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 9. 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:

$$N \approx L \cdot 4 \cdot (h \cdot x + h \cdot h + h) \quad (17)$$

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

$$N \approx L \cdot 3 \cdot (h \cdot x + h \cdot h + h) \quad (18)$$

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 for RNN models

The calculation of FLOPs for LSTM and GRU models incorporates several architectural and training parameters, including the number of layers, the size of hidden units (h), the size of input features (x), the sequence length (T), and the batch size (B) used during training. At each layer, the FLOPs include computations required to process all sequences in the batch for the entire input sequence length. These calculations are then scaled by the number of layers (L) and summed up to determine the overall computational cost. The formula for estimating the number of FLOPs in LSTM and GRU models can be expressed as:

$$\mathrm{C}_{lstm} = L \cdot B \cdot T \cdot (4 \cdot (2 \cdot (h + x) \cdot h + 2 \cdot h) + 5 \cdot h) \quad (19)$$

$$\mathrm{C}_{gru} = L \cdot B \cdot T \cdot (3 \cdot (2 \cdot (h + x) \cdot h + 2 \cdot h) + 5 \cdot h) \quad (20)$$

7.4 Transformer Models.

To estimate the floating point operations per second (FLOPs) of the Transformer model, we calculate the computational cost of each component and aggregate it across the layers and batch size (see Table 10 in the supplementary materials). For the Multi-Head Attention mechanism, the FLOPs are derived from the matrix multiplications required for projecting the input into query, key, and value spaces, computing the attention scores, and projecting the outputs. Specifically, the attention mechanism involves operations for query and key projections, attention weight computation, and the final output projection. These operations are scaled by the number of attention heads and the feedforward layers. The key parameters are the model dimension (d_{model}), the number of attention heads (num_attention_heads), the feedforward dimension ($\text{dim}_{\text{feedforward}}$), the number of layers (num_layers), input size (W), sequence length (L), and batch size.

Table 10. FLOPs estimation for Transformer components

Component	Formula
Query and Key Projections	$\text{FLOPs}_{QK} = 2 \cdot W \cdot d^2_{\text{model}}$
Value Projection	$\text{FLOPs}_{V} = W \cdot d^2_{\text{model}}$
Attention Weights	$\text{FLOPs}_{\text{att}} = W^2 \cdot d_{\text{model}}$
Output Projection	$\text{FLOPs}_{\text{out}} = W \cdot d^2_{\text{model}}$
Multi-Head Attention (MHA)	$\text{FLOPs}_{\text{MHA}} = \text{num\_heads} \cdot (\text{FLOPs}_{QK} + \text{FLOPs}_{V} + \text{FLOPs}_{\text{att}} + \text{FLOPs}_{\text{out}})$
Feedforward (FF)	$\text{FLOPs}_{\text{FF}} = 2 \cdot (\text{dim}_{\text{feedforward}} \cdot d_{\text{model}})$
Single Transformer Layer	$\text{FLOPs}_{\text{layer}} = \text{FLOPs}_{\text{MHA}} + \text{FLOPs}_{\text{FF}}$
Total FLOPs (C)	$\text{Total FLOPs} = \text{num\_layers} \cdot \text{batch\_size} \cdot L \cdot \text{FLOPs}_{\text{layer}}$

7.4.1 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 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. 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 favorable 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.

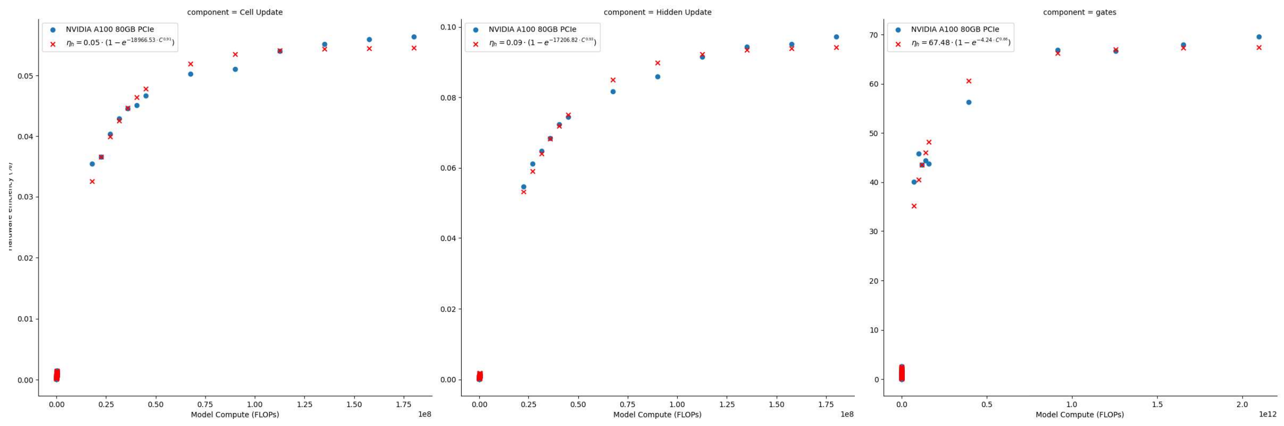

Figure 5. Hardware efficiency factors of elementary operations in LSTM and GRU layer.

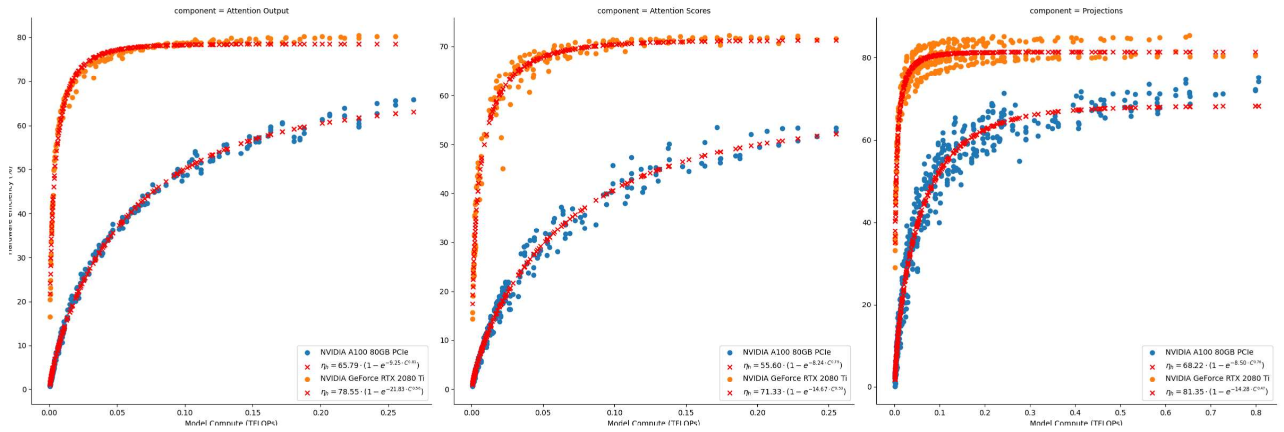

Figure 6. Hardware efficiency factors of elementary operations in transformer layer.