizations form a foundation for workload-aware and phase-aware optimization strategies in future LLM serving systems.

Our measurements show that semantic query characteristics provide stronger signals of workload heterogeneity than input length alone, and that the decode phase dominates inference time and energy consumption while remaining largely insensitive to GPU frequency scaling. A case study further demonstrates how combining workload-aware model selection with phase-aware frequency configuration can substantially improve inference energy efficiency without sacrificing latency.

In summary, this paper makes the following contributions:

- **Workload characterization:** We analyze lexical, syntactic, and semantic query features and show that semantic properties (e.g., entity density) explain inference difficulty better than input length (Section V).
- **DVFS characterization:** We benchmark GPU DVFS across five models and four workloads, revealing a consistent phase-level asymmetry in which the decode phase is comparatively frequency-insensitive (Section VI).
- **Implications via case study:** We demonstrate how workload-aware model selection and phase-aware DVFS can be combined to improve energy efficiency while maintaining quality (Section VII).

Our results provide a benchmarking foundation for energy-efficient LLM inference policies grounded in workload heterogeneity and phase-aware hardware behavior rather than fixed conservative defaults.

II. Background

A. Transformer-based LLM Inference

Transformer decoders generate tokens autoregressively using stacked attention and feed-forward layers [24]. Modern open LLMs span a wide range of sizes and architectures, including GPT-style decoders [25], OPT-style baselines [26], LLaMA-family models [27], and recent instruction-tuned families such as Qwen and Mistral [28], [29]. Inference efficiency is strongly influenced by memory traffic from model weights and the KV cache, as well as attention implementations that reduce HBM (High Bandwidth Memory) accesses [9].

B. Prefill and Decode Phases

Decoder-only inference consists of two phases. During **prefill**, the model processes the entire input sequence to compute hidden states and populate the KV cache, typically exhibiting high arithmetic intensity. During **decode**, output tokens are generated sequentially, requiring repeated access to model weights and the growing KV cache, which often makes decode memory-dominated [4], [9], [19]. Serving systems optimize these phases through batching and KV-cache management, as implemented in high-throughput engines such as vLLM [4], [30].

C. GPU DVFS and Energy–Performance Tradeoffs

Dynamic Voltage and Frequency Scaling (DVFS) adjusts GPU operating frequency to trade energy for performance. Compute-bound kernels typically scale with frequency, while memory-bound kernels are limited by bandwidth and show weaker scaling. This distinction motivates phase-aware analysis for LLM inference. Prior work has explored energy–performance tradeoffs using DVFS, power limits, and configuration search [8], [31]–[33]. In this study, we focus on DVFS characterization for LLM inference and measure GPU power using NVML telemetry [34].

D. Workload Heterogeneity in LLM Inference

Benchmarking LLM inference requires representative workloads and standardized metrics. MLPerf Inference highlights the importance of end-to-end comparability across systems [7]. LLM workloads further exhibit substantial heterogeneity: query streams vary widely in semantic difficulty, and generation tasks increasingly shift execution toward decode-dominated regimes. We use established NLP benchmarks that capture complementary difficulty profiles, including BoolQ and HellaSwag (classification), TruthfulQA (misconception-prone factuality), and NarrativeQA (long-context comprehension) [16], [21]–[23].

III. Related Work

Prior work on efficient LLM inference spans benchmarking methodologies, system-level optimizations, adaptive inference, and energy-aware execution. The growing energy and environmental footprint of deep learning has motivated measurement-driven approaches and "Green AI" principles [5], [6], while benchmark suites such as MLPerf Inference emphasize standardized, end-to-end evaluation of inference performance across platforms and configurations [7]. Efficient LLM serving systems improve throughput and latency through batching and KV-cache management, as shown by vLLM and its PagedAttention mechanism [4], and through optimized attention kernels such as FlashAttention that reduce HBM traffic [9]. Other systems and toolchains explore optimized and distributed inference runtimes, including TensorRT-based stacks and large-scale distributed frameworks [19]. A complementary line of work reduces decode cost through speculative execution or multi-head decoding [35], [36], post-training quantization techniques such as SmoothQuant [37], or adaptive inference strategies including model cascades and cost-aware routing [38], [39]. Separately, DVFS and power-aware execution have been explored as knobs for improving energy efficiency in GPU-based ML workloads, using configuration search or power–performance tradeoff analysis [8], [31]–[33]. In contrast to these efforts, our work provides a measurement-driven, phase-aware DVFS characterization for LLM inference across model sizes and workloads, and explicitly connects hardware sensitivity to workload and query heterogeneity.

IV. Design and Methodology

This section describes the design and methodology of our study, and provides details about the hardware platform, models, datasets, and evaluation metrics used in our benchmarking study. All experiments use an offline, replay-based inference setup to ensure reproducibility.

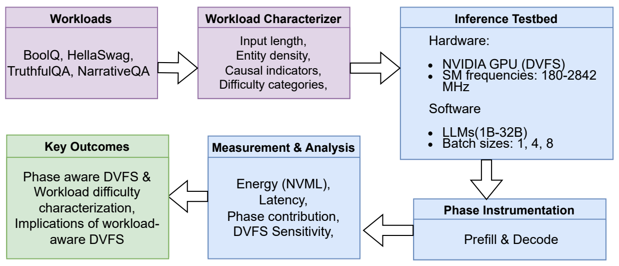
Workloads
BoolQ, HellaSwag, TruthfulQA, NarrativeQA
Workload Characterizer
Input length, Entity density, Causal indicators, Difficulty categories,
Inference Testbed
Hardware:
NVIDIA GPU (DVFS)
SM frequencies: 180-2842 MHz
Software
LLMs(1B-32B)
Batch sizes: 1, 4, 8
Phase Instrumentation
Prefill & Decode
Measurement & Analysis
Energy (NVML), Latency, Phase contribution, DVFS Sensitivity,
Key Outcomes
Phase aware DVFS & Workload difficulty characterization, Implications of workload-aware DVFS

Fig. 1. Study workflow.

A. Design

Figure 1 shows the benchmarking workflow. We conduct an offline, measurement-driven study that evaluates representative NLP workloads using a DVFS-controlled GPU testbed. Queries are first characterized using lightweight semantic features, after which inference is executed while varying model size, batch size, and SM frequency under a fixed decoding configuration. Inference is instrumented to separate prefill and decode phases, and GPU power and latency are recorded via NVML. These measurements are aggregated to derive phase-aware DVFS behavior, workload difficulty characterization, and implications of workload aware DVFS.

B. Testbed and Measurement Infrastructure

All measurements are performed on a single NVIDIA RTX PRO 6000 (Blackwell) GPU with 96 GB memory, which supports explicit dynamic voltage and frequency scaling (DVFS). The streaming multiprocessor (SM) clock is fixed via NVIDIA's management interface for the duration of each experiment. We evaluate seven SM frequency levels: 180, 487, 960, 1500, 2000, 2505, and 2842 MHz, while keeping GPU memory frequency at its default setting to isolate SM scaling effects. Unless stated otherwise, we use the maximum supported SM frequency (2842 MHz) as the baseline configuration for all comparisons. Power consumption is measured using NVIDIA Management Library (NVML) telemetry via `nvidia-smi`, sampled at 10 ms and integrated to compute per-request energy in joules. All timing measurements use `torch.cuda.synchronize()` to ensure GPU operations complete before recording timestamps. Each configuration is repeated three times, and we report mean values. Experiments follow an offline replay-based setup on an otherwise idle system, using batch sizes of 1, 4, and 8 to ensure controlled and repeatable energy and latency measurements.

C. Models

We evaluate five decoder-only large language models spanning 1B to 32B parameters, representing widely used open-source architectures (Table I). All models use publicly released pre-trained weights without fine-tuning. Inference is performed in FP16 precision with identical decoding configurations across models to ensure comparability.

TABLE I
MODELS AND DATASETS USED IN EVALUATION

Models			Datasets (Avg. Tokens)			
Model	**Params**	**Arch.**	**Dataset**	**Task**	**Input**	**Output**
Llama-3.2-1B	1B	Decoder-only	BoolQ	Class.	100	LL
Llama-3.2-3B	3B	Decoder-only	HellaSwag	Class.	166	LL
Llama-3.1-8B	8B	Decoder-only	NarrativeQA	Gen.	336	100
Qwen2.5-14B	14B	Decoder-only	TruthfulQA	Gen.	13	100
Qwen2.5-32B	32B	Decoder-only				

LL = log-likelihood evaluation (no token generation).

For generation tasks, all inference experiments use greedy decoding (`do_sample=False`, `temperature=0`), with a maximum of 100 generated tokens and early stopping on the end-of-sequence token. For classification tasks (BoolQ and HellaSwag), we use log-likelihood evaluation by comparing answer option probabilities without token generation, ensuring deterministic and comparable energy measurements across models.

D. Datasets and Metrics

We evaluate inference workloads using four widely used NLP benchmarks: BoolQ, HellaSwag, NarrativeQA, and the generative variant of TruthfulQA (TruthfulQA_GEN, referred to as TruthfulQA in the remainder of the paper). These datasets span both classification and generative tasks and exhibit substantial variation in input and output lengths. BoolQ and HellaSwag represent classification-style workloads with short outputs, while NarrativeQA and TruthfulQA require free-form generation with longer decode phases. Each dataset is evaluated across all five models using identical decoding configurations, unless explicitly stated otherwise. Table I summarizes key dataset characteristics.

We evaluate each inference configuration using metrics that capture both efficiency and output quality. **Energy** is measured as total GPU energy consumption in joules by integrating power over inference duration. **Latency** is measured as end-to-end inference time from input submission to completion of output generation. **Quality** is measured using accuracy for classification tasks and ROUGE-L for generative tasks. We additionally report the **Energy–Delay Product (EDP= Energy × Latency)** to capture the tradeoff between energy consumption and latency.

V. WORKLOAD CHARACTERIZATION

A. Motivation and Scope

Understanding input query characteristics is essential for designing energy-efficient LLM inference systems, as queries differ widely in their computational demands and response to model scaling. A key motivation of this work is that not all queries require the same model capacity: simple factual queries can often be handled effectively by smaller models, while complex reasoning queries benefit from larger models.

Building on the benchmark datasets described in Section IV-D, we focus on observable properties of input queries and generated outputs that influence inference behavior and

TABLE II
INPUT LENGTH STATISTICS (TOKENS)

Dataset	Mean	Std	Min	Max	Range
TruthfulQA	12.6	5.7	5	52	10.4×
BoolQ	102.9	46.0	24	294	12.2×
HellaSwag	163.8	56.0	49	265	5.4×
NarrativeQA	339.1	34.3	208	396	1.9×

energy consumption. We analyze lightweight structural, linguistic, and semantic features to distinguish easy and hard queries, validate that these features capture semantic difficulty beyond surface-level properties such as input length, and examine how query difficulty interacts with model scale. All features are extracted prior to inference and incur negligible runtime overhead. We do not claim to propose an optimal or deployable difficulty predictor; rather, we demonstrate that lightweight linguistic and semantic features provide stronger explanatory and predictive signals of inference difficulty than length-based heuristics.

B. Input Length and Structural Properties

Input query length varies substantially across datasets, reflecting different workload profiles. Table II summarizes token length statistics across 3,817 queries (1,000 per dataset, 817 for TruthfulQA).

Input length spans a 26.8× range across datasets, from short factual queries in TruthfulQA to long-context inputs in NarrativeQA. Short-context queries incur minimal prefill cost but rely on parametric knowledge, while long-context queries shift computation toward prefill and enable in-context retrieval. However, despite this large variation, input length alone does not capture differences in linguistic structure or reasoning complexity. We therefore analyze higher-level linguistic and semantic features that reflect query difficulty beyond token counts.

C. Linguistic and Semantic Complexity

We characterize each query using lexical, syntactic, and semantic features. We intentionally focus on lightweight, interpretable features that can be extracted online with negligible overhead, rather than learned difficulty predictors that require additional inference cost. To identify the most informative ones, we compute correlations between each feature and output quality across all models, retaining features with $|r| > 0.10$ and low redundancy (pairwise correlation < 0.7). This yields five key features capturing complementary aspects of query complexity(Table III):

- **Complexity Score** (0–1): Weighted combination of normalized token entropy, unique token ratio, entity density, and average sentence length.
- **Reasoning Complexity** (0–1): Density of causal and comparison markers (e.g., "because", "therefore", "however"), normalized by word count.
- **Entity Density** (0–1): Ratio of named entities to total tokens, computed using spaCy's `en_core_web_sm` NER model (PERSON, ORG, GPE, LOC types).

TABLE III
INPUT COMPLEXITY FEATURES BY DATASET (MEAN VALUES)

Feature	BoolQ	HellaSwag	TruthfulQA	NarrativeQA
Complexity Score	0.54	0.47	0.53	0.56
Reasoning Complexity	0.06	0.11	0.07	0.12
Entity Density	0.20	0.12	**0.34**	0.18
Token Entropy	5.82	6.31	3.50	**7.16**
Causal Questions (%)	2.4	4.4	10.2	**33.6**

TABLE IV
CAUSAL QUESTION DISTRIBUTION BY DATASET

Dataset	Causal Questions (%)	Dominant Query Type
BoolQ	2.4	Factual verification
HellaSwag	4.4	Sequence prediction
TruthfulQA	10.2	Factual and causal
NarrativeQA	33.6	Comprehension and causal

- **Token Entropy** (bits): Shannon entropy of the token distribution, $H = -\sum_i p_i \log_2 p_i$, where p_i is the relative frequency of token i
- **Causal Question Score** (0–100%): Percentage of causal question words ("why", "how", "explain", "justify", "prove") relative to total question count.

Across datasets, distinct semantic profiles emerge. TruthfulQA exhibits the highest entity density (0.34), reflecting reliance on parametric factual recall , while BoolQ shows substantially lower entity density (below 20%). NarrativeQA displays higher token entropy (typically exceeding 4.5 bits/token) and a larger fraction of causal questions (approximately 30–35%), indicating multi-step reasoning over provided context. In contrast, HellaSwag and NarrativeQA show higher reasoning complexity scores than BoolQ, consistent with their focus on commonsense inference and narrative continuation rather than binary factual verification.

NarrativeQA contains substantially more causal questions (33.6%), requiring multi-step reasoning across narrative context, while BoolQ primarily contains factual verification questions (2.4% causal). To assess whether these linguistic and semantic features provide information beyond input length, we next analyze their independence from basic structural properties and validate their explanatory power.

D. Feature Independence and Validation

A key question is whether the proposed workload features capture semantic difficulty beyond input length alone. We evaluate whether these features provide information independent of length, rather than acting as proxies for length.

1) Length Independence Analysis: Table V reports correlations between each feature and input length. While token entropy is strongly correlated with length ($r = +0.88$), most semantic features, including entity density, causal question score, and reasoning complexity, exhibit weak correlations with length. This indicates that these features capture properties beyond input size.

Notably, input length itself shows near-zero correlation with output quality ($r = 0.002$), as shown in Figure 2. This

TABLE V
FEATURE INDEPENDENCE FROM INPUT LENGTH

Feature	Corr. with Length	Independent?
Entity Density	$r = -0.44$	✓Yes
Causal Question Score	$r = +0.31$	✓Yes
Reasoning Complexity	$r = +0.19$	✓Yes
Token Entropy	$r = +0.88$	× No
Complexity Score	$r = +0.16$	✓Yes
Length → Quality	$r = 0.002$ (near zero)	

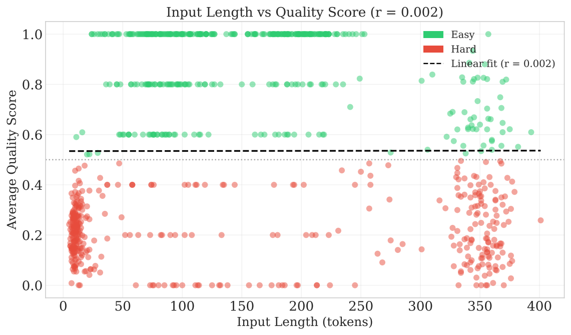

Fig. 2. Input length vs quality score. The near-zero correlation ($r = 0.002$) demonstrates that length alone cannot predict query difficulty. Easy/hard labels indicate whether normalized mean quality across models exceeds 0.5.

demonstrates that length alone is not a reliable indicator of query difficulty. In contrast, semantic features retain meaningful associations with quality even after controlling for length through partial correlation analysis, confirming that they capture semantic workload properties independently of input length.

2) Ablation Study: We define query difficulty using a binary label. Queries are labeled as *easy* if the normalized average quality score across all five models exceeds the dataset median, and *hard* otherwise. Quality scores are min-max normalized within each dataset prior to averaging, ensuring comparable scales between accuracy (classification tasks) and ROUGE-L (generation tasks). This yields a balanced split of 49% easy and 51% hard queries. We train a logistic regression classifier with L2 regularization (C=1.0) using 5-fold stratified cross-validation; all features are standardized to zero mean and unit variance before training. The baseline uses a simple length threshold of 150 tokens, which achieves near-random performance (51.1%). We include a simple length-only heuristic for comparison.

Table VI summarizes classification accuracy for different feature sets. Adding semantic features improves accuracy by 17.5 percentage points over the length-only baseline. Notably, semantic features alone achieve the same accuracy as the combined feature set, confirming that input length provides no additional predictive value beyond semantic workload characteristics.

E. Query Difficulty Categories

Based on the validated features, we develop a model-size-aware difficulty classification by analyzing per-query performance across five models (1B to 32B parameters).

TABLE VI
FEATURE ABLATION: DIFFICULTY CLASSIFICATION ACCURACY

Feature Set	Accuracy (5-fold CV)
Length only (>150 tokens)	51.1%
+ Entity density	66.6%
+ Causal question score	68.4%
Features only (no length)	**68.6%**

TABLE VII
QUALITY SCORES BY MODEL AND DATASET

Dataset	1B	3B	8B	14B	32B	Avg
BoolQ	0.685	0.785	0.855	0.785	0.815	0.785
HellaSwag	0.640	0.755	0.805	0.830	0.860	0.778
TruthfulQA	0.208	0.211	0.207	0.243	0.252	0.224
NarrativeQA	0.161	0.306	0.368	0.474	0.455	0.353
Model Avg	**0.423**	**0.514**	**0.559**	**0.583**	**0.596**	**0.535**

Note: Classification reports accuracy; generation reports ROUGE-L.

1) Baseline Quality by Dataset: Table VII shows that model scaling consistently improves quality, with average scores increasing from 0.423 (1B) to 0.596 (32B). Dataset difficulty varies substantially, with BoolQ and HellaSwag achieving higher accuracy than TruthfulQA and NarrativeQA, indicating different underlying difficulty characteristics.

2) Feature-Quality Correlations by Model Size: Table VIII presents correlation coefficients between input features and quality scores for each model size tier.

The analysis reveals two primary difficulty predictors. Entity density and causal question score emerge as the most consistent predictors of difficulty across all model sizes, exhibiting stable negative correlations with quality (typically $r \in [-0.18, -0.30]$ for entity density and $r \in [-0.15, -0.25]$ for causal questions). In contrast, other features show weaker or less consistent correlations (generally $|r| < 0.10$) and are excluded to maintain a lightweight difficulty model.

Feature Selection Rationale. Of the five candidate features analyzed, we retain only **entity density** and **causal question score** as primary difficulty predictors for classification. The other features are excluded for the following reasons:

- **Token Entropy**: Highly correlated with input length ($r = +0.88$, Table V), making it redundant with a surface-level property we have shown to be non-predictive of difficulty.
- **Reasoning Complexity**: Near-zero correlation with quality ($r = -0.03$), providing negligible predictive power despite theoretical relevance.
- **Complexity Score**: Weak correlation with quality ($r = -0.05$), insufficient to justify inclusion in a classifier.

Entity density and causal question score satisfy both criteria for effective difficulty prediction: they are independent from input length ($|r| < 0.5$) and show consistent negative correlations with quality across all model sizes.

3) Scaling Pattern Analysis: By analyzing performance trajectories across model sizes, we categorize queries into four scaling patterns. Quality scores are min-max normalized

TABLE VIII
FEATURE-QUALITY CORRELATIONS BY MODEL SIZE

Feature	1B	3B	8B	14B	32B
Entity Density	−0.20	−0.29	−0.32	−0.31	−0.32
Causal Question	−0.16	−0.17	−0.18	−0.13	−0.18
Token Entropy	+0.11	+0.22	+0.29	+0.33	+0.31

TABLE IX
QUERY SCALING PATTERNS ACROSS MODEL SIZES

Pattern	%	Mean Feature Profile
Always Easy	44.5	Entity=0.17, Causal=0.05, Entropy=6.13
Scaling Helps	15.5	Entity=0.18, Causal=0.13, Entropy=6.32
Always Hard	32.6	Entity=0.27, Causal=0.22, Entropy=4.94
Inconsistent	7.4	Architecture-dependent

within each dataset before classification, ensuring that accuracy (0–1) and ROUGE-L scores are on comparable scales. Table IX summarizes the resulting distribution.

- **44.5% of queries are easy for all models**: These can be safely routed to small models (1–3B) for maximum energy efficiency.
- **15.5% benefit from scaling**: These fail on small models but succeed on 8B+, representing optimal routing opportunities.
- **32.6% are hard for all models**: Even 32B models struggle; routing to larger models provides marginal benefit at significant energy cost.

4) Classification Validation: We validate the feature-based difficulty classification against empirical model performance. Queries are classified as *easy* if entity density < 0.20 and causal question score < 0.05, and *hard* otherwise. This yields 406 easy (50.8%) and 394 hard (49.2%) queries. Table X shows that all five models consistently achieve higher quality scores on easy queries, with an average gap of +0.256 (63% relative improvement), validating the feature-based classification.

F. Main Workload Characterization Insights

Our workload characterization yields the following insights for energy-efficient LLM inference:

1) **Input length is a weak difficulty predictor** ($r = 0.002$), while semantic features such as entity density and causal question presence provide stronger signals.
2) **Entity density is the dominant predictor** ($r = -0.29$), disproportionately challenging smaller models.
3) **44.5% of queries are easy across all model sizes**, enabling substantial energy savings via routing to smaller models.
4) **32.6% of queries are hard across all models**, where scaling yields diminishing quality returns.

These properties shape inference energy consumption by influencing model-scale sensitivity and the prefill–decode balance. In the next section, we shift focus from workload properties to hardware-level behavior and analyze how LLM inference energy and latency respond to GPU frequency scaling under these heterogeneous workloads.

TABLE X
CLASSIFICATION VALIDATION: QUALITY BY DIFFICULTY CATEGORY

Model	Easy	Hard	Gap	Valid?
Llama-3.2-1B	0.516	0.328	+0.188	✓
Llama-3.2-3B	0.637	0.388	+0.250	✓
Llama-3.1-8B	0.695	0.419	+0.276	✓
Qwen2.5-14B	0.713	0.450	+0.263	✓
Qwen2.5-32B	0.745	0.441	+0.304	✓
Average	**0.661**	**0.405**	**+0.256**	✓

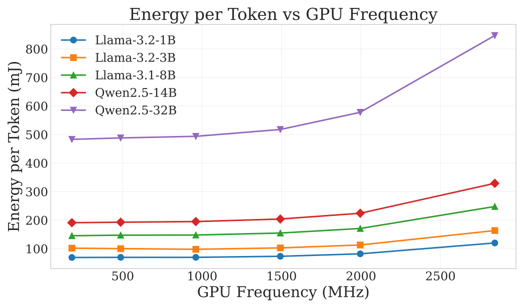

Fig. 3. Energy consumption per generated token across GPU frequencies. Lower frequencies achieve better energy efficiency (higher tokens per joule) due to the memory-bound nature of the decode phase.

VI. DVFS CHARACTERIZATION

This section characterizes the impact of GPU dynamic voltage and frequency scaling (DVFS) on LLM inference energy and performance. Our analysis reveals a fundamental phase-level asymmetry: the decode phase, which dominates inference time, is memory-bound and largely insensitive to GPU frequency, enabling substantial energy savings with minimal latency penalty. We quantify this behavior across five models (1B–32B parameters), four datasets, and seven discrete GPU SM frequency settings (180, 487, 960, 1500, 2000, 2505 and 2842 MHz), spanning the full supported operating range of the device.

A. Experimental Setup

This section builds on the experimental setup described in Section IV. We evaluate five decoder-only LLMs (1B–32B parameters) across four datasets, using fixed GPU SM frequencies spanning 180–2842 MHz. Energy and latency are measured at phase granularity (prefill and decode) under batch sizes of 1, 4, and 8. Unless stated otherwise, results are averaged over three runs with 1,000 queries per dataset.

B. Energy Reduction via DVFS

Figure 3 shows energy consumption per generated token as a function of GPU SM frequency. Lowering frequency monotonically reduces energy consumption. At minimum frequency (180 MHz), we observe **42% average energy savings** relative to baseline, ranging from 39.9% (Llama-3.2-3B) to 44.2% (Llama-3.2-1B at batch 8). Table XI shows results across

TABLE XI
DVFS Results at 180 MHz vs. Baseline (2842 MHz).

Model	B	E↓	LΔ	PreΔ	DecΔ	Pre%	Dec%
Llama-1B	1	42.3%	+5.6%	+52.4%	+0.7%	9.6%	90.4%
	4	42.4%	+5.3%	+34.5%	+1.2%	12.4%	87.6%
	8	44.2%	+3.5%	+21.7%	−0.3%	17.0%	83.0%
Llama-3B	1	39.9%	+4.2%	+37.4%	+0.8%	9.4%	90.6%
	4	42.3%	+3.3%	+16.1%	+1.0%	14.7%	85.3%
	8	43.5%	+1.6%	+8.5%	−0.1%	20.0%	80.0%
Llama-8B	1	42.2%	+2.5%	+23.3%	+0.1%	10.0%	90.0%
	4	42.4%	+0.8%	+7.6%	−0.5%	16.9%	83.1%
	8	42.9%	+0.3%	+3.4%	−0.7%	24.4%	75.6%
Qwen-14B	1	41.0%	+1.3%	+11.2%	−0.2%	13.8%	86.2%
	4	41.7%	+0.8%	+3.4%	+0.2%	17.6%	82.4%
	8	43.3%	+0.0%	+1.4%	−0.5%	25.9%	74.1%
Qwen-32B	1	44.0%	+0.3%	+4.4%	−0.2%	11.5%	88.5%
	4	43.0%	+0.3%	+1.7%	+0.0%	19.2%	80.8%
	8	44.1%	−0.1%	+0.5%	−0.3%	27.5%	72.5%
Avg B=1		**41.9%**	**+2.8%**	**+25.7%**	**+0.2%**	**10.9%**	**89.1%**
Avg B=4		**42.4%**	**+2.1%**	**+12.7%**	**+0.4%**	**16.2%**	**83.8%**
Avg B=8		**43.6%**	**+1.1%**	**+7.1%**	**−0.4%**	**23.0%**	**77.0%**

Notes: B = batch size; E↓ = energy reduction vs. baseline; LΔ = end-to-end latency change; PreΔ = prefill latency change; DecΔ = decode latency change; Pre%, Dec% = fraction of inference time.

three batch sizes. Energy savings are remarkably consistent: **41.9–43.6%** across all batch sizes. Latency increases are small: 2.8% at batch 1, decreasing to just 1.1% at batch 8. Critically, decode latency changes are **negligible** ($\pm 1\%$) across all configurations, confirming decode's memory-bound nature. Prefill slowdown decreases with batch size (25.7% $\rightarrow$ 7.1%) as larger batches amortize the compute-bound prefill overhead. Larger models show smaller latency penalties due to higher memory-boundedness. This favorable tradeoff provides substantial energy savings with minimal performance penalty across all batch sizes motivates our investigation into why LLM inference is so amenable to frequency scaling.

C. Why DVFS Works: Phase-Level Analysis

The key to understanding DVFS effectiveness lies in the distinct behavior of the two inference phases. Table XI reveals a striking asymmetry:

- **Prefill latency increases by 7–26%** at minimum frequency, with smaller increases at larger batch sizes, indicating compute-bound behavior.
- **Decode latency changes by <1%**, indicating memory-bound behavior largely insensitive to frequency.

This asymmetry arises from fundamental differences in how each phase utilizes hardware. Prefill processes all input tokens in parallel through dense matrix multiplications, achieving high arithmetic intensity that benefits from faster clocks. Decode generates tokens one at a time and requires repeated memory accesses to the KV cache and model weights. As a result, it is memory-bandwidth-limited and largely insensitive to compute frequency.

Table XI (rightmost columns) quantifies phase contribution: at batch size 1, decode accounts for **86–91% of total time**.

TABLE XII
Optimal EDP Frequency by Model and Batch Size (vs. 2842 MHz).

	B=1			B=4			B=8		
Model	**Freq**	**E↓**	**L**	**Freq**	**E↓**	**L**	**Freq**	**E↓**	**L**
Llama-1B	960	44.8%	−4.4%	960	42.3%	+1.2%	180	44.2%	+3.5%
Llama-3B	960	47.6%	−10.4%	487	42.7%	+2.7%	960	42.8%	+0.2%
Llama-8B	960	47.4%	−11.0%	180	42.4%	+0.8%	960	45.2%	−5.6%
Qwen-14B	960	51.5%	−19.0%	487	41.8%	+0.5%	180	43.3%	+0.0%
Qwen-32B	960	63.2%	−36.5%	487	43.4%	+0.2%	180	44.1%	−0.1%

Notes: Freq = SM frequency (MHz) minimizing EDP; E↓ = energy reduction vs. 2842 MHz; L = end-to-end latency change.

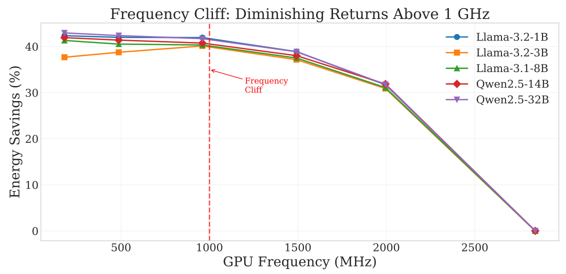

Fig. 4. The frequency cliff: energy savings plateau below ~1000 MHz. All models achieve 40–45% savings in the plateau region, with diminishing returns at lower frequencies. The optimal operating point lies at ~960 MHz where energy savings are maximized without significant latency penalty.

As batch size increases, prefill's share grows: at batch 8, prefill contributes 17–28% of time (vs 9–14% at batch 1). Decode throughput (tokens/second) remains constant across all frequencies, while power consumption increases significantly at maximum frequency. Higher frequencies during the decode phase increase energy consumption without providing measurable performance benefits.

D. The Frequency Sweet Spot

While minimum frequency maximizes energy savings, it incurs prefill slowdown. We identify the optimal operating point by analyzing the energy-delay product (EDP = Energy $\times$ Latency) across frequencies.

Table XII shows optimal EDP frequencies across batch sizes. At batch size 1, all models converge to approximately **960 MHz** (34% of maximum), delivering 43–49% energy savings with latency improvements. At larger batch sizes, optimal frequencies vary more across models, with some achieving even greater savings. Figure 4 visualizes the frequency cliff phenomenon across all evaluated models. The sweet spot exists because:

1) Energy savings plateau below ~1000 MHz (the "frequency cliff"), yielding diminishing returns at lower frequencies.
2) Prefill slowdown accelerates below 1000 MHz, increasing latency disproportionately.

E. Effect of Output Length and Model Size

DVFS effectiveness depends on output length and model size, which determine decode's contribution to total latency.

TABLE XIII
DVFS Effectiveness by Output Length and Model Size (180 MHz, B=1).

Dataset	E↓	L↑	Model	E↓	L↑
BoolQ (LL)	41.4%	+7.7%	Small (1–3B)	40.9%	+4.8%
HellaSwag (LL)	42.6%	+4.1%	Medium (8B)	42.2%	+2.5%
TruthfulQA (100)	42.5%	+0.7%	Large (14–32B)	43.2%	+0.6%
NarrativeQA (100)	43.1%	+0.9%			

LL = log-likelihood evaluation (no token generation).

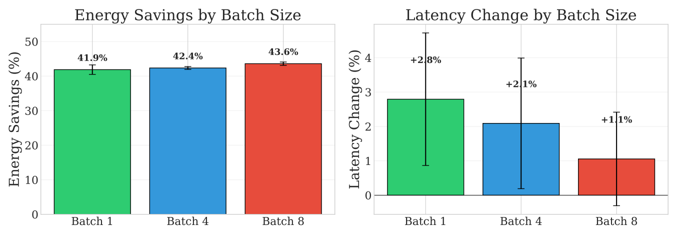

Fig. 5. Effect of batch size on DVFS effectiveness. Energy savings remain consistent (42–44%) across all batch sizes. Latency penalties decrease with larger batches as prefill overhead is amortized over more tokens.

Table XIII shows these effects. For short-output datasets (BoolQ, HellaSwag), prefill accounts for 31–39% of time, resulting in latency increases of 4–8%. For long-output datasets (TruthfulQA, NarrativeQA), prefill drops to 3–12%, and DVFS achieves **42–43% energy savings with under 1% latency penalty**. Model size affects latency impact: **small models (1–3B)** show 2–5% latency increase, **medium models (8B)** achieve 0.3–2.5% increase, and **large models (14–32B)** show negligible impact (<1%). All categories remain memory-bound with consistent 41–44% energy savings.

F. Effect of Batch Size

Contrary to initial expectations, batch size has minimal impact on DVFS effectiveness. As shown in Table XI, energy savings remain consistent at **41.9–43.6%** across batch sizes 1, 4, and 8 (Figure 5). Latency impact decreases with larger batches, from +2.8% at batch 1 to +1.1% at batch 8, as prefill overhead is amortized. This robustness arises because the decode phase remains memory-bound even at larger batch sizes. While prefill becomes more compute-intensive with batching, the decode phase that dominates total time remains bottlenecked by memory bandwidth for loading model weights and the KV cache. Overall, DVFS provides **consistent 42% energy savings** with modest latency impact (1–3%) across batch sizes, making low-frequency operation suitable for offline and latency-tolerant scenarios.

G. Summary and Recommendations

Table XIV consolidates our key findings. The central insight is that **LLM inference is dominated by memory-bound decode operations that are largely insensitive to GPU frequency**, enabling **42% energy savings with only 1–3% latency increase**. While aggressive DVFS (down to 180 MHz) proved effective in our controlled setting, these results motivate *phase-aware* GPU frequency policies rather than a universal operating point.

TABLE XIV
Summary of phase-level DVFS effects on energy and latency.

Aspect	Observation
DVFS at 180 MHz (vs. 2842 MHz)	
Energy savings	40–44% (avg. 42%)
Latency change	+1–3%
Prefill / Decode slowdown	+7–26% / ±1%
Decode time fraction	77–91%
Effect of Batch Size (B=1/4/8)	
Energy savings	41.9 / 42.4 / 43.6%
Latency impact	+2.8 / +2.1 / +1.1%
Operating Point Tradeoffs	
Max savings (180 MHz)	42% savings, +1–3% latency
EDP-optimal (∼960 MHz)	45–50% savings, −5–10% latency
Baseline (2842 MHz)	Reference point

TABLE XV
Routing Strategy Based on Scaling Patterns.

Pattern	%	Model	Rationale
Always Easy	44.5	1–3B	Similar quality across sizes
Scaling Helps	15.5	8B+	Quality improves with scale
Always Hard	32.6	1–3B	Limited benefit from scaling
Inconsistent	7.4	8B	Architecture-dependent

VII. Case Study: Implications of Workload-Aware DVFS

To illustrate the practical implications of our workload characterization and DVFS findings, we present a case study exploring how query difficulty features could inform energy optimization decisions. This analysis examines the potential synergy between model routing and frequency scaling and provides empirical estimates of achievable energy savings.

A. Mapping Query Difficulty to Model Scale

Our scaling pattern analysis (Section V-E) revealed that query difficulty varies systematically with input features. We explore how these patterns might inform model selection decisions. The four scaling patterns identified in our workload characterization suggest a natural mapping to model tiers. Queries classified as “Always Easy” (44.5% of our dataset) achieved comparable quality scores across all model sizes, suggesting they could potentially be handled by smaller models. Conversely, “Scaling Helps” queries (15.5%) showed measurable quality improvements with larger models, while “Always Hard” queries (32.6%) exhibited limited benefit from increased model capacity. Table XV summarizes how these patterns could map to routing decisions.

The energy consumption measurements from Section VI provide context for these routing decisions. At baseline frequency (2842 MHz), average energy per query ranged from 2.9 J for the 1B model to 21.0 J for the 32B model, a 7.2× difference. If easy queries were routed to 3B models instead of 32B, this would represent an 80% energy reduction per query for that subset of the workload.

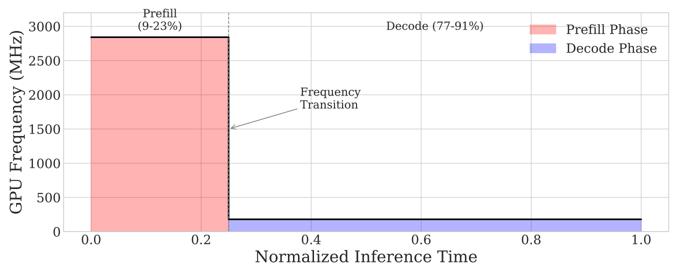

Fig. 6. Phase-aware frequency profile during inference. The transition occurs after prefill completion, exploiting the memory-bound nature of decode.

TABLE XVI
DVFS Energy Savings by Model (2842 MHz → 180 MHz)

Model	Baseline (J)	Low Freq (J)	Savings	Latency
Llama-3.2-1B	2.92	1.69	42.3%	+5.6%
Llama-3.2-3B	4.19	2.52	39.9%	+4.2%
Llama-3.1-8B	6.24	3.61	42.2%	+2.5%
Qwen2.5-14B	8.32	4.91	41.0%	+1.3%
Qwen2.5-32B	20.97	11.74	44.0%	+0.3%
Average			**41.9%**	**+2.8%**

B. Phase-Aware Frequency Scaling

The distinct behavior of prefill and decode phases (Section VI) motivates phase-aware frequency scaling. Since decode dominates inference time (77–91%) and is largely memory-bound, we consider a policy that uses high frequency during prefill (2842 MHz) and switches to low frequency during decode (180 MHz). Across five models, this reduces energy consumption by 41.9% on average with a 2.8% latency increase (Table XVI). Larger models incur smaller penalties (0.3% for 32B vs. 5.6% for 1B), and long-output workloads achieve 43–44% savings with under 1% latency impact. Results at 180 MHz represent an upper bound; in practice, EDP-optimal frequencies as shown in Table XII may be preferred.

C. Combined Optimization Potential

Combining model routing with phase-aware frequency scaling could yield multiplicative energy reductions. We estimate the potential savings by applying both techniques to our workload distribution. For each query category, we compute expected savings by combining the model routing benefit (relative to always using the 32B model at 2842 MHz) with the DVFS benefit for the target model tier. Table XVII presents these projections. These projections suggest that workload-aware optimization could reduce energy consumption by approximately 87% compared to a baseline of always using the largest model at maximum frequency.

1) Quality Considerations: An important consideration is the quality impact of routing queries to smaller models. Our validation analysis found that overall classification accuracy (averaged across BoolQ and HellaSwag) was 77.0% on the 3B model versus 83.8% on the 32B model, a gap of 6.8 percentage points. Whether this tradeoff is acceptable depends on application requirements. Table XVIII positions different optimization strategies on the energy-quality frontier. The combined approach achieves the highest energy savings but incurs a modest quality reduction concentrated in the subset of queries routed to smaller models. Figure 7 visualizes these tradeoffs, showing that DVFS-only optimization preserves quality while routing introduces a quality-energy tradeoff.

TABLE XVII
Estimated Combined Energy Savings

Category	%	Model	Freq	Est. Savings
Always Easy	44.5%	3B	180 MHz	88%
Scaling Helps	15.5%	14B	180 MHz	77%
Always Hard	32.6%	3B	180 MHz	88%
Inconsistent	7.4%	8B	180 MHz	83%
Weighted Average				**87%**

TABLE XVIII
Energy-Quality Tradeoff Across Strategies

Strategy	Energy	Quality	Est. Savings
Baseline (32B, 2842 MHz)	20.97 J	83.8%	—
DVFS only (32B, 180 MHz)	11.74 J	83.8%	44%
Routing only (3B, 2842 MHz)	4.19 J	77.0%	80%
Combined (3B, 180 MHz)	2.52 J	77.0%	88%

D. Main Insights from the Case Study

This case study illustrates how workload characterization findings could inform practical energy optimization. Three key observations emerge:

1) **Model routing potential:** The finding that 44.5% of queries achieve similar quality across model sizes suggests significant routing opportunities. Routing to 3B instead of 32B yields 80% energy savings with a 6.8 percentage point quality reduction.
2) **DVFS viability:** Reducing GPU frequency from 2842 MHz to 180 MHz achieves 42% average energy savings with 3% latency increase, preserving quality. Larger models incur smaller latency penalties due to their memory-bound nature.
3) **Combined potential:** Workload-aware optimization combining both techniques could achieve up to 87% energy reduction, though quality implications warrant application-specific evaluation.

Threats to validity. Our evaluation uses offline replay-based benchmarking on a single GPU and does not capture production serving dynamics such as continuous batching, speculative decoding, or multi-GPU execution. Energy measurements reflect GPU consumption only and exclude system-level overheads. Extending this analysis to distributed inference, heterogeneous hardware, and non-English workloads is an important direction for future work. Moreover, these findings are intended as empirical observations rather than deployment prescriptions. Real-world implementation would require additional considerations, including classification accuracy, switching overhead, and service-level requirements.

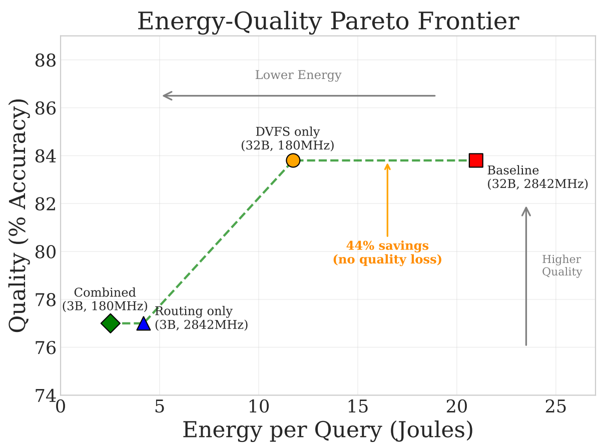

Fig. 7. Energy-quality Pareto frontier. DVFS provides "free" energy savings; routing introduces quality tradeoffs for additional gains.

VIII. Conclusion

We present a measurement-driven characterization of LLM inference energy efficiency across workload and hardware dimensions. We showed that semantic query features predict inference difficulty better than input length, with 44.5% of queries achieving comparable quality across model sizes, and that the decode phase dominates inference time (77–91%) while remaining largely insensitive to GPU frequency. This enables up to 42% energy savings at 180 MHz with only 1–3% latency increase. These findings challenge the assumptions that inference cost scales primarily with token count or requires a uniform GPU frequency. A case study further illustrated that combining workload-aware model selection with DVFS can reduce energy consumption by up to 87%, providing an empirical foundation for phase-aware and workload-aware optimization in future LLM serving systems.

Future work will explore phase-aware runtime DVFS control, lightweight query classification for model routing, and extensions to multi-GPU inference and emerging accelerator architectures.

References

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.

[3] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[4] W. Kwon, Z. Li, S. Zhuang *et al.*, "Efficient memory management for large language model serving with pagedattention," *arXiv preprint arXiv:2309.06180*, 2023.

[5] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *ACL*, 2019.

[6] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[7] V. J. Reddi, C. Cheng, D. Kanter *et al.*, "Mlperf inference benchmark," *arXiv preprint arXiv:1911.02549*, 2019.

[8] M. Revel *et al.*, "Optimml: Joint control of inference latency and server power consumption," *ACM Transactions on Computer Systems*, 2024.

[9] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *arXiv preprint arXiv:2205.14135*, 2022.

[10] Z. Li *et al.*, "Alpaserve: Statistical multiplexing with model parallelism for efficient llm serving," in *USENIX OSDI*, 2023.

[11] A. Talmor *et al.*, "Commonsenseqa: A question answering challenge targeting commonsense knowledge," in *NAACL*, 2019.

[12] M. Geva *et al.*, "Did aristotle use a laptop? a question answering benchmark with implicit reasoning," in *ACL*, 2021.

[13] K. Valmeekam *et al.*, "Planbench: An extensible benchmark for evaluating planning and reasoning," in *NeurIPS*, 2022.

[14] Y. Bai *et al.*, "Longbench: A benchmark for long-context language models," *arXiv preprint arXiv:2308.14508*, 2023.

[15] M. Zhong *et al.*, "Qmsum: A new benchmark for query-based multi-domain meeting summarization," in *NAACL*, 2021.

[16] T. Kočiský, J. Schwarz, P. Blunsom *et al.*, "The narrativeqa reading comprehension challenge," in *Transactions of the Association for Computational Linguistics (TACL)*, 2018.

[17] K. Ethayarajh *et al.*, "Understanding dataset difficulty with v-information," in *ICLR*, 2022.

[18] F. Liu *et al.*, "What makes prompts hard?" *arXiv preprint arXiv:2310.01462*, 2023.

[19] M. Shoeybi, M. Patwary, R. Puri *et al.*, "Megatron-lm: Training multi-billion parameter language models using model parallelism," in *NeurIPS*, 2019.

[20] P. Patel, E. Choukse, C. Zhang, A. Shah, Goiri, S. Maleki, and R. Bianchini, "Splitwise: Efficient generative llm inference using phase splitting," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 118–132.

[21] C. Clark, K. Lee, M.-W. Chang *et al.*, "Boolq: Exploring the surprising difficulty of natural yes/no questions," in *NAACL*, 2019.

[22] R. Zellers, A. Holtzman, Y. Bisk *et al.*, "Hellaswag: Can a machine really finish your sentence?" in *ACL*, 2019.

[23] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," *arXiv preprint arXiv:2109.07958*, 2022.

[24] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *NeurIPS*, 2017.

[25] A. Radford, J. Wu, R. Child *et al.*, "Language models are unsupervised multitask learners," *OpenAI Technical Report*, 2019.

[26] S. Zhang, S. Roller, N. Goyal *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[27] H. Touvron, T. Lavril, G. Izacard *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[28] Qwen Team, "Qwen2.5 technical report," *arXiv preprint*, 2024, model family technical report.

[29] A. Q. Jiang, A. Sablayrolles, A. Roux *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[30] L. Zheng, L. Jin, Z. Li *et al.*, "Orca: A distributed serving system for transformer-based generative models," in *OSDI*, 2022.

[31] J. You, J. W. Chung *et al.*, "Zeus: Understanding and optimizing gpu energy consumption of dnn training," in *NSDI*, 2023.

[32] J. W. Chung *et al.*, "Reducing energy bloat in large model training," in *SOSP*, 2024.

[33] S. M. Nabavinejad *et al.*, "Batchsizer: Power-performance trade-off for dnn inference," *arXiv preprint*, 2021.

[34] NVIDIA, "Nvidia management library (nvml) api reference," 2025, gPU deployment & management documentation.

[35] Y. Levine *et al.*, "Accelerating llm inference with staged speculative decoding," *arXiv preprint arXiv:2308.04623*, 2023.

[36] H. Cai, Z. Li *et al.*, "Medusa: Simple llm inference acceleration framework with multiple decoding heads," *arXiv preprint arXiv:2401.10774*, 2024.

[37] G. Xiao, J. Lin, M. Seznec, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *ICML*, 2023, often cited as arXiv:2211.10438.

[38] L. Chen, M. Zaharia, and J. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," *arXiv preprint arXiv:2305.05176*, 2023.

[39] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*, 2016.