Assessing the Environmental Impact of ML Application

PROGRAMMING 6

Romana Mahjabin Eshita

1 Introduction

Machine learning (ML) applications have rapidly grown in complexity and scale, raising concerns about their environmental impact. This report evaluates the carbon footprint of different ML approaches, focusing on data processing tools and model architecture choices. For the data processing tools, I have selected my ML project predicting the histology of lung cancer using clinical and gene expression data [1]. For the larger dataset, I have used the MNIST [5] handwritten digit dataset. I analyzed the environmental impact of the data processing using Polars and Pandas libraries and digit recognition using Convolutional Neural Networks (CNNs) and identified strategies that can balance the higher performance with less environmental impact.

Part 1: Impact of ML applications Data Processing with Polars and Pandas

Data processing is the crucial part of any ML application. The less carbon footprint it produces, the better it would be for the environment. In this part, we analyzed the data processing of the lung cancer dataset. We compared the same data processing of two different tools, like pandas and polars. Pandas is a widely used data processing and manipulation library in Python, known for its flexibility and ease of use. On the other hand, Polars is a fast DataFrame library designed for high-performance data processing, especially efficient with larger datasets.

Dataset:

- **Clinical Data:** There are 89 samples of lung cancer patients' data in CSV format, including 16 clinical variables such as tumor source location and tumor stage.
- **Gene Expression Data:** It is a matrix format data containing normalized expression levels of 66,000 genes for 89 samples.

Hardware Setup:

- **CPU Model:** Intel(R) Xeon(R) Gold 6248 @ 2.50GHz
- **CPU Count:** 80
- **RAM:** 880.35 GB

Environmental Impact Assessment: Table 1 represents the data processing time and environmental effect for pandas and polars for the lung cancer datasets.

Tool	Run Time (s)	Emissions (gCO_2)
Pandas	17.58	0.54
Polars	9.19	0.32

Table 1: Comparison of Processing Time and Carbon Emissions for Pandas and Polars

From the experiment, we could see that polars processed the dataset almost twice as fast as Pandas. Polars took 9.19s; on the other hand, pandas took 17.58s. Polars emitted 0.32 g, and pandas emitted 0.54 g of carbon. In both cases, polars demonstrated both faster processing and lower carbon emissions compared to Pandas, indicating that optimized multi-threaded processing can enhance sustainability. In this case, changing the pandas with polars would be best, as it serves the purpose and has less impact on the environment. However, from this paper[3], we know that multiprocessing is not always the most carbon efficient option, as the benefits depend on workload characteristics. We always need to balance the processing time with the multiprocessors to get the optimal time and core, which emits less carbon.

Part 2: Digit Recognition with Convolutional Neural Networks (CNN)

Model Description

Our ML application focuses on a convolutional neural network (CNN) model for digit recognition using the MNIST dataset. A convolutional neural network (CNN) is a regularized feedforward neural network that learns features on its own using filter (or kernel) optimization. This deep learning network has been used to process and predict a variety of data types, including text, pictures, and audio [4]. CNNs are highly effective for digit recognition due to their specialized architecture designed to process visual data. But it also requires significant computational resources and a large amount of data to train and achieve high accuracy. The MNIST dataset contains 60,000 training images and 10,000 test images, each of size 28x28 pixels. The study compared three CNN models of varying sizes:

- **Small CNN:** 118,538 trainable parameters
- **Medium CNN:** 831,242 trainable parameters
- **Large CNN:** 1,985,610 trainable parameters

Experiment 1: Comparison of carbon emissions in different model sizes

In this experiment, we aimed to see how varying neural network model sizes impact both accuracy and carbon emissions. Our objective was to identify the model size that delivers optimal accuracy with minimal environmental impact. To make a fair comparison, we kept the following for all experiments.

- optimizer (SGD)
- learning rate (1e-3)
- batch size (20)
- number of training rounds (5 epochs)

We didn't use any techniques to prevent overfitting. This simple setup let us clearly see how model size alone affects both performance and environmental impact.

Environmental Impact Assessment: From Table 2, we could see that the small CNN model takes less time and emits less carbon. The medium model takes more time than the small model, but accuracy is less than the small model. On the other hand, the small model gives high accuracy but emits the most carbon. This is expected from a large model to emit the most carbon as they have the highest parameters.

Model	Run Time (s)	Emissions (gCO_2)	Accuracy
Small CNN	154	5.28	99.10%
Medium CNN	179	6.14	99.06%
Large CNN	238	8.16	98.85%

Table 2: Comparison of Run Time, Carbon Emissions, and Accuracy Across Different CNN Model Sizes

Experiment 2: Effect of Learning Rate on Carbon Emissions:

In this experiment, we investigated how different learning rates affect carbon emissions and model performance. We hypothesized that extremely small or large learning rates might lead to inefficient training processes, resulting in increased carbon emissions without corresponding performance benefits. We selected a small CNN model for this experiment. We tested three distinct learning rates while keeping all other parameters constant, including model architecture, batch size, and evaluation metrics. Each model was trained with early stopping with a patience of 2 and a maximum of 10 epochs.

Environmental Impact Assessment: As shown in Table 3, learning rate selection impacts both training efficiency and environmental footprint. The smallest learning rate (1e-10) produced the lowest emissions at 3.16 gCO_2 while achieving 99.09% accuracy. The moderate learning rate (1e-5) achieved the highest accuracy of 99.68% but at a higher environmental cost—more than triple the emissions of the smallest learning rate. Most notably, the largest learning rate (3e-1) not only failed to converge with only 10.2% accuracy but also generated emissions comparable to the moderate learning rate. This shows how improperly selected hyperparameters can result in unsatisfactory performance and computational waste.

Learning Rate	Run Time (s)	Emissions (gCO_2)	Accuracy
1e-10	92.12	3.16	99.09%
1e-5	307.72	10.54	99.68%
3e-1	304.88	10.44	10.2%

Table 3: Effect of Learning Rate on Run Time, Carbon Emissions, and Accuracy

Experiment 3: Optimizer Comparison:

This experiment compared the environmental impact and performance of two popular optimization algorithms: Stochastic Gradient Descent (SGD) and Adam. Our hypothesis was that Adam's adaptive learning rate mechanism might converge faster than SGD, potentially reducing computational requirements and associated carbon emissions. We maintained identical model architecture, hyperparameters, and training conditions for both optimizers.

Environmental Impact Assessment: The results in table 4 show that the Adam optimizer achieved a slightly higher accuracy compared to stochastic gradient descent (SGD). However, this improvement came with a 0.05 g increase in carbon emissions and a negligible difference in run time. For this particular task, choosing between SGD and Adam doesn't really impact the environmental footprint. But in situations where every bit of accuracy counts, Adam's small performance edge might be worth its slightly higher emissions.

Optimizer	Run Time (s)	Emissions (gCO_2)	Accuracy
Stochastic	154	5.29	99.22%
Adam	156	5.34	99.56%

Table 4: Comparison of Run Time, Carbon Emissions, and Accuracy Between Stochastic and Adam Optimizers

Experiment 4: Early Stopping vs. No Early Stopping

In our last experiment, we tested early stopping—a technique that automatically ends training when the model stops improving. We expected that early stopping would considerably cut carbon emis-

sions by preventing unnecessary computation cycles, potentially with minimal impact on final model accuracy. We compared identical model architectures and hyperparameters with and without early stopping, measuring the resulting differences in training time, carbon emissions, and final accuracy.

Environmental Impact Assessment: Table 5 shows that early stopping cut carbon emissions by almost 60% from 14.35 to 5.80 (gCO_2) and training time by about 60% from 419 to 169 seconds. These findings position early stopping as an effective strategy for reducing the environmental footprint

Early Stopping	**Run Time (s)**	**Emissions (gCO_2)**	**Accuracy**
With	169.37	5.80	99.3%
Without	419.00	14.35	99.51%

Table 5: Comparison of Run Time, Carbon Emissions, and Accuracy With and Without Early Stopping

of machine learning applications. The slight trade-off in performance is outweighed by the significant reductions in training time and carbon emissions. This technique is especially beneficial for large models or extensive datasets, where energy savings can be even more pronounced.

Conclusion

From our anaanalysis, we could see that minor technical decisions in machine learning (ML) application development can have significant environmental impacts. We can significantly minimize the carbon footprint of ML applications by selecting efficient tools, appropriately sized models, and optimized training processes.

Using efficient data processing libraries, selecting smaller model architectures, implementing early stopping, and carefully selecting hyperparameters are some of the most effective strategy . These improvements, when combined, have the potential to reduce carbon emissions by 60-70% compared to traditional strategies that prioritize performance alone.

These findings align with the "Ten Simple Rules" proposed by this paper [2], particularly their emphasis on algorithm efficiency, appropriate hardware selection, and measuring environmental impact. By integrating these considerations into the ML development workflow, we can create more sustainable ML applications that balance performance with environmental responsibility.

References

[1] Hugo J. W. L. Aerts, Emmanuel Rios Velazquez, Ralph T. H. Leijenaar, Chintan Parmar, Patrick Grossmann, Sara Carvalho, Johan Bussink, René Monshouwer, Benjamin Haibe-Kains, Derek Rietveld, Frank Hoebers, Michelle M. Rietbergen, C. René Leemans, Andre Dekker, John Quackenbush, Robert J. Gillies, and Philippe Lambin. Data from nsclc-radiomics-genomics, 2015.

[2] Loïc Lannelongue, Jason Grealey, Alex Bateman, and Michael Inouye. Ten simple rules to make your computing more environmentally sustainable. *PLOS Computational Biology*, 17(9):e1009324, September 2021.

[3] Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green algorithms: Quantifying the carbon footprint of computation. *Advanced Science*, 8(12), May 2021.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[5] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. Mnist handwritten digit database. *ATT Labs [Online]*, 2, 2010.