Cross-Machine Comparable Benchmark for Machine Learning Energy Consumption

CS4575 Sustainable Software Engineering - Group 24

Luc Dop
Delft University of Technology
Delft, The Netherlands
L.T.J.Dop@student.tudelft.nl

Sabina Grădinariu
Delft University of Technology
Delft, The Netherlands
S.M.Gradinariu@student.tudelft.nl

Nawmi Nujhat
Delft University of Technology
Delft, The Netherlands
N.Nujhat@student.tudelft.nl

Vincent van Vliet
Delft University of Technology
Delft, The Netherlands
B.V.vanVliet@student.tudelft.nl

***Abstract*—With growing environmental concerns and energy costs, efficiency is a key issue. To reflect on such efficiency, measuring and comparing energy consumption plays an important role, as it allows developers and organizations to identify the most energy-efficient software and hardware combinations. However, energy usage comparisons between systems can be challenging due to the inherent differences between these systems. These include differences in hardware, power measurement tools, the way energy consumption is reported, and background processes. Thus, transparent and standardized benchmarks are needed to motivate simple energy efficiency measurement and comparison. Our paper presents a machine agnostic benchmark built upon the foundation of MLPerf, an industry-standard benchmarking suite for machine learning performance. Energy consumption is measured through an aggregation of EnergiBridge and CarbonTracker measurements, allowing for straightforward comparison of different machine learning models.**

***Index Terms*—sustainable software engineering, computer vision, machine learning, energy efficiency benchmarking**

I. Introduction

As machine learning (ML) continues to advance, the environmental impact of this field has become an increasing concern. Deep learning models [1], including models within the area of computer vision, require substantial computational resources for training and inference, leading to high energy consumption and consequently high carbon footprints [2]. This growing energy demand raises critical questions about the sustainability of ML research and deployment. While model accuracy and efficiency have traditionally been the main focus point of ML research [3], energy consumption remains a relatively new aspect. Existing ML benchmarks primarily evaluate performance metrics like speed and accuracy [4] but often lack detailed energy profiling. Without a standardized methodology to compare machine learning models in terms of energy efficiency, it becomes difficult to optimize ML workloads for sustainability [5].

The environmental impact of machine learning models is substantial and growing. For example, training a single large language model like GPT-3 [6] can produce approximately 552 metric tons of CO_2 equivalent, comparable to the yearly emissions of 120 passenger vehicles [7]. In a similar way, the training process for a large Transformer model using neural architecture search can emit up to 626.155 pounds (approximately 284 metric tons) of CO_2, which is nearly five times the lifetime emissions of an average American car [8]. Even smaller models have a significant impact: training a single BERT [9] base model on GPU is estimated to emit about 652 kg of CO_2, equivalent to a round-trip flight for one passenger between New York and San Francisco [10]. These numbers underscore the immediate need for energy-efficient machine learning practices and the importance of developing standardized benchmarks for measuring and comparing energy consumption across different models and hardware configurations.

For this reason, we propose the development of a cross-system benchmark designed to provide reproducible energy consumption reports that are independent of specific hardware and software environments. The goal is to make energy benchmarking simpler, more accessible, and easier to interpret, especially for smaller workloads or experiments run on low-power devices.

It does this through the standardization of energy consumption measurements across systems by implementing a CPU-intensive workload with support for diverse operating systems and architectures: Windows, Linux, macOS, x86, and ARM (depending on availability).

Energy consumption is measured using built-in tools, such as `nvidia-smi` [11] for GPUs or system-level power

statistics when available. The benchmark automatically selects the most reliable measurement source on the system. To allow fair comparisons between hardware platforms with different performance and power characteristics, we apply normalization techniques such as scaling energy between observed extremes and using FLOPS-based adjustments. The results are stored in standardized formats (e.g., JSON or CSV) for easy analysis.

Our initial experiment focuses on evaluating energy consumption for two widely used computer vision models EfficientNet [12] and Vision Transformers (ViT) [13], on Imagenette, [14], a subset of the ImageNet-1K dataset consisting of 10 classes. The choice is a standard point for evaluating how system parameters affect energy per inference, latency and accuracy levels. By supporting smaller workloads and enabling consistent measurements across platforms, our benchmark fills a gap left by large-scale efforts like MLPerf [15] and supports more sustainable and interpretable ML development.

The further chapters of this paper are as follows: Section II provides background information on energy consumption in ML and existing benchmarking approaches. Section III explains the methodology, including our energy measurement techniques and normalization methods. Section IV presents our results, and in Section V, we discuss our findings, limitations, threats and potential implications. Lastly, Section VI concludes the paper and suggests directions for future work.

II. Related Works

The benchmarking tool known as MLPerf [15] provides reliable assessment capabilities for machine learning. The benchmark system focuses its evaluation process on the run-time speed and accuracy of models while performing image classification, object detection or translation tasks. Its benchmarks are designed for consistent hardware setups where everything is carefully controlled. While MLPerf does include some energy-related measurements, its main goal is performance, especially speed and throughput.

In our project, we propose a different approach. We aim to support flexible, lightweight experiments that can run on any system instead of requiring a strict setup. Our benchmark can work with the tools and data available on each machine, and it includes techniques for comparing results across different hardware, even when conditions vary.

Other projects examine energy use across models and systems, including comparisons based on FLOPS [16], complete training cycles, or scaling between high and low-energy-consuming models. These studies offer useful methods for measuring and interpreting energy data, and we draw inspiration from them in the manner we normalize our results and design our experiments.

In addition, Latenergy [17] presents a framework for real-time energy-efficient inference benchmarking, which is in line with our interest in making energy performance more accessible and interpretable. While their focus is on real-time applications, we adopt some of their ideas such as focusing on per-inference energy and using standardized formats for output, to inform the design of our benchmark.

Latenergy sets itself apart by focusing on low-latency inference tasks and offering detailed energy measurements at the level of individual inference operations. This allows users to assess trade-offs between performance and energy cost. This ability to balance trade-offs is especially helpful in situations when strict energy budgets are important. This trade-off is closely related to our research since it allows for more intelligent decision-making about the trade-off when the user has a better grasp of the energy consumption of models.

III. Methodology

The following section presents the design of our benchmarking framework for evaluating the energy efficiency of computer vision models across different hardware platforms. Our goal is to enable fair and reproducible comparisons of energy usage, even when experiments are run on systems with varying performance and power profiles. We achieve this through a combination of containerized execution, multi-source energy measurement, and normalization techniques that make results cross-machine comparable. The methodology has five key components: benchmark setup, model selection and dataset, evaluation metrics, normalization techniques, and output usability.

A. Benchmark Setup

Our benchmark was created to be cross-platform, capable of running on Windows, macOS, and Linux systems, across both x86 and ARM architectures. In order to keep consistency and minimize external noise, all models were executed inside Docker containers. This decision minimized the impact of background processes and system variability.

Energy consumption was measured using a combination of tools: `EnergyBridge` and `CarbonTracker`. EnergyBridge runs on the host system and logs energy consumption at the hardware level, while CarbonTracker tracks emissions and power draw during model inference from within the container. Depending on hardware availability, we selected the most appropriate energy source dynamically (e.g., `nvidia-smi` for GPUs or CPU-level power readings when available).

B. Models and Dataset

For our experiments, we selected two widely used image classification architectures: EfficientNet[12] and Vision Transformer (ViT)[13]. These models were chosen due to their popularity in energy-aware computer vision tasks and their distinct approaches to optimizing performance. This

experiment used both models without further training on the chosen dataset.

EfficientNet is a family of CNN models that scale depth, width, and resolution in a compound manner to achieve high accuracy with fewer parameters and lower computational cost. It is known for being lightweight and performant, making it a good candidate for energy benchmarking.

Vision Transformer (ViT) represent a shift from traditional convolutional architecture by applying the transformer architecture to image patches. ViTs process images as sequences of patches, enabling global context modeling early in the network. While transformers typically require more computational resources than CNNs, they offer competitive accuracy on large datasets [13]

The models are evaluated using a subset of the ImageNet-1K dataset [14]. ImageNet-1K is a widely used benchmark for classification tasks and offers a lightweight, low-resource workload that makes it ideal for cross-platform evaluation. All images are resized to a resolution of 224×224, matching the expected input dimensions of both models. Models were specifically trained on the ImageNet-1K dataset for 1000 subclasses, so the performance is relatively high.

Since the original MLPerf repository supports only a limited set of models, we extend it by integrating our own inference scripts for EfficientNet and ViT, while preserving MLPerf's logging and measurement structure where applicable.

C. Evaluation Metrics

Our benchmark evaluates models using the following metrics:

- **Total Energy Consumption (J)** — The total amount of energy used during inference, measured using EnergyBridge or CarbonTracker.
- **Latency (ms)** — The time taken per inference.
- **Normalized Energy Score** — Energy consumption normalized against a duo reference baseline (see next subsection).
- **Penalty Score** — A compound metric that increases with high energy usage or latency, useful for detecting outliers.

D. Normalization Techniques

As our focus is to enable a fair comparison across different hardware setups, we used normalization techniques for accuracy and general energy consumption metrics. For energy normalization, we adopt the formula: For energy, we report the average energy consumed per inference:

$$\text{Energy}_{\text{per\_inference}} = \frac{\text{Total Energy Consumption (J)}}{\text{Number of Inferences}}$$

This metric abstracts away differences in batch size or inference time, allowing model comparisons on a per-decision basis. Where relevant, we also compute energy per FLOP to estimate computational efficiency:

$$\text{Energy}_{\text{per\_FLOP}} = \frac{\text{Total Energy Consumption}}{\text{FLOPs}}$$

In addition, we adopt a min-max normalization approach inspired by MLPerf's methodology [15]:

$$\text{Normalized Energy Score} = \frac{E_{\text{model}} - E_{\text{min}}}{E_{\text{max}} - E_{\text{min}}}$$

Here, Emin and Emax are defined from the observed lower and upper bounds of energy usage across all evaluated models. This relative metric helps us compare models on a common scale, mitigating skew from hardware variability.

The preprocessing methods follow a standardized procedure to make the data usable among different models. The pre-trained models require inputs with dimensions 224×224. Hence, they were resized and normalized using Standard normalization technique (mean and standard deviation).

The analysis of energy-performance trade-offs becomes more comprehensive by reporting accuracy as absolute (e.g. top-1 accuracy) values and relative to a chosen baseline model.

The normalization strategies as a whole allow model assessments which maintain independence while producing results that align with actual deployment conditions.

E. Experimental Setup

The experiment is implemented as a stand-alone Python script. Its primary function is to measure the energy consumption of various different implementations of image classification models, using the MLPerf Loadgen interface in combination with CodeCarbon and EnergiBridge. The script is designed to be modular to allow for the input of data, preprocessing functions, and model types given as command-line arguments. Accepted arguments are as follows:

- **model** — Either the path to a local model file, or the model name on Huggingface.
- **dataset** — The path to the dataset to be used for inference.
- **model_type** — Used to designate the file type of the locally stored model, or to state that it is hosted on huggingface. Accepted types are PyTorch, onnx, TensorFlow, and HuggingFace
- **scenario** — The MLPerf scenario type. Specifies the inference workload pattern, to model real-world application settings. Accepted scenarios are SingleStream and Offline.
- **mode** - The MLPerf test mode. Defines the goal of a given test. Accepted modes are PerformanceOnly and AccuracyOnly.
- **preprocess_fn_file** - File path to the custom user-given preprocessing function.

- **task_type** - The type of machine learning task to run. Currently only classification is implemented.
- **flops** - The user-defined number of FLOPs (Floating Point Operations). Used to measure the efficiency of the model.
- **labels_dict** - The path to the directory containing the classification labels for a dataset.
- **energiBridge** - The path to the EnergiBridge executable.
- **model_architecture** - The model architecture to load weights into. This is only relevant for PyTorch models where only the weights are saved.

The core benchmark is build upon MLPerf Loadgen, with which we implement a custom SystemUnderTest (SUT) class. This class loads a pre-trained model from either a file or imports it using Huggingface. It also performs data handling by loading input samples from the provided dataset and applying either a user defined or default preprocessing function. Finally, it performs the inference execution. It receives queries, processes them through the model and returns the results. Loadgen is configured using the command-line arguments 'scenario' and 'mode'.

To capture energy consumption during inference, the script activates two logging tools:

- **CodeCarbon** – Provides estimated energy usage based on CPU/GPU utilization and regional carbon intensity. It is initialized at the start of inference and stopped upon completion.
- **Energibridge** – Offers more precise hardware telemetry by logging CPU/GPU/RAM power draw. It uses a context manager that wraps around the LoadGen execution block and saves metrics to a CSV file.

When the experiment is finished a csv file is generated with the results. These are dependent on the mode of the experiment. In case of performancyOnly mode, it will give the sample id's with the amount of latency in ms for each. In the case of accuracyOnly mode it gives the following:

- **experiment_name** – Name of the model that was tested.
- **model_architecture** – Architecture that was implemented to the weights, if those were given.
- **accuracy** – The accuracy of the model on the given dataset.
- **energy_wh** – The total consumed energy during inference in watt-hours.
- **normalized_energy** – The normalized energy score as mentioned in section 3D.
- **penalty_factor** – The penalty score as mentioned in section 3C.
- **ede_score** – The Energy–Delay–Efficiency (EDE) score. Used to create a composite score that shows how well the model performs both in terms of accuracy and resource consumption.
- **flops** – The amount of FLOPs (Floating Point Operations) given by the user.
- **task_type** – The machine learning task that was performed.
- **timestamp** – The timestamp the experiment finished.

We performed this experiment in PerformancyOnly once for both models on a laptop running Windows and another time for both models on a MacBook Air. We ran the AccuracyOnly expiriment twice for each model on a laptop running Windows and three times for each models on a Macbook Air. The laptop we used was an ASUS Vivobook with the following specifications:

- Windows 11 home.
- 12th Gen Intel(R) Core(TM) i7-12650H 2.30 GHz processor
- 16 GB RAM.
- Brightness level of 30
- Resolution of 1920 x 1200.
- Wifi turned off.

The Macbook Air we used for the experiments had these specifications:

- macOS Sonoma 14.6.1
- Apple M3 chip
- 16 GB RAM.
- Full brightness level with auto-adjustment turned off
- Resolution of 2560 x 1664
- Power-saving mode turned off
- Wifi turned off.

Further instructions can be found in the `README.md` found in `vision/general` in the repository (see Section A).

IV. Results

This section presents the results of our energy benchmarking experiments across different CNNs and hardware configurations. We analyze the performance of EfficientNet and ViT on ImageNet-1k dataset, reduced to 10 classes for faster experimentation, focusing on key metrics such as energy consumption, inference latency, and accuracy. By applying our normalization techniques, we provide a cross-platform comparison highlighting the trade-offs between energy efficiency and model performance. The results are discussed in the context of both absolute measurements and relative scores to ensure hardware-independent insights.

On Windows, we see an inverse trend(Figure5): ViT is doing better in terms of accuracy, but at obviously higher cost. This suggests that platform-specific factors, such as driver optimization, system background processes, influence energy efficiency and model behavior. Overall, the accuracy vs. energy plots reveal that EfficientNet maintains, overall a better balance between resource usage and output quality across platforms. These results support our benchmark's goal: enabling fair and reproducible energy-performance trade-offs

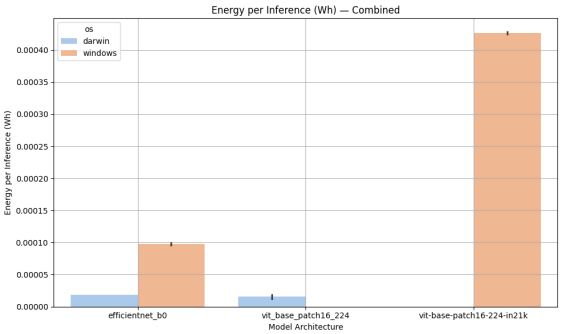

Fig. 1: Energy per inference for EfficientNet and ViT across macOS and Windows platforms.

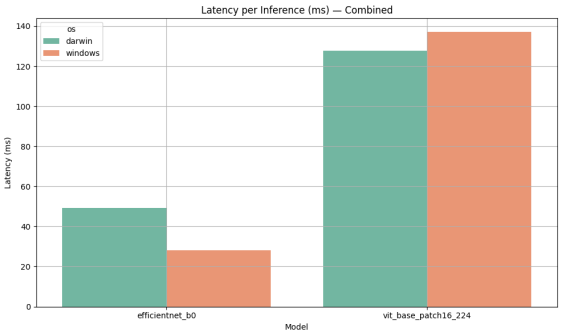

Fig. 2: Latency per inference (in milliseconds) averaged across platforms.

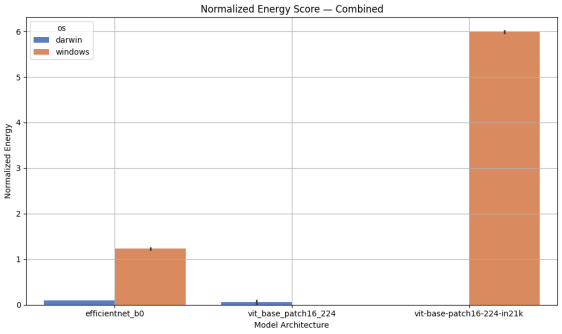

Fig. 3: Normalized energy scores based on FLOPs and energy consumption across platforms.

through normalized metrics and multi-platform analysis.

On macOS (Darwin), EfficientNet-B0 consistently outperforms ViT in energy efficiency, latency, and accuracy. It achieves slightly higher accuracy (99.86%) while consuming moderately more energy per inference. ViT, despite its transformer-based architecture, shows slightly lower accuracy and better energy usage per inference, but falls short in overall normalized energy score. Latency measurements reveal that EfficientNet offers faster inference with lower variability, while ViT suffers from increased computational demands. MLPerf's LoadGen ensures consistent query evaluation,

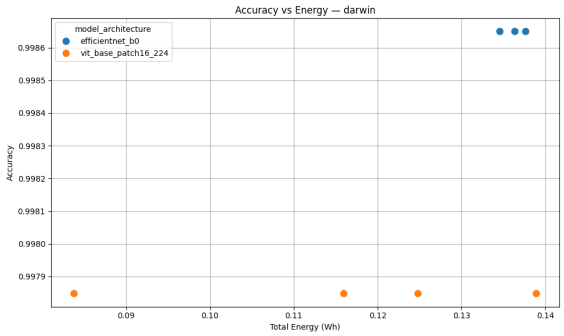

Fig. 4: Accuracy vs. Total Energy for macOS (darwin).

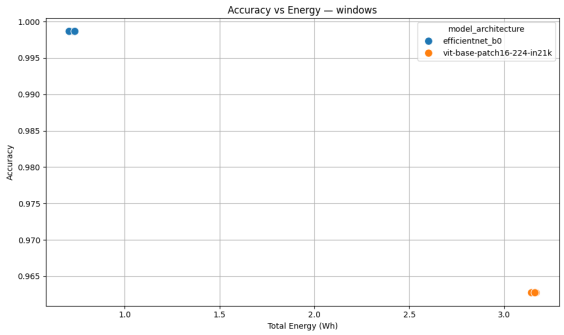

Fig. 5: Accuracy vs. Total Energy for Windows.

allowing Darwin's optimized hardware and software stack, especially on Apple Silicon, to showcase EfficientNet's strengths in real-time, energy-aware scenarios.

Different FLOPs values were assigned to each model to reflect their inherent architectural complexity and enable fair energy normalization. EfficientNet-B0 was set to approximately 390 million FLOPs, aligning with its lightweight convolutional design optimized for mobile inference. In contrast, ViT-base (patch16-224) was assigned 17 billion FLOPs to account for its transformer-based architecture, which includes heavy matrix multiplications and attention mechanisms. These FLOPs estimates allowed MLPerf's normalized energy score to adjust for computational load, ensuring comparisons reflect efficiency relative to model size and not just raw energy usage.

V. Discussion

This section will discuss the interpretation of the results, as well as the limitations of our project. Furthermore, threats to validity will be discussed and, finally, the justification and implications will be explored.

A. Interpretation of Results

The results across all figures reveal consistent differences in the energy and performance characteristics of the two evaluated models, EfficientNet and ViT across both macOS and Windows platforms.

Figure 1 shows that EfficientNet requires significantly less energy per inference compared to ViT, which aligns with EfficientNet's lightweight architecture. This makes it a suitable candidate for energy-constrained environments such as mobile or embedded systems.

Latency measurements in Figure 2 indicate that EfficientNet also performs faster inference on average, particularly on macOS. This suggests that EfficientNet maintains both low energy usage and quick responsiveness, while ViT's attention-based architecture introduces higher computational overhead.

Figure 3 further highlights these trends using normalized energy scores. EfficientNet consistently scores higher, indicating that our normalization technique effectively enables the benchmarking. Even when accounting for FLOPs and hardware variability, EfficientNet remains more energy efficient.

Figures 4 and 5 present the trade-off between accuracy and total energy consumption. Despite neither model being fine-tuned, EfficientNet demonstrates a more favourable energy-to-accuracy ratio on both platforms. ViT consumes more energy for only marginal improvements—or even lower accuracy in some cases—making it less optimal in low-power scenarios.

Taken together, these results reinforce the notion that model architecture has a substantial impact on energy and latency. EfficientNet emerges as the more sustainable and practical choice for deployment in diverse environments, while ViT may be justified in scenarios where transformer-based performance is critical and energy is less constrained.

B. Limitations

Our implementation is an extension of the MLPerf Inference repository, but it is entirely separate from MLCommons' MLPerf Training benchmark[1]. For this reason, our implementation is meant solely for inference tasks and lacks the opportunity to work with models that still require proper training. The implementations from the MLPerf Training benchmark are measured by the time to train to a defined quality target [18], while the inference benchmark focuses on how efficiently the system can perform inference. To properly compare models, it would also be necessary to take into account the time it takes to train those models on the inference data set, which was beyond the scope of this project.

Another important limitation is the lack of a warm-up phase prior to energy measurement. In real-world deployment, machine learning systems often experience a warm-up period during which hardware resources ramp up to stable operating conditions [19]. During this phase, energy consumption can fluctuate significantly due to the cold-start of the hardware. Since our benchmark begins measuring energy consumption immediately with the first inference request, the results may include consequences from cold-start overhead, such as unusually high power usage or slower inference times in the first few iterations. This can lead to results that are not representative of the actual energy consumptions, especially for short runs or single-stream scenarios. Future iterations of our benchmark could incorporate a defined warm-up period to ensure more stable and realistic measurement conditions [20]. This recommendation is consistent with best practices observed in other studies like Wang et al. (2020)[21], which emphasize the need for system stabilization before collecting performance and energy data.

[1]https://github.com/mlcommons/training

C. Threats to Validity

Our study considers three main types of threats to validity as described by Cook and Campbell [22]: internal, external, and construct validity.

Internal validity is threatened by the restricted set of hardware platforms tested. While our benchmark is cross-platform by design, actual results were gathered from only a few systems due to time constraints. This impacts the generalizability of our findings.

Construct validity could be influenced by potential inconsistencies in power measurement tools (e.g., different sampling rates, or metrics between CarbonTracker [23] and EnergyBridge [24]). We selected the suitable measurements per system, but discrepancies may still exist.

D. Justification and Mitigation

We can justify our choice of tools and normalization methods based on simplicity, reproducibility, and compatibility across systems. To mitigate variability across devices, we used containerized execution and normalization by FLOPs and per-inference metrics. Time constraint was also a basis of our choices.

VI. Conclusion

A. Future work

As mentioned in Section V, our current implementation only made use of the MLPerf Inference benchmark. In the future, an additional integration of the MLPerf Training benchmark could be useful. This additional feature would need a proper plan for integrating both repositories, but the possible outcome could be a stronger benchmark that measures energy consumption in an accurate way during both the training and inference steps of a machine learning pipeline. Ultimately, this would allow for a more proper comparison between models.

One possible improvement for future iterations of this project is the use of containerization. Due to time constraints, we were

unable to package the benchmarking tool in a Docker container. Providing a containerized version of the project would greatly improve usability and reproducibility by allowing users to easily run the benchmark on their own systems without dealing with dependency or environment issues. Containerization would also allow for more consistent benchmarking across machines by minimizing configuration drift and make it easier to deploy the tool in cloud or edge environments. For these reasons, we believe this could be a strong feature to add in the future.

B. Conclusion

This paper proposed a novel extension of the MLPerf Inference benchmark, designed to evaluate machine learning models in a machine-agnostic and energy-aware manner. While traditional benchmarks often overlook hardware variability and sustainability considerations, our approach emphasizes reproducibility and energy efficiency across diverse computing environments. This method allows for a simplified comparison of different machine learning models by allowing the user to upload their own models and datasets to execute the benchmark on. Although other features could be implemented in the future, this paper represents a step towards machine-agnostic ML model comparison in the sustainable software field.

Appendix

A. Link to repository

The source code from this paper can be found here. This repository is a fork of the official MLCommons Inference Benchmark repository and has been extended to include additional energy measurement and normalization functionality.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[2] Lucía Bouza, Aurélie Bugeau, and Loıc Lannelongue. "How to estimate carbon footprint when training deep learning models? A guide and review". In: *Environmental Research Communications* 5.11 (2023), p. 115014.

[3] Lukas Budach et al. "The effects of data quality on machine learning performance". In: *arXiv preprint arXiv:2207.14529* (2022).

[4] Jeyan Thiyagalingam et al. "Scientific machine learning benchmarks". In: *Nature Reviews Physics* 4.6 (2022), pp. 413–420.

[5] Peter Henderson et al. "Towards the systematic reporting of the energy and carbon footprints of machine learning". In: *Journal of Machine Learning Research* 21.248 (2020), pp. 1–43.

[6] Tom B Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[7] Tom B Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1877–1901.

[8] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3645–3650.

[9] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[10] David Patterson et al. "Carbon Emissions and Large Neural Network Training". In: *arXiv preprint arXiv:2104.10350* (2021).

[11] NVIDIA Corporation. *NVIDIA System Management Interface*. https : / / developer . nvidia . com / nvidia - system-management-interface. Accessed: YYYY-MM-DD. 2023.

[12] Mingxing Tan and Quoc V Le. "EfficientNet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 6105–6114.

[13] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[14] Jeremy Howard. *Imagenette: A smaller subset of ImageNet for quick experiments*. https://github.com/fastai/imagenette. Accessed: 2025-04-04. 2019.

[15] Vijay Janapa Reddi et al. "Mlperf inference benchmark". In: *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE. 2020, pp. 446–459.

[16] Constance Douwes and Romain Serizel. "Normalizing Energy Consumption for Hardware-Independent Evaluation". In: *arXiv preprint arXiv:2409.05602* (2024). URL: https://doi.org/10.48550/arXiv.2409.05602.

[17] Jason M. Pittman. "Latenergy: Model Agnostic Latency and Energy Consumption Prediction for Binary Classifiers". In: *arXiv preprint arXiv:2412.19241* (2024). URL: https://doi.org/10.48550/arXiv.2412.19241.

[18] Peter Mattson et al. "Mlperf training benchmark". In: *Proceedings of Machine Learning and Systems* 2 (2020), pp. 336–349.

[19] Ana Carolina Moises, Andreia Malucelli, and Sheila Reinehr. "Practices of energy consumption for sustainable software engineering". In: *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*. IEEE. 2018, pp. 1–6.

[20] Luís Cruz. *Scientific Guide for Reliable Energy Experiments*. Lecture CS4575 Sustainable Software Engineering. TU Delft, 2025.

[21] Gu-Yeon Wang, Kim Hazelwood, David Hsu, et al. "Benchmarking Deep Learning Inference: Characteriz-

ing and Optimizing DL Inference Workloads at Datacenter Scale". In: *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*. 2020, pp. 88–98.

[22] Janet Siegmund, Norbert Siegmund, and Sven Apel. "Views on internal and external validity in empirical software engineering". In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. IEEE. 2015, pp. 9–19.

[23] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models". In: *arXiv preprint arXiv:2007.03051* (2020).

[24] Jie Chen et al. "EnergyBridge: A Mobile-Cloud Platform for Energy Analysis and Optimization". In: *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE. 2022, pp. 1–10.