Modeling Energy Consumption in Deep Learning Architectures Using Power Laws

Mansour Zoubeirou a Mayaki[a,*] **and Victor Charpenay**[a]

[a]Mines Saint-Etienne, Université Clermont Auvergne, Clermont Auvergne INP, CNRS, LIMOS

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 power 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 [26, 13, 27]. 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 [5]. 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 a pressing issue. 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: Long Short-Term Memory units (LSTMs), Gated Recurrent Units (GRUs), 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 Natural Language Processing (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, FLOPs count and hardware efficiency, which quantifies actual throughput relative to maximum throughput. 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 our work can be summarized as follows:

- 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.
- We release a dataset on the real-world energy consumption of Transformer-based models (BERT and GPT-2), trained on two GPU architectures: NVIDIA A100 (80GB) and GeForce RTX 2080 Ti [1]. Despite the scarcity of such data in the field, our dataset includes over 98,000 runs across 24,000 unique configurations, varying in layers , embedding size, attention heads, and sequence lengths. Each run includes energy usage, training time, and other metrics, collected using CodeCarbon. This resource supports benchmarking and research on energy-efficient machine learning.

2 Related Work

This section reviews prior work in three key areas: energy consumption of DL models, computational efficiency, and optimization strategies for energy-efficient training [26, 7, 3, 12, 11, 27, 16].

2.1 Energy Consumption of Deep Learning 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. [21] quantified the carbon footprint of designing and training NLP pipelines, showing that such pipelines can emit

* Corresponding Author. Email: mansour.zoubeirou-a-mayaki@univ-lyon2.fr

[1] https://github.com/MansMayaki/MLEnergyConsumption

as much greenhouse gases as an American household over a year. The LLMCarbon framework by Faiz et al. [6] 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. [17] 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 [28].

Carbon emissions remain hard to estimate, though. A case study on the Evolved Transformer [20] revealed an 88x overestimation in prior 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. [22] 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.

2.2 *Computational Efficiency of Architectures*

Comparative analyses 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 [23] 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 [10, 1, 17, 15] 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 [14] 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 [19] and MobileNet [8, 18] 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 introduce 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. [22], 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, 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, traditionally measured as a FLOPs 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[2].

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 the GPU, achievable for a purely parallel, purely arithmetic operation. The HEF η_Θ of operation Θ is defined as follows:

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

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.

[2] 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.

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. The general expression for e is as follows:

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

where

$$t_i = \frac{c_i}{v_{\max} \cdot \eta_{\Theta_i}} \tag{3}$$

and t_i is the duration (time) of the i-th operation, which itself is a function of its workload c_i. The constant h_i (or elasticity) scales energy with respect to that operation's duration. It measure the increase in total energy (all else equal) due to the increase in operation 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 the execution duration t.

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

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

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) [25, 24, 9].

Transformer Layer. Figure 1 and Table 1 give examples of power laws for operations inside a Transformer layer, as defined by Vaswani et al. [23]. Attention between matrices Q, K and V can be calculated in these steps:

$$\texttt{QKV Projections} \quad Q', K', V' \leftarrow QW^Q, KW^K, VW^V \tag{5}$$

$$\texttt{Attention Scores} \quad A \leftarrow \text{softmax}(\frac{Q'(K')^T}{\sqrt{d}}) \tag{6}$$

$$\texttt{Attention Output} \quad B \leftarrow AV' \tag{7}$$

$$\texttt{Final Projection} \quad O \leftarrow BW^O \tag{8}$$

where Q', K', V', A, B, O are intermediate tensors stored in memory, and W^Q, W^K, W^V, W^O are learned projection matrices. Figure 1 shows power laws of the *hardware efficiency factor* for each of these elementary operations.

All operations saturate at a level below 100%, reflecting the fact that memory management imposes a significant overhead compared to purely arithmetic throughput. It is also clear from the figure that different hardware platforms achieve varying efficiency levels. Across all measured Transformer layer components, the highest observed efficiency on the A100 80GB GPU is about 70.4% (Final Projection), while the GeForce RTX 2080 Ti reaches up to 81.9% (Final Projection). Despite the RTX 2080 Ti's higher peak percentage, the newer A100 remains faster in absolute terms because its maximum theoretical throughput is much greater (156 TFLOP/s vs. 13.45 TFLOP/s).

Table 1. Fitted efficiency parameters ($\eta_{\max}, k, \alpha$) for Transformer layer operations across NVIDIA A100 80GB PCIe and GeForce RTX 2080 Ti GPUs.

Component	GPU	$\eta_{\max}$	k	α
`Attention Output`	A100 80GB PCIe	66.83	8.65	0.80
`Attention Scores`	A100 80GB PCIe	56.47	8.09	0.80
`QKV Projections`	A100 80GB PCIe	69.38	10.37	0.78
`Final Projection`	A100 80GB PCIe	70.43	6.24	0.77
`Attention Output`	RTX 2080 Ti	78.55	21.77	0.56
`Attention Scores`	RTX 2080 Ti	71.42	14.25	0.52
`QKV Projections`	RTX 2080 Ti	81.45	18.94	0.52
`Final Projection`	RTX 2080 Ti	81.93	9.21	0.40

In the above example, `QKV Projections`,`Attention Output` and `Final Projection` can be considered elementary but `Attention Scores` is not. It itself involves a projection, where $\eta_{\texttt{QKV}}$ may be reused to estimate its energy consumption. The calculation may be further decomposed into a sequence of four operations:

$$\texttt{T} \quad A \leftarrow K^T \tag{9}$$

$$\texttt{proj} \quad B \leftarrow QA \tag{10}$$

$$\texttt{div} \quad C \leftarrow B/\sqrt{d} \tag{11}$$

$$\texttt{softmax} \quad D \leftarrow \text{softmax}(C) \tag{12}$$

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 attention mechanism, Equation 2 requires a HEF law for five operations: `proj`, `mul`, `T`, `div`, and `softmax`. Feed-forward network layers, also present in the Transformer architecture, also consist of matrix multiplication on which $\eta_{\texttt{proj}}$ applies.

Strictly speaking, the transpose operation yields no FLOP. We however adopt a generalized definition of FLOPs, which includes pure memory management operations. Other FLOPs counts are given in Appendix 7.3.

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:

$$\text{Hidden State Update} \quad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{13}$$

$$\text{Output Gate} \quad \mathbf{o}_t = \sigma(\text{Linear}_{d_h}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \tag{14}$$

$$\text{Cell State update} \quad \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{15}$$

$$\text{Forget Gate} \quad \mathbf{f}_t = \sigma(\text{Linear}_{d_h}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \tag{16}$$

$$\text{Input Gate} \quad \mathbf{i}_t = \sigma(\text{Linear}_{d_h}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \tag{17}$$

$$\text{Candidate Cell State} \quad \tilde{\mathbf{c}}_t = \tanh(\text{Linear}_{d_h}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \tag{18}$$

where $\odot$ denotes element-wise multiplication of vectors, t is the current timestep, and $\mathbf{h}_t$ is the outputted hidden state. $[\mathbf{x}_t, \mathbf{h}_{t-1}]$ represents the concatenation of the input vector $\mathbf{x}_t$ at time step t with the previous hidden state $\mathbf{h}_{t-1}$. $\mathbf{c}_t$ is the cell state, which carries information across time steps, and $\tilde{\mathbf{c}}_t$ is the candidate cell state that will be added to the cell state. The intermediate results $\mathbf{h}_t, \mathbf{o}_t, \mathbf{c}_t, \mathbf{f}_t, \mathbf{i}_t$ 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 d_h representing the hidden dimension of the model, the dimensions of weight matrices are $W_{d_h d_h} \in \mathbb{R}^{d_h \times 4d_h}$ and $W_{id_h} \in \mathbb{R}^{d \times 4d_h}$. All these operations are detailed in appendix 7.4. Table 2 shows the hardware efficiency

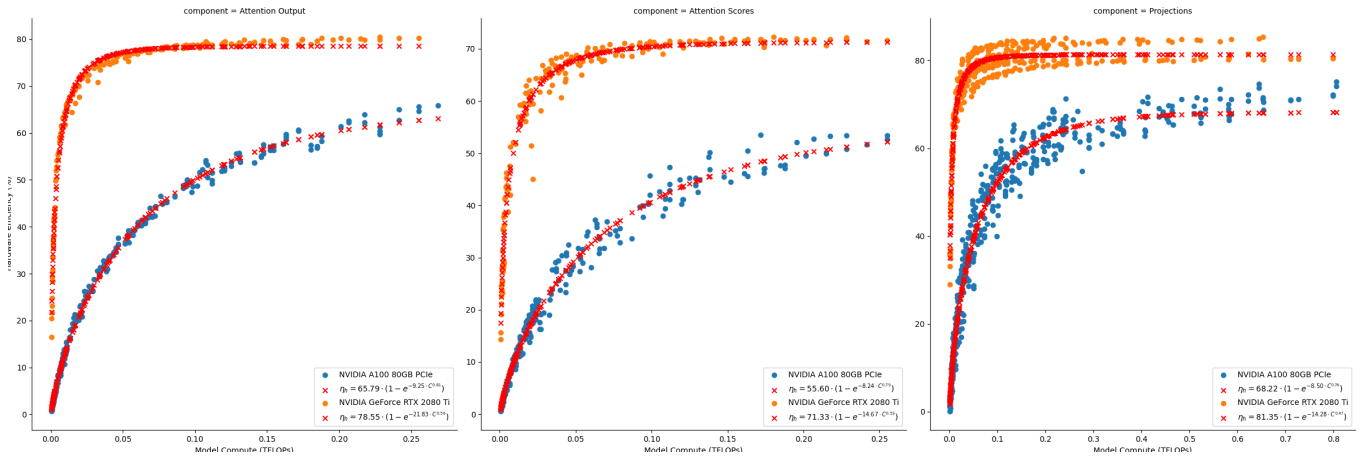

Figure 1. Hardware efficiency factors of elementary operations in transformer layer.

trends for various elementary operations across RNN components 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 80%) due to their heavier computational demands. In contrast, operations like `Cell Update` and `Activations` (σ and $tanh$) exhibit lower peak efficiencies, 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`	0.0965	17544.28	0.94
`Cell Update`	0.0554	12823.88	0.88
`Hidden Update`	0.0977	10286.70	0.89
`Compute Gates`	41.03	21.57	0.86
`Hidden Gates`	80.04	10.08	0.79

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. Note that, for a full Transformer model, one might add the feed-forward network (FFN) component; however, our experiments showed that doing so did not improve predictive accuracy. The reason is twofold: (i) FFN FLOPs are nearly a fixed multiple of the matrix multiplications we already model, making the new term highly collinear and thus redundant for regression; and (ii) in our measurement regime, runtime and energy are primarily driven by memory movement in attention (score/probability tensors and key–value traffic), not by the compute-bound FFN. Consistent with this, augmenting the model with I/O-oriented features (such as approximate bytes moved and precision/FlashAttention switches) yields larger gains than refining FFN arithmetic. We therefore retain the simpler FFN approximation and focus model capacity on I/O-sensitive components.

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. 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 available on Github [3].

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 6 in the appendix). 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

[3] https://github.com/MansMayaki/MLEnergyConsumption

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.

- 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_i of each elementary operation within the model using equations (4) and (3). Subsequently, these estimated durations t_i served as input features for our proposed energy consumption model, as described by equation (2). To enhance numerical stability and improve interpretability of regression coefficients, we expressed the duration in microseconds. This was necessary because the original duration values were very small (on the order of microseconds), which can lead to numerical issues in model fitting. Additionally, all energy measurements were converted from watt-hours (Wh) to joules (J) using the standard conversion factor: $1\ \text{Wh} = 3,600\ \text{J}$. These transformations do not affect the validity of the regression results but ensure better numerical conditioning and clarity in reporting. The details of estimation procedure is describe in appendix 7.6.

The coefficients should be interpreted with caution. Each term in the model reflects the marginal effect of its corresponding variable while holding all other variables constant. The negative coefficients from table 3 and table 9 result from the multivariate linear model, where each term reflects the marginal effect of a variable while holding others constant. Due to multicollinearity among features common in neural network operations where components are highly interdependent some coefficients may turn negative to balance overlapping effects and improve overall fit. These values do not necessarily imply that the associated operations are inherently energy-reducing; rather, they indicate how energy consumption changes when that variable increases, assuming all other factors remain fixed. Predictions from our model (2) were compared against the measurements from code carbon using the coefficient of determination (R^2). This allowed us to assess both the accuracy and the linear alignment of the model's predictions with empirical data. The estimated coefficients may also vary with dataset size because the regression is sensitive to the distribution of configurations and runtimes in the sample. Sign changes occur when correlated terms (e.g., projection and attention score FLOPs) redistribute variance between them, a common statistical effect that does not imply a reversal of the underlying physical relationship and does not affect the model's predictive performance.

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 ($R^2 = 0.96$). Specifically, the estimated constant term (3.63 joules) provides a baseline for energy use independent of computational operations. Among the analyzed components, the duration of the final projection operation ($t_{\texttt{Final Projection}}$) has the strongest positive effect, increasing energy consumption by approximately 0.56 joules per additional microsecond. Attention score computations ($t_{\texttt{score}}$) and attention output operations ($t_{\texttt{Attention Output}}$) both show consistent positive contributions of about 0.30 joules per microsecond. In contrast, QKV projections ($t_{\texttt{Projections}}$) exhibit a small but significant negative coefficient (-0.14 joules per microsecond), indicating that longer runtimes in these operations are associated with reduced incremental energy costs likely reflecting overlap with memory-bound stages or hardware-level optimizations. The final energy consumption model for transformers architectures is written as follows:

$$\begin{aligned} E_{\text{trans}} \approx & 3.6318 - 0.1377 \cdot t_{\text{qkvProj}} + 0.5637 \cdot t_{\text{Score}} \\ & + 0.3041 \cdot t_{\text{FinalProj}} + 0.3041 \cdot t_{\text{AttnOut}} \end{aligned} \tag{19}$$

Table 3. Estimated coefficients for energy consumption of Transformer models. The last two columns give the lower and upper bounds of the 95% confidence intervals. With a coefficient of determination $R^2 = 0.9584$. Durations are expressed in microseconds and the energy in joules.

Operation	Estimate	$\text{CI}_{0.025}$	$\text{CI}_{0.975}$
constant (h)	3.6318	3.4088	3.8858
$h_{\texttt{qkv Projections}}$	-0.1377	-0.1455	-0.1297
$h_{\texttt{score}}$	0.3041	0.3016	0.3065
$h_{\texttt{Final Projection}}$	0.5637	0.5396	0.5880
$h_{\texttt{Attention Output}}$	0.3041	0.3016	0.3065

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 9) had a fit (R^2) = 0.95), suggesting that these operations explain a significant portion of energy consumption, but accuracy could be improved. We observed that each additional micro-second spent on gating operations reduced energy consumption by approximately 169 Joules, suggesting efficient computational characteristics of gating processes. Each micro second in $Cell\_Update$ and $Hidden\_Update$ also increases the energy consumption respectively by 4.52×10^4 and 2.95×10^4. Conversely, activation computations reduces energy usage by about $-7.66\times10^4 J/\mu s$.

Table 4. Estimated coefficients, standard errors, and 95% confidence intervals for RNNs (LSTM and GRU). With a coefficient of determination $R^2 = 0.95$. Durations are expressed in microseconds and energy in Joules.

Operation	Estimate	Std. Error	$\text{CI}_{0.025}$	$\text{CI}_{0.975}$
constant (h)	3367.3304	8.375	3350.882	3383.779
$h_{Activations}$	-7.66×10^4	7021.058	-9.04×10^4	-6.28×10^4
h_{Cell_Update}	4.52×10^4	4221.466	3.70×10^4	5.35×10^4
h_{Gates}	169.0733	9.072	151.255	186.892
h_{Hidden_Update}	2.95×10^4	2813.017	2.39×10^4	3.50×10^4

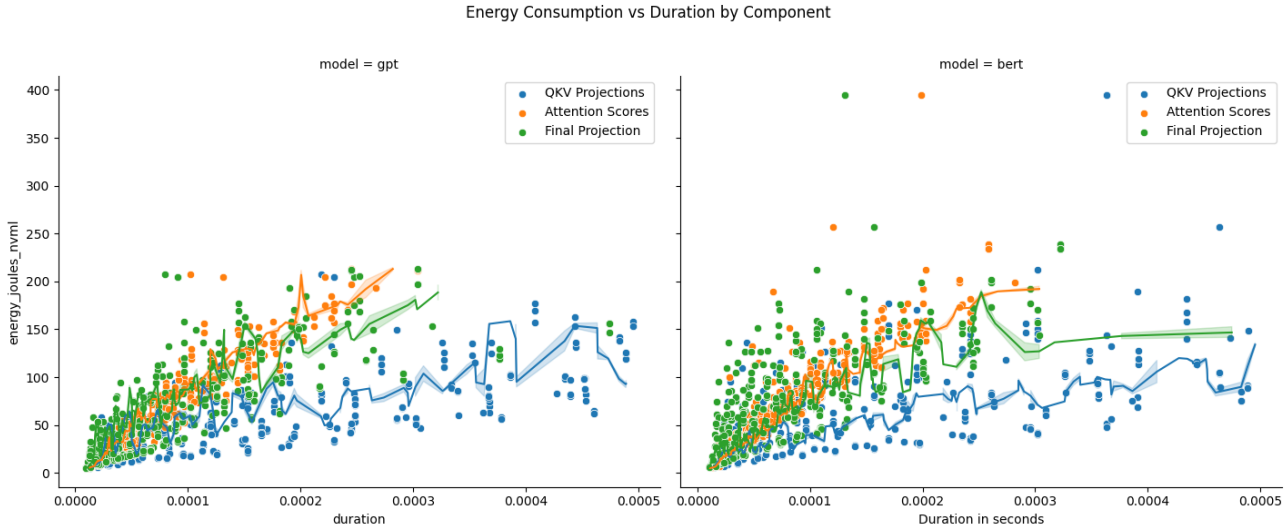

Figure 2. Plot of energy consumption versus operation duration for key Transformer components, with moving average trend lines. The data is shown separately for GPT and BERT models. The trend lines highlight the average energy scaling behavior as operation duration increases.

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.

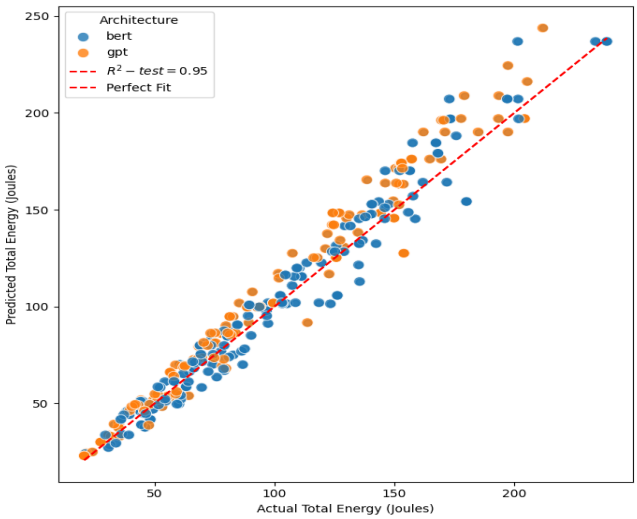

Figure 3. Quality of energy consumption estimates.

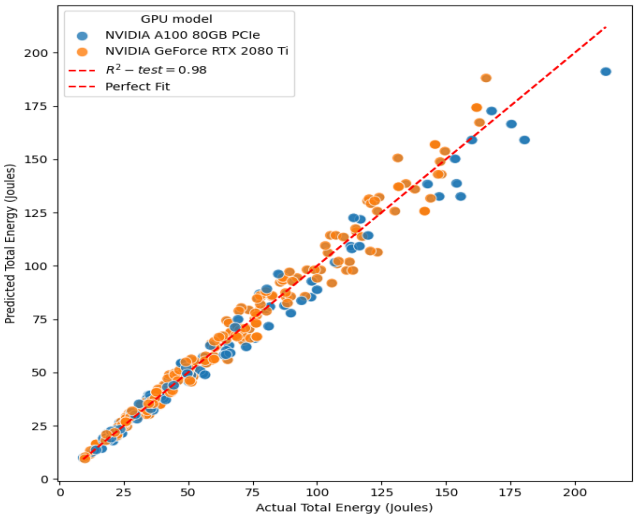

Figure 4. Quality of energy consumption estimates by GPU Model.

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 ($R^2 = 0.95$). Importantly, our method remained consistent across 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 behaviour 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 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. The third column of Table 5 gives the computational load associated with Transformer operations: projections, attention score computation, and output operations in this example.

Step 2: Hardware efficiency and duration Using the fitted parameters from Table 1, the hardware efficiency and the duration for each operation in Table 5 were computed with Equations 3 and 4.

Table 5. FLOPs (in teraflops), efficiency values $\eta_\Theta(c)$, and operation durations $t_\Theta(c)$ (in micro-seconds) for each model and operation.

Operation	Model	FLOPs (TF)	$\eta_\Theta(c)$	$t_\Theta(c)$ (μ s)
QKV Projections	A	0.0322	35.31	35.08
Attention Scores	A	0.00674	7.75	33.28
Attention output	A	0.00674	9.76	26.43
Final Projection	A	0.01074	12.19	33.87
QKV Projections	B	0.0724	51.19	108.90
Attention Scores	B	0.01	10.43	74.21
Attention output	B	0.01	13.11	59.05
Final Projection	B	0.024	21.04	88.31

Step 3: Energy Estimation We now apply Equation 19 to estimate the total energy consumption of the two Transformer configurations: $E_A \approx 36\,\text{J}$ and $E_B \approx 79\,\text{J}$. Despite being evaluated under identical input conditions, Model B is estimated to consume more than **twice energy per epoch** than Model A. While a single-epoch difference of roughly 42 joules may seem modest, the effect becomes pronounced at scale: over one million training epochs, this difference would accumulate to more than **11 kWh**. Such a gap is non-negligible in battery-powered, embedded, or high-throughput environments, where energy efficiency directly constrains deployment feasibility. The higher cost of Model B is largely attributable to its **increased dimensionality and number of layers**, which disproportionately amplify the computational burden of projection and score computations relative to Model A. Beyond highlighting the energy trade-offs between model sizes, this example illustrates how our method enables **early-stage architectural comparisons** and **energy-aware design choices** without requiring execution on actual hardware. The results in Table 10 and Table 11 in appendix show that energy consumption is largely insensitive to the number of attention heads, with only negligible variation across configurations. In contrast, the number of layers and the dimensionality have a pronounced impact, with deeper models consuming substantially more energy. This suggests that model depth and the dimensionality of the input and output of all the sub-layers are the dominant factor in transformer energy efficiency.

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 [15].

6 Conclusion

In this work, we proposed and validated a method to estimate the energy consumption of machine learning models using the elementary operations in the model architecture and power laws. The results showed a strong correlation between the estimated and measured energy consumption, with a high R^2 value, particularly for lower and mid-complexity configurations. 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.

Acknowledgements

This work was supported by the European Lighthouse to Manifest Trustworthy and Green AI (ENFIELD) project funded by the European Union's HORIZON Research and Innovation Programme. Grant agreement No: 101120657. The code and the supplementary materials for this paper is available at https://github.com/MansMayaki/MLEnergyConsumption.

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] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, J. Chen, J. Chen, J. Yuan, J. Qiu, J. Li, J. Song, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Wang, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Wang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Zhang, R. Pan, R. Wang, R. Xu, R. Zhang, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Pan, T. Wang, T. Yun, T. Pei, T. Sun, W. L. Xiao, W. Zeng, W. Zhao, W. An, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, X. Q. Li, X. Jin, X. Wang, X. Bi, X. Liu, X. Wang, X. Shen, X. Chen, X. Zhang, X. Chen, X. Nie, X. Sun, X. Wang, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yu, X. Song, X. Shan, X. Zhou, X. Yang, X. Li, X. Su, X. Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Y. Zhang, Y. Xu, Y. Xu, Y. Huang, Y. Li, Y. Zhao, Y. Sun, Y. Li, Y. Wang, Y. Yu, Y. Zheng, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Tang, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Wu, Y. Ou, Y. Zhu, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Zha, Y. Xiong, Y. Ma, Y. Yan, Y. Luo, Y. You, Y. Liu, Y. Zhou, Z. F. Wu, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Huang, Z. Zhang, Z. Xie, Z. Zhang, Z. Hao, Z. Gou, Z. Ma, Z. Yan, Z. Shao, Z. Xu, Z. Wu, Z. Zhang, Z. Li, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Gao, and Z. Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

[21] 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.

[22] 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.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

7 Appendix

7.1 Model Configurations

In our experiments, 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. This approach allowed us to capture how different settings influenced both the training dynamics and resource utilization. By standardizing the number of epochs across all models, we ensured a fair comparison, isolating the impact of architectural differences and hyperparameter choices on energy efficiency and computational performance. These evaluations provided insights into the trade-offs between model accuracy, training time, and resource consumption, offering a comprehensive perspective on the practical implications of deploying these models in real-world scenarios. The hyperparameters explored include:

Table 6. Experimental parameters and their ranges

Parameter	Range and Purpose
Batch Sizes	64 to 640
Number of Layers	2 to 12, to evaluate the impact of model depth on capturing data complexity.
Hidden Dimensions (LSTM/GRU)	64 to 1280, to assess the influence of hidden state size on energy consumption and hardware usage.
d_{model} **(Transformer)**	128 to 1280, corresponding to the dimensionality of input embeddings and internal representations.
Number of Heads (Transformer)	2 to 12, to analyze the effects of multi-head attention on data representation.

7.2 Number of parameters in RNN models

To compute the number of parameters in LSTM and GRU models, we need to account for the weights and biases involved in the input-to-hidden and hidden-to-hidden connections, as well as the bias terms for each gate. For an LSTM with a hidden size h and an input size x, each layer has four gates: input, forget, cell, and output gates. Each gate requires a weight matrix for the input (W_x) and a weight matrix for the hidden state (W_h), along with their respective biases (b). Thus, the total number of parameters for one LSTM layer is:

$$N \approx L \cdot 4 \cdot (h \cdot x + h \cdot h + h) \tag{20}$$

For GRU models, which have three gates (reset, update, and candidate activation), the number of parameters for one layer is slightly reduced:

$$N \approx L \cdot 3 \cdot (h \cdot x + h \cdot h + h) \tag{21}$$

In both cases, if embedding layers or fully connected layers are included, their parameters should be added to the total.

7.3 Floating point operations (FLOPs) for Transformer Models

To estimate the floating-point operations (FLOPs) of a Transformer model, we calculate the computational cost of each component and aggregate it across layers and batch size (see Table 7 in the supplementary materials). For the Multi-Head Attention mechanism, FLOPs are derived from the matrix multiplications required for projecting the input into query, key, and value (QKV) spaces, computing the attention scores, and projecting the outputs. Specifically, the attention mechanism involves: (i) query, key, and value projections, (ii) attention weight computation, and (iii) the final output projection. These operations are scaled by the number of attention heads and include the feed-forward layers.

The key parameters are the model dimension (d_{model}), number of attention heads (num_attention_heads), feed-forward dimension (d_{ff}), number of layers (num_layers), input size (W), sequence length (seq_len), and batch size. For a single pass (either forward or backward) on a batch of data, FLOPs for each major sub-operation are as follows:

1. **QKV Projections**
   This consists of three matrix multiplications projecting the Transformer's input $x \in \mathbb{R}^{(\text{batch\_size}, \text{seq\_len}, d_{\text{model}})}$ into Q, K, and V matrices. Each projection uses a weight matrix of shape $(d_{\text{model}}, d_{\text{model}})$. FLOPs for a single projection (e.g., Q):

   $$2 \times (\text{batch\_size} \times \text{seq\_len}) \times d_{\text{model}} \times d_{\text{model}}$$

   Total FLOPs for all three projections:

   $$3 \times \left[2 \times (\text{batch\_size} \times \text{seq\_len}) \times d_{\text{model}} \times d_{\text{model}}\right]$$

2. **Attention Scores**
   The attention scores are computed by multiplying Q with $K^\top$. Here, $Q, K \in \mathbb{R}^{(\text{batch\_size}, \text{heads}, \text{seq\_len}, d_k)}$ with $d_k = d_{\text{model}}/\text{heads}$.
   FLOPs:

   $$2 \times \text{batch\_size} \times \text{heads} \times \text{seq\_len} \times d_k \times \text{seq\_len}$$

   Simplifying:

   $$2 \times \text{batch\_size} \times \text{seq\_len}^2 \times d_{\text{model}}$$

3. **Attention Output**
   After applying softmax, the attention weights are multiplied by V to obtain the final attention output. This multiplies a tensor of shape $(\text{batch\_size}, \text{heads}, \text{seq\_len}, \text{seq\_len})$ with $(\text{batch\_size}, \text{heads}, \text{seq\_len}, d_k)$.
   FLOPs:

   $$2 \times \text{batch\_size} \times \text{heads} \times \text{seq\_len} \times \text{seq\_len} \times d_k$$

   Which again simplifies to:

   $$2 \times \text{batch\_size} \times \text{seq\_len}^2 \times d_{\text{model}}$$

4. **Final Projection**
   The concatenated attention outputs are projected back to d_{model} using a weight matrix of shape $(d_{\text{model}}, d_{\text{model}})$.
   FLOPs:

   $$2 \times (\text{batch\_size} \times \text{seq\_len}) \times d_{\text{model}} \times d_{\text{model}}$$

5. **Feed-Forward Network (FFN)**
   Each Transformer block contains a position-wise feed-forward network consisting of two linear transformations with an activation in between (commonly GELU or ReLU). The input of shape $(\text{batch\_size}, \text{seq\_len}, d_{\text{model}})$ is first projected to an intermediate dimension d_{ff}, then projected back to d_{model}.
   The first linear layer FLOPs:

   $$2 \times (\text{batch\_size} \times \text{seq\_len}) \times d_{\text{model}} \times d_{\text{ff}}$$

   The second linear layer FLOPs:

   $$2 \times (\text{batch\_size} \times \text{seq\_len}) \times d_{\text{ff}} \times d_{\text{model}}$$

   Since both layers are of similar complexity, the total FFN FLOPs are:

   $$4 \times \text{batch\_size} \times \text{seq\_len} \times d_{\text{model}} \times d_{\text{ff}}$$

Table 7. FLOPs estimation for Transformer components (per forward or backward pass). Here, $W = \text{batch\_size} \times seq\_len$ is the total number of tokens processed in the batch.

Component	Formula
Query Projection	$\text{FLOPs}_Q = 2 \cdot W \cdot d_{\text{model}}^2$
Key Projection	$\text{FLOPs}_K = 2 \cdot W \cdot d_{\text{model}}^2$
Value Projection	$\text{FLOPs}_V = 2 \cdot W \cdot d_{\text{model}}^2$
QKV Total	$\text{FLOPs}_{\text{QKV}} = 6 \cdot W \cdot d_{\text{model}}^2$
Attention Scores	$\text{FLOPs}_{\text{att\_score}} = 2 \cdot \text{batch\_size} \cdot seq\_len^2 \cdot d_{\text{model}}$
Attention Output	$\text{FLOPs}_{\text{att\_out}} = 2 \cdot \text{batch\_size} \cdot seq\_len^2 \cdot d_{\text{model}}$
Final Projection	$\text{FLOPs}_{\text{proj}} = 2 \cdot W \cdot d_{\text{model}}^2$
Multi-Head Attention (MHA)	$\text{FLOPs}_{\text{MHA}} = \text{FLOPs}_{\text{QKV}} + \text{FLOPs}_{\text{att\_score}} + \text{FLOPs}_{\text{att\_out}} + \text{FLOPs}_{\text{proj}}$
Feed-Forward Network (FFN)	$\text{FLOPs}_{\text{FFN}} = 4 \cdot W \cdot d_{\text{model}} \cdot d_{\text{ff}}$ (two linear layers)
Single Transformer Layer	$\text{FLOPs}_{\text{layer}} = \text{FLOPs}_{\text{MHA}} + \text{FLOPs}_{\text{FFN}}$
Total Model FLOPs	$\text{FLOPs}_{\text{total}} = \text{num\_layers} \cdot (\text{FLOPs}_{\text{layer}})$

7.4 Floating Point Operations (FLOPs) in LSTM Models

To estimate the floating-point operations (FLOPs) required by a Long Short-Term Memory (LSTM) layer, we account for the computations performed in its gating mechanisms, non-linear activations, and the element-wise updates of the hidden and cell states. An LSTM cell enhances a standard RNN by introducing gating mechanisms that control information flow, thereby mitigating the vanishing and exploding gradient problems.

Each LSTM cell maintains four gates: the *input gate* (i_t), the *forget gate* (f_t), the *cell candidate* ($\tilde{c}_t$), and the *output gate* (o_t). These gates regulate which information from the current input x_t and the previous hidden state h_{t-1} should be stored, forgotten, or propagated to the next state. Mathematically, each gate involves two linear projections (input-to-hidden and hidden-to-hidden) followed by a non-linear activation. LSTMs are computed as follows:

$$
\begin{aligned}
\text{(Hidden State)} \quad & \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \\
\text{(Output Gate)} \quad & \mathbf{o}_t = \sigma(\text{Linear}_{dh}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \\
\text{(Cell State Recurrence)} \quad & \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\text{(Forget Gate)} \quad & \mathbf{f}_t = \sigma(\text{Linear}_{dh}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \\
\text{(Input Gate)} \quad & \mathbf{i}_t = \sigma(\text{Linear}_{dh}([\mathbf{x}_t, \mathbf{h}_{t-1}])) \\
\text{(Candidate Cell State)} \quad & \tilde{\mathbf{c}}_t = \tanh(\text{Linear}_{dh}([\mathbf{x}_t, \mathbf{h}_{t-1}]))
\end{aligned}
$$

Let B denote the batch size, T the sequence length, I the input dimension, and H the hidden size. The FLOPs can be broken down as follows:

1. **Hidden-to-Hidden Projections ($h_{t-1}W_{hh}^T$)**
   Each gate computes a linear transformation of the previous hidden state $h_{t-1} \in \mathbb{R}^{B \times H}$ using a weight matrix $W_{hh} \in \mathbb{R}^{H \times H}$. Since there are four gates, the total cost per timestep is:

   $$4 \times \left[2 \times (B \times H) \times H\right] = 8 \times B \times H^2$$

   For T timesteps:

   $$8 \times B \times H^2 \times T$$

2. **Input-to-Hidden Projections ($x_t W_{ih}^T$)**
   Each gate also projects the input $x_t \in \mathbb{R}^{B \times I}$ with $W_{ih} \in \mathbb{R}^{I \times H}$. FLOPs per timestep:

   $$4 \times \left[2 \times (B \times I) \times H\right] = 8 \times B \times I \times H$$

   For T timesteps:

   $$8 \times B \times I \times H \times T$$

3. **Non-linear Activations**
   Each gate output passes through a non-linearity: sigmoid functions for i_t, f_t, o_t and $\tanh$ for $\tilde{c}_t$. We approximate each element-wise activation as 1 FLOP. FLOPs per timestep:

   $$4 \times (B \times H)$$

   For T timesteps:

   $$4 \times B \times H \times T$$

4. **Cell State Update**
   The cell state c_t is updated via:

   $$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

   This requires two element-wise multiplications and one addition per element. FLOPs per timestep:

   $$2 \times (B \times H) + (B \times H) = 3 \times B \times H$$

   For T timesteps:

   $$3 \times B \times H \times T$$

5. **Hidden State Update**
   The hidden state is computed as:

   $$h_t = o_t \odot \tanh(c_t)$$

   This involves one $\tanh$ activation and one element-wise multiplication per element. FLOPs per timestep:

   $$(B \times H) + (B \times H) = 2 \times B \times H$$

   For T timesteps:

   $$2 \times B \times H \times T$$

Total LSTM FLOPs per layer:

$$
\begin{aligned}
&\left[8BH^2 + 8BIH + 4BH + 3BH + 2BH\right] \times T \\
&= \left[8BH^2 + 8BIH + 9BH\right] \times T
\end{aligned}
$$

This expression highlights that the computational cost of an LSTM layer is dominated by the matrix multiplications ($O(H^2)$ and $O(IH)$), while element-wise activations and state updates contribute comparatively less. Consequently, the hidden size H and input dimension I are the primary factors driving FLOP complexity, with the batch size B and sequence length T scaling the total cost linearly.

7.5 Hardware Efficiency Model.

Figures 6 illustrates the hardware efficiency scaling of three core Transformer operations Attention Output, Attention Scores, and Projections across two GPU architectures: NVIDIA A100 and RTX 2080 Ti. Each subplot presents empirical efficiency measurements (dots) plotted against operation size in TFLOPs, with fitted curves showing how efficiency approaches saturation as computational load increases. Across all components, the RTX 2080 Ti achieves higher peak efficiencies than the A100, with values exceeding 80% for Projections and Attention Output. The fitted models, of the form $\eta(c) = \eta_{\max}(1 - e^{-kc^{\alpha}})$, accurately capture the nonlinear relationship between compute size and utilization. Notably, while both GPUs show similar efficiency trends at low compute loads, the RTX 2080 Ti maintains a steeper gain and reaches saturation faster, likely due to architectural differences in how parallelism is exploited. These results validate the use of our fitted efficiency model across hardware

platforms and highlight its utility in energy estimation and model selection.

Figure 5 shows the hardware efficiency trends for various elementary operations across RNN components executed on the NVIDIA A100 80GB PCIe GPU. Each subplot represents a specific operation Activations, Cell Update, Hidden Update, and Gates and plots empirical efficiency (blue dots) against compute size in FLOPs. The fitted curves (red crosses) follow the function $\eta(c) = \eta_{\max}(1 - e^{-kc^{\alpha}})$, capturing 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 80%) due to their heavier computational demands. In contrast, operations like Cell Update and Activations exhibit lower peak efficiencies, likely due to their smaller or less parallel workloads. These fitted curves validate the use of our analytical efficiency model and show that it captures the non-linear scaling of utilization with operation size across different RNN components.

LSTM and Transformer architectures exhibit fundamentally different energy and computational profiles, driven by their underlying design and operational patterns. LSTMs rely on sequential computation with gating mechanisms, where energy consumption is dominated by recurrent updates and memory cell operations. Their efficiency scales moderately with compute size but remains relatively low due to limited parallelism and smaller operation granularity. In contrast, Transformers leverage highly parallelizable operations such as multi-head attention and dense matrix projections leading to more efficient hardware utilization at scale. Empirical results show that Transformer components (e.g., Attention Output and Projections) achieve higher peak efficiencies (often exceeding 70–80%) on modern GPUs compared to LSTM gates, which typically saturate below 70%. Moreover, Transformer workloads benefit more from hardware accelerators like the A100 and RTX 2080 Ti due to their larger batch-friendly compute patterns. While LSTMs may still be favor able for low-latency or small-scale tasks, Transformers offer superior energy scalability and performance in high-throughput settings, making them more suited for modern deep learning workloads.

7.6 Extensive analysis: Modelling energy consumption

To estimate the energy consumption of Transformer architectures, we developed a regression-based modeling framework that balances predictive accuracy with coefficient stability. The dependent variable is measured energy consumption (in joules), and the predictors are the execution durations of the main layer operations of the model architecture.

We split the dataset into training and test partitions (67/33 split), stratified by model type and number of layers to ensure representative sampling across architectures. Two competing model specifications were considered:

- **Ordinary Least Squares (OLS)** regression with standardized features.
- **Ridge regression** with an L_2 penalty ($\alpha = 5.0$).

Both models were implemented as pipelines consisting of a StandardScaler to normalize feature magnitudes, followed by the linear estimator. To recover interpretable coefficients in the original feature space, we explicitly inverted the scaling transformation when extracting parameter estimates and intercepts.

To assess model robustness, we applied a nonparametric bootstrap with $B = 1000$ resamples. For each resample, the model was refit on the training set and evaluated on the holdout test set. This procedure yielded:

- Median coefficient and intercept estimates,
- 95% confidence intervals for each parameter,
- Distributional summaries of test-set Mean Absolute Error (MAE) and coefficient of determination (R^2).

We compared models based on both predictive performance and parameter stability. Specifically, we first identified the model with the lowest test MAE, and then considered any competing specification within 1% of that MAE as acceptable. Among these candidates, we selected the final model according to the smallest total width of the bootstrap confidence intervals across all coefficients, thereby favoring parsimonious and stable parameterizations.

This procedure ensures that our final model not only explains energy consumption with high predictive accuracy ($R^2 \approx 0.96$), but also provides stable and interpretable coefficients that reflect the contribution of each Transformer operation to overall energy usage.

For Transformer model. The bootstrap estimation results for the Ridge specification (Table 8) confirm that the model provides both accurate and stable predictions of Transformer energy consumption. The intercept is estimated at 3.63 J, representing the baseline energy cost independent of operation durations. Among the coefficients, the duration of the final projection has the largest positive effect (0.56 J per µs,), followed by attention scores (0.30 J per µs) and attention output (0.30 J per µs). In contrast, QKV projections exhibit a negative effect (-0.14 J per µs), suggesting that longer projection times may be associated with efficiency trade-offs in hardware execution. The narrow confidence intervals across all features, along with a high mean R^2 of 0.96 and a low mean absolute error of 6.30, indicate that the model is both robust and highly predictive, making it well suited for characterizing energy efficiency in Transformer architectures.

Table 8. Bootstrap estimates for the Ridge regression model of Transformer energy consumption. Reported values include the median coefficient estimates, 95% confidence intervals (CI), and performance metrics.

Operation	Estimate	$CI_{0.025}$	$CI_{0.975}$
Intercept	3.6292	3.4041	3.8759
duration_QKV Projections	-0.1378	-0.1455	-0.1299
duration_Attention Scores	0.3041	0.3015	0.3065
duration_Final Projection	0.5641	0.5404	0.5872
duration_Attention Output	0.3041	0.3015	0.3065

Model performance: Mean MAE = 6.3010, Mean $R^2 = 0.9584$. Total CI width = 0.0721.

For LSTM model. The regression analysis in Table 9 shows that the model achieves a high goodness of fit with $R^2 = 0.95$, indicating strong explanatory power for energy consumption in RNNs. The constant term is stable and significant, capturing the baseline overhead of execution. Among the operational components, gate computations (h_{Gates}) exhibit the smallest positive contribution, while hidden state updates (h_{Hidden_Update}) and cell state updates (h_{Cell_Update}) contribute substantially to energy usage, with coefficients on the order of 10^4. Interestingly, the activation functions ($h_{Activations}$) show a negative coefficient, suggesting that their relative energy cost is overshadowed by savings or overlaps in execution compared to other operations. Overall, the results highlight that state update operations dominate the energy profile of LSTM and GRU models, while gate operations play a comparatively minor role.

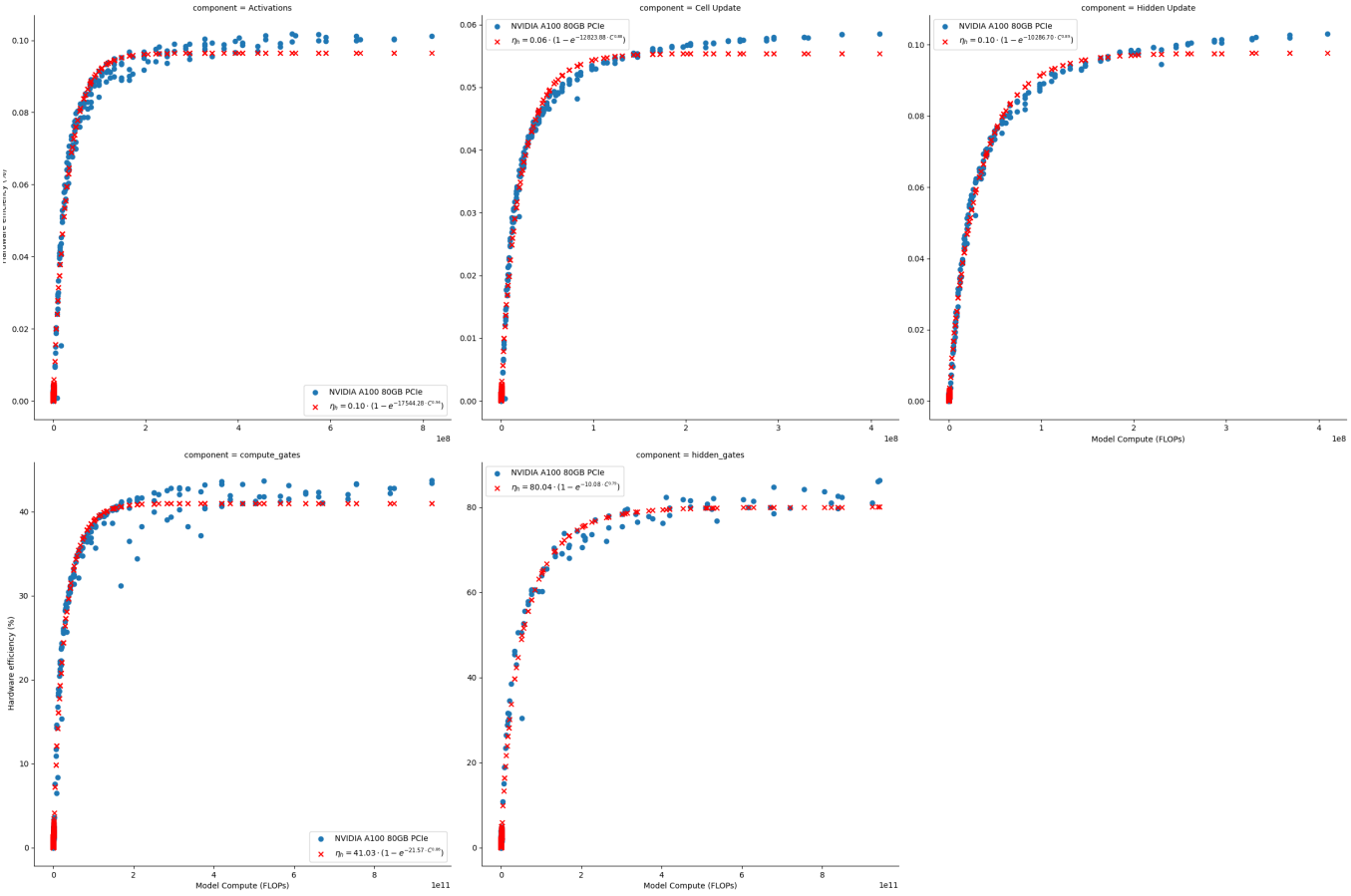

Figure 5. Hardware efficiency factors of elementary operations in LSTM and GRU layer.

From a hardware optimization perspective, this suggests that accelerators targeting RNN workloads should prioritize reducing the cost of hidden and cell state updates, for example through optimized memory access or specialized update kernels. Since gate computations are relatively inexpensive, energy savings from gate-level optimizations would be marginal. Instead, tailoring hardware design to the dominant update operations could lead to more meaningful efficiency gains in practical deployments.

Table 9. Estimated coefficients, standard errors, and 95% confidence intervals for RNNs (LSTM and GRU). With a coefficient of determination $R^2 = 0.95$. Durations are expressed in microseconds and energy in Joules.

Operation	Estimate	Std. Error	$CI_{0.025}$	$CI_{0.975}$
constant (h)	3367.3304	8.375	3350.882	3383.779
$h_{Activations}$	-7.66×10^4	7021.058	-9.04×10^4	-6.28×10^4
h_{Cell_Update}	4.52×10^4	4221.466	3.70×10^4	5.35×10^4
h_{Gates}	169.0733	9.072	151.255	186.892
h_{Hidden_Update}	2.95×10^4	2813.017	2.39×10^4	3.50×10^4

7.7 Energy Consumption Scaling with Model Depth

The results in Table 10 and Table 11 highlight a clear trend in how transformer architecture parameters affect energy consumption. While increasing the number of attention heads has only a marginal effect on total energy demand with variations remaining within fractions of a joule the number of layers strongly drives energy consumption upward. For example, moving from 2 to 24 layers increases estimated energy usage nearly tenfold, whereas increasing heads from 2 to 16 at a fixed depth produces almost indistinguishable values. Similarly, scaling up the hidden dimension also drives consumption upward, though in a smoother fashion for example, with 12 layers, energy use climbs from 36.5 J at $d_{\text{model}} = 64$ to nearly 95 J at $d_{\text{model}} = 1280$ (see Table 11). This indicates that the computational and energy complexity of transformer models is dominated by depth rather than width. While both architectural dimensions contribute to energy consumption, the data suggests that adding more layers is markedly less efficient than increasing the dimensionality (d_{model}). Therefore, careful control over the number of layers is the primary lever for energy-aware architecture optimization, suggesting that model design should weigh the disproportionately high energy demands of deeper architectures against the benefits of wider ones.

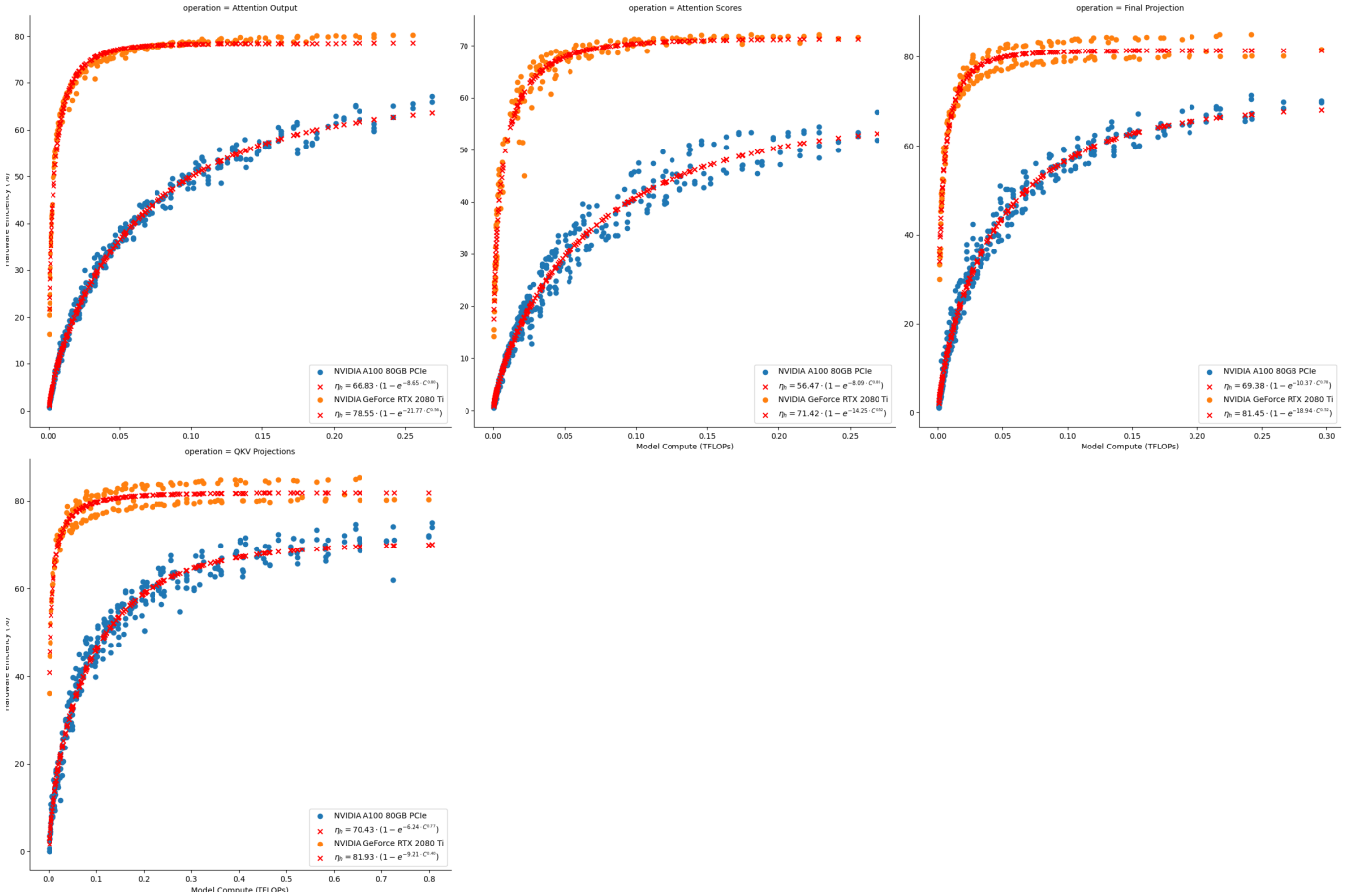

Figure 6. Hardware efficiency factors of elementary operations in transformer layer.

Table 10. Estimated energy consumption (in joules) as a function of Transformer layers and attention heads.

Layers	2	4	6	8	10	12	14	16
2	14.44	14.44	14.43	14.44	14.43	14.42	14.42	14.44
4	25.25	25.25	25.24	25.25	25.24	25.20	25.20	25.25
6	36.06	36.06	36.05	36.06	36.05	35.99	35.99	36.06
8	46.88	46.88	46.85	46.88	46.85	46.78	46.78	46.88
10	57.69	57.69	57.66	57.69	57.66	57.56	57.56	57.69
12	68.50	68.50	68.46	68.50	68.46	68.35	68.35	68.50
14	79.31	79.31	79.27	79.31	79.27	79.14	79.14	79.31
16	90.12	90.12	90.07	90.12	90.07	89.92	89.92	90.12
18	100.93	100.93	100.88	100.93	100.88	100.71	100.71	100.93
20	111.74	111.74	111.68	111.74	111.68	111.50	111.50	111.74
22	122.56	122.56	122.49	122.56	122.49	122.28	122.28	122.56
24	133.37	133.37	133.29	133.37	133.29	133.07	133.07	133.37

Table 11. Table showing the estimated energy consumption (in Joules) of the transformer model by varying the number of layers and the dimensionality (d_{model}) of the model architecture.

d_model	64	128	192	256	320	384	448	512	576	640	704	768	832	896	960	1024	1088	1152	1216	1280
layers																				
2	9.11	10.43	11.37	12.12	12.79	13.39	13.92	14.43	14.91	15.35	15.78	16.18	16.56	16.93	17.28	17.61	17.93	18.25	18.53	18.82
4	14.58	17.22	19.11	20.60	21.95	23.15	24.22	25.24	26.20	27.07	27.93	28.74	29.49	30.23	30.94	31.59	32.24	32.86	33.44	34.01
6	20.06	24.02	26.85	29.09	31.11	32.92	34.51	36.05	37.48	38.79	40.07	41.30	42.42	43.53	44.59	45.57	46.54	47.48	48.34	49.21
8	25.54	30.81	34.60	37.57	40.27	42.68	44.80	46.85	48.77	50.51	52.22	53.85	55.35	56.83	58.25	59.55	60.85	62.10	63.24	64.40
10	31.01	37.61	42.34	46.06	49.43	52.44	55.10	57.66	60.05	62.23	64.37	66.41	68.28	70.13	71.90	73.53	75.15	76.71	78.15	79.59
12	36.49	44.41	50.08	54.55	58.59	62.20	65.39	68.46	71.34	73.95	76.52	78.96	81.21	83.43	85.55	87.51	89.46	91.33	93.05	94.78
14	41.97	51.20	57.82	63.03	67.75	71.97	75.68	79.27	82.62	85.67	88.67	91.52	94.13	96.73	99.21	101.49	103.77	105.95	107.96	109.97
16	47.44	58.00	65.56	71.52	76.91	81.73	85.98	90.07	93.91	97.39	100.82	104.07	107.06	110.03	112.86	115.47	118.07	120.56	122.86	125.17
18	52.92	64.79	73.30	80.00	86.07	91.49	96.27	100.88	105.19	109.11	112.97	116.63	119.99	123.33	126.52	129.45	132.38	135.18	137.76	140.36
20	58.40	71.59	81.05	88.49	95.23	101.25	106.56	111.68	116.48	120.83	125.11	129.19	132.92	136.63	140.17	143.43	146.68	149.80	152.67	155.55
22	63.87	78.39	88.79	96.98	104.39	111.02	116.85	122.49	127.76	132.55	137.26	141.74	145.85	149.93	153.83	157.41	160.99	164.41	167.57	170.74
24	69.35	85.18	96.53	105.46	113.55	120.78	127.15	133.29	139.04	144.27	149.41	154.30	158.78	163.23	167.48	171.39	175.29	179.03	182.48	185.94
26	74.83	91.98	104.27	113.95	122.71	130.54	137.44	144.10	150.33	155.99	161.56	166.85	171.71	176.53	181.13	185.37	189.60	193.65	197.38	201.13
28	80.30	98.78	112.01	122.44	131.87	140.30	147.73	154.90	161.61	167.71	173.71	179.41	184.64	189.83	194.79	199.35	203.90	208.26	212.28	216.32
30	85.78	105.57	119.75	130.92	141.03	150.06	158.03	165.71	172.90	179.43	185.86	191.96	197.57	203.13	208.44	213.33	218.21	222.88	227.19	231.51
32	91.26	112.37	127.50	139.41	150.19	159.83	168.32	176.51	184.18	191.15	198.00	204.52	210.50	216.43	222.10	227.31	232.51	237.50	242.09	246.70
34	96.73	119.16	135.24	147.89	159.34	169.59	178.61	187.32	195.47	202.87	210.15	217.08	223.43	229.73	235.75	241.29	246.82	252.11	257.00	261.90
36	102.21	125.96	142.98	156.38	168.50	179.35	188.91	198.13	206.75	214.59	222.30	229.63	236.36	243.03	249.41	255.27	261.12	266.73	271.90	277.09
38	107.69	132.76	150.72	164.87	177.66	189.11	199.20	208.93	218.04	226.31	234.45	242.19	249.29	256.33	263.06	269.25	275.43	281.35	286.80	292.28
40	113.16	139.55	158.46	173.35	186.82	198.88	209.49	219.74	229.32	238.03	246.60	254.74	262.22	269.63	276.71	283.23	289.73	295.96	301.71	307.47
42	118.64	146.35	166.20	181.84	195.98	208.64	219.79	230.54	240.61	249.75	258.75	267.30	275.14	282.93	290.37	297.21	304.04	310.58	316.61	322.66
44	124.12	153.14	173.94	190.33	205.14	218.40	230.08	241.35	251.89	261.47	270.90	279.85	288.07	296.23	304.02	311.19	318.34	325.20	331.52	337.86
46	129.59	159.94	181.69	198.81	214.30	228.16	240.37	252.15	263.17	273.19	283.04	292.41	301.00	309.53	317.68	325.17	332.65	339.81	346.42	353.05
48	135.07	166.74	189.43	207.30	223.46	237.93	250.67	262.96	274.46	284.91	295.19	304.97	313.93	322.83	331.33	339.15	346.95	354.43	361.32	368.24
50	140.55	173.53	197.17	215.78	232.62	247.69	260.96	273.76	285.74	296.63	307.34	317.52	326.86	336.13	344.99	353.13	361.26	369.05	376.23	383.43
52	146.02	180.33	204.91	224.27	241.78	257.45	271.25	284.57	297.03	308.35	319.49	330.08	339.79	349.43	358.64	367.11	375.66	383.66	391.13	398.63
54	151.50	187.13	212.65	232.76	250.94	267.21	281.55	295.37	308.31	320.07	331.64	342.63	352.72	362.73	372.29	381.09	389.87	398.28	406.04	413.82
56	156.98	193.92	220.39	241.24	260.10	276.98	291.84	306.18	319.60	331.79	343.79	355.19	365.65	376.03	385.95	395.07	404.17	412.90	420.94	429.01
58	162.46	200.72	228.14	249.73	269.26	286.74	302.13	316.98	330.88	343.51	355.93	367.74	378.58	389.33	399.60	409.05	418.48	427.51	435.84	444.20
60	167.93	207.51	235.88	258.21	278.42	296.50	312.43	327.79	342.17	355.23	368.08	380.30	391.51	402.63	413.26	423.03	432.78	442.13	450.75	459.39
62	173.41	214.31	243.62	266.70	287.58	306.26	322.72	338.60	353.45	366.95	380.23	392.86	404.44	415.93	426.91	437.02	447.09	456.75	465.65	474.59