Modeling Energy Consumption in Deep Learning Architectures Using Power Laws

Abstract. Modern Deep Learning architectures such as LSTM, GRU, and Transformers achieve remarkable performance in various sequence processing tasks. Yet, their high computational cost and energy consumption have raised concerns about their environmental impact and the sustainability of Deep Learning.

In this paper, we present an empirical study assessing the efficiency of training LSTM, GRU, and Transformer models on a GPU. By evaluating these models under various configurations, we characterize the relationship between energy consumption and pre-defined quantities such as hardware efficiency and the number of floating point operations (FLOPs) required for inference. We show that it is possible to derive scaling laws that make energy consumption predictable, given an architecture and a GPU model.

1 Introduction

Recent studies have expressed concerns about the sustainability of Deep Learning (DL) research [25, 12, 26]. The increasing use of larger datasets, more complex models, and prolonged training periods has led to a significant rise in energy usage for training. For instance, training GPT-3 required 3.14×10^{23} floating-point operations (FLOPs) [1], which amounts to at least 20 years of computation on a single H100 GPU platform (one of the fastest Nvidia GPUs). Training DeepSeek-v3, allegedly more cost-efficient than GPT-3 and its successors, was equivalent to 318 years of uninterrupted computation on that same GPU [**? ?**]. In reaction, critical questions have been raised about their economic and ecological implications. As DL systems continue to grow in scale, understanding and mitigating their energy demands has become an urgent area of research. Yet, comprehensive studies comparing energy consumption across different model architectures, training configurations, and hyperparameter choices remain limited.

This paper bridges this gap by conducting a detailed empirical investigation into the energy efficiency of three widely used model architectures: LSTM, GRU, and Transformers. These architectures are fundamentally different in their computational design. LSTMs and GRUs are recurrent neural networks (RNNs) optimized for processing sequential data through memory mechanisms, while Transformers employ self-attention mechanisms, enabling superior performance on tasks involving long-range dependencies. Despite these differences, all three architectures are widely used in applications ranging from NLP to time series forecasting, making them ideal candidates for a comparative energy consumption study.

Our study focuses on measuring the energy consumption of these architectures during training under varying model configurations, such as the number of layers, hidden dimensions, and attention heads (for Transformers). Beyond raw energy measurements, we also explore the relationship between energy usage and model performance metrics, such as hardware efficiency and the number of floating point operations (FLOPs). Our goal is to show that it is possible to derive a general law that makes energy consumption predictable, given a model architecture.

The main contributions of this paper are the following:

- We demonstrate that energy consumption can be estimated using laws that depend on the computational cost of a model, expressed in FLOPs, and a hardware efficiency measure. We empirically found that modeling hardware efficiency was critical to properly model energy consumption of a given model.
- We show that hardware efficiency for a given elementary operation, such as matrix multiplication, can be succinctly captured using a power law featuring exponential decay with a saturation offset.

2 Related Work

This section reviews prior work in three key areas: energy consumption in DL models, architectural efficiency comparisons, and optimization strategies for energy-efficient training [25, 6, 3, 11, 10, 26, 15].

2.1 Energy Consumption in DL Models

The growing size and complexity of DL models have led to a dramatic rise in energy consumption, with several studies highlighting the environmental implications of large-scale model training. For instance, Strubell et al. [20] quantified the carbon footprint of designing and training natural language processing (NLP) pipelines, showing that such pipelines can emit as much greenhouse gases as an American household over a year. The LLMCarbon framework by Faiz et al. [5] expand on this by modeling the total carbon footprint of large language models, incorporating both operational and embodied carbon emissions. It provides a predictive tool to estimate emissions across training, inference, and storage, highlighting the importance of architectural optimization, renewable energy integration, and hardware efficiency. Similarly, Patterson et al. [16] highlight the significant energy consumption and carbon footprint of training large NLP models and present strategies for reducing their environmental impact. The study reports that training GPT-3 emitted 552 tCO2e, of the same order of magnitude as a transcontinental

flight. It also observes that sparse neural networks, such as mixture-of-experts architectures, may reduce energy consumption by more than 90% compared to dense models while maintaining accuracy. However, the mixture-of-experts approach has been leveraged to increase the size of language models, not to keep it constant [27].

Carbon emissions remain hard to estimate, though. A case study on the Evolved Transformer [19] revealed an 88x overestimation in prior emissions calculations, underscoring the need for accurate reporting and optimized configurations. These results advocate for transparent reporting of energy and carbon metrics in DL research and prioritization of efficiency improvements to address environmental concerns. Tripp et al. [21] contribute to this reporting effort by making the BUTTER-E dataset available, which provides real-world energy consumption measurements of training fully connected neural networks across various architectures, sizes, and hardware configurations. The study reveals complex relationships between dataset size, network structure, and energy use, highlighting the significant impact of cache effects and challenging the assumption that reducing parameters or FLOPs necessarily leads to greater energy efficiency. The authors also propose an energy model that accounts for network size, computing, and memory hierarchy. In this paper, we introduce a more general energy model and evaluate it on more DL architectures, including Transformers and recurrent neural networks (RNN).

2.2 Comparative Analyses of Architectures

Comparative studies of different neural network architectures have provided valuable insights into their computational efficiency. For example, RNNs such as LSTM and GRU have traditionally been favored for sequential data processing due to their ability to capture temporal dependencies. However, the introduction of Transformers [22] has revolutionized the field, offering superior performance on tasks involving long-range dependencies. While Transformers are more computationally intensive, their parallelizable architecture has made them the dominant choice in NLP and other sequential data processing tasks. Recent work [9, 1, 16, 14] has examined the trade-offs between these architectures in terms of computational cost, memory usage, and inference latency, but few studies have focused explicitly on energy consumption during training.

2.3 Optimization Strategies for Energy Efficiency

Efforts to reduce energy consumption in DL training have focused on both algorithmic and hardware optimizations. Techniques such as mixed-precision training [13] and gradient checkpointing [2] have been widely adopted to lower computational requirements without sacrificing model performance. On the architectural side, lightweight models such as DistilBERT [18] and MobileNet [7, 17] have been designed to balance performance and efficiency. Some of these techniques, such as distillation, still require to fully train an initial model.

2.4 Positioning of This Study

While significant progress has been made in understanding the energy demands of DL models, most studies focus on specific architectures in isolation, lacking comprehensive comparisons under a unified framework. The relationship between energy efficiency and model design choices (number of layers, attention heads, hidden dimensions, etc.) also remains poorly understood. This study addresses these gaps by systematically evaluating the energy consumption of various configurations of LSTM, GRU, and Transformer models during training. We also introduced a methodology to estimate energy consumption prior to training the model, enabling researchers to predict resource requirements and optimize design choices in advance.

Contrary to the work by Tripp et al. [21], which relies on detailed measurements from a specific hardware setup, our approach has minimal hardware dependence. Our work also expands its focus by examining RNNs and Transformers, in addition to multi-layer perceptrons (MLPs), offering a broader perspective on energy consumption across different fundamental neural network architectures. This allows for a more generalized understanding of energy consumption trends and facilitates more informed architectural choices in the context of energy efficiency.

3 Methodology

It is well-known that the energy consumption of computation on GPU is not proportional to computational cost, measured as a FLOP count. However, energy consumption is highly correlated with execution time which, in turn, depends on the level of parallelization of elementary operations on the GPU. A fully parallelizable operation such as the Hadamard product will take less time than a matrix multiplication, even if they require the same amount of FLOPs. Their exact execution time will depend on the number of cores on the GPU. In addition, GPUs also spend several processing cycles to manage memory, moving tensors across main memory, GPU memory and GPU registers. Memory management operations also depend on hardware characteristics of the GPU, such as memory size and cache size. Parallelization and memory management both tie any energy consumption estimate to a specific hardware platform.

Tensor manipulation frameworks such as PyTorch and TensorFlow associate elementary tensor operations (including Hadamard product and matrix multiplication) to kernels, i.e. pieces of code executed on GPU cores. At the hardware level, a training or inference pipeline consists in a sequence of kernel executions, such that the execution time of the entire pipeline can be estimated by providing an estimate for each kernel execution, in a compositional approach[1].

Our approach consists in providing a *hardware efficiency factor* (HEF) for each elementary operation. This HEF captures the behavior of a kernel both in terms of parallelization and memory management by comparing the actual compute throughput for that operation, in FLOP/s, with the maximum theoretical throughput on some GPU, achievable for a purely parallel, purely arithmetic operation. The HEF η_Θ of operation Θ is defined as follows:

$$\eta_\Theta(c) = \frac{c/t}{v_{\max}} \tag{1}$$

where c is the computational cost of the operation in FLOPs, t is its execution time in seconds and $v_{\max}$ is the maximum theoretical throughput (or velocity) of the GPU in FLOP/s.

The HEF quantity relates execution time and computational cost in a reliable way. The energy consumption e of training some arbitrary model can then be estimated via a sum over all elementary operations needed to train a single batch, scaled by a hardware-dependent factor h such that an execution time of 1 s gives an energy consumption of h Wh. The general expression for e is as follows:

[1] Compilation techniques for deep learning pipelines have been recently introduced in the Accelerated Linear Algebra (XLA) compiler. For simplicity, we assume that pipelines are not compiled.

$$e \approx h + \sum_{i=1}^{n} h_{\Theta_i} \cdot t_{\Theta_i}(c_i) \tag{2}$$

where

$$t_{\Theta_i}(c_i) = \frac{c_i}{v_{\max} \cdot \eta_{\Theta_i}(c_i)} \tag{3}$$

and t_{Θ_i} is the duration (time) of the i-th operation, which itself is a function of its workload c_i. h_{Θ_i} is the scaling constant (or elasticity) of energy with respect to that operation's duration. It measure the increase in total energy (all else equal) due to the increase in operation Θ_i .

The main challenge to properly estimate e is thus to find an expression for $\eta_{\Theta_1}, \eta_{\Theta_2}, \ldots, \eta_{\Theta_n}$ that does not depend on t.

We empirically found that the HEF of most elementary operations follows a power law that only depends on their FLOP count, with the following parametric expression:

$$\eta_\Theta(c) \approx \eta_{\max}\left(1 - e^{-kc^\alpha}\right) \tag{4}$$

where $\eta_{\max}$ is the maximum hardware efficiency factor (saturation limit) for operation Θ, k is a decay rate, determining how quickly the hardware approaches its maximum efficiency as compute increases, and α is an additional law parameter to better fit observations. The exponential decay with a saturation offset effectively models hardware efficiency, capturing the rapid initial improvements in resource utilization followed by diminishing returns as physical or architectural limits are approached (e.g., memory bandwidth or thermal constraints) [24, 23, 8].

For transformer layer. Figure 1 and Table 1 give examples of power laws for operations inside a Transformer layer, as defined by Vaswani [22]. Attention between matrices Q, K and V can be calculated in three steps:

$$\Theta_{\texttt{proj}} \qquad Q', K', V' \leftarrow QW^Q, KW^K, VW^V \tag{5}$$

$$\Theta_{\texttt{score}} \qquad A \leftarrow \text{softmax}(\frac{Q'(K')^T}{\sqrt{d}}) \tag{6}$$

$$\Theta_{\texttt{mul}} \qquad B \leftarrow AV' \tag{7}$$

where Q', K', V', A, B are intermediary results stored in memory. The figure shows power laws for $\eta_{\Theta_{\texttt{proj}}}$, $\eta_{\Theta_{\texttt{score}}}$ and $\eta_{\Theta_{\texttt{mul}}}$.

All operations saturate at a level below 100%, which captures the fact that memory management introduces a significant overhead with respect to purely arithmetic operations. It is also clear from the figure that different hardware platforms achieve varying efficiency levels. An A100 80GB GPU peaks at 66% efficiency on matrix multiplication whereas a GeForce RTX 2080 Ti is above 78%. Note that the more recent A100 GPU would still be faster because its maximum throughput is significantly higher (156 TFLOP/s vs. 13.45 TFLOP/s).

Table 1. Fitted efficiency parameters $(\eta_{\max}, k, \alpha)$ for Transformer layer operations across A100 and RTX 2080 Ti GPUs.

Component	GPU	$\eta_{\max}$	k	α
$\Theta_{\texttt{mul}}$	A100	65.79	9.25	0.81
$\Theta_{\texttt{score}}$	A100	55.60	8.24	0.73
$\Theta_{\texttt{proj}}$	A100	68.22	8.50	0.70
$\Theta_{\texttt{mul}}$	RTX 2080 Ti	78.55	21.83	0.56
$\Theta_{\texttt{score}}$	RTX 2080 Ti	71.33	14.67	0.57
$\Theta_{\texttt{proj}}$	RTX 2080 Ti	81.35	14.28	0.47

In the above example, $\Theta_{\texttt{proj}}$ and $\Theta_{\texttt{out}}$ can be considered elementary but $\Theta_{\texttt{score}}$ is not. It itself involves a projection, where $\eta_{\Theta_{\texttt{proj}}}$ may be reused to estimate its energy consumption. The calculation may be further decomposed into a sequence of four operations:

$$\Theta_{\texttt{T}} \qquad A \leftarrow K^T \tag{8}$$

$$\Theta_{\texttt{proj}} \qquad B \leftarrow QA \tag{9}$$

$$\Theta_{\texttt{div}} \qquad C \leftarrow B/\sqrt{d} \tag{10}$$

$$\Theta_{\texttt{softmax}} \qquad D \leftarrow \text{softmax}(C) \tag{11}$$

where d is the number of columns in Q, K and V, and A, B, C, D are intermediate results stored in memory. To estimate the energy consumption of the full QKV operation, Equation 2 requires a HEF law for five operations: $\eta_{\Theta_{\texttt{proj}}}$, $\eta_{\Theta_{\texttt{mul}}}$, $\eta_{\Theta_{\texttt{T}}}$, , $\eta_{\Theta_{\texttt{div}}}$ and $\eta_{\Theta_{\texttt{softmax}}}$. Feed-forward network layers, also present in the Transformer architecture, also consist of matrix multiplication on which $\eta_{\Theta_{\texttt{proj}}}$ applies.

Strictly speaking, the transpose operation yields no FLOP. We however adopt a generalized definition of FLOPs, which includes pure memory management operations. Computing the transpose of a matrix of size $d_1 \times d_2$ would take $d_1 d_2$ generalized FLOPs. Other FLOP counts are defined as usual. For instance, multiplying two matrices of size $d_1 \times d_2$ and $d_2 \times d_3$ would take $d_1 (d_2)^2 d_3$ FLOPs. The calculation for other operations is given in Appendix.

For LSTM and Gru layer. For LSTM and GRU architectures, the primary elementary operations revolve around their gating mechanisms, which manage information flow within the network. These key elementary operations can be expressed as follows:

$$\Theta_{\texttt{gates}} \qquad E \leftarrow h \cdot W_{hh}^T + x\_t \cdot W_{ih}^T \tag{12}$$

$$\Theta_{\texttt{softmax}} \qquad F \leftarrow \text{softmax}(E) \tag{13}$$

$$\Theta_{\texttt{cell update}} \qquad G \leftarrow \text{update}(F) \tag{14}$$

$$\Theta_{\texttt{hidden update}} \qquad H \leftarrow \text{update}(G) \tag{15}$$

Here, intermediate results E, F, G, H are temporarily stored in memory during computation. Given input data $x \in \mathbb{R}^{B_s \times T \times d}$, with B_s representing the batch size, T time steps and d the input dimensionality, and h representing the hidden dimension of the model, the dimensions of weight matrices are $W_{hh} \in \mathbb{R}^{h \times 4h}$ and $W_{ih} \in \mathbb{R}^{d \times 4h}$. The total computational complexity in terms of floating-point operations (FLOPs) per time step for the gating operations Θ_gates is $2 \times B_s \times (h \times 4h) + 2 \times B_s \times (d \times 4h)$. The softmax operation involves $4 \times B_s \times h$ flops one for each gate. The updating operations take $2 \times B_s \times h$ flops. Table 2 shows the hardware efficiency trends for various elementary operations across RNN components (Activations, Cell Update, Hidden Update, and Gates) executed on the NVIDIA A100 80GB PCIe GPU. The fitted curves (red crosses see figure 5 in appendix), captures how efficiency improves with increasing workload. As expected, efficiency saturates at higher compute loads, with operations like Gates achieving substantially higher peak efficiency (up to 67%) due to their heavier computational demands. In contrast, operations like Cell Update and Activations exhibit lower peak efficiencies (5–10%), likely due to their smaller or less parallel workloads.

Table 2. Fitted efficiency parameters $(\eta_{\max}, k, \alpha)$ for LSTM operations on A100 GPU.

Component	$\eta_{\max}$	k	α
Activations	0.10	12164.49	1.00
Cell Update	0.05	1986.55	1.00
Hidden Update	0.09	1780.62	1.00
Gates	67.48	8.14	0.75

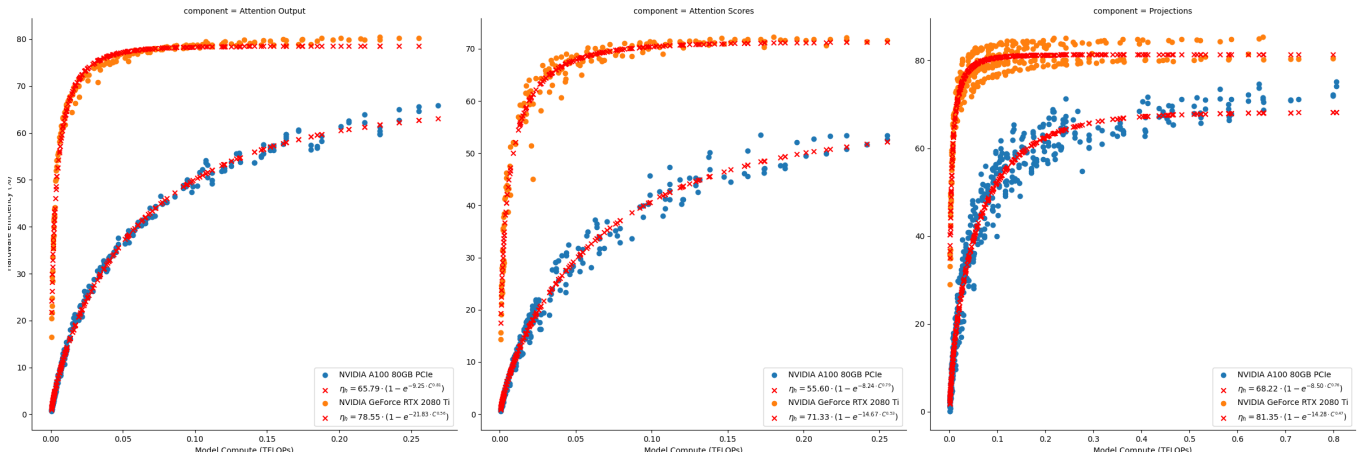

Figure 1. Hardware efficiency factors of elementary operations in transformer layer.

For a given hardware platform, HEF power law coefficients must be calculated only once per operation. Once a HEF exists for the usual elementary operations, the energy consumption of any deep learning model can be estimated via Equation 2, at design time. In the next section, we evaluate the generality of our approach on Transformers and other well-known model architectures.

4 Experiments

4.1 Learning Task and Architectures

In our experiments, we used simulated sequential data to systematically evaluate the performance and efficiency of various models under different conditions. We varied key parameters such as the number of samples, the sequence length (128 to 1024), and the dimensionality of the input data (64 to 640). By adjusting the number of samples, we explored how the models scale with increasing dataset size. Varying the dimensionality of the input enabled us to evaluate how well the models manage increased feature complexity. For the transformers based architectures, we used the pre-trained GPT-2 and BERT models, sourced from the Hugging Face Transformers library. For each architecture, we varied several key hyperparameters and data characteristics, including: the number of training samples (to assess scalability with increasing dataset size); the sequence length, ranging from 128 to 1024 tokens (to evaluate the capacity to process and learn from varying context lengths); and the dimensionality of the input data, spanning from 64 to 640 features. Furthermore, we explored the impact of internal model configurations by adjusting the number of layers (2, 4, 6, and 12), the embedding dimensionality (64 to 640), and the number of attention heads (2, 4, and 8). We trained over 2500 transformer models with number of parameters varying from 3 millions to 300 millions and 2000 RNNs (LSTM and GRU) of size varying from 200 thousands to 278 millions parameters. The experimental data sets and codes are included in the supplementary materials.

4.2 Experimental Setup

Model Configurations We trained each model for 5 epochs across a diverse range of hyperparameter configurations to thoroughly evaluate their energy consumption and computational efficiency. The hyperparameter configurations included variations in batch sizes, and model-specific parameters such as the number of hidden units, layers and number of heads (see details in Table **??** in the appenix (supplementary materials)). This approach allowed us to capture how different settings influenced both the training dynamics and resource utilization.

Energy Measurement. Energy consumption during training is tracked using a combination of tools that provide detailed metrics for both GPU and CPU usage. These tools allow for precise monitoring and aggregation of energy usage across the entire training process for each model and configuration. The total energy consumption is reported in kilowatt-hours (kWh). The following tools are used:

- **NVIDIA Nsight Systems:** Provides detailed metrics on GPU power consumption, enabling fine-grained monitoring of energy usage at the GPU level.
- **CodeCarbon**[4]. Tracks the energy consumption of the entire training process, including both GPU and CPU usage. Additionally, it estimates the carbon footprint (CO_2 emissions) associated with the energy consumed.
- **psutil**. Monitors system-level CPU utilization, memory usage, and energy consumption, offering insights into overall system resource usage during training.
- **GPUtil**. Tracks GPU utilization and power consumption specifically for NVIDIA GPUs. This tool provides fine-grained measurements that help attribute energy usage directly to GPU operations.

These tools collectively ensure that energy consumption and hardware metrics are measured accurately and comprehensively, enabling robust analysis of the computational cost associated with different training configurations.

Used Hardware The experiments are conducted on an NVIDIA A100 GPU with 80GB memory and GeForce RTX 2080 Ti. The energy consumption is measured for both the GPU, CPU and RAM to provide a comprehensive analysis. For each configuration, energy consumption was recorded over 5 epochs and averaged. This methodology ensures a consistent and fair comparison across models while providing detailed insights into the factors influencing energy consumption.

4.3 Results

For these experiments, we initially estimated the duration t_{Θ_i} of each elementary operation within the model using equations (4) and (3).

Subsequently, these estimated durations t_{Θ_i} served as input features for our proposed energy consumption model, as described by equation (2). To enhance numerical stability and interpretability, durations are expressed in mega-seconds (10^6 seconds) and the energy in joules (1 Wh is equivalent to 3,600 joules).

Energy consumption per elementary operation. Figure 2 illustrates the relationship between operation duration and energy consumption for different attention-related components across two Transformer models: GPT and BERT. The plots reveal a consistent positive correlation between duration and energy consumption, particularly for Attention Scores and Final Projection operations, across both models. Notably, BERT shows a denser clustering at shorter durations, suggesting generally faster execution times compared to GPT. However, both models exhibit similar energy scaling behavior, with QKV Projections consistently consuming less energy relative to the other components. The lines smooth out fluctuations, confirming that longer operation durations lead to higher energy usage, albeit with varying slopes depending on the component and model architecture.

Energy estimation model for Transformers. The experimental results for Transformer architectures (Table 3 and Figure 3) confirm that our proposed model accurately predicts energy consumption, with a high coefficient of determination ($R^2 = 0.93$). Specifically, the estimated constant term (23.8962 joules) provides a clear baseline for energy use independent of computational operations. Among the analyzed operations, the attention score computation duration ($t_{\Theta_{\text{score}}}$) has the most significant positive impact, increasing energy usage substantially by approximately 0.4743 joules per additional megasecond. In comparison, projection computations ($t_{\Theta_{\text{Projections}}}$) moderately elevate energy consumption by about 0.1089 joules per megasecond increase. Interestingly, the duration of matrix multiplication operations ($t_{\Theta_{\text{mul}}}$) negatively influences energy usage, decreasing consumption by about 0.1029 joules per megasecond. This negative correlation suggests that extended durations in multiplication operations might correspond to scenarios involving optimized computational resources or hardware efficiencies, effectively moderating overall energy demands.

Table 3. Estimated coefficients for energy consumption of Transformer models (BERT and GPT). The last two columns give the lower and upper bounds of the 95% confidence intervals. With a coefficient of determination $R^2 = 0.93$. Durations are expressed in 10^6 seconds and the energy in joules.

Operation	Estimate	Std. Error	$\text{CI}_{0.025}$	$\text{CI}_{0.975}$
constant (h)	23.8962	0.672	22.579	25.214
$t_{\Theta_{\text{Projections}}}$	0.1089	0.011	0.087	0.131
$t_{\Theta_{\text{score}}}$	0.4743	0.006	0.463	0.486
$t_{\Theta_{\text{mul}}}$	-0.1029	0.036	-0.173	-0.033

Energy estimation model for RNNs. We also tested our method on LSTM and GRU architectures. Initially, we focused on the core gating operations and hidden-state updates. This initial model (Table 5) had a moderate fit (R^2) = 0.83), suggesting that these operations explain a significant portion of energy consumption, but accuracy could be improved. We observed that each additional megasecond spent on gating operations reduced energy consumption by approximately 1.21 Wh, suggesting efficient computational characteristics of gating processes. Conversely, hidden-state updates significantly increased energy usage by about 4041.57 Wh per megasecond, highlighting this step as particularly energy-intensive. Incorporating FLOPs and the number of parameters improved accuracy, raising (R^2) to 0.907. As detailed in the second part of Table 5, in this enhanced model, the negative impact of gating operations became even more pronounced, reducing energy consumption by 11.59 Wh per mega-second. Hidden-state updates continued to be strongly energy-intensive, increasing consumption dramatically by 23,810 Wh per mega-second. Moreover, the number of parameters was positively correlated with energy usage, with each additional million parameters increasing consumption by about 0.1173 Wh, whereas an increase in FLOPs slightly reduced energy use (by approximately 0.1181 Wh per tera-FLOP), likely reflecting hardware efficiency at higher computational workloads.

Table 4. Estimated coefficients, standard errors, and 95% confidence intervals for RNNs (LSTM and GRU). With a coefficient of determination $R^2 = 0.83$. Durations are expressed in 10^6 seconds and the energy in Wh.

Operation	Estimate	Std. Error	$\text{CI}_{0.025}$	$\text{CI}_{0.975}$
constant (h)	0.3601	0.015	0.330	0.391
$t_{\Theta_{\texttt{gates}}}$	-1.2099	0.196	-1.596	-0.823
$t_{\Theta_{\texttt{update}}}$	4041.5679	410.269	3234.431	4848.705

Table 5. Estimated coefficients, standard errors, and 95% confidence intervals for RNNs (LSTM and GRU). With a coefficient of determination $R^2 = 0.907$. Durations are expressed in 10^6 seconds, parameters in millions, flops in tera (10^{12} flops) and the energy in Wh.

Operation	Estimate	Std. Error	$\text{CI}_{0.025}$	$\text{CI}_{0.975}$
constant (h)	0.2228	0.013	0.197	0.248
$t_{\Theta_{\texttt{gates}}}$	-11.5928	0.436	-12.450	-10.736
$t_{\Theta_{\texttt{update}}}$	2.381e+04	741.694	2.24e+04	2.53e+04
Fops	-0.1181	0.003	-0.123	-0.113
nb_params	0.1173	0.003	0.112	0.123

Model generalization across GPU architectures. Figure 4 demonstrates the robustness and generalizability of our proposed energy estimation method across different GPU architectures. In this experiment, we evaluated the model using two distinct GPUs: the NVIDIA A100 80GB PCIe and the NVIDIA GeForce RTX 2080 Ti. The plot compares predicted total energy consumption against actual measured values, with each point representing a single model inference. The model achieves an R^2 score of 0.98 on the test set, indicating that it explains 98% of the variance in energy consumption across both devices. This high level of accuracy across heterogeneous hardware platforms highlights the method's ability to capture fundamental computational behaviors rather than overfitting to a specific architecture. These results validate the portability and scalability of our approach for estimating energy consumption in real-world, multi-device settings.

5 Discussion

5.1 Interpretation

The results of our experiments demonstrate that the proposed energy estimation model is both accurate and generalizable across diverse neural network architectures and hardware platforms. For Transformer architectures, the model captured key elements of energy use, most notably the attention score computation as reflected by high explanatory power ($R^2 = 0.93$). Similarly, for RNN architectures, the model effectively identified hidden-state updates as the dominant contributor to energy consumption, and incorporating additional features like FLOPs and parameter count further improved accuracy ($R^2 = 0.907$). Importantly, our method remained consistent across

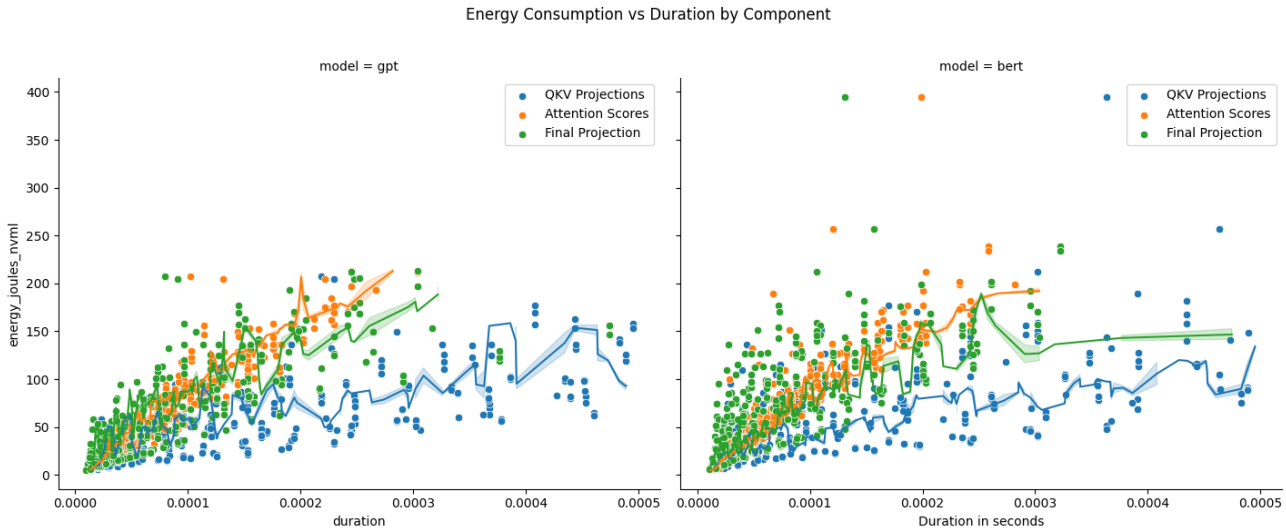

Figure 2. Plot of energy consumption versus operation duration for key Transformer components, with moving average trend lines. The data is shown separately for GPT and BERT models. The trend lines highlight the average energy scaling behavior as operation duration increases.

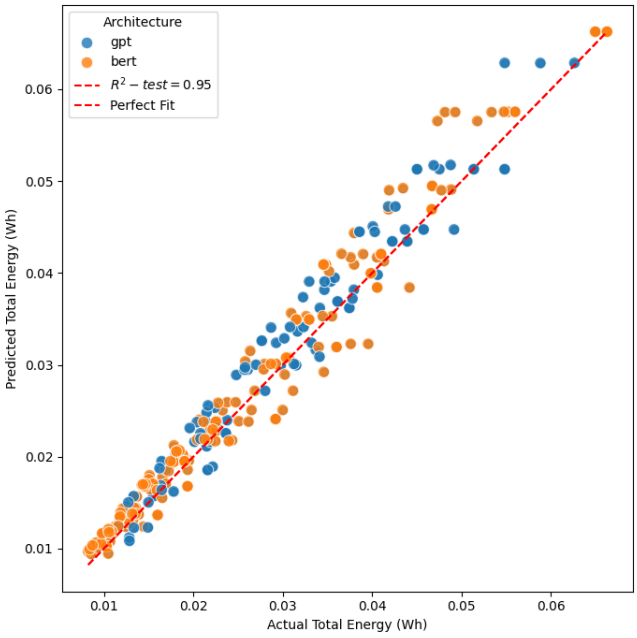

Figure 3. Quality of energy consumption estimates.

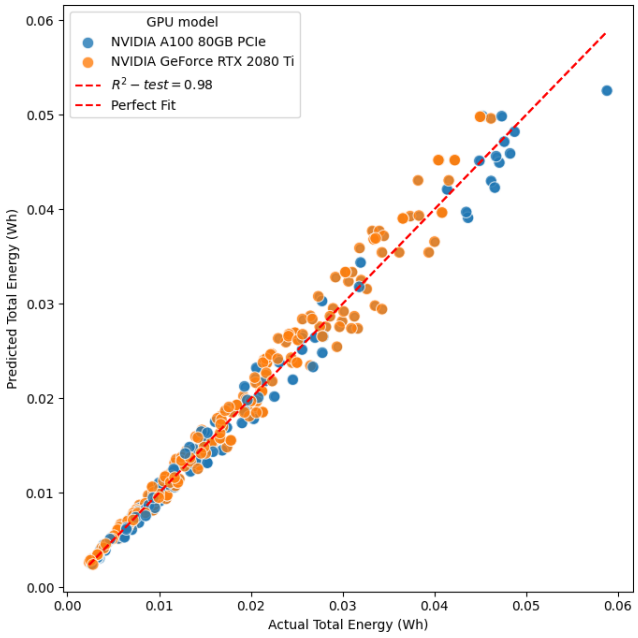

Figure 4. Quality of energy consumption estimates by GPU Model.

different GPU architectures, when tested on both the NVIDIA A100 and RTX 2080 Ti platforms. This confirms that the model captures underlying patterns in computational behavior rather than hardware-specific effects. These findings suggest that the method could be applied reliably in various real-world deployment settings, offering a lightweight and scalable tool to estimate energy consumption without the need for intrusive hardware measurements.

5.2 *Application: Energy-Aware Model Selection*

This section presents a practical use of our energy estimation method to support model selection based on energy efficiency. We compare two Transformer configurations on a GPU (NVIDIA A100) with maximum throughput of 156 TFLOPs/s. Rather than benchmarking, our method uses architectural parameters to estimate energy consumption.

Hardware and input settings. All the following steps are encapsulated in modular functions, allowing practitioners to simply define their model configuration and workload parameters (batch size, sequence length, layers, heads and embedding dimensionality d_{model}). Once these inputs are provided, the system automatically computes the estimated energy consumption for a single training epoch.

- **Max. Throughput:** $v_{\max} = 156 \times 10^{12}$ FLOPs/s

- **Batch Size:** 64
- **Sequence Length:** 320

Model configurations. Let us compare two GPT models.

- **Model A:** 6 layers, $d_{\text{model}} = 512$, 8 attention heads
- **Model B:** 12 layers, $d_{\text{model}} = 768$, 12 attention heads

Step 1: FLOPs estimation

Using the analytical formulas for elementary operations described in Section 3, we compute the number of floating-point operations (FLOPs) required for each component of the Transformer architecture. Table 6 gives the computational load associated with Transformer operations: projections, attention score computation, and output operations in this example

Table 6. FLOPs per operation for Model A and Model B under batch size 64 and sequence length 320.

Operation	Model A FLOPs	Model B FLOPs
QKV Projections	6.44×10^{10}	3.43×10^{11}
Final Projection	4.30×10^{10}	2.29×10^{11}
Attention Scores	6.74×10^{9}	2.69×10^{10}
Attention Output	6.74×10^{9}	2.69×10^{10}
Total Projections	1.07×10^{11}	5.72×10^{11}
Total Scores	6.74×10^{9}	2.69×10^{10}
Total Output	6.74×10^{9}	2.69×10^{10}

Step 2: Hardware efficiency and duration estimation

The hardware efficiency for each operation is computed using Equation (4). with fitted parameters (see figure 1) :

Table 7. Fitted efficiency parameters $(\eta_{\max}, k, \alpha)$ for each elementary operation.

Operation	$\eta_{\max}$	k	α
Projections	68.43	8.14	0.754
Attention Scores	55.70	8.43	0.800
Attention Output	67.10	8.49	0.794

Table 8. FLOPs (in teraflops), efficiency values $\eta_\Theta(c)$, and operation durations $t_\Theta(c)$ (in mega-seconds) for each model and operation.

Operation	Model	FLOPs (TF)	$\eta_\Theta(c)$	$t_\Theta(c)$ (10^6-s)
Projections	A	0.1074	35.45	19.4
Attention Scores	A	0.00674	5.15	0.76
Attention Output	A	0.00674	6.68	0.64
Projections	B	0.572	57.65	63.6
Attention Scores	B	0.0269	11.98	1.23
Attention Output	B	0.0269	15.37	1.12

Step 3: Energy Estimation

We now apply the regression model with coefficients from table 3:

$$E = 23.8962 + 0.1089 \cdot t_{\text{Proj}} + 0.4743 \cdot t_{\text{Score}} - 0.1029 \cdot t_{\text{Mul}} \quad (16)$$

$$E_A = 23.8962 + 0.1089 \cdot 19.4 + 0.4743 \cdot 0.76 - 0.1029 \cdot 0.64 \approx \textbf{26.30 Joules}$$

$$E_B = 23.8962 + 0.1089 \cdot 63.6 + 0.4743 \cdot 1.23 - 0.1029 \cdot 1.12 \approx \textbf{31.28 Joules}$$

Under identical input conditions, Model B is estimated to consume **19% more energy per epoch** than Model A. Over one million epochs, this difference would accumulate to more than **1.3 kWh**, which is significant in battery-powered or high-throughput systems. This example demonstrates how our method enables early-stage architecture comparisons and energy-aware deployment planning without executing the models on hardware.

5.3 Limitations and Future Directions

The main limitation of our study is that all experiments were conducted using single-GPU setups. While this approach provides valuable insights into energy consumption for a controlled environment, it does not fully capture the complexities and efficiencies of training modern state-of-the-art models, which often rely on multiple GPUs or TPUs operating in parallel. These large-scale setups benefit from optimized hardware utilization, distributed computation, and lower energy consumption per unit of computation due to specialized accelerators. Consequently, our results may not fully reflect the energy efficiency or scalability of these systems in real-world, multi-device training scenarios. Future work could extend the experiments to distributed environments with multiple GPUs or TPUs [14].

6 Conclusion

In this work, we proposed and validated a method to estimate the energy consumption of machine learning models using scaling laws. Our analysis revealed that while energy consumption remains relatively stable across layers for RNN models it varies significantly for Transformer-based architectures, highlighting the complexity of these models. Overall, this study provides a practical framework for estimating energy consumption before training, offering valuable insights for optimizing model design and promoting energy-efficient machine learning practices. The proposed method can serve as a useful tool for researchers and practitioners aiming to balance model performance with environmental sustainability. Future work could expand on this study by validating the proposed energy estimation method in distributed training environments, where factors such as inter-node communication, load balancing, and data sharding significantly impact energy usage. Incorporating additional aspects such as parallel efficiency, memory bandwidth limitations, and mixed precision training would further enhance the accuracy and generalizability of the model. Additionally, we plan to apply and adapt our method to emerging architectures that are designed for greater efficiency, such as mixture-of-experts models, which introduce dynamic sparsity and routing mechanisms that pose new challenges for precise energy estimation.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[3] B. Cottier, R. Rahman, L. Fattorini, N. Maslej, and D. Owen. The rising costs of training frontier ai models. *arXiv preprint arXiv:2405.21015*, 2024.

[4] B. Courty, V. Schmidt, S. Luccioni, Goyal-Kamal, MarionCoutarel, B. Feld, J. Lecourt, LiamConnell, A. Saboni, Inimaz, supatomic, M. Léval, L. Blanche, A. Cruveiller, ouminasara, F. Zhao, A. Joshi, A. Bogroff, H. de Lavoreille, N. Laskaris, E. Abati, D. Blank, Z. Wang, A. Catovic, M. Alencon, Michał Stęchły, C. Bauer, L. O. N. de Araújo,

JPW, and MinervaBooks. mlco2/codecarbon: v2.4.1, May 2024. URL https://doi.org/10.5281/zenodo.11171501.

[5] A. Faiz, S. Kaneda, R. Wang, R. C. Osi, P. Sharma, F. Chen, and L. Jiang. LLMCarbon: Modeling the end-to-end carbon footprint of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=aIok3ZD9to.

[6] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248): 1–43, 2020.

[7] A. G. Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[8] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[9] J. Kaplan, S. McCandlish, T. H. OpenAI, T. B. B. OpenAI, B. C. OpenAI, R. C. OpenAI, S. G. OpenAI, A. R. OpenAI, J. W. OpenAI, and D. A. OpenAI. Scaling laws for neural language models, 2020.

[10] A. S. Luccioni, S. Viguier, and A.-L. Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253):1–15, 2023.

[11] S. Luccioni, Y. Jernite, and E. Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 85–99, 2024.

[12] V. Mehlin, S. Schacht, and C. Lanquillon. Towards energy-efficient deep learning: An overview of energy-efficient approaches along the deep learning lifecycle. *arXiv preprint arXiv:2303.01980*, 2023.

[13] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[14] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

[15] C. Ogbogu, M. Abernot, C. Delacour, A. Todri-Sanial, S. Pasricha, and P. P. Pande. Energy-efficient machine learning acceleration: From technologies to circuits and systems. In *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–8, 2023. doi: 10.1109/ISLPED58423.2023.10244360.

[16] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[17] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler. vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.

[18] V. Sanh. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[19] D. So, Q. Le, and C. Liang. The evolved transformer. In *International conference on machine learning*, pages 5877–5886. PMLR, 2019.

[20] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13693–13696, 2020.

[21] C. E. Tripp, J. Perr-Sauer, J. Gafur, A. Nag, A. Purkayastha, S. Zisman, and E. A. Bensen. Measuring the energy consumption and efficiency of deep neural networks: An empirical analysis and design recommendations. *arXiv preprint arXiv:2403.08151*, 2024.

[22] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[23] J. Végh. How amdahl's law limits the performance of large artificial neural networks: why the functionality of full-scale brain simulation on processor-based simulators is limited. *Brain informatics*, 6(1):4, 2019.

[24] S. Williams, A. Waterman, and D. Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[25] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

[26] T. Yarally, L. Cruz, D. Feitosa, J. Sallou, and A. Van Deursen. Uncovering energy-efficient practices in deep learning training: Preliminary steps towards green ai. In *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*, pages 25–36. IEEE, 2023.

[27] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Q. V. Le, J. Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.