Capturing the Carbon Impact of Deep Learning

Alexis Corona, Sanmukh Kuppannagari
Department of Computer and Data Sciences, Case Western Reserve University
Contact: alexiscorona606@gmail.com, sxk1942@case.edu

***Abstract*—Modern Deep Learning training is extremely carbon and energy intensive. Existing tools to evaluate carbon and energy consumption are either too coarse-grained, making them inaccurate or too sophisticated, making them inaccessible. This work presents a framework that enables users to capture accurate carbon and energy consumption of their deep learning training runs.**

***Index Terms*—Deep Learning; Carbon Impact; Profiling**

I. Introduction

As machine learning models are becoming more widely integrated in society, it is important to be aware of their growing costs. For example, GPT-3, a language model, released a predicted 502 tonnes of carbon emissions during its training. [1] The developers of more expensive models such as Gemini Ultra and GPT-4 have not released carbon emissions data, but of the data reported in 2023, the most carbon-intensive machine learning model released almost 300 tonnes of CO2, and consumed 400 MWh. [1]

Tools such as CodeCarbon [2] and the Machine Learning Emissions Calculator [3] exist to better understand and reduce the environmental impact of high-performance computing. However, they are either not well integrated with existing training pipelines [2], requiring time consuming manual integration, or they require users to input metadata such as computing hours and device information [3], leading to coarse-grained inaccurate predictions. A framework to automatically collect and report energy usage and carbon emissions during the training process can significantly improve the availability of environmental impact data.

Thus, we propose a framework for seamlessly enabling users to measure the carbon impact of their machine learning models. Our tool allows a user to select a model and dataset, automatically preprocess their data, fine-tune for multiple tasks, and output energy and carbon usage information with their trained model. It can be launched within an HPC environment or a local platform and has an an easy to use web-based framework. Our hope is that our tool makes machine-learning models easy to train while also making users more aware of the environmental impact associated with computing time.

II. Background

A. CodeCarbon

CodeCarbon is a tool for tracking carbon emissions while running code. Designed to be as light as possible, it measures the power consumption of the system's GPU, CPU, and RAM every 15 seconds, to track electricity used by computer hardware. Afterward, it multiplies this total electricity consumed by the "carbon intensity" of the region - the average amount of CO2 produced per kilowatt-hour. [2]

With CodeCarbon installed, a decorator can be applied to a python function to track the emissions generated while it executes. An emission-tracking object can also be put into a function, started and stopped manually, and CodeCarbon can be started from the command line to track carbon emissions system-wide.

B. Relevant Related Works

The Machine Learning Emissions Calculator is a tool for estimating the raw carbon emissions produced while training a machine learning model. In order to get this estimation, it takes as input the hardware type, hours used, cloud provider, and the region where the computation is being performed. While useful, this tool is only capable of estimating carbon emissions and carbon offset, as opposed to measuring the data as the training is being done. [3]

Microsoft Azure cloud tools have been used to track the carbon emissions of various machine learning models for natural language processing and image analysis, and while these measurements were not done with the same technologies that our framework uses, they provide valuable insight into cloud-based machine learning energy usage and carbon output. [4] It's also important to be aware that carbon emissions can vary greatly from region to region. For example, a model trained in the United States has considerably more environmental impact than a model trained in France, due to the areas' differing carbon intensity. [4]

III. Framework

Our framework allows a user to load machine learning models and datasets directly from Hugging Face and train for multiple language processing tasks, including text classification, language modeling (casual and masked), token classification, extractive QA, translation, and summarization. The program can be launched either on an HPC environment or a local platform - the setup process (utilizing TensorFlow and Pytorch) is able to analyze its environment and adjust the functionality accordingly.

The model and dataset are specified by the user (via a link to the Hugging Face website) and downloaded automatically using the transformers library. The user is also required to select the kind of training they want to perform, and they may be required to input more information about the dataset. For example, if the data is split into multiple subsets, they may have to select which subset of the data they want to train with. Certain tasks require other input, such as block size for language modeling or source and target languages for

Our framework is available at: https://github.com/KLab-AI3/ai-carbon

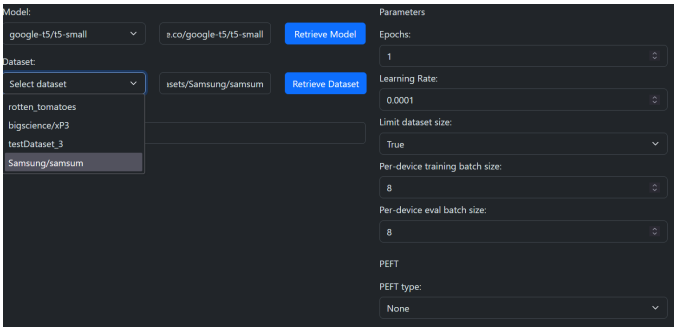

Fig. 1: User Interface

translation. Finally, the user can adjust different parameters, including the number of epochs, the learning rate, the training batch size, evaluation batch size, and PEFT type.

Dataset preprocessing is performed automatically based on the task. For example, during text classification, the dataset's features, labels, and keys are all detected by their contents position within the dataset if the user declines to specify them. Afterwards, they are put into the proper format for the model selected. This might require conversion between data types (such as boolean values into int values), renaming columns, or combining labels into matrices. During language modeling, texts are grouped and split according to the maximum length that the model can effectively train with before being encoded. Each task preprocesses data differently - our goal was to make the framework easy to use as possible for a wide variety of datasets and models.

For measuring the carbon impact of these machine learning models, our implementation uses a callback function and the CodeCarbon library to track energy. CodeCarbon allows us to compute the carbon emissions based on carbon intensity and geolocation data. Our framework then displays the carbon emission data in a downloadable CSV format within our frontend application.

IV. Experimental Evaluations

To test our framework, we fine-tuned several different machine learning models for multiple natural language processing tasks. These tests were done with smaller datasets of only 2000 entries each. Our local platform utilizes an NVIDIA GeForce GTX 1660 GPU and an Intel Core i7-9750H processor, and all of these tests were performed in the Central United States within the SPNO eGRID subregion, which has an above-average emission rate for CO2 compared to the national average. While the emissions presented in this section are dependent on the carbon intensity of the region, it is possible calculate the carbon emissions of other regions as well from the raw power usage data dependent on our hardware. The models were selected for each task based on their popularity on Hugging Face. Table I demonstrates the results.

Model	Task	Duration (minutes)	Energy Usage (kWh)	Carbon Emissions (g)
t5-small	Summarization	49.02	0.0229	12.43
distilbert-base-uncased	Text Classification	7.78	0.0036	1.98
opus-mt	Translation	17.17	0.008	4.36
distilbert-base-uncased	Extractive QA	54.73	0.0256	13.87
distilbert-base-uncased	Token Classification	6.3	0.0029	1.6
distilled-gpt2	Language Modeling	15.1	0.007	3.8

TABLE I: Carbon Emissions for Fine-Tuning

V. Conclusion

We have developed an easy to implement framework for training and fine-tuning machine learning models for a variety of different natural language processing tasks. We hope that our framework can allow AI developers to experiment with different models and parameters to better understand and minimize the carbon impact of their work. The manufacture, transportation, and end-of-life phases for the computer hardware can also impact a model's carbon footprint [5], but these areas are outside the scope of our framework.

While our tests serve to demonstrate the implementation, our framework could be used to conduct a more comprehensive evaluation by comparing deep learning models' energy usage with larger datasets, different computational resources, and different PEFT parameters run over a longer period of time. We plan to support deep learning models other than natural language processing in the future iterations.

Acknowledgements

This research was supported in part by NSF Award 2425535. Additionally, this work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. We also thank Fletcher Li, Samuel Cook, Xochitl Villalvazo, and Joel Hottinger for their work on the implementation.

References

[1] N. Maslej, L. Fattorini, R. Perrault, V. Parli, A. Reuel, E. Brynjolfsson, J. Etchemendy, K. Ligett, T. Lyons, J. Manyika, J. C. Niebles, Y. Shoham, R. Wald, and J. Clark, "Artificial intelligence index report 2024," 2024. [Online]. Available: https://arxiv.org/abs/2405.19522

[2] 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. Stechły, C. Bauer, Lucas-Otavio, JPW, and MinervaBooks, "mlco2/codecarbon: v2.4.1," May 2024. [Online]. Available: https://doi.org/10.5281/zenodo.11171501

[3] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," *arXiv preprint arXiv:1910.09700*, 2019.

[4] J. Dodge, T. Prewitt, R. T. D. Combes, E. Odmark, R. Schwartz, E. Strubell, A. S. Luccioni, N. A. Smith, N. DeCario, and W. Buchanan, "Measuring the carbon intensity of ai in cloud instances," 2022. [Online]. Available: https://arxiv.org/abs/2206.05229

[5] A. Bichsel and G. Skladman, "Microsoft cloud for sustainability api calculation methodology," 08 2024. [Online]. Available: https://learn.microsoft.com/en-us/industry/sustainability/api-calculation-method