Energy-Efficient GPU Frequency Scaling Characterization for SLM Fine-Tuning on Embedded Platforms

by

Aidar Amangeldi

Submitted to the Department of Data Science
in partial fulfillment of the requirements for the degree of

Master of Science in Data Science

at the

NAZARBAYEV UNIVERSITY

June 2026

Author .....
Department of Data Science
27.04.2026

Certified by .....
Jurn-Gyu Park
Assistant Professor
Thesis Supervisor

Accepted by .....
Elizaveta Arkhangelsky
Dean, School of Engineering and Digital Sciences

Energy-Efficient GPU Frequency Scaling Characterization for SLM Fine-Tuning on Embedded Platforms

by

Aidar Amangeldi

Submitted to the Department of Data Science
on 27.04.2026, in partial fulfillment of the
requirements for the degree of
Master of Science in Data Science

Abstract

While embedded GPU dynamic voltage and frequency scaling (DVFS) is well-studied for inference workloads, fine-tuning exhibits different memory access patterns and runs 100–1000× longer, making inference-derived policies inappropriate. We present the first per-frequency characterization of transformer fine-tuning across three model scales (BERT-tiny 14M, BERT-base 110M, DeBERTa-xlarge 900M) on the NVIDIA Jetson AGX Orin, sweeping GPU frequencies from 306 to 1300 MHz on SST-2 and QNLI benchmarks. Across 77 experiments, optimal frequencies fall consistently in the 612–1020 MHz range, with production-scale models achieving 22–32% energy savings over the default governor. We develop a GPU-utilization-guided frequency selection algorithm requiring only 30 profiling steps that achieves a 1.5% average gap from the true optimum across 13 validation workloads, versus 21% energy waste for the default governor.

Thesis Supervisor: Jurn-Gyu Park
Title: Assistant Professor

Acknowledgments

I thank my advisor Professor Jurn Gyu Park for his guidance and feedback throughout this work, and my committee members and readers for their time and constructive comments.

I am grateful to my colleagues and collaborators for helpful discussions and technical assistance. I thank ISSAI for access to its NVIDIA Jetson Orin platform and its members for their support during the experimental work.

Finally, I thank my family, friends, and close people for their encouragement and patience. Their support and belief in me made it possible to stay focused through the hardest parts of this work. I am grateful for their presence and for being there when it mattered.

Contents

1 Introduction **13**

2 Motivation, Background and Related Work **15**

2.1 Motivation 15

2.2 Background 16

2.3 Related Work 19

3 Methodology **23**

3.1 Experimental Platform 24

3.2 Models and Datasets 24

3.3 DVFS Characterization Protocol 25

3.4 Metrics and Validity 26

4 Results Analysis for Characterization **27**

4.1 BERT-tiny Results (14M Parameters) 27

4.2 BERT-base Results (110M Parameters) 29

4.3 DeBERTa-xlarge Results (900M Parameters) 32

4.4 Summary of Fine-Tuning Time and Energy–Time Trade-off 34

4.5 Cross-Model Patterns and Summary 36

5 DVFS Governor Policy and Validation **41**

5.1 Empirical Basis based on key fidnings 41

5.2 Algorithm Description 41

5.3 Algorithm Validation 44

6 Discussion, Future Work, and Conclusion **47**

6.1 Discussion and Future Work 47

6.2 Conclusion 48

List of Figures

2-1 Motivating Example: The optimal energy consumption occurs at 714 MHz (green star) in the Full-FT method on BERT-base energy profiles across 10 GPU frequencies on Jetson AGX, 64GB. (Horizontal dashed lines denote the unconstrained maximum-performance default (purple) and the 30W power-cap default (orange); upright dotted line indicates energy consumptions in each frequency). 16

3-1 Methodology overview: Four-stage pipeline for DVFS characterization and algorithm validation. (1) Workload selection: 3 models (14M, 110M, 900M) × 2 tasks (SST-2, QNLI) = 6 combinations. (2) Execution: 11-point frequency sweep (306–1300 MHz) + 2 baselines on Jetson Orin = 77 experiments. (3) Characterization: Energy-frequency curves and GPU utilization clustering. (4) Algorithm: Profile at 714 MHz, branch on GPU utilization, validate on 13 workloads (1.5% gap from optimal). 23

4-1 Energy consumption versus GPU frequency for BERT-tiny (left), BERT-base (center), and DeBERTa-xlarge (right) on SST-2 (top) and QNLI (bottom). Mode0 and Mode2 baselines shown as horizontal dashed lines; optimal points marked with stars. 37

4-2 Energy savings of manual optimization versus Mode0 and Mode2 baselines. 38

List of Tables

3.1 Experimental Platform Configuration 24

3.2 Model Specifications and Training Configuration 25

4.1 BERT-tiny: Results on SST-2 and QNLI 28

4.2 BERT-base: Results on SST-2 and QNLI 31

4.3 DeBERTa-xlarge: Results on SST-2 and QNLI 33

4.4 Energy saving and training-time change at the optimal frequency relative to Mode0 and Mode2. Negative time values indicate the optimal is faster than the baseline. †Mode2 SST-2 excluded for DeBERTa-xlarge (batch-size incomparability). 36

4.5 Summary: Optimal Frequencies and Energy Savings vs Baselines . . . 36

5.1 Profiling Overhead: 30 Steps at 714 MHz 44

5.2 Algorithm Validation table 45

Chapter 1

Introduction

Edge AI applications increasingly require on-device machine learning, where pre-trained transformer models are adapted to local data directly on embedded hardware. Use cases ranging from clinical NLP in hospitals to real-time language processing in autonomous systems require fine-tuning on the device itself to preserve data privacy, reduce latency, and enable operation without cloud connectivity. Embedded GPU platforms such as the NVIDIA Jetson family are now the primary targets for these workloads, supporting models from compact 14-million-parameter architectures to production-scale systems approaching one billion parameters.

However, fine-tuning transformers on embedded platforms is energy-intensive. A single BERT-base training run on the Jetson AGX Orin consumes 25–35 kJ, while larger models such as DeBERTa-xlarge exceed 500 kJ, making energy a practical constraint for battery-powered or thermally-constrained deployments. Dynamic Voltage and Frequency Scaling (DVFS) is a hardware technique that adjusts the GPU frequency to balance between performance and power, provides a direct route to decreasing this energy cost. However, the default power settings on embedded systems use fixed, workload-independent frequency settings and no systematic analysis has been conducted regarding the impact of the frequency of the GPU on the energy efficiency of transformer fine-tuning on embedded systems.

Existing DVFS research targets datacenter GPUs and inference workloads, neither of which transfers to the embedded fine-tuning setting. Datacenter frameworks

depend on power capping APIs and multi-GPU pipeline parallelism unavailable on embedded SoCs. Inference-oriented DVFS policies are designed for millisecond-scale execution with fixed computational graphs, whereas fine-tuning runs span multiple hours and involve backpropagation, gradient accumulation, and dynamic memory access patterns. Model-level efficiency techniques such as pruning and parameter-efficient fine-tuning reduce computational requirements, but they operate at the algorithmic level, orthogonal to hardware frequency selection. This leaves a gap: no prior work characterizes GPU DVFS for transformer fine-tuning across model scales on power-constrained embedded platforms.

This thesis addresses that gap through two contributions. First, we present the first systematic per-frequency characterization of transformer fine-tuning energy consumption on the NVIDIA Jetson AGX Orin. Through 77 controlled experiments spanning three model scales (BERT-tiny 14M, BERT-base 110M, DeBERTa-xlarge 900M), two GLUE benchmark tasks (SST-2, QNLI), and 11 GPU frequencies (306–1300MHz), we establish that energy-optimal frequencies lie consistently in the 612–1020MHz range (50–60% below the hardware maximum), yielding 22–32% energy savings over the default 30W mode without accuracy degradation. Second, grounded in the finding that GPU utilization at a reference frequency separates workloads into predictable groups, we develop a lightweight utilization-guided frequency selection algorithm. Validation across 13 workloads shows a 1.5% average energy gap from the true optimal, versus 21% for the default platform governor.

The remainder of this thesis is organized as follows. Chapter 2 presents a motivational example and reviews background and related work on GPU DVFS, energy-aware machine learning, and embedded platform benchmarking. Chapter 3 describes the experimental setup, DVFS characterization protocol, and measurement methodology. Chapter 4 presents per-model results and cross-model analysis addressing our research questions. Chapter 5 introduces the frequency selection algorithm and reports validation results. Chapter 6 discusses limitations and future work, then summarizes the findings and contributions.

Chapter 2

Motivation, Background and Related Work

2.1 Motivation

To illustrate energy opportunities as a motivating example, Figure 2-1 shows full fine-tuning of BERT-base (110M parameters) on the SST-2 sentiment classification task across 11 GPU frequencies from 306 to 1300MHz. All frequency configurations reach identical validation accuracy (92.5%), while training time and power draw vary with frequency. The energy-optimal configuration occurs at 714MHz (green star), consuming 25.2 kJ—25.2% below the platform's default 30W power-limited mode (Mode2, orange line) and 22.9% below the unconstrained maximum-performance mode (Mode0, purple line).

Energy consumption follows a U-shape: it rises at low frequencies due to extended training time and at high frequencies due to elevated power draw. The minimum at 714MHz yields 22.9% energy savings over Mode0 (the unconstrained maximum-performance baseline with dynamic frequency scaling up to 1300MHz) at no accuracy cost. Default high-frequency configurations therefore waste 22–25% of energy that frequency selection alone can recover.

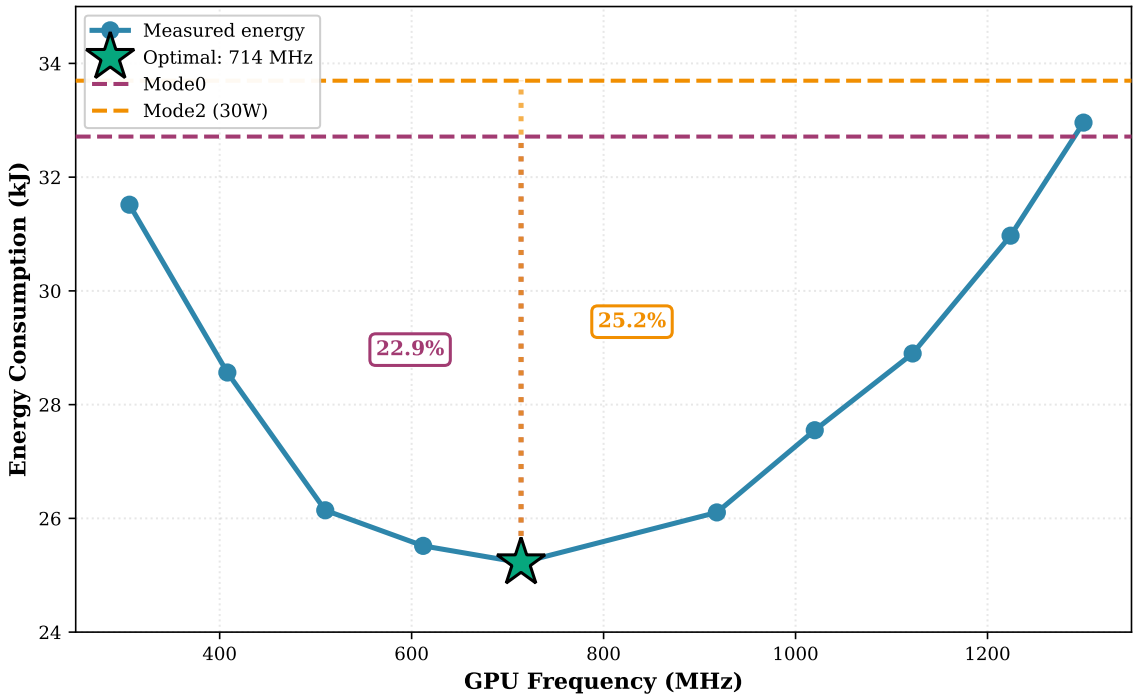

Figure 2-1: Motivating Example: The optimal energy consumption occurs at 714 MHz (green star) in the Full-FT method on BERT-base energy profiles across 10 GPU frequencies on Jetson AGX, 64GB. (Horizontal dashed lines denote the unconstrained maximum-performance default (purple) and the 30W power-cap default (orange); upright dotted line indicates energy consumptions in each frequency).

Our research questions are as follows:

- **Parameter Size**: *RQ1) Does the energy-optimal frequency generalize across transformer models of different parameter sizes (14M, 110M, and 900M)?*
- **Sequence Length**: *RQ2) Does the sequence length (128 tokens for SST-2 versus 256 tokens for QNLI)-shift the optimal frequency?*
- **Algorithm**: *RQ3) What opportunities exist for DVFS design (or preliminary results) based on our characterization findings?*

2.2 Background

Edge AI deployments increasingly require on-device transformer fine-tuning for privacy, latency, and offline operation. Transformer models have

reshaped natural language processing through pre-training and fine-tuning, enabling task adaptation with minimal labeled data [7, 8]. As edge AI deployments grow, on-device fine-tuning has become a key requirement for adapting pre-trained models to local contexts while satisfying privacy constraints, reducing inference latency, and supporting offline operation [4, 27]. In healthcare AI, for instance, fine-tuning clinical NLP models on hospital-specific records enables personalized medical language understanding without transmitting sensitive data to cloud servers. Similar requirements arise in finance, manufacturing, and autonomous systems where data locality and real-time adaptation matter most. This shift toward edge-based fine-tuning introduces a hard constraint: energy efficiency on battery-powered and thermally-limited embedded platforms.

Fine-tuning transformers on embedded platforms consumes substantial energy, yet DVFS guidance for training workloads remains absent. Fine-tuning transformers on embedded platforms consumes substantial energy: a BERT-base model training for 5 epochs requires 25–35kJ on the NVIDIA Jetson AGX Orin, while larger models such as DeBERTa-xlarge (900M parameters) can exceed 500kJ per run. This cost is multiplicative with the repeated fine-tuning steps typical of production deployments (hyperparameter tuning, continual learning, multi-task adaptation), posing a challenge to sustainable edge AI. Dynamic voltage and frequency scaling (DVFS) provides a single lever: a trade-off between computational throughput and lower power. But its use in fine-tuning of transformers is not well understood. Current definitions of DVFS focus on inference workloads [23], CPU-side policies [10], or generic compute kernels, but not training workloads. There are differences that are important to fine-tuning: runs take hours, not milliseconds, backward propagation generates bursty memory access, and accuracy needs to be maintained across epochs. Such differences nullify inference-based DVFS policies and must be characterized by training.

Prior GPU DVFS research targets inference workloads on server platforms, leaving embedded training uncharacterized. Existing GPU DVFS research in server environments such as [10, 36] were able to achieve 15–25% energy

savings for inference through a so called adaptive frequency selection approaches [23]. These studies target datacenter GPUs with different power envelopes and memory hierarchies than embedded platforms, focus on inference-time optimization under SLO constraints, or characterize single model configurations without examining scale-dependent behavior. For embedded platforms specifically, existing work covers inference throughput on vision workloads [2, 22] or provides generic DVFS guidelines for compute-intensive kernels [43]. To our knowledge, no prior study has systematically characterized GPU DVFS for transformer fine-tuning across model scales on power-constrained embedded platforms, compared practical baseline configurations, or identified optimal frequency regimes for production deployment. This gap matters because embedded deployments increasingly require local fine-tuning, yet practitioners have no evidence-based guidance for GPU frequency selection.

We systematically characterize GPU DVFS for transformer fine-tuning across model scales on embedded platforms. This paper addresses that gap through systematic GPU frequency characterization for transformer fine-tuning on embedded platforms. We run controlled experiments across three transformer architectures spanning two orders of magnitude in parameter count (BERT-tiny 14M, BERT-base 110M, DeBERTa-xlarge 900M) and two GLUE tasks (SST-2, QNLI) on the NVIDIA Jetson AGX Orin, sweeping 11 discrete GPU frequencies (306–1300MHz) and comparing against practical baseline modes (Mode0 unconstrained, Mode2 30W default). Across 69 experiments, energy-optimal frequencies fall consistently in the 612–1020MHz range (50–60% below the hardware maximum), with production-scale models achieving 22–32% energy savings over the 30W default at no accuracy cost. We also find that the energy-optimal frequency is independent of model size for short sequences and that the dynamic governor matches manual tuning in several cases, challenging the assumption that static frequency selection always wins. These results give practitioners a concrete, evidence-based starting point for energy-efficient transformer deployment on embedded hardware.

Our paper makes the following main contributions:

- Design and implement systematic small language model (SLM) workloads char-

acterization of GPU frequency scaling for on-device fine-tuning using an embedded platform.

- Propose opportunities for GPU DVFS Design based on characterization key findings, identifying a potentially valuable frequency selection algoithm.

- Lastly, we demonstrate 22–32% energy savings against the default 30W mode without accuracy degradation. Reproducible codes and data are here [1].

2.3 Related Work

GPU DVFS for General Computing. DVFS has been extensively studied for general-purpose computing, where adaptive frequency control improves energy proportionality under varying workloads [21, 37]. Mei et al. explored GPU DVFS characterizations and showed energy-saving potential across Fermi and Maxwell architectures [19]. Tang et al. found that optimal core frequencies conserve 8.7–23.1% energy for CNN training across Pascal, Volta, and Turing GPUs, identifying valley-shaped energy curves at mid-range frequencies [33]. Our work extends these findings to transformer fine-tuning on embedded systems, where shared CPU–GPU memory, thermal throttling, and the absence of per-GPU power capping APIs create a different optimization landscape than the server-class GPUs studied by Tang et al. (P100, V100, GTX 2080Ti). Mendes et al. [20] explored GPU voltage underscaling alongside frequency scaling for CNN training, achieving up to 38% energy savings, showing that DVFS dimensions beyond frequency alone can yield further gains on datacenter hardware. Karzhaubayeva et al. [11] characterized CNN inference workloads on NVIDIA Jetson TX2 and proposed integrated CPU–GPU DVFS governor policies using utilization-based frequency-pair selection, achieving up to 16.7% EDP improvement with less than 2% performance degradation on YOLO object detection applications. Recent work has extended GPU DVFS to transformer and LLM workloads on datacenter hardware. Maliakel et al. [17] performed a systematic GPU

[1] Project repository: `https://github.com/aidarjpg/llm-dvfs`

frequency sweep across five decoder-only LLMs (1B–32B parameters) and found that the decode phase dominates inference time (77–91%) and is largely insensitive to GPU frequency, enabling up to 42% energy savings with minimal latency impact. Liu et al. [15] proposed GreenLLM, an SLO-aware DVFS framework for LLM inference serving that exploits the compute-bound/memory-bound asymmetry between prefill and decode phases, achieving up to 34% energy reduction on A100 GPUs. At finer granularity, Lammertyn et al. [13] applied kernel-level DVFS to LLM training on datacenter GPUs, demonstrating that iteration-level frequency control leaves substantial savings unrealized—kernel-level control recaptures up to 14.6% energy without performance loss, confirming that granularity of frequency control matters even for training workloads. While these studies confirm that GPU frequency selection is consequential for transformer workloads, all target datacenter-class GPUs (A100, H100) and rely on profiling APIs (NVML, nvprof) unavailable on embedded SoCs; none addresses fine-tuning workloads where backpropagation and optimizer state updates dominate memory and compute patterns. Recent work also uses GPU frequency scaling for deadline-aware scheduling [10] and SLO-constrained LLM serving, with 22–45% inference energy savings. These works are aimed at inference workloads that require milliseconds to execute and fixed accuracy, unlike multi-hour fine-tuning runs where energy and accuracy trade-offs accumulate through each epoch.

Energy-Aware Machine Learning. The environmental cost of large-model training has driven renewed focus on energy efficiency [29, 31]. Patterson et al. [24] quantified the carbon footprint of training large neural networks, while Luccioni et al. [16] extended this to the full lifecycle of BLOOM (176B parameters), including manufacturing and deployment emissions. Both works motivate energy optimization across the training pipeline, from datacenter pre-training to edge fine-tuning. Zeus [41] proposed an online framework that jointly tunes GPU power limits and batch sizes for recurring training jobs, reducing energy by 15–75% on datacenter GPUs. Perseus [6] extended this to pipeline-parallel training by removing energy bloat through graph-cut-based scheduling, achieving up to 30% savings on GPT-3 without throughput loss. EnvPipe [5] exploits pipeline bubbles to lower SM frequency during

slack time, saving up to 28% energy with under 1% slowdown. Koszczal et al. [12] showed that GPU power capping in multi-GPU training yields 17–25% energy-delay product improvements. Transformer efficiency research also targets model-level modifications (pruning [7], distillation, and parameter-efficient fine-tuning such as LoRA [9] and BitFit [42]) or inference-time compression [8]. Avinash [3] profiled LoRA/QLoRA fine-tuning energy on a consumer GPU (RTX 4060), reporting 0.15 J/token, but no equivalent measurement exists for embedded platforms. These algorithmic techniques reduce computational requirements but are orthogonal to hardware-level DVFS during fine-tuning. All datacenter-focused DVFS works (Zeus, Perseus, EnvPipe) target server-class GPUs (A100, A40, V100) with power capping APIs absent on embedded platforms, making them inapplicable to edge deployments.

Embedded Platform Studies. Jetson-focused benchmarks report performance-power sensitivity across NVP modes [2, 22, 32]. Prashanthi et al. [26] characterized power modes and DVFS behavior across four Jetson generations (Nano, NX, AGX, Orin), showing that enabling DVFS reduces idle and training power on larger devices but has negligible impact on smaller ones, informing our choice of the AGX Orin platform. Prashanthi et al. [25] measured containerized DNN training on Jetson Orin AGX, reporting power across NVP modes for CNN workloads, but did not examine transformer architectures or per-frequency characterization. Several studies characterize transformer deployment on embedded platforms—including vision transformer inference on embedded SoCs [18], edge ViT surveys [28], learning-based DVFS for edge-cloud inference [40], and compressed model deployment on Jetson and ARM [30]—but all focus exclusively on inference optimization. Li et al. [14] addressed transformer fine-tuning on device-edge architectures with privacy constraints, demonstrating demand for on-device adaptation but without DVFS analysis. Most recently, Xu et al. [39] proposed Camel, a multi-armed bandit framework that jointly selects GPU frequency and batch size for LLM inference on the Jetson AGX Orin—the same platform used in our study. Their frequency sweep (306–930 MHz, overlapping our characterization range) and 12.4–29.9% energy-delay product improvements confirm that GPU frequency selection yields substantial gains on this platform. However,

Camel targets inference with models under 3B parameters via llama.cpp, leaving fine-tuning workloads—which involve backpropagation, gradient accumulation, and optimizer states absent in inference—entirely uncharacterized. The closest existing work to ours is Woisetschläger et al. [38], who fine-tuned FLAN-T5 models (80M–3B parameters) with LoRA on a 10-device Jetson AGX Orin 64GB testbed for federated learning. While they share our platform and fine-tuning task, their study operated exclusively under the default power mode, did not vary GPU frequency, and measured training throughput and communication cost rather than energy-frequency relationships. Our work differs by systematically sweeping 11 GPU frequencies per workload, characterizing the resulting U-shaped energy curves, and developing a frequency selection algorithm from the observed GPU utilization patterns. No prior work provides per-frequency energy measurements for transformer fine-tuning at multiple model scales with controlled baseline comparisons.

Research Gap. To our best knowledge, no prior study has characterized GPU DVFS for transformer fine-tuning across model scales on power-constrained embedded platforms. Datacenter DVFS frameworks (Zeus, Perseus, EnvPipe) rely on power capping APIs and multi-GPU pipeline parallelism unavailable on embedded SoCs. Embedded platform studies characterize inference or federated training logistics, not the energy-frequency relationship during fine-tuning. Model-level efficiency techniques (LoRA, pruning, distillation) reduce computation but do not address hardware frequency selection. A systematic review of Green AI [34] confirms that hardware-level energy optimization on embedded platforms remains underexplored relative to algorithmic and datacenter-focused approaches. As recently as 2026, the most advanced GPU DVFS work for LLM training operates at kernel level on datacenter hardware [13], and the most recent embedded DVFS work on the Jetson AGX Orin targets inference [39]. The intersection of these two directions—transformer fine-tuning on embedded platforms—remains unaddressed, and this paper provides the first systematic characterization.

Chapter 3

Methodology

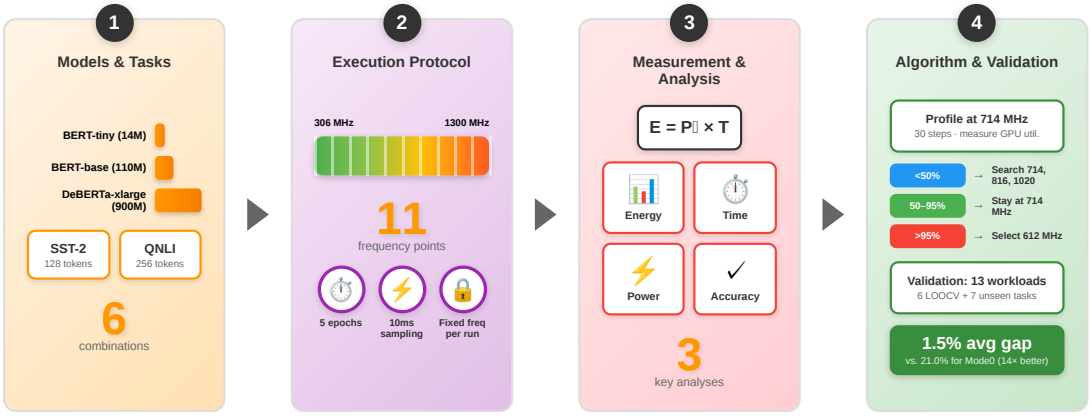

Figure 3-1: Methodology overview: Four-stage pipeline for DVFS characterization and algorithm validation. (1) Workload selection: 3 models (14M, 110M, 900M) × 2 tasks (SST-2, QNLI) = 6 combinations. (2) Execution: 11-point frequency sweep (306–1300 MHz) + 2 baselines on Jetson Orin = 77 experiments. (3) Characterization: Energy-frequency curves and GPU utilization clustering. (4) Algorithm: Profile at 714 MHz, branch on GPU utilization, validate on 13 workloads (1.5% gap from optimal).

Figure 3-1 presents our experimental methodology across four stages: workload selection, execution protocol, characterization analysis, and algorithm validation. Platform configuration and hyperparameter settings appear in Tables 3.1 and 3.2 respectively. The following sections detail model selection (Section 3.2), training protocol (Section 3.3), and metrics collection (Section 3.4).

3.1 Experimental Platform

Experiments run on the NVIDIA Jetson AGX Orin Developer Kit with 64GB LPDDR5 memory (204GB/s bandwidth), an Ampere GPU with 2048 CUDA cores and 64 Tensor Cores, and a 12-core ARM Cortex-A78AE CPU. The platform exposes 11 discrete GPU frequencies from 306 to 1300 MHz under user-space governor control. All non-essential background processes are disabled and the CPU governor is fixed to performance mode.

Table 3.1: Experimental Platform Configuration

Parameter	**Value / Description**
Hardware	
Platform	NVIDIA Jetson AGX Orin
CPU	Arm Cortex-A78AE, 8 active cores
GPU	NVIDIA Ampere
CUDA Cores	2048
Memory	64 GB LPDDR5, unified CPU/GPU
Memory Bandwidth	204.8 GB/s
Frequency Range	306 MHz – 1300 MHz
Software Stack	
OS / Kernel	Linux 5.15 (Tegra)
CUDA	12.6
NVIDIA Driver	540.4.0
PyTorch	2.5.0 (NV24.08 build)
Python	3.10.12
HuggingFace `transformers`	via `pip` (venv)
Measurement Tool	`tegrastats`

3.2 Models and Datasets

We evaluate three transformer architectures spanning two orders of magnitude in parameter count: BERT-tiny (14M parameters, 4 layers, 312 hidden size), BERT-base (110M parameters, 12 layers, 768 hidden size), and DeBERTa-v2-xlarge (900M parameters, 24 layers, 1536 hidden size). Fine-tuning targets two GLUE tasks [35]: SST-2 (sentiment analysis, 67K training examples, 128-token sequences) and QNLI

(question-answer natural language inference, 104K training examples, 256-token sequences). This design separates model size (14M vs. 900M parameters) from dataset characteristics (short vs. long sequences).

Table 3.2 summarizes model architecture, checkpoint source, and training hyperparameters.

Table 3.2: Model Specifications and Training Configuration

Model	**Params** (M)	**Layers**	**Hidden**	**Heads**	**Batch**	**Ep.**	**LR**	**Opt.**
BERT-tiny	14	2	128	2	128	5	2e-5	AdamW
BERT-base	110	12	768	12	128	5	2e-5	AdamW
DeBERTa-v2-xlarge	900	24	1536	24	128/16*	5	2e-5	AdamW

*Batch size 16 for QNLI due to memory constraints. Fixed seed (42), FP16 mixed precision, gradient clipping (1.0), and linear warmup over 10% of steps are used across all runs. Checkpoints: `prajjwal1/bert-tiny`, `bert-base-uncased`, `microsoft/deberta-v2-xlarge`.

3.3 DVFS Characterization Protocol

For each model-dataset combination, we sweep GPU frequencies from 306 to 1300MHz at approximately 100MHz intervals, giving 11 frequency points. Each configuration is measured once and the variance is controlled by: (1) running identical training pipelines across frequencies, differing only in the pinned GPU frequency; (2) measuring power with `tegrastats` at 10 ms sampling; (3) synchronizing power measurements to active training windows, excluding data loading and evaluation; and (4) computing total energy as:

$$E_{\text{total}} = \left(\frac{1}{N} \sum_{i=1}^{N} P_{\text{CPU},i} + P_{\text{GPU},i} \right) \times T_{\text{train}}, \tag{3.1}$$

where $P_{\text{CPU},i}$ and $P_{\text{GPU},i}$ are instantaneous CPU and GPU power at sample i, and T_{train} is wall-clock training time.

We compare manual frequency selection against two baseline configurations: Mode0 (unconstrained, dynamic frequency scaling up to 1300MHz) and Mode 2 (30W

power budget with dynamic scaling constrained to the power target). The study comprises 77 experiments in total: 57 frequency sweeps and 12 baseline measurements.

3.4 Metrics and Validity

We report wall-clock training time, validation accuracy after 5 epochs, average CPU/GPU power, CPU/GPU utilization, and total energy. Accuracy measurements confirm that frequency scaling does not affect convergence: across all 67 experiments, accuracy varies by less than $\pm 0.5\%$ from baseline values.

Threats to validity: Single-run measurements provide datapoints without confidence intervals. But to our observations, a typical variance is approximately 3–5% for energy metric. Also, note that results may not generalize to decoder-only architectures (e.g., GPT, LLaMA, etc.), platforms with different memory hierarchies, or sequences exceeding 512 tokens.

Chapter 4

Results Analysis for Characterization

4.1 BERT-tiny Results (14M Parameters)

GPU utilization stays low across all tested frequencies, ranging from 10–51%, while training time decreases by 2.2× from the lowest to the highest frequency point.

For SST-2, the optimal frequency is 1020 MHz with 92 J total energy, achieving 29.1% savings over Mode2 (130 J) and 9.6% over Mode0 (102 J). The energy-frequency curve in Figure 4-1 shows a flat profile across mid-to-high frequencies (714–1224 MHz), with energy varying by only ±8 J in this range, indicating limited sensitivity to precise frequency selection.

For QNLI, the optimal frequency drops to 816 MHz consuming 535 J. The Mode0 baseline (751 MHz, 519 J) outperforms manual selection by 3.1%, showing that dynamic governors can find near-optimal frequencies for workloads with low GPU utilization (27–51%). This contradicts prior claims [10] that manual tuning always exceeds dynamic governors.

Classification accuracy holds at 65.5% (SST-2) and 72.7% (QNLI) across all GPU frequencies, confirming that frequency scaling does not affect model quality.

Table 4.1: BERT-tiny: Results on SST-2 and QNLI

Dataset	Freq. (MHz)	Acc. (%)	Time (s)	CPU Freq (MHz)	GPU Freq (MHz)	CPU Util. (%)	GPU Util. (%)	Total Pwr (mW)	Energy (J)	ΔT vs Mode 0 (%)
SST-2	306	65.5	25.9	1267	306	58	29	**5426**	140	+67.1
	408	65.5	21.4	1361	408	61	24	5571	119	+38.1
	510	65.5	19.7	1375	510	59	20	5711	112	+27.1
	612	65.5	17.5	1466	612	60	18	6075	106	+12.9
	714	65.5	15.9	1416	714	58	16	6098	97	+2.6
	816	65.5	15.4	1436	816	57	14	6252	96	−0.6
	918	65.5	13.9	1600	918	66	16	7262	100	−10.3
	1020	65.5	13.1	1463	1020	55	13	7041	**92**	**−15.5**
	1122	65.5	12.0	1608	1122	62	13	8371	100	−22.6
	1224	65.5	11.7	1440	1224	53	10	8206	96	−24.5
	1300	65.5	**11.6**	1656	1300	65	13	9377	108	−25.2
	Mode0	65.5	15.5	1539	665	63	16	6585	102	0.0
	Mode2	65.8	15.9	1396	1055	68	17	8186	130	+2.6
QNLI	306	72.7	110.8	1159	306	56	51	6572	727	+99.3
	408	72.7	91.5	1259	408	60	43	6925	633	+64.6
	510	72.7	78.7	1362	510	63	42	7523	592	+41.5
	612	72.7	68.9	1426	612	63	36	8123	559	+23.9
	714	72.7	61.5	1538	714	63	35	8812	541	+10.6
	816	72.7	55.2	1663	816	66	34	9712	**535**	**−0.7**
	918	72.7	51.7	1726	918	67	32	10452	540	−7.0
	1020	72.7	49.6	1696	1020	66	30	10809	536	−10.8
	1122	72.7	47.8	1683	1122	64	27	11500	549	−14.0
	1224	72.7	46.5	1628	1224	60	24	11788	547	−16.4
	1300	72.7	46.2	1709	1300	69	27	13074	604	−16.9
	Mode0	72.7	55.6	1712	751	72	29	9344	519	0.0
	Mode2	72.4	63.1	1418	1196	70	34	11059	697	+13.5

Mode0: maximum performance baseline; Mode2: 30W default. Bolded and green-colored row frequency is the optimal. ΔT vs Mode0: training-time change relative to Mode0 (%); negative = faster than Mode0.

Energy–Training-Time Trade-off (BERT-tiny) The more practically relevant comparison is against the two platform defaults available to practitioners. Against Mode2 (30W power cap), the optimal frequency dominates on *both* dimensions simultaneously: on SST-2, 1020 MHz is 29.2% more energy-efficient *and* 17.6% faster (15.9 s → 13.1 s); on QNLI, 816 MHz saves 23.2% energy and runs 12.5% faster (63.1 s → 55.2 s). Against Mode0, the optimal frequency also dominates on SST-2 (9.8% energy saving, 15.5% faster). The sole exception is QNLI, where Mode0 (519 J, 55.6 s) edges the manual optimum (535 J, 55.2 s) by 3.1% in energy at near-identical

training time, as discussed in Key Finding 3. These results indicate that for BERT-tiny, the energy–training-time trade-off is favourable relative to realistic deployment choices: no slowdown cost is incurred versus Mode2, and the dominant comparison against Mode0 applies to the more common short-sequence workload (SST-2).

Key Findings. BERT-tiny demonstrates three characteristics relevant to frequency selection strategy:

1) GPU utilization stays low across all tested frequencies (10–51% on SST-2, 27–51% on QNLI), with CPU utilization moderately higher (55–68% and 56–72% respectively). Neither processor reaches saturation, suggesting BERT-tiny's small computational footprint does not fully occupy the platform at any frequency point.

2) Mode0's 3.1% energy advantage over manual fixed frequency on QNLI (519 J vs. 535 J at 816 MHz) contradicts the assumption [10] that fixed frequencies always dominate for training workloads. Adaptive governors can outperform static tuning when GPU utilization is low.

3) The narrow energy spread across 714–1224 MHz (±8 J) on SST-2 shows that small models are insensitive to precise frequency selection in the mid-range, reducing the value of manual tuning for this workload class.

4.2 BERT-base Results (110M Parameters)

Table 4.2 reports BERT-base results, representative of production-scale transformer deployment. Unlike BERT-tiny, GPU utilization is consistently high (86–94%) across all frequencies, while CPU utilization remains moderate (39–56%).

For SST-2, the energy minimum occurs at 714 MHz with 25,221 J, yielding 25.2% savings over Mode2 (33,696 J) and 22.9% over Mode0 (32,713 J). The energy curve (Figure 4-1) follows a well-defined U-shape: energy rises by 30.7% as frequency increases from 714 MHz to 1300 MHz (elevated power draw) and by 25.0% as frequency falls from 714 MHz to 306 MHz (extended training time).

For QNLI, the optimal frequency shifts to 816 MHz (21,034 J), achieving 28.0% savings over Mode2 (29,230 J). The Mode0 baseline (805 MHz, 20,071 J) again

matches manual optimization, operating only 11 MHz from the optimal point and consuming 4.8% less energy than the manually-selected 816 MHz. For production-scale models on long-sequence workloads, the Jetson dynamic governor finds near-optimal frequencies without manual intervention.

Mode2 (30W power budget) performs worse than Mode0 for SST-2, consuming 3.0% more energy despite the power constraint. The 30W limit forces operation at 1287 MHz – well into the inefficient high-frequency regime—while extending training time to 1425 s versus 978 s for Mode0. The power reduction does not compensate for the time penalty, showing that static power budgets do not guarantee energy efficiency.

Energy–Training-Time Trade-off (BERT-base) The trade-off becomes pronounced at production scale, and its shape depends strongly on the baseline. Against Mode2 (30W power cap), the optimal frequency dominates on *both* dimensions on both tasks: on SST-2, 714 MHz saves 25.1% energy at essentially the same training time (1432.5 s vs. 1425.0 s, +0.5%); on QNLI, 816 MHz saves 28.0% energy *and* runs 13.4% faster (1208.9 s $\rightarrow$ 1047.5 s). Against Mode0, the picture splits. On SST-2, reaching the optimum costs a real slowdown: 22.9% energy saving at the price of 46.4% longer training (978.3 s $\rightarrow$ 1432.5 s, +454 s absolute). On QNLI, the dynamic governor lands within 11 MHz of the manual optimum and edges it by 4.8% in energy at near-identical time (1056.0 s vs. 1047.5 s), consistent with the Section V-B finding that Mode0 self-adapts well on long-sequence workloads. For BERT-base, the trade-off is therefore genuine but baseline-dependent: practitioners using Mode2 face no slowdown cost regardless of task, while those using Mode0 must accept a $\sim$46% time penalty on short sequences in exchange for $\sim$23% energy savings.

Key Findings. BERT-base reveals three properties critical for mid-scale transformer deployment:

1) GPU utilization is consistently high (86–94%) across all tested frequencies. The energy-optimal frequency for SST-2 is 714 MHz, and QNLI's optimum (816 MHz) is one frequency step away. For workloads with GPU utili. in this range, 714 MHz provides a near-optimal default without further search.

Table 4.2: BERT-base: Results on SST-2 and QNLI

Dataset	Freq. (MHz)	Acc. (%)	Time (s)	CPU Freq (MHz)	GPU Freq (MHz)	CPU Util. (%)	GPU Util. (%)	Total Pwr (mW)	Energy (J)	ΔT vs Mode0 (%)
SST-2	306	92.5	3101.9	1512	306	47	94	10160	31515	+217.1
	408	92.5	2351.0	1506	408	47	94	12150	28564	+140.3
	510	92.5	1930.6	1514	510	47	93	13541	26142	+97.3
	612	92.5	1638.4	1491	612	45	92	15574	25516	+67.5
	714	92.5	1432.5	1500	714	45	91	17607	**25221**	**+46.4**
	816	92.5	1290.3	1501	816	44	91	20000	25800	+31.9
	918	92.5	1171.7	1503	918	43	89	22280	26104	+19.8
	1020	92.5	1094.2	1470	1020	42	88	25178	27550	+11.8
	1122	92.5	1045.0	1425	1122	40	87	27654	28899	+6.8
	1224	92.5	998.8	1401	1224	39	86	31007	30970	+2.1
	1300	92.5	976.8	1435	1300	42	86	33741	32958	−0.2
	Mode0	92.5	978.3	1429	1270	40	86	33438	32713	0.0
	Mode2	92.0	1425.0	1131	1287	56	90	23647	33695	+45.7
QNLI	306	89.9	2576.0	942	306	5	95	10312	26564	+143.9
	408	89.9	1955.3	943	408	5	94	12065	23591	+85.2
	510	89.9	1595.8	944	510	5	93	13447	21459	+51.1
	612	89.9	1350.9	1343	612	53	92	15669	21166	+27.9
	714	89.9	1183.7	1346	714	54	91	17774	21038	+12.1
	816	89.9	1047.5	1374	816	53	91	20082	**21034**	**−0.8**
	918	89.9	967.0	1358	918	54	90	22535	21791	−8.4
	1020	89.9	890.5	1371	1020	54	89	25720	22903	−15.7
	1122	89.9	851.4	1348	1122	55	89	28368	24153	−19.4
	1224	89.9	810.0	950	1224	5	88	32108	26006	−23.3
	1300	89.9	798.8	1354	1300	57	88	34417	27490	−24.4
	Mode0	89.9	1056.0	1318	805	54	84	19007	20071	0.0
	Mode2	89.3	1208.9	1231	1287	57	91	24179	29230	+14.5

Mode0: maximum performance baseline; Mode2: 30W default. Bold frequency is the optimal. Single-run measurements; typical variance <5%. ΔT vs Mode0: training-time change relative to Mode0 (%); negative = faster than Mode0.

2) The U-shaped energy curve is steep: deviating from the 714 MHz optimum leads to 25–31% energy penalties. This contrasts with BERT-tiny's flat profile, showing that precise frequency selection matters more as model size grows.

3) Mode2 constantly underperforms (25.2–28.0% energy waste), while Mode0 effectiveness depends on the workload, i.e. it matches manual tuning on QNLI but not on SST-2. These patterns motivate the utilization-guided frequency selection strategy in Section 5.

4.3 DeBERTa-xlarge Results (900M Parameters)

Table 4.3 presents DeBERTa-xlarge results, our largest model with nearly 8× more parameters than BERT-base. GPU utilization is near-saturated (95–98%) across all frequencies for both tasks, while CPU utilization holds steady at 65–72%.

For SST-2, the optimal frequency is 714 MHz consuming 40,927 J, yielding 21.8% savings over Mode0 (52,317 J). This matches BERT-base's optimal frequency despite the 8× difference in model size. The energy curve has a broad minimum spanning 612–816 MHz with less than 0.3% variation, so frequency selection is forgiving within this range.

For QNLI, the optimal frequency shifts *downward* to 612 MHz (536,503 J), achieving 31.5% savings over Mode2 (783,258 J) and 2.9% over Mode0 (813 MHz, 552,288 J). This downward shift contrasts with BERT-base's upward shift on QNLI (714→816 MHz). Training time decreases by only 1.94× from 612 to 1300 MHz, substantially less than BERT-base's 3.18× over a comparable range despite similarly high GPU utilization. Weak time scaling means that raising GPU frequency yields diminishing reductions in training time while power continues to rise, pushing the energy-optimal point toward lower frequencies.

The Mode2 baseline for SST-2 shows anomalous behavior (262,768 J versus 52,317 J for Mode0, a 5× increase). The 30W power cap forced batch size reduction from 128 to 16, increasing training steps by 8× and energy proportionally. This configuration is not directly comparable and is excluded from the primary analysis.

Energy–Training-Time Trade-off (DeBERTa-xlarge) At 900M parameters, the energy–time trade-off is the most pronounced in our study. On SST-2, reaching the 714 MHz optimum saves 21.8% energy versus Mode0 but costs 51.3% more wall-clock time (1516.2 s → 2294.4 s, +778 s absolute, or ∼13 min); the Mode2 baseline is not directly comparable due to a forced batch size change, so it is excluded from this comparison. On QNLI, the trade-off divides cleanly along the baseline: against Mode2, 612 MHz delivers a substantial 31.5% energy saving but adds 11.2% to training time (33,408 s → 37,154 s, +62 min); against Mode0, the energy saving

Table 4.3: DeBERTa-xlarge: Results on SST-2 and QNLI

Dataset	Freq. (MHz)	Acc. (%)	Time (s)	CPU Freq (MHz)	GPU Freq (MHz)	CPU Util. (%)	GPU Util. (%)	Total Pwr (mW)	Energy (J)	ΔT vs Mode0 (%)
SST-2	306	96.3	5136.8	1691	306	71	97	10764	55292	+238.8
	408	96.3	3878.6	1700	408	71	97	12250	47512	+155.8
	510	96.3	3144.9	1707	510	72	97	13681	43025	+107.4
	612	96.3	2638.6	1711	612	71	97	15555	41044	+74.0
	714	96.3	2294.4	1700	714	71	96	17838	**40927**	**+51.3**
	816	96.3	2023.4	1710	816	72	96	20254	40981	+33.5
	918	96.3	1849.2	1707	918	71	96	22741	42053	+22.0
	1020	96.3	1716.5	1706	1020	71	96	25855	44378	+13.2
	1122	96.3	1627.7	1699	1122	71	95	28444	46298	+7.4
	1224	96.3	1549.0	1697	1224	71	95	32270	49986	+2.2
	1300	96.3	1516.5	1694	1300	72	95	34604	52475	+0.0
	Mode0	96.3	1516.2	1696	1270	71	95	34505	52317	0.0
	Mode2*	96.7	11714.0	1358	1295	66	97	22432	262768	+672.6
QNLI	306	95.4	73055.8	1627	306	66	98	9494	693592	+159.6
	408	95.3	54942.0	1632	408	65	98	10860	596670	+95.3
	510	95.2	44244.3	1631	510	65	98	12339	545931	+57.2
	612	95.3	37154.0	1615	612	65	98	14440	**536503**	**+32.0**
	714	95.2	32238.2	1607	714	64	98	16729	539312	+14.6
	816	95.2	28136.8	1613	816	64	98	19534	549624	−0.0
	918	95.4	25135.8	1612	918	65	98	22765	572217	−10.7
	1020	95.3	22971.8	1613	1020	65	98	26575	610475	−18.4
	1122	95.2	21401.6	1610	1122	65	97	29964	641276	−23.9
	1224	95.4	19963.2	1630	1224	65	97	34984	698391	−29.1
	1300	95.3	19161.8	1623	1300	66	97	38464	737041	−31.9
	Mode0	95.3	28139.2	1614	813	65	98	19627	552288	0.0
	Mode2	95.3	33408.3	1366	1296	66	98	23445	783258	+18.7

ΔT vs Mode0: training-time change relative to Mode0 (%); negative = faster than Mode0.
*Mode2 SST-2 used batch=16 (vs batch=128); not directly comparable.

collapses to only 2.9% while the slowdown grows to 32.0% (+9,015 s, or ∼2.5 h on a single fine-tuning run). For DeBERTa-xlarge, this means the practical energy saving is strong only when measured against the 30W default—a configuration that already pays a heavy time penalty for short sequences—whereas against an unconstrained dynamic governor, the marginal energy gain is too small to justify the multi-hour slowdown for time-sensitive deployments. The QNLI optimum is best interpreted as the energy-floor of the workload rather than a balanced operating point.

Key Findings. DeBERTa-xlarge reveals three properties that distinguish large-scale transformers from smaller models.

1) Despite an 8× parameter difference from BERT-base (900M vs. 110M), DeBERTa reaches the same 714 MHz optimum on SST-2. Combined with the broad energy minimum (612–816 MHz, <0.3% variation), this suggests a single reference frequency can serve models across a wide range of scales on short-sequence workloads.

2) The optimal frequency decreases from 714 MHz (SST-2) to 612 MHz (QNLI), opposite to BERT-base's 714→816 MHz increase. In the algorithm of Section 5, this downward shift corresponds to the >95% GPU utilization branch, where utilization distinguishes DeBERTa workloads from BERT-base (86–94%) and motivates selecting a lower frequency.

3) Training time scaling is substantially weaker than for BERT-base: 1.94× versus 3.18× over a comparable frequency range, despite similarly high GPU utilization. Throughput gains diminish at higher frequencies while power continues to rise, so the energy balance favors lower frequencies at this model scale—consistent with the algorithm's 612 MHz selection for the >95% utilization regime.

4.4 Summary of Fine-Tuning Time and Energy–Time Trade-off

Fine-tuning time characteristics. Training time decreases monotonically with GPU frequency across all workloads, but the degree of scaling differs substantially by model scale. Over the full 306–1300 MHz range, BERT-tiny time compresses by 2.2–2.4×, BERT-base by 3.2×, and DeBERTa-xlarge by 3.4–3.8× (Tables 4.1–4.3). Absolute training durations span three orders of magnitude: from ∼12 s (BERT-tiny SST-2 at 1300 MHz) to ∼20 h (DeBERTa-xlarge QNLI at 306 MHz). Despite this range, the U-shaped energy curve means that neither endpoint minimises energy: running at maximum frequency wastes power faster than it saves time, while running at minimum frequency extends training so long that energy climbs again.

Trade-off summary across model scales. Table 4.4 consolidates the energy

saving and training-time change at the optimal frequency relative to each baseline. Three regimes emerge.

No trade-off (vs. Mode2, all workloads). Against the default 30W cap, the optimal frequency delivers 23–32% energy savings with no slowdown penalty—and often a speedup. Mode2 forces high GPU clocks while throttling overall power, landing on the inefficient right-hand side of the energy–frequency curve and incurring unnecessary training-time overhead.

Conditional trade-off (vs. Mode0, short sequences). Against the unconstrained dynamic governor, the trade-off is real on short-sequence workloads: BERT-base SST-2 requires 46.4% more training time (+454 s) for 22.9% energy saving, and DeBERTa-xlarge SST-2 requires 51.3% more time (+778 s, ∼13 min) for 21.8% energy saving. Whether this is acceptable depends on deployment cadence: for nightly fine-tuning pipelines or batch continual-learning jobs where wall-clock time is not critical, a 13–20% energy reduction across thousands of runs represents a meaningful operational saving. For latency-sensitive retraining (e.g., rapid adaptation to a new user), Mode0 may be preferred.

Unfavourable trade-off (DeBERTa-xlarge QNLI vs. Mode0). The sole case where the energy saving is too small to justify the slowdown is DeBERTa QNLI against Mode0: only 2.9% energy saving (16 kJ) at the cost of 32% longer training (+2.5 h). Here the dynamic governor already operates close to the energy optimum (∼813 MHz vs. 612 MHz optimal), and the marginal gain from manual frequency selection does not justify the time cost.

Across all three models, the energy–training-time trade-off follows a consistent pattern determined by model scale and baseline choice. Against Mode2, the optimal frequency *never* incurs a slowdown: it dominates on both energy and time for every model-task combination where Mode2 is comparable, yielding 23–31% energy savings while training faster or at equal speed. Against Mode0, the trade-off is genuine and grows with model scale: BERT-tiny avoids any slowdown on SST-2 (−15.5% time, +9.8% energy saving), BERT-base incurs a moderate penalty on SST-2 (+46.4% time, +22.9% energy saving), and DeBERTa-xlarge faces the steepest cost on SST-2

Table 4.4: Energy saving and training-time change at the optimal frequency relative to Mode0 and Mode2. Negative time values indicate the optimal is faster than the baseline. †Mode2 SST-2 excluded for DeBERTa-xlarge (batch-size incomparability).

			vs. Mode0		vs. Mode2	
Model	**Task**	**Opt. (MHz)**	**ΔE**	**ΔT**	**ΔE**	**ΔT**
BERT-tiny	SST-2	1020	+9.6%	+2.4%	+29.1%	−0.1%
BERT-tiny	QNLI	816	−3.1%	−0.8%	+23.2%	−12.5%
BERT-base	SST-2	714	+22.9%	+46.4%	+25.2%	+0.5%
BERT-base	QNLI	816	−4.8%	−0.8%	+28.0%	−13.4%
DeBERTa-xlarge	SST-2	714	+21.8%	+51.3%	†n/a	
DeBERTa-xlarge	QNLI	612	+2.9%	+32.0%	+31.5%	+11.2%

ΔE: energy saving at optimal vs. baseline (positive = more efficient). ΔT: training-time change (positive = slower, negative = faster).

(+51.3% time, +21.8% saving) and an unfavorable ratio on QNLI (+32.0% time for only +2.9% saving).

4.5 Cross-Model Patterns and Summary

Figure 4-1 shows energy-frequency relationships for all six model-dataset combinations. All six follow a U-shaped profile: energy rises at low frequencies as extended training time dominates, and at high frequencies as elevated power dominates, with minima consistently in the 612–1020 MHz range.

Table 4.5: Summary: Optimal Frequencies and Energy Savings vs Baselines

Model	**Dataset**	**Freq (MHz)**	**Opt. E (J)**	**M0 E (J)**	**M2 E (J)**	**ΔM0 (%)**	**ΔM2 (%)**
BERT-tiny	SST-2	1020	92	102	130	+9.6	+29.1
BERT-base	SST-2	714	25,221	32,713	33,696	+22.9	+25.2
DeBERTa-xlarge	SST-2	714	40,927	52,317	-*	+21.8	-
BERT-tiny	QNLI	816	536	520	698	−3.1	+23.2
BERT-base	QNLI	816	21,035	20,071	29,230	−4.8	+28.0
DeBERTa-xlarge	QNLI	612	536,504	552,289	783,258	+2.9	+31.5

ΔM0/ΔM2: savings versus Mode0/Mode2. *Mode2 baseline unavailable for DeBERTa SST-2 due to batch size incomparability.

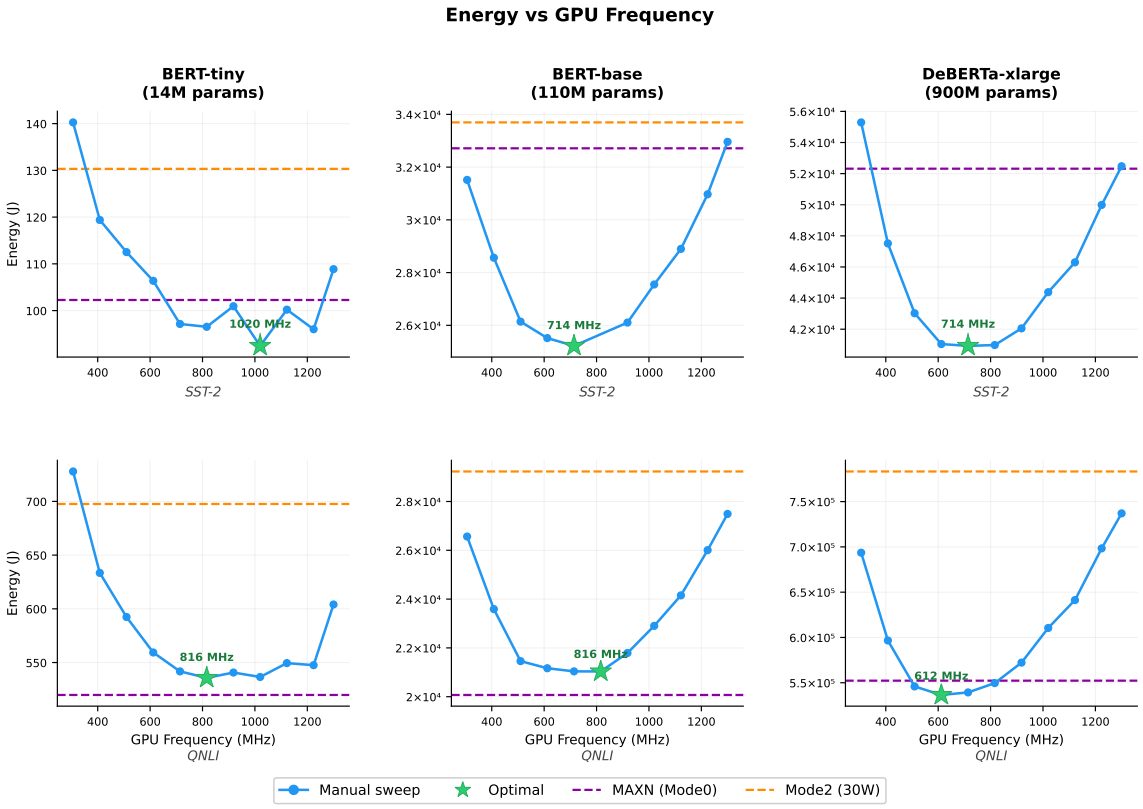

Figure 4-1: Energy consumption versus GPU frequency for BERT-tiny (left), BERT-base (center), and DeBERTa-xlarge (right) on SST-2 (top) and QNLI (bottom). Mode0 and Mode2 baselines shown as horizontal dashed lines; optimal points marked with stars.

Table 4.5 aggregates optimal configurations and baseline comparisons. Three patterns emerge.

First, optimal frequencies fall in the 612–1020 MHz range across all workloads, never at the extremes. Within this range, 714 MHz achieves the minimum for three of six configurations and falls within 5.1% of the true optimum for all six, making it a reliable reference frequency across model scales.

Second, BERT-base and DeBERTa-xlarge both minimize energy at 714 MHz on SST-2 despite an 8× parameter gap, showing that optimal frequency is not determined by model size alone. A single reference frequency can serve models across a wide range of scales under short-sequence conditions.

Third, the frequency shift between SST-2 and QNLI differs by model: BERT-tiny decreases from 1020 to 816 MHz (−204 MHz), BERT-base increases from 714

to 816 MHz (+102 MHz), and DeBERTa-xlarge decreases from 714 to 612 MHz (−102 MHz). The opposing directions for medium and large models show that sequence length affects the energy-optimal frequency differently depending on model scale, a distinction that any frequency selection policy must account for.

GPU utilization as a workload classifier. When GPU utilization at 714 MHz is examined across all six workloads, three groups appear with no overlap: BERT-tiny (16–35%), BERT-base (86–94%), and DeBERTa-xlarge (96–98%). None of our workloads lie in the 36–85% range. This leaves a big gap between groups. As such, each group corresponds to a different optimal frequency direction: below 60% utilization the true optimum lies *above* 714 MHz, at 86–94% it is near 714 MHz; above 95% it shifts *down* to 612 MHz. This key finding makes a threshold-based frequency selection policy feasible.

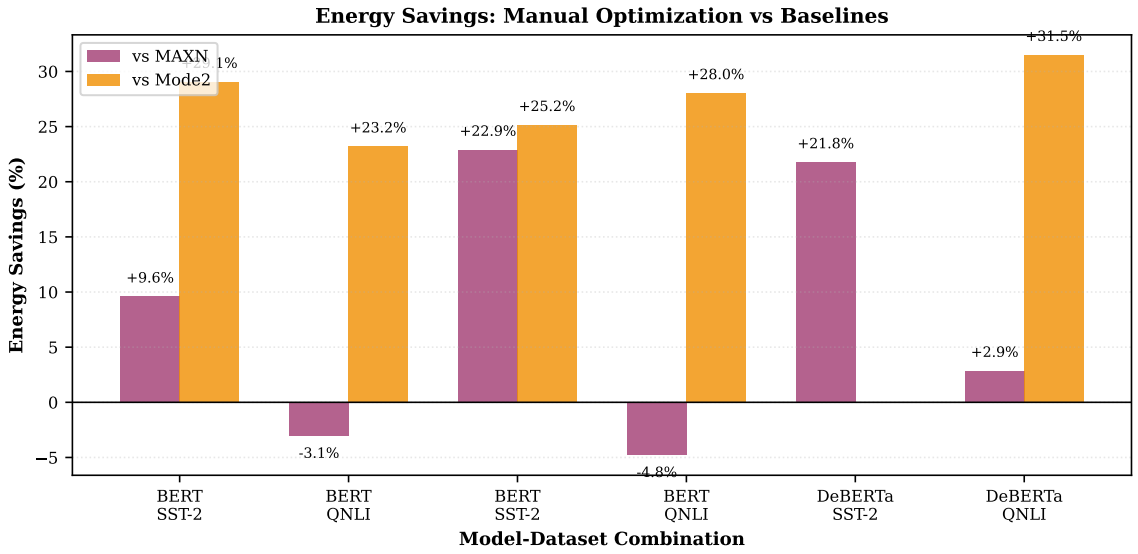

Figure 4-2: Energy savings of manual optimization versus Mode0 and Mode2 baselines.

Figure 4-2 shows savings relative to both baselines. Mode2 comparisons are consistently positive (23–32%), confirming that the default 30W configuration is uniformly suboptimal. Mode0 comparisons vary: manual selection yields 22–23% savings for BERT-base and DeBERTa-xlarge on SST-2, but negative savings (-3.1% and -4.8%) for BERT-tiny and BERT-base on QNLI, where the dynamic governor already operates near-optimally.

Chapter 5

DVFS Governor Policy and Validation

5.1 Empirical Basis based on key fidnings

Two patterns from the characterization inform the algorithm design:

Mid-range convergence. Optimal frequencies across all six workloads fall in the 612–1020 MHz range, with 714 MHz achieving the minimum for three of six configurations. For the remaining three, substituting 714 MHz for the true optimum incurs at most a 5.1% energy penalty (Table 4.5), establishing 714 MHz as a reliable reference frequency.

GPU utilization predicts optimal direction. GPU utilization measured at 714 MHz separates workloads into three groups with no overlap: below 60% (BERT-tiny, 16–35%), the true optimum lies *above* 714 MHz; at 86–94% (BERT-base), it sits near 714 MHz; above 95% (DeBERTa-xlarge, 96–98%), it shifts *below* to 612 MHz.

5.2 Algorithm Description

We propose a frequency selection algorithm (Algorithm 1) that requires at most 90 training steps of GPU utilization profiling.

Each design choice maps to an empirical observation. The reference frequency

$f_{\text{ref}} = 714$ MHz follows from mid-range convergence: it is the most common true optimum and minimizes worst-case error as a universal default. GPU utilization serves as the classifier because it cleanly separates workload classes, and the search directions follow the observed energy curve shifts.

Technical justification for GPU utilization as classifier. A natural concern is that GPU utilization is itself a function of frequency: as the clock rises, the GPU completes work faster per unit time and therefore appears less loaded. Our measurements confirm this effect but show it is small relative to the cluster separation. Across the full 306–1300 MHz sweep, within-workload utilization varies by at most 27 percentage points for BERT-tiny QNLI, 8 pp for BERT-base, and 2 pp for DeBERTa-xlarge (Table 4.1–4.3). Crucially, no workload crosses a threshold boundary at any frequency: BERT-tiny remains below 51% even at 306 MHz, well clear of the 60% lower threshold; BERT-base stays between 86–95%, within the 60–95% mid-range band; DeBERTa stays at 95–98%, above the upper threshold. Measuring utilization at the single reference point $f_{\text{ref}} = 714$ MHz is therefore sufficient: the 36–85% inter-cluster gap provides a margin larger than any frequency-induced shift observed in our data.

Moreover, using utilization as the primary signal for DVFS frequency selection is well-established: Linux kernel governors use utilization thresholds to trigger frequency transitions [10], Karzhaubayeva et al. [11] derived integrated CPU–GPU policies from characterized utilization medians on embedded Jetson platforms, and Ali et al. [1] confirmed via mutual information analysis that GPU utilization is the feature most correlated with power and execution time for optimal frequency selection.

Threshold selection. Both thresholds derive from natural clustering in the characterization data. At 714 MHz, GPU utilization falls into three non-overlapping groups: 16–35% (BERT-tiny), 86–94% (BERT-base), and 96–98% (DeBERTa-xlarge). The lower threshold of 60% is the midpoint of the unoccupied 36–85% range, providing maximum margin from both clusters. The upper threshold of 95% bisects the narrower gap between BERT-base (up to 94%) and DeBERTa-xlarge (96% and above).

Algorithm 1 GPU-Utilization-Guided Frequency Selection

Require: Transformer workload (model M, dataset D)
Ensure: Energy-optimal GPU frequency GPU_{freq}

Phase 1: Profile
Set GPU frequency to $f_{\text{ref}} = 714$ MHz
Run $N = 30$ training steps
Record mean GPU utilization GPU_{util}

Phase 2: Select frequency
if $GPU_{\text{util}} < 60\%$ **then**
    Run N steps each at 816 MHz and 1020 MHz
    $GPU_{\text{freq}} \leftarrow \arg\min_{f \in \{714, 816, 1020\}} E(f)$
else if $GPU_{\text{util}} \leq 95\%$ **then**
    $GPU_{\text{freq}} \leftarrow 714$ MHz
else
    $GPU_{\text{freq}} \leftarrow 612$ MHz

Validation confirms that these thresholds produce identical classifications in every fold, and the 13-workload validation (Table 5.2) shows no workload in the 36–85% gap, supporting threshold robustness. For workloads that fall in this unoccupied range, the algorithm defaults to 714 MHz—the most common optimum across the study—as a conservative fallback.

The opposite frequency shifts on QNLI—BERT-base upward (714→816 MHz) and DeBERTa-xlarge downward (714→612 MHz)—are explained by asymmetric time scaling under frequency increase. From 612 to 1300 MHz, BERT-base training time decreases by 3.18× on QNLI, so higher frequencies reduce time faster than they raise power, shifting the energy optimum upward. DeBERTa achieves only 1.94× time reduction over the same range: power rises superlinearly while time reduction saturates, shifting the optimum downward. This asymmetry is consistent with DeBERTa's near-saturated GPU utilization (95–98%) across all frequencies—a signature of compute and memory pressure leaving little headroom to benefit from higher clocks. We specualte that the longer sequences (256 tokens on QNLI versus 128 on SST-2) increase activation tensor sizes and attention computation, likely exacerbating memory pressure for the larger model while BERT-base, with 8× fewer parameters, retains scheduling slack that higher frequencies can exploit. Direct measurement of mem-

ory bandwidth utilization during training would be needed to confirm this inference, which we identify as future work.

Profiling cost. In order to define the search direction, the algorithm needs to figure out the stabilized GPU utilization for the run - that is called profiling cost. Our measurements confirm that GPU utilization stabilizes in the first few seconds of fine-tuning and maintains that utilization level until the end. Thus, we use 30 steps (just enough to stabilize) to accumulate sufficient `tegrastats` samples for a stable GPU util. average. Table 5.1 shows the profiling overhead is minimal for all workloads and doesn't hurt the performance.

Table 5.1: Profiling Overhead: 30 Steps at 714 MHz

Model	**Task**	**Time/step (s)**	**30 steps (s)**	**Full train (s)**	**Overhead (%)**
BERT-tiny	SST-2	0.003	0.1	15.9	0.5
BERT-tiny	QNLI	0.006	0.2	61.5	0.3
BERT-base	SST-2	0.23	6.8	1,433	0.5
BERT-base	QNLI	0.12	3.6	1,184	0.3
DeBERTa	SST-2	0.36	10.9	2,294	0.5
DeBERTa	QNLI	3.28	98.5	32,238	0.3

5.3 Algorithm Validation

We validate the algorithm on unseen three additional GLUE tasks MRPC, RTE, CoLA.

Methodology. For each validation workload, we ran the full experimental pipeline from Section 3: training at 7–11 GPU frequency points spanning 306–1300 MHz for BERT-tiny and BERT-base, and 612–816 MHz for DeBERTa-xlarge, with `tegrastats` monitoring plus a Mode0 baseline. This yields the true optimal frequency and energy for each workload. This yields the true optimal frequency and energy for each workload. We then compare the algorithm's frequency results against the true optimal.

For cross-validation, thresholds are derived from five characterization workloads

and tested on the held-out sixth, repeated for all six workloads. The utilization gap (36–85%) persists across all six held-out evaluations, and the 60% threshold falls well within this unoccupied range.

Table 5.2: Algorithm Validation table

Model	**Task**	**GPU (%)**	**Util. Range**	**Algo (MHz)**	**True Opt.**	**Gap vs. Opt. (%)**	**Gap vs. Mode0 (%)**
Characterization workloads (training tasks)							
BERT-tiny	SST-2	16	<60%	1020	1020	0.0	+9.6
BERT-tiny	QNLI	35	<60%	816	816	0.0	−3.1
BERT-base	SST-2	91	60–95%	714	714	0.0	+22.9
BERT-base	QNLI	91	60–95%	714	816	0.0	−4.8
DeBERTa	SST-2	96	>95%	612	714	−0.3	+21.5
DeBERTa	QNLI	98	>95%	612	612	0.0	+2.9
Avg.						0.0	+8.1
Validation workloads (unseen tasks)							
BERT-tiny	MRPC	32	<60%	714	612	−12.2	+1.0
BERT-tiny	RTE	25	<60%	816	612	−1.7	+1.2
BERT-base	CoLA	90	60–95%	714	510	−4.3	+27.9
BERT-base	MRPC	95	>95%	510	510	0.0	+30.1
BERT-base	RTE	96	>95%	510	714	0.1	+28.9
DeBERTa	MRPC	97	>95%	612	612	0.0	+24.8
DeBERTa	RTE	98	>95%	612	612	0.0	+24.2
Avg.						**−2.6**	**+19.7**

Gap vs. Opt. $= (E_{\text{algo}} - E_{\text{opt}})/E_{\text{opt}} \times 100\%$. Gap vs. Mode0 (by the same token) (+) sign indicates the algorithm uses less energy (improvement) and (−) sign indicates more (degradation).

Results. Table 5.2 reports validation across 7 workloads. The algorithm achieves a -2.6% average energy gap from the true optimum, compared to an average $+19.7\%$ improvement over Mode0. The worst case (-12.2% on BERT-tiny MRPC) occurs on a small dataset (2.4K samples) with a flat, noisy energy curve: the true optimum at 612 MHz (89.4 J) differs from the algorithm's 714 MHz selection by only 10.9 J. On larger datasets, gaps fall below 4%, and the algorithm matches the true optimum exactly on 6 of 13 workloads (0.0% gap).

DeBERTa workloads are classified correctly in all four cases, with the algorithm selecting 612 MHz each time and achieving 0.0–0.3% gap. BERT-base workloads show 0.0–4.3% gap; the 4.3% case (CoLA) arises because the smaller dataset shifts the true optimum below 714 MHz to 510 MHz. Mode0 wastes 40–45% energy on these same BERT-base workloads, showing that the default governor is poorly suited to high-utilization transformer training.

Chapter 6

Discussion, Future Work, and Conclusion

6.1 Discussion and Future Work

Discussion of limitations. The proposed algorithm has several potential weaknesses. First, the utilization thresholds (60%, 95%) are derived from a limited workload set on a single platform; workloads falling in the unoccupied 36–85% GPU utilization range would be classified by untested boundaries. Second, GPU utilization is the sole classifier, but validation showed that dataset size also influences the optimal frequency: smaller datasets (MRPC 3.7K, CoLA 8.5K samples) shifted the true optimum below 714 MHz to 510–612 MHz, a factor the current algorithm does not capture. Third, the low-utilization branch (<60%) searches only three candidate frequencies (714, 816, 1020 MHz), which produced the worst-case 12.2% gap on BERT-tiny MRPC where the true optimum (612 MHz) lay outside the search set.

Toward a second-generation algorithm. Algorithm 1 uses GPU utilization as its sole workload classifier, which captures compute intensity per step but ignores training duration. Validation on small datasets (MRPC 3.7K, RTE 2.5K, CoLA 8.5K samples) revealed that short training runs shift the energy-optimal frequency below 714 MHz, outside the algorithm's search range, producing the worst-case 12.2% gap on BERT-tiny MRPC. When total training is short, power dominates over time in

the energy balance, pulling the optimum to lower frequencies regardless of per-step utilization.

A direct extension would add training duration as a second input alongside GPU utilization. Duration is a static property (computable from dataset size, batch size, and epoch count before training begins) and is therefore unaffected by the frequency-induced variation that Algorithm 1 must account for in utilization. Short-duration workloads would be routed to a lower candidate frequency range (510–714 MHz), while longer workloads would follow the existing utilization-guided logic unchanged. This two-signal design would also reduce exposure to the unoccupied 36–85% utilization gap, since many short training runs exhibit low and variable utilization and would be classified by duration before the utilization threshold is reached.

Future work. Several extensions could address these weaknesses. First, expanding the low-utilization search set to include frequencies below 714 MHz (e.g., 612, 510 MHz) would eliminate the blind spot responsible for the worst-case gap. Second, incorporating dataset size or total training steps as a secondary input alongside GPU utilization could improve selection for small-dataset workloads. Third, extending validation to decoder-only architectures (GPT-2, LLaMA) and parameter-efficient methods (LoRA, QLoRA) would test whether utilization-based classification generalizes to workloads with different compute patterns. Fourth, cross-platform calibration on devices with different GPU architectures and memory subsystems (e.g., Jetson Orin Nano, Xavier NX) would determine whether the thresholds transfer or require per-platform tuning.

6.2 Conclusion

This paper presented a systematic per-frequency characterization of GPU DVFS for transformer fine-tuning on the NVIDIA Jetson AGX Orin. Through 77 controlled experiments spanning three model scales (BERT-tiny 14M, BERT-base 110M, DeBERTa-xlarge 900M) and two GLUE tasks (SST-2, QNLI), we established that energy-optimal GPU frequencies fall consistently in the 612–1020 MHz range (50–

60% below the 1300 MHz hardware maximum), yielding 22–32% energy savings over the default 30W mode without accuracy degradation.

The characterization answered two empirical research questions. For RQ1, the U-shaped energy-frequency pattern holds across model scales: both BERT-base and DeBERTa-xlarge minimize energy at 714 MHz on SST-2 despite an 8× parameter gap, showing that optimal frequency is not determined by model size alone. For RQ2, sequence length shifts the optimum in model-dependent directions: BERT-base shifts upward (714→816 MHz) while DeBERTa-xlarge shifts downward (714→612 MHz) on QNLI, and GPU utilization at 714 MHz separates workloads into three non-overlapping groups that predict these directions.

For RQ3, we developed a GPU-utilization-guided frequency selection algorithm grounded in these findings. The algorithm profiles GPU utilization at 714 MHz for 30 training steps (<0.5% overhead), then selects a frequency based on utilization thresholds derived from the observed clustering. Validation across 13 workloads—including 7 unseen task configurations—shows a 1.5% average energy gap from the true optimum versus 21.0% for the default governor (Mode0), a 14× reduction in energy waste.

Limitations include single-run measurements, evaluation on encoder-only architectures, and thresholds derived from one embedded platform. Future work should extend characterization to decoder-only architectures (GPT, LLaMA) and parameter-efficient methods (LoRA, QLoRA), validate threshold transferability across platforms with different memory hierarchies, and incorporate dataset size as an additional input to address cases where small datasets shift the optimum below the 714 MHz reference.

Bibliography

[1] Ghazanfar Ali, Mert Side, Sridutt Bhalachandra, Nicholas J. Wright, and Yong Chen. Performance-aware energy-efficient GPU frequency selection using DNN-based models. In *Proceedings of the 52nd International Conference on Parallel Processing (ICPP)*, pages 433–442, 2023.

[2] Antmicro Ltd. Benchmarking deep neural networks on nvidia jetson agx orin with kenning. `https://antmicro.com/blog/2022/12/benchmarking-dnn-on-nvidia-jetson-agx-orin-with-kenning/`, December 2022. Accessed: 2026-03-26.

[3] MSR Avinash. Profiling LoRA/QLoRA fine-tuning efficiency on consumer GPUs: An RTX 4060 case study. *arXiv preprint arXiv:2509.12229*, 2025.

[4] Zhongwei Chen, Jiaheng Liu, Yibo Zhang, et al. A review on edge large language models: Design, execution, and applications. *ACM Computing Surveys*, September 2024.

[5] Sangjin Choi, Inhoe Koo, Jeongseob Ahn, Myeongjae Jeon, and Youngjin Kwon. EnvPipe: Performance-preserving DNN training framework for saving energy. In *2023 USENIX Annual Technical Conference (USENIX ATC)*, pages 851–864. USENIX Association, 2023.

[6] Jae-Won Chung, Yile Gu, Insu Jang, Luoxi Meng, Nikhil Bansal, and Mosharaf Chowdhury. Perseus: Removing energy bloat from large model training. In *Proceedings of the 30th ACM Symposium on Operating Systems Principles (SOSP)*, 2024.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, 2019.

[8] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations (ICLR)*, 2021.

[9] Edward J. Hu, Yelong Shen, Phillip Wallis, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[10] Shashikant Ilager, Rajeev Muralidhar, Kotagiri Rammohanrao, and Rajkumar Buyya. A data-driven frequency scaling approach for deadline-aware energy efficient scheduling on graphics processing units (gpus). In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 579–588, 2020.

[11] Meruyert Karzhaubayeva, Aidar Amangeldi, and Jurn-Gyu Park. CNN workloads characterization and integrated CPU–GPU DVFS governors on embedded systems. *IEEE Embedded Systems Letters*, 2023.

[12] Grzegorz Koszczał, Jakub Dobrosolski, Mateusz Matuszek, and Paweł Czarnul. Performance and energy aware training of a deep neural network in a multi-GPU environment with power capping. In *Euro-Par 2023: Parallel Processing Workshops*, volume 14352 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2024.

[13] Niels Lammertyn, Robbe Decorte, Femke Ongenae, Pieter Simoens, and Mathijs Dhoedt. Reducing compute waste in LLMs through kernel-level DVFS, 2026.

[14] Hao Li, Gang Xia, Yuxuan Zhang, Zhiwen Cai, Hui Li, and Chunpeng Yu. Efficient deployment and fine-tuning of transformer-based models on the device-edge. In *Euro-Par 2024: Parallel Processing Workshops*, volume 15386 of *LNCS*. Springer, 2025.

[15] Qunyou Liu, Darong Huang, Marina Zapater, and David Atienza. GreenLLM: SLO-aware dynamic frequency scaling for energy-efficient LLM serving, 2025.

[16] 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.

[17] Paul Joe Maliakel, Shashikant Ilager, and Ivona Brandic. Characterizing LLM inference energy-performance tradeoffs across workloads and GPU scaling, 2025.

[18] Iván Martín-Salinas, José M. Badia, Óscar Valls, et al. Evaluating and accelerating vision transformers on GPU-based embedded edge AI systems. *The Journal of Supercomputing*, 81:349, 2025.

[19] Xinxin Mei, Qiang Wang, and Xiaowen Chu. A survey and measurement study of GPU DVFS on energy conservation. *Digital Communications and Networks*, 3(2):89–100, 2017.

[20] Francisco Mendes, Pedro Tomás, and Nuno Roma. Decoupling gpgpu voltage-frequency scaling for deep-learning applications. *Journal of Parallel and Distributed Computing*, 165:32–51, 2022.

[21] Seyed Morteza Nabavinejad, Hassan Hafez-Kolahi, and Sherief Reda. Coordinated dvfs and precision control for deep neural networks. *IEEE Computer Architecture Letters*, 18(2):136–140, 2019.

[22] NVIDIA Corporation. Jetson benchmarks. `https://developer.nvidia.com/embedded/jetson-benchmarks`, 2024. Accessed: 2026-03-26.

[23] Vraj Patel, Hui Cheng, Eiman Ebrahimi Khan, et al. Slo-aware gpu frequency scaling for energy efficient llm inference serving. *arXiv preprint arXiv:2408.05235*, August 2024.

[24] David Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[25] S. K. Prashanthi, Vineeth Hegde, Karthik Patchava, Anirban Das, and Yogesh Simmhan. Performance characterization of containerized DNN training and inference on edge accelerators. In *IEEE International Conference on High Performance Computing (HiPC)*, 2023.

[26] S. K. Prashanthi, Vineeth Hegde, Karthik Patchava, Anirban Das, and Yogesh Simmhan. Characterizing the performance of accelerated jetson edge devices for training and inference. *arXiv preprint arXiv:2509.20160*, 2025.

[27] Miguel Rodriguez, Ana Santos, and Vijay Kumar. Deploying ai on edge: Advancement and challenges in edge intelligence. *Mathematics*, 13(11):1878, June 2025.

[28] Soumya Saha and Long Xu. Vision transformers on the edge: A comprehensive survey. *Neurocomputing*, 2025.

[29] Siddharth Samsi, Dan Zhao, Joseph McDonald, et al. From words to watts: Benchmarking the energy costs of large language model inference. *arXiv preprint arXiv:2310.03003*, October 2023.

[30] Asad Ullah Samson. Lightweight transformer architectures for edge devices in real-time applications. *arXiv preprint arXiv:2601.03290*, 2026.

[31] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650. Association for Computational Linguistics, 2019.

[32] Tushar Prasanna Swaminathan, Ahan MR, and Abhishek Joshi. Benchmarking deep learning models on nvidia jetson nano for real-time systems: An empirical investigation. *arXiv preprint arXiv:2406.17749*, June 2024.

[33] Zhenheng Tang, Yuxin Wang, Qiang Wang, and Xiaowen Chu. The impact of gpu dvfs on the energy and performance of deep learning: An empirical study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems (e-Energy '19)*, pages 315–325, New York, NY, USA, 2019. ACM.

[34] Roberto Verdecchia, June Sallou, and Luís Cruz. A systematic review of green AI. *WIREs Data Mining and Knowledge Discovery*, 13(4):e1507, 2023.

[35] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, pages 353–355, 2018.

[36] Farui Wang, Weizhe Zhang, Shichao Lai, Meng Hao, and Zheng Wang. Dynamic gpu energy optimization for machine learning training workloads. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2943–2954, 2022.

[37] Qiang Wang and Xiaowen Chu. Gpgpu performance estimation with core and memory frequency scaling. *IEEE Transactions on Parallel and Distributed Systems*, 31(12):2865–2881, 2020.

[38] Herbert Woisetschläger, Alexander Isenko, Shiqiang Wang, Ruben Mayer, and Hans-Arno Jacobsen. Federated fine-tuning of LLMs on the very edge: The good, the bad, the ugly. In *NeurIPS 2024 Workshop on Federated Foundation Models*, 2024.

[39] Hao Xu, Penglong Jiao, et al. Camel: Energy-aware LLM inference on resource-constrained devices, 2025.

[40] Peng Yang, Nuo Xu, Kai Huang, et al. DVFO: Learning-based DVFS for energy-efficient edge-cloud collaborative inference. *IEEE Transactions on Mobile Computing*, 23(10):9396–9413, 2024.

[41] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. Zeus: Understanding and optimizing GPU energy consumption of DNN training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 119–139. USENIX Association, 2023.

[42] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, June 2021.

[43] Jakov Zidar, Tomislav Matic, Ivan Aleksi, and Zeljko Hocenski. Dynamic voltage and frequency scaling as a method for reducing energy consumption in ultra-low-power embedded systems. *Electronics*, 13(5):826, February 2024.