WattLayer: Get Layers Right to Estimate Inference Energy of Neural Networks

Adrien Sardi[1,2,3], Marie Line Alberi-Morel[1], Sara Alouf[2], Frédéric Giroire[3] and Joanna Moulierac[3]

[1]*Bell Labs, Nokia Networks France*
[2]*Inria, Université Côte d'Azur, France*
[3]*Université Côte d'Azur, CNRS, Inria, I3S, France*

Abstract

The widespread adoption of Artificial Intelligence (AI) has led to increasing concerns about energy consumption, yet there is a lack of standardized methodologies to accurately estimate AI inference energy consumption, particularly across various tasks and architectures. In this study, we propose a task independent, layer wise energy estimation model for AI architectures. Our model is evaluated on a large dataset of more than 100,000 layers for 295 neural network architectures across 3 widely used tasks and 3 distinct hardware platforms. Our approach achieves a median error of 19.6%, outperforming state-of-the-art methods. We further show that layer-wise decomposition generalizes to new tasks without complete retraining, by leveraging shared layers across architectures. It offers tools, insights, and a precise methodology to empower stakeholders in designing energy-efficient AI systems.

Keywords

Layer Decomposition, Energy Measurement, Inference, Neural Networks

1. Introduction

Artificial Intelligence (AI) has been rapidly adopted across industries due to recent advances in model accuracy [1]. This widespread adoption has led to a substantial increase in energy demand, particularly for large-scale models. For instance, Google reported that machine learning workloads accounted for 10% to 15% of its total energy consumption between 2019 and 2021 [2]. AI energy consumption can be broadly divided into two phases: training and inference. While most efficiency efforts have historically focused on training due to its high absolute cost compared to a single inference [3], the large-scale deployment of AI services has shifted this balance. At scale, inference—the phase where models are used—can dominate overall energy consumption. Indeed, reports from Google [2] and Meta [4] indicate that inference can account for approximately 60–65% of total energy use, reaching up to 90% in cloud deployments such as Amazon Web Services [5]. As AI systems become increasingly integrated into real-world applications, energy-aware model design is gaining importance among developers. However, despite this growing need, there is still a lack of standardized tools and methodologies to estimate the energy consumption of AI systems without executing the models. While companies such as Google [6] and Mistral AI [7] have reported energy usage for large language models, the absence of unified estimation frameworks makes comparisons difficult and limits practical use in local or personal deployment scenarios.

This paper introduces WattLayer: a task-independent layer-wise energy estimation model for neural

SuRE'26: 1st Workshop on Sustainability and Resource-Efficiency of Artificial Intelligence, co-located with IJCAI 2026, August 17, 2026, Bremen, Germany
adrien.sardi@nokia.com (A. Sardi); marie_line.alberi-morel@nokia-bell-labs.com (M. L. Alberi-Morel); sara.alouf@inria.fr (S. Alouf); frederic.giroire@cnrs.fr (F. Giroire); joanna.moulierac@univ-cotedazur.fr (J. Moulierac)
https://aupeka.github.io/ (A. Sardi); https://www-sop.inria.fr/members/Sara.Alouf/ (S. Alouf); https://www-sop.inria.fr/members/Frederic.Giroire/ (F. Giroire); https://www-sop.inria.fr/members/Joanna.Moulierac/ (J. Moulierac)
0009-0004-2226-6253 (A. Sardi); 0000-0002-3843-7617 (M. L. Alberi-Morel); 0009-0009-6643-0568 (S. Alouf); 0000-0002-3727-051X (F. Giroire); 0000-0002-4367-5839 (J. Moulierac)

network architecture. We propose and evaluate our methodology to estimate energy consumption over different hardwares and experimental set-up. By outperforming state-of-the-art (SOTA) models in terms of accuracy, we demonstrate that a layer-based approach can enhance prediction quality across multiple tasks and generalize to entirely new tasks without requiring re-training, distinguishing it from traditional models. This paper addresses the emerging need for methodologies and tools that enable users to predict the energy consumption of AI services proactively. It provides accurate predictions of neural network energy consumption at the layer level, enabling stakeholders to make informed decisions about energy efficiency during AI development and deployment. Our key contributions are as follows:

- **Experimental protocol** We establish a rigorous experimental protocol to ensure reliable and reproducible measurements of the energy consumption of AI algorithms and their individual layers on GPUs.
- **Extensive dataset** We collect energy consumption data for neural networks architectures across 3 widely used tasks and 3 hardware with a maximum of 295 architectures, totaling over 100,000 energy measurements, creating one of the most comprehensive datasets for energy estimation in AI considering full-architecture and individual layer consumption.
- **Architecture features extraction** We develop a Python-based framework capable of rigorously extracting the layer-wise decomposition of any PyTorch architecture, including the layers and their main characteristics. This enables detailed analysis and modeling of energy consumption at the layer level.
- **Layer-wise energy estimation methodology** We propose WattLayer, a task-agnostic methodology to estimate the energy consumption of AI algorithms.By breaking down architectures to a level of granularity sufficiently fine to eliminate task-specific dependencies and unlike SOTA models which require retraining for each task, our approach generalizes across diverse algorithms without customization, achieving superior accuracy and adaptability.
- **Zero-shot generalization to large LLMs** Finally, we test the zero-shot generalization capacity of our layer-wise energy model on a small number of LLMs. The model successfully estimates energy consumption for LLMs without fine-tuning (Mean Absolute Percentage Error MAPE $\leq 30\%$), showcasing its generalization capability and the promising potential as a reliable tool for sustainable AI initiatives.

2. Related Work

Researchers have long worked on energy estimation models, initially for general programs [8] and more recently for AI algorithms [9]. [10] approximate the value of the energy consumption by measuring power over discrete intervals $\Delta\tau$ and assuming a constant power between two measurements. Given a set of $k \in \mathbb{N}$ intervals of length $\Delta\tau \in \mathbb{R}$, $E = \sum_k P(k \cdot \Delta\tau)\Delta\tau$ illustrates the link between the energy consumption E and the power P. Starting from this definition, [11] seek to understand the internal mechanisms of the energy consumption measurement provided by NVIDIA-smi, a command-line utility provided by NVIDIA that can be used for monitoring, managing, and querying GPU statistics. They propose some very useful guidelines to help collect power and energy data through NVIDIA-smi. [12] propose an evaluation of the energy consumption of several Convolutional Neural Network (CNN) on GPU and CPU. They evaluate the impact of several experimental setups on the resulting energy consumption including libraries, batch size and hardware settings. They also propose a first evaluation of the energy consumption of the layers of several CNN. They obtain that, among convolutional, pooling, fully connected, and ReLU layers, the convolutional layers are by far the most consuming layers (87% of the overall energy consumption).

A common way to estimate the energy consumption is to use the number of operations performed as features of the energy consumed by a neural network: FLOPs or MACs [13, 14, 15, 16]. However, this feature alone is often insufficient and requires either supplementing the model with other features

Table 1
SOTA models summary with features, model types and adressed tasks.

Model	Features					Layer-wise model	Task		
	MACs or FLOPs	Activations	Parameters	Input size	Batch size		Vision	NLP	Audio
NeuralPower [19]	✓			✓	✓	✓	✓		
Getzner [16]	✓					✓	✓		
Mac, e.g. [15]	✓						✓		
HJ [18]	✓	✓	✓				✓		
WattLayer	✓	✓	✓	✓	✓	✓	✓	✓	✓

or modifying the model itself [17]. Whats more, using architecture-level estimation model prevents generalization on different tasks. As our experiments in Section 4 show, adding new tasks to the database weakens the estimation performance of the final model. [18] collect the energy consumption of 1,200 vision models. They propose a log-linear regression based on 3 hyper-parameters of the model to estimates its consumption: number of parameters, gmacs and activation count. Overall, their model has an error of 34%. While this model offers a robust benchmark and demonstrates strong performance in controlled conditions, its scope is limited to vision architectures. The authors also propose a second estimation model that utilizes throughput as a feature, delivering exceptional results with an error rate of less than 1%. They suggest this metric as a viable option for estimation in scenarios where GPU information is unavailable but throughput data is accessible. However, measuring GPU throughput during an experiment is as challenging as measuring energy consumption, which led to the decision not to compare our model with this second approach.

Another approach estimates the energy consumption of a neural network by modeling the energy of its individual layers. This strategy has shown promising results [19, 16, 20], but existing studies remain limited in terms of hardware coverage and execution settings. NeuralPower proposes a polynomial regression framework for CNNs executed on GPUs, but considers only convolutional, fully connected, and pooling layers, and relies on fixed GPU frequency and voltage settings. [16] introduces a linear regression model based on features such as MACs but focuses on CPU execution and shows reduced accuracy on real-world architectures. These limitations highlight the need for a more general methodology that supports modern GPU-based workloads under realistic hardware configurations. We summarize SOTA models in Table 1.

3. Methodology and Experimental Set-up

3.1. Experimental Protocol

The experiments conducted concern short time processes. Indeed, the execution of a layer of a neural network, using the PyTorch library, may take less than a second. To ensure high-quality experimental data and improve reproducibility, it is essential to establish a rigorous experimental protocol. This includes defining the data collection tool, the frequency of measurement and the number of repetitions of one execution.

Data Collection Tool Several applications are available to measure the energy consumption or power usage of NVIDIA GPUs. Most, if not all, of these applications rely on the same tool for estimating NVIDIA GPU energy consumption: NVIDIA-smi. [11] have investigated the internal mechanisms of energy measurement provided by NVIDIA-smi and found that the tool has an overall measurement error of 5%. Based on this finding, we adopt NVIDIA-smi for our experiments. Specifically, we use CodeCarbon [21] to measure energy consumption, which combines Intel RAPL for CPU energy measurement and NVIDIA-smi for GPU energy measurement.

Frequency of Measurement The sampling frequency f_{mes} is set to 10 Hz to get closer to the value recommended by [11] without raising errors from CodeCarbon.

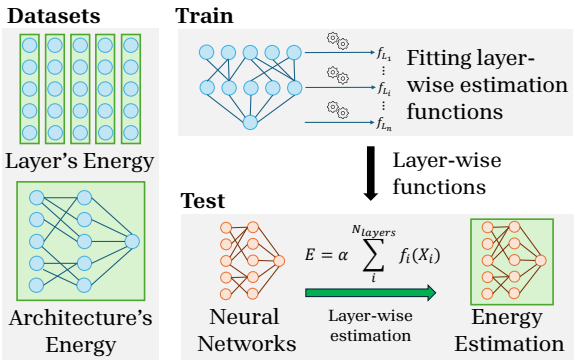


Figure 1: Overview of WattLayer for neural network energy estimation. Energy measurements are collected for complete architecture and individual layers. During training, layers are grouped by type and a dedicated model is fitted to each group. During inference, a target architecture is decomposed into layers, each layer's energy is estimated with the corresponding model, and the total energy is obtained by aggregating all layer estimates.

Repetition To obtain reliable energy measurements, the duration of each experiment must be substantially longer than the sampling interval of the power sensor. We therefore repeat each inference multiple times and measure the cumulative energy consumption. As shown in Section 4.1, the number of repetitions directly affects measurement precision. Based on these experiments, we define a minimum number of forward passes, denoted N_{mes}, and set it to 4,000 for most experiments and 15,000 for natural language processing (NLP) layers.

3.2. Layer-Wise Energy Estimation Framework

Framework Our methodology estimates the energy consumption of a neural network by decomposing the architecture into individual layers. A dedicated estimation model is trained for each layer type using the parameters of each layer. The total energy consumption of the architecture is then obtained by aggregating the estimated energy of all constituent layers. Figure 1 illustrates the methodology schematically.

Layer-Wise Energy Estimation Let $\mathcal{A}$ be a neural network architecture with $N(\in \mathbb{N})$ layers $L_1, ..., L_N$. $E_{\mathcal{A}}$ and $\{E_{L_i}\}_i$ are respectively the energy consumption of the architecture and of the layers. We make the assumption that the total energy $E_{\mathcal{A}}$ may be approximated by the sum of the energy consumption of its layers $\{E_{L_i}\}_i$ up to a correction factor α: $E_{\mathcal{A}} \approx \alpha \sum_{i=1}^{N} E_{L_i}$. Despite their large diversity, existing architectures use a limited number of types of layers (e.g. Linear, Conv2d, Embedding,...). We note type(i) the type of layer i. Then, for each type, we define a layer-specific function $f_{\text{type}(i)}(\mathbf{x}_i)$ that estimates the energy consumption of layer i, where $\mathbf{x}_i$ denotes the relevant parameters impacting the energy consumption of layer i. These parameters are: #MAC, #Activation, #Parameters, Input shape, Batch size and Kernel size for convolutional layers (see Appendix A). Finally, we write

$$E_{\mathcal{A}} \approx \alpha \sum_{i=1}^{N} E_{L_i} = \alpha \sum_{i=1}^{N} f_{\text{type}(i)}(\mathbf{x}_i) \ , \tag{1}$$

where α is a correction factor to take into account the gap between architecture energy and layer's energy summation.

Data Collection To train the layer estimation function, we need to construct a training dataset of layer energy consumption. For each layer, energy data is collected by running the layer alone using the experimental protocol with randomly generated tensor. During the evaluation phase, testing data is also generated based on randomly generated input tensor with the same distribution. Associated parameters required for training layer-wise model are also collected.

In their work on estimating the energy consumption of vision neural network layers, [16] generate random layers to build their training dataset. This approach enabled them to produce sufficient data to train statistical models for various layer types. (e.g., Convolutional, Linear, MaxPooling). However, this method sacrifices real-world layer specificities, resulting in strong performance on random architectures but poorer results on real-world architectures. To address this limitation, we choose to collect data from real-world architectures to build our training dataset. While this approach is more challenging,

requiring the extraction of architecture structures without prior knowledge, it provides a more accurate basis for estimating layer energy consumption. To automate the extraction of neural network structures across multiple architectures, we develop a Python script that sets hooks on each layer. A *layer* is defined as a leaf module in the architecture hierarchy. The architecture is recursively traversed, following the forward pass of the architecture, and only these terminal modules are retained. This results in a fine-grained decomposition of the network, typically corresponding to elementary operations such as convolutional layers, fully connected layers, activation functions, and normalization layers. To ensure accurate extraction, it is crucial to account for the actual utilization of layers during inference, rather than relying solely on the layers' names listed in the PyTorch structure (as proposed by [16]). Our approach captures the true layer decomposition and considers the sequential behavior of layers, including jumps and bypasses that may occur during inference. Information regarding our Python scripts can be found in the supplementary materials.

Model Training The constructed dataset includes energy consumption measurements for architecture layers along with features that characterize energy usage. Layers are grouped by type, independently of the architecture or the task from which they were extracted. This fine-grained, task-independent decomposition enables the aggregation of data from multiple tasks into a larger training set. Ultimately, it aims to enhance prediction quality when transitioning across multiple tasks, rather than compromising it, as is often the case with other models. A dedicated layer-estimation function is then trained for each group using the collected features. To ensure the model remains lightweight and interpretable, we prioritize simple statistical models such as linear regression and multi-linear regression. For specific layers, we also employ log-linear regression similarly to [18].

Model Testing To evaluate the performance of our model, we test it on architectures whose energy consumption data was not included in the training dataset. The estimation process for each architecture involves three steps: (i) extracting the layer decomposition of the architecture, (ii) estimating the energy consumption of each layer using the trained layer-estimation functions, and (iii) reconstructing the total energy consumption of the entire architecture by summing the individual layer estimations.

4. Results

In this section, we present our results in 4 parts. First, we study the number of executions needed to correctly measure the energy consumption of a process. Next, we assess the layer-wise decomposition hypothesis. We show that a correction factor needs to be applied for our methodology to be accurate. We then evaluate WattLayer on several tasks. We demonstrate that, by leveraging layer-wise energy consumption modeling, it outperforms state-of-the-art models. Finally, we demonstrate that our methodology paves the way for task-agnostic estimation model by showcasing WattLayer's ability to accurately estimate the energy consumption of LLMs, despite no model of this family being included in the training dataset.

4.1. Impact of Measurement Time

We repeat the forward passes N_{mes} times and compute the average energy consumption: $E_{\text{collected}} = E_{\text{experiment}}/N_{\text{mes}}$. By defining the experimental error as: $\text{Error}(N_{\text{mes}}) = \frac{|E_{\text{collected}}(N_{\text{mes}}) - E_{\text{collected}}(N_{\max})|}{E_{\text{collected}}(N_{\max})}$ we set N_{mes} to the minimum value that guarantees an error lower than a desired threshold. We evaluate N_{mes} for 2 tasks and on 2 GPUs (see Figure 2). We demonstrate that a minimum value of N_{mes} is required to ensure an error below 20%. Furthermore, setting N_{mes} correctly depends on the process that is run. If a specific process requires only a few calculation steps, then N_{mes} needs to be set to a larger value.

4.2. Assessing the Layer-Wise Decomposition Hypothesis

Our methodology relies on the assumption that the energy of an architecture can be approximated by the aggregation of the energy consumption of its layers (see Equation 1). Our experimental findings, detailed in Figure 3, show that a simple summation of the energy consumption of layers tends to

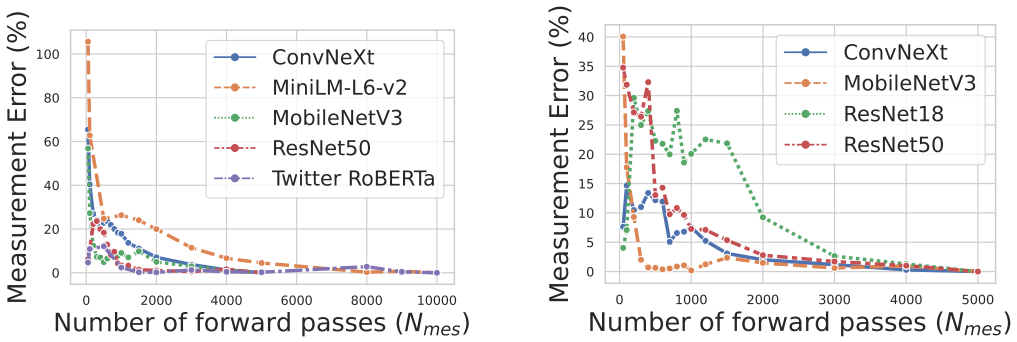


Figure 2: Error(N_{mes}) with respect to the number of repetitions of a forward pass N_{mes}. Experiments ran on an NVIDIA GPU RTX 6000 (left) and NVIDIA GPU TITAN X (right) with batch size equal to 1.

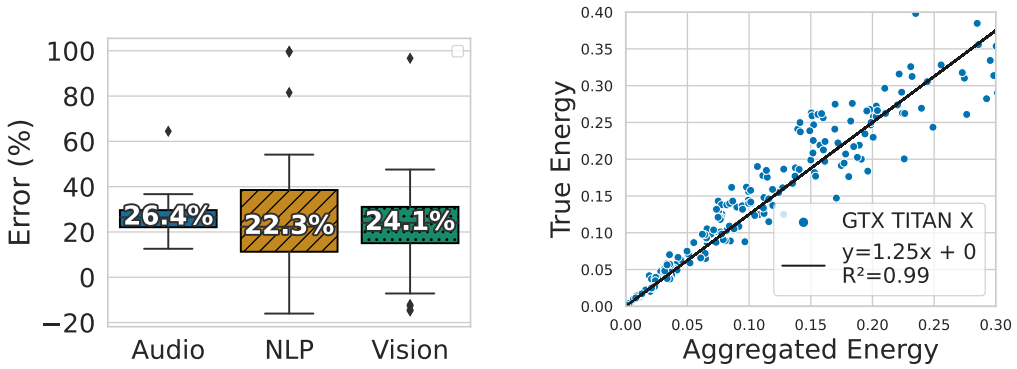


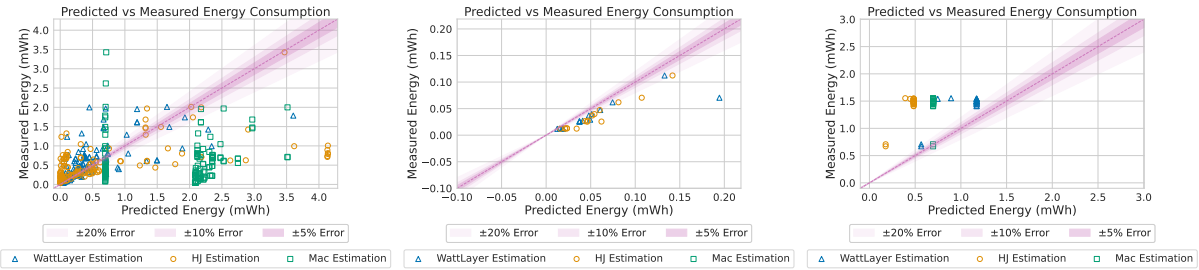
Figure 3: Layer decomposition error (left) and correction factor calibration (right).

underestimate the total consumption by approximately 25%. Different from [16] findings, we attribute this discrepancy to measurement granularity and system-level overheads, such as memory management, data movement or GPU frequency setting, that are not captured when profiling layers in isolation. Consequently, we calibrate the WattLayer correction factor α in (1) to 1.25. In our experiments, this adjustment is stable across hardware platform and tasks. Moreover, it yields robust predictive performance while preserving the model's universality and adaptability to new tasks, as demonstrated throughout the subsequent results.

4.3. Statistical Model to Estimate Layer-Wise Energy Consumption

Evaluation Framework The training dataset was built by collecting the energy consumption data for 92 image-classification models from Timm [22] and 29 text-classification models and 29 audio-classification models from Transformers [23] across three batch sizes: 1, 32, and 64 for a total of 164 models and 99,411 samples. Most parameters were kept at their default values, corresponding to the standard use case. In Hugging Face workflows, models are typically initialized from pretrained configurations, and users generally rely on these default settings unless they explicitly specify alternative values. To validate our model, we test it on 93 image-classification models from Timm and TorchVision [24], 17 text-classification and 35 audio-classification models from Transformers that were excluded from the training dataset for a total of 145 models and 207 samples. To evaluate accuracy, we use four metrics: mean absolute percent error (MAPE), median absolute percent error (MedAPE), maximum absolute percent error (MaxAPE), and minimum absolute percent error (MinAPE).

Baselines We compare WattLayer against two strong and widely applicable baselines. The first is a linear regression model using the number of MACs as the sole predictor, a common approach in prior work and practice. The second is the state-of-the-art "HJ" model proposed by [18], which employs a log-linear regression based on three architectural features: #MACs, #Parameters, and #Activations. We



(a) 94 vision architectures (b) 17 NLP architectures (c) 35 audio architectures

Figure 4: Measured vs. predicted energy with WattLayer, HJ, and Mac estimation models for (a) Vision, (b) NLP and (c) Audio architectures. The evaluation is conducted on architectures sourced from the widely used Python libraries Torchvision and Transformers.

Table 2
Performance metrics of WattLayer compared to SOTA models on image-, text- and audio-classification models.

Metric	WattLayer	HJ-V	HJ-B	MAC-V	MAC-B
Testing on 93 image-classification models					
MAPE	**40.1**%	63.9%	99.8%	132.6%	833.7%
MedAPE	**33.6**%	48.7%	86.8%	52.9%	229.0%
MaxAPE	**168.9**%	259.2%	870.1%	3,073.3%	18784.3%
MinAPE	0.4%	1.9%	**0.02**%	0.7%	8.2%
Std dev	**35.1**%	56.7%	130.4%	353.7%	2199.7%
Testing on 17 text-classification models					
MAPE	**45.6**%	341.4%	152.9%	–	3,091.2%
MedAPE	**46.2**%	265.4%	68.1%	–	2,526.3%
MaxAPE	**175.8**%	1,158.1%	1,510.5%	–	5,713.4%
MinAPE	**4.7**%	105.5%	26.2%	–	518.6%
Std dev	**35.9**%	320.9%	341.7%	–	1832.8%
Testing on 43 audio-classification models					
MAPE	**26.6**%	74.6%	67.7%	–	58.3%
MedAPE	**22.2**%	68.6%	67.5%	–	53.8%
MaxAPE	**61.1**%	100%	75.1%	–	93.6%
MinAPE	15.6%	64.1%	63.4%	–	**1.72**%
Std dev	10.1%	12.6%	**3.0**%	–	23.7%

do not directly compare against earlier layer-wise methods as their assumptions limit applicability in our setting. NeuralPower is restricted to a fixed GPU configuration and a three layer types, limiting its applicability beyond vision models. [16] supports a wider range of layer types but relies on randomly generated layer configurations and extracts layers from the static PyTorch module hierarchy rather than the effective execution graph, limiting its ability to handle modern architectures with skip connections such as ResNet [25]. In contrast, WattLayer is based on effective forward-pass execution and a unified layer-wise model covering diverse layer types and tasks, enabling scalable and accurate estimation across architectures and hardware (see Appendix B). To have a fair comparison we create a balanced train dataset with 25 architectures from each task with which we train both SOTA models (MAC-balanced, HJ-balanced) for all tasks. In addition, for vision only, we also train specific models called HJ-Vision and MAC-Vision because the number of architectures was sufficient to have a fair comparison between our models and the SOTA models. Having those different models ensures an evaluation of good quality and it emphasises the adaptability of WattLayer. Figure 4 provides a visualization of the model performance against the two SOTA models.

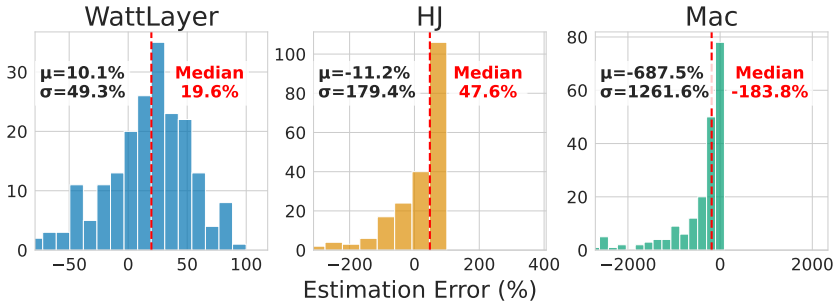


Figure 5: Distribution of estimation error for WattLayer, HJ and Mac model across all architectures in the test dataset. The evaluation is conducted on architectures sourced from the widely used Python libraries: TorchVision, Timm and Transformers.

Vision Models We train statistical models for each layer type to predict energy consumption. Three types of models are evaluated: linear regression, multi-linear regression, and log-linear regression. Linear regression is applied to most layer types due to its simplicity and effectiveness. However, for layers that are highly represented in the dataset or accounted for the majority of energy consumption, more complex models, such as multi-linear and log-linear regression, are employed to enhance prediction accuracy. Results, presented in Table 2, show that **our model outperforms both models with MAPE = 40.1% and MedAPE = 33.6%** due to several factors: (i) our approach incorporates more specific parameters, enabling finer-grained layer-wise decomposition for improved accuracy; (ii) batch size, a critical factor influencing energy consumption per image, is explicitly considered, whereas other models overlook this dependency despite prior findings by [18] that larger batch sizes reduce energy consumption per image. and (iii) while training datasets use 100 different models, state-of-the-art models use only 300 samples by focusing on the energy consumption of the full architecture (100 models $\times$ 3 batch sizes), whereas our framework generates thousands of samples by also accounting for the energy consumption of individual layers.

NLP and Audio We extend our energy estimation framework to text and audio classification models by jointly training on Vision, NLP, and Audio datasets. This multi-domain setup enables learning of shared layer-wise energy patterns across heterogeneous tasks, moving towards a task-agnostic formulation of energy estimation. Unlike prior work, which typically requires task-specific training or retraining (Table 2), our approach relies on a unified layer-wise model applicable across domains. By exploiting common layer types across modalities (e.g., linear layers in vision, NLP, and audio, or convolutional layers in vision and audio), the model learns transferable energy functions without task-specific redesign. This generalization capability not only simplifies the estimation process but also ensures superior performance and scalability across various applications. We observe that **our methodology outperforms SOTA models with MAPE = 46.6% and MedAPE 46.2% for text-classification and with MAPE = 26.6% and MedAPE = 22.2% for audio-classification**. Figure 5 provides a detailed visualization of the error distribution across the three estimation models. Notably, for all tasks, WattLayer exhibits a significantly more centralized error distribution, with values closer to zero compared to the other models.

Finally, we test our methodology on other GPUs from NVIDIA: H100 and A100 to assess the adaptability of our model on other hardware. We show that WattLayer has similar MAPE on other hardware platforms (see Figure 6).

4.4. Evaluating Zero-Shot Generalization of WattLayer to LLMs

After validating that WattLayer accurately predicts the energy consumption of unseen architectures within its training domain, we evaluate its zero-shot generalization to large language models (LLMs), without any fine-tuning or LLM-specific training data. This experiment is particularly relevant because LLMs are widely deployed in real-world applications and are known for their substantial energy

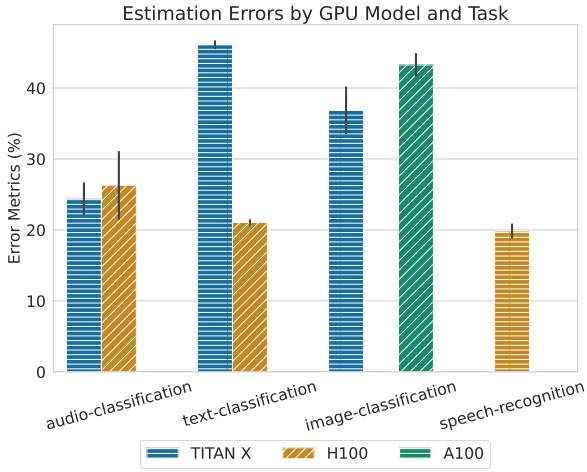


Figure 6: WattLayer performance for other GPUs: NVIDIA H100 (5 NLP architectures and 18 audio architectures are used for training and 116 models across NLP and Audio for testing) and A100 (19 Vision architectures are used for training and 19 for testing).

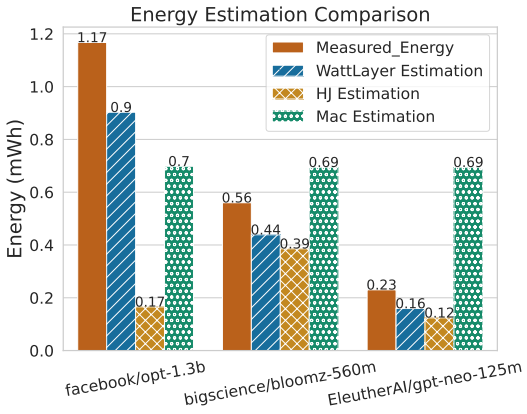


Figure 7: Energy estimation for 3 LLMs architectures facebook/opt-1.3b (1.3B parameters), bigscience/bloomz-560m (560M parameters), and EleutherAI/gpt-neo-125m (125M parameters) using WattLayer and SOTA models on GPU TITAN X for batch size equal to 1.

demands. Collecting dedicated energy measurements for LLMs is costly, as it requires repeated inference of computationally intensive models.

In this setting, the model relies exclusively on the layer-wise estimation functions learned during joint training on vision, text-classification, and audio-classification architectures. No data from text-generation models is included in the training set. We apply the framework to three Transformer-based models from the Transformers library: facebook/opt-1.3b [26] (1.3 billion parameters), bigscience/bloomz-560m [27] (560 million parameters), and EleutherAI/gpt-neo-125m [28, 29] (125 million parameters). Each model generates text from a single prompt, and measurements are repeated according to the protocol described in Section 4.1.

The results, presented in Figure 7, demonstrate that the layer-wise formulation transfers effectively to this previously unseen family of architectures. The estimates leverage layer types encountered during training, including linear layers shared across Vision, NLP, and Audio, embedding layers from text-classification models, and one-dimensional convolutions commonly used in Audio architectures. These findings suggest that the proposed framework captures fundamental hardware-operation relationships that remain invariant across model scales and architectures paradigms. Furthermore, WattLayer outperforms SOTA models, showcasing its potential as a reliable tool for sustainable AI initiatives.

5. Discussion

The proposed estimation model offers several notable advantages. Its modular and flexible design allows for the seamless addition of new parameters, enabling continuous improvement of the method. Furthermore, if new types of layers that significantly impact energy consumption are identified in the future, then specific data can be collected, and a new model be trained to account for these layers. This adaptability ensures that the method remains relevant and can evolve alongside advances in neural network architectures. Additionally, the lightweight nature of the model ensures that its own energy consumption remains minimal, making it efficient and suitable for practical use.

To contextualize these advantages, certain boundaries of the current scope should be noted. While the proposed methodology already demonstrates strong performance across several hardware configurations, it is currently trained for one target machine at a time. This design enables accurate predictions by adapting the model to the characteristics of a specific GPU. Extending the approach to a new platform requires collecting a relatively small amount of additional data, as demonstrated in our cross-GPU ex-

periments. Future work will focus on improving hardware generalization by incorporating GPU-specific descriptors—such as architectural and performance characteristics—similarly to the approach proposed by [30]. Such features could reduce the amount of data required when targeting new devices, improve the estimation of the correction factor α across hardware platforms, and move the framework toward a hardware-agnostic formulation. A broader cross-hardware evaluation of α will also be conducted as part of this effort. In addition, we plan to extend the methodology to distributed systems, enabling energy estimation across multiple machines simultaneously.

Although pruning is accounted for in the model due to its impact on layer shapes, other acceleration techniques such as quantization or sparsification have not been explicitly included in the methodology. Our primary objective is to derive estimation models that are applicable to the widest range of real-world use cases, in particular those relying on widely adopted libraries such as Transformers and Timm, executed on commodity GPUs with workflows that often includes by default the available accelerations. Considering specialized hardware that can skip zero-valued computations or incorporating quantization as a feature in future iterations of the model could further enhance its accuracy and applicability but falls outside the scope of our work. Looking ahead, we also plan to extend our results for more LLMs architectures but also on other generative algorithms, as those models are known to be very energy consuming.

6. Conclusion

We proposed a task independent layer-wise methodology to estimate the energy consumption of AI algorithms. Unlike state-of-the-art models which require specific training or fine-tuning for each task, our approach generalizes across diverse algorithms without additional customization. By incorporating finer-grained layer-wise decomposition, considering critical factors such as batch size, and leveraging a larger dataset for training, our methodology achieves superior accuracy and adaptability across various AI tasks and architectures.

By addressing the lack of standardized tools for energy estimation, WattLayer contributes to the advancement of sustainable AI practices. It empowers developers and organizations to proactively evaluate the energy impact of their models, fostering a culture of transparency and accountability in AI design. This research has the potential to influence both academic and industrial approaches to AI energy management, encouraging the integration of energy efficiency considerations into the life cycle of AI systems. By bridging the gap between energy estimation and practical implementation, WattLayer supports the global effort to reduce the environmental footprint of AI technologies while maintaining innovation and performance.

Acknowledgments

This work is part of the SmartNet challenge project within the Inria–Nokia Joint Lab and has been supported by the French government National Research Agency (ANR) through the UCA JEDI (ANR-15-IDEX-01) and EUR DS4H (ANR-17-EURE-004), by the France 2030 program under Grant Agreements Nos. ANR-23-PECL-0003 and ANR-22-PEFT-0002, through the 3IA Côte d'Azur Investments in the Future project with reference number ANR-23-IACL-0001, and by the European Network of Excellence dAIEDGE under Grant Agreement No. 101120726.

Declaration on Generative AI

During the preparation of this work, the authors used GPT5.1 in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] N. Maslej, L. Fattorini, R. Perrault, Y. Gil, V. Parli, N. Kariuki, E. Capstick, A. Reuel, E. Brynjolfsson, J. Etchemendy, K. Ligett, T. Lyons, J. Manyika, J. C. Niebles, Y. Shoham, R. Wald, T. Walsh, A. Hamrah, L. Santarlasci, J. B. Lotufo, A. Rome, A. Shi, S. Oak, Artificial intelligence index report 2025, 2025. URL: https://arxiv.org/abs/2504.07139. arXiv:2504.07139.

[2] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, J. Dean, The carbon footprint of machine learning training will plateau, then shrink, 2022. URL: http://arxiv.org/abs/2204.05149. doi:10.48550/arXiv.2204.05149. arXiv:2204.05149 [cs].

[3] S. Luccioni, Y. Jernite, E. Strubell, Power hungry processing: Watts driving the cost of AI deployment?, in: Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, Rio de Janeiro, Brazil, 2024, pp. 85–99. doi:10.1145/3630106.3658542.

[4] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. A. Behram, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. S. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, K. Hazelwood, Sustainable AI: Environmental implications, challenges and opportunities, 2022. URL: http://arxiv.org/abs/2111.00364. doi:10.48550/arXiv.2111.00364. arXiv:2111.00364 [cs].

[5] B. De Chateauvieux, E. Pick, D. Ferguson, B. Sisson, Optimize AI/ML workloads for sustainability: Part 3, deployment and monitoring, 2022. URL https://aws.amazon.com/blogs/architecture/optimize-ai-ml-workloads-for-sustainability-part-3-deployment-and-monitoring/.

[6] C. Elsworth, K. Huang, D. Patterson, I. Schneider, R. Sedivy, S. Goodman, B. Townsend, P. Ranganathan, J. Dean, A. Vahdat, B. Gomes, J. Manyika, Measuring the environmental impact of delivering AI at Google scale, 2025. URL: http://arxiv.org/abs/2508.15734. doi:10.48550/arXiv.2508.15734. arXiv:2508.15734 [cs].

[7] MistralAI, Our contribution to a global environmental standard for AI, 2025. URL https://mistral.ai/news/our-contribution-to-a-global-environmental-standard-for-ai.

[8] M. Dubois, M. Annavaram, P. Stenström, Parallel Computer Organization and Design, Cambridge University Press, 2012. doi:10.1017/CBO9781139051224.

[9] C. Rodriguez, L. Degioanni, L. Kameni, R. Vidal, G. Neglia, Evaluating the energy consumption of machine learning: Systematic literature review and experiments, 2024. URL: http://arxiv.org/abs/2408.15128. doi:10.48550/arXiv.2408.15128. arXiv:2408.15128 [cs].

[10] R. Saborido, V. V. Arnaoudova, G. Beltrame, F. Khomh, G. Antoniol, On the impact of sampling frequency on software energy measurements, 2015. URL: https://peerj.com/preprints/1219v2. doi:10.7287/peerj.preprints.1219v2.

[11] Z. Yang, K. Adamek, W. Armour, Part-time power measurements: nvidia-smi's lack of attention, 2024. URL: http://arxiv.org/abs/2312.02741v2. doi:10.48550/arXiv.2312.02741. arXiv:2312.02741 [cs].

[12] D. Li, X. Chen, M. Becchi, Z. Zong, Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs, in: IEEE BDCloud-SocialCom-SustainCom, 2016, pp. 477–484. doi:10.1109/BDCloud-SocialCom-SustainCom.2016.76.

[13] C. Rodrigues, G. Riley, M. Luján, SyNERGY: An energy measurement and prediction framework for convolutional neural networks on Jetson TX1, in: 24th International Conference on Parallel and Distributed Processing Techniques and Applications, 2018.

[14] S. Goel, M. Balakrishnan, R. Sen, EnergyNN: Energy estimation for neural network inference tasks on DPU, in: 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, 2021, pp. 64–68. doi:10.1109/FPL53798.2021.00019.

[15] R. Desislavov, F. Martínez-Plumed, J. Hernández-Orallo, Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning, Sustainable Computing: Informatics and Systems 38 (2023) 100857. doi:10.1016/j.suscom.2023.100857.

[16] J. Getzner, B. Charpentier, S. Günnemann, Accuracy is not the only metric that matters: Estimating the energy consumption of deep learning models, 2023. URL: http://arxiv.org/abs/2304.00897.

doi:10.48550/arXiv.2304.00897. arXiv:2304.00897 [cs].

[17] V. Sze, Y.-H. Chen, T.-J. Yang, J. S. Emer, Efficient processing of deep neural networks: A tutorial and survey, Proceedings of the IEEE 105 (2017) 2295–2329. doi:10.1109/JPROC.2017.2761740.

[18] Z. Yang, W. Armour, The hidden Joules: Evaluating the energy consumption of vision backbones for progress towards more efficient model inference, in: ICML 2025 - 42nd International Conference on Machine Learning, 2025. URL: https://bytez.com/docs/icml/45063/paper.

[19] E. Cai, D.-C. Juan, D. Stamoulis, D. Marculescu, NeuralPower: Predict and deploy energy-efficient convolutional neural networks, 2017. URL: http://arxiv.org/abs/1710.05420. doi:10.48550/arXiv.1710.05420. arXiv:1710.05420 [cs].

[20] J. Zhang, Z. Wang, H. Wang, T. Song, H.-a. Su, R. Chen, Y. Hua, X. Zhou, R. Ma, M. Pan, H. Guan, AMPERE: A generic energy estimation approach for on-device training, SIGMETRICS Perform. Eval. Rev. 53 (2025) 27–32. doi:10.1145/3764944.3764951, aCM SIGMETRICS 2025 Workshop - AI Crossroads: Systems, Energy, and Applications.

[21] 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, M. Stechly, C. Bauer, L. O. N. de Araújo, JPW, MinervaBooks, mlco2/codecarbon: v2.4.1, 2024. URL: https://doi.org/10.5281/zenodo.11171501. doi:10.5281/zenodo.11171501.

[22] R. Wightman, Pytorch image models, https://github.com/rwightman/pytorch-image-models, 2019. doi:10.5281/zenodo.4414861.

[23] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 2020, pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.

[24] TorchVision maintainers, contributors, Torchvision: Pytorch's computer vision library, https://github.com/pytorch/vision, 2016.

[25] K. He, X. Zhang, S. Ren, J. Sun, [resnet] deep residual learning for image recognition, 2015. URL: http://arxiv.org/abs/1512.03385. doi:10.48550/arXiv.1512.03385. arXiv:1512.03385 [cs].

[26] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, L. Zettlemoyer, OPT: Open pre-trained transformer language models, 2022. URL: http://arxiv.org/abs/2205.01068. doi:10.48550/arXiv.2205.01068. arXiv:2205.01068 [cs].

[27] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf, X. Tang, D. Radev, A. F. Aji, K. Almubarak, S. Albanie, Z. Alyafeai, A. Webson, E. Raff, C. Raffel, Crosslingual generalization through multitask finetuning, 2022. URL: http://arxiv.org/abs/2211.01786. doi:10.48550/arXiv.2211.01786.

[28] S. Black, G. Leo, P. Wang, C. Leahy, S. Biderman, GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, 2021. URL: https://doi.org/10.5281/zenodo.5297715. doi:10.5281/zenodo.5297715.

[29] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al., The Pile: An 800GB dataset of diverse text for language modeling, 2020. URL: https://doi.org/10.48550/arXiv.2101.00027. doi:10.48550/arXiv.2101.00027.

[30] M. Z. a. Mayaki, V. Charpenay, Modeling energy consumption in deep learning architectures using power laws, IOS Press, 2025. URL: https://hal.science/hal-04977474. doi:10.3233/FAIA250900.

[31] V. Sovrasov, ptflops: a flops counting tool for neural networks in pytorch framework, 2018-2024. URL: https://github.com/sovrasov/flops-counter.pytorch.

A. Features of the Estimation Model

#MACs (Multiply–Accumulate Operations). The number of MACs measures the computational complexity of a layer and is computed using `ptflops` [31]. It captures the number of arithmetic operations performed during inference. **#Parameters.** The number of parameters corresponds to the total number of learnable weights and biases in a layer. It serves as a proxy for memory footprint and weight-transfer cost during inference. **#Activations.** The number of activations is the total number of output elements produced by a layer (computed as `output.numel()`). It reflects the amount of intermediate data generated and transferred through memory [17]. **Batch Size.** Batch size determines the number of inputs processed simultaneously. Our experiments show that it significantly affects the energy consumption per input and is therefore included as a feature. **Input Size.** Input size represents the dimensionality of the input (e.g., image resolution or prompt length). It directly impacts both computation and memory usage and strongly influences energy consumption.

B. Comparison against SOTA

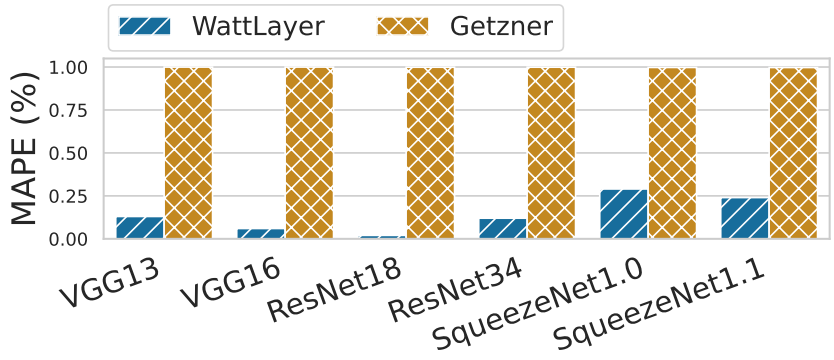


Figure 8: Comparison to SOTA model 'Getzner' [16].