An Analysis of Optimizer Choice on Energy Efficiency and Performance in Neural Network Training

Tom Almog
University of Waterloo
Waterloo, Ontario, Canada
Email: talmog@uwaterloo.ca

***Abstract*—As machine learning models grow increasingly complex and computationally demanding, understanding the environmental impact of training decisions becomes critical for sustainable AI development. This paper presents a comprehensive empirical study investigating the relationship between optimizer choice and energy efficiency in neural network training. We conducted 360 controlled experiments across three benchmark datasets (MNIST, CIFAR-10, CIFAR-100) using eight popular optimizers (SGD, Adam, AdamW, RMSprop, Adagrad, Adadelta, Adamax, NAdam) with 15 random seeds each. Using CodeCarbon for precise energy tracking on Apple M1 Pro hardware, we measured training duration, peak memory usage, CO_2 emissions, and final model performance. Our findings reveal substantial trade-offs between training speed, accuracy, and environmental impact that vary across datasets and model complexity. We identify AdamW and NAdam as consistently efficient choices, while SGD demonstrates superior performance on complex datasets despite higher emissions. These results provide actionable insights for practitioners seeking to balance performance and sustainability in machine learning workflows.**

***Index Terms*—Energy efficiency, Machine learning, Optimizers, Sustainable AI, Carbon emissions, Deep learning**

I. Introduction

The rapid advancement of machine learning (ML) has led to increasingly complex models that require substantial computational resources for training. Recent studies estimate that training large language models can produce carbon emissions equivalent to hundreds of transatlantic flights [1], raising critical questions about the environmental sustainability of current ML practices. As the field moves toward more responsible AI development, understanding the energy implications of fundamental training decisions becomes paramount.

Optimizer choice represents one of the most fundamental decisions in neural network training, directly affecting convergence speed, final performance, and computational requirements. While extensive research has focused on optimizing for accuracy and training speed, the environmental implications of different optimization algorithms remain understudied. This gap is particularly concerning given that optimizer selection can significantly impact both training duration and computational intensity.

Previous studies on ML energy efficiency have primarily focused on model architecture design [2] or hardware utilization [3]. However, few studies have systematically examined how optimizer choice affects energy consumption across different problem complexities and dataset scales. The limited existing work [4] has been constrained by small-scale experiments or limited optimizer coverage.

A. Contributions

This paper makes several key contributions to understanding the energy efficiency implications of optimizer choice:

- **Comprehensive empirical evaluation**: We present results from 360 controlled experiments across three datasets with varying complexity, using eight popular optimizers with 15 random seeds per configuration.
- **Multi-dimensional analysis**: We examine the relationships between optimizer choice and multiple metrics including accuracy, training time, CO_2 emissions, memory usage, and convergence behavior.
- **Robust evaluation**: We use 15 random seeds per configuration to ensure reliable and reproducible results, addressing a key limitation in prior work.
- **Practical guidelines**: We provide actionable recommendations for practitioners to make informed optimizer choices considering both performance and environmental impact.
- **Reproducible methodology**: We present detailed experimental protocols and make our data collection framework available to enable replication and extension of this work.

II. Related Work

A. Energy Efficiency in Machine Learning

The environmental impact of machine learning has gained increasing attention in recent years. Strubell et al. [1] highlighted the substantial carbon footprint of training large neural networks, estimating that training a transformer model with neural architecture search produces approximately 284,000 kg of CO_2 equivalent. This seminal work catalyzed broader interest in sustainable ML practices.

Patterson et al. [3] provided comprehensive guidelines for measuring and reporting carbon emissions in ML research, emphasizing the importance of transparency in environmental impact reporting. Their work established methodological standards that we adopt in this study.

Schwartz et al. [5] introduced the concept of "Green AI," advocating for research that considers computational efficiency

alongside accuracy. They proposed metrics for evaluating the efficiency of ML models and called for greater emphasis on environmentally conscious research practices.

B. Optimizer Performance Analysis

Optimization algorithms are fundamental to neural network training, with extensive research comparing their performance characteristics. Ruder [6] provided a comprehensive survey of gradient descent optimization algorithms, establishing the theoretical foundations for understanding optimizer behavior.

Schmidt et al. [7] conducted empirical comparisons of popular optimizers across computer vision tasks, focusing primarily on convergence speed and final accuracy. However, their work did not consider energy consumption or environmental impact.

Choi et al. [8] examined optimizer performance across different network architectures, finding that optimal choices vary significantly based on model complexity and dataset characteristics. Our work extends this finding to include environmental considerations.

C. Sustainable Computing and Green Software

The broader computer science community has increasingly emphasized sustainable computing practices. Koomey et al. [9] established early frameworks for measuring computational energy consumption, providing methodological foundations for modern energy-aware computing research.

In the software engineering domain, Pinto and Castor [10] demonstrated that seemingly minor algorithmic choices can have significant energy implications, supporting our hypothesis that optimizer selection may substantially impact training energy consumption.

III. Methodology

A. Experimental Design

Our experimental design follows a factorial approach with three primary factors: dataset (3 levels), optimizer (8 levels), and random seed (15 levels), resulting in 360 total experiments. This design enables robust analysis while controlling for various sources of variability.

1) Datasets and Model Architectures: We selected three benchmark datasets representing different complexity levels:

- **MNIST**: Handwritten digit classification (60,000 training samples, 28×28 grayscale images). We used a simple CNN with 421,642 parameters consisting of two convolutional layers followed by two fully connected layers.
- **CIFAR-10**: Natural image classification (50,000 training samples, 32×32 RGB images, 10 classes). We employed a modern CNN architecture with 3,249,994 parameters incorporating batch normalization and dropout for improved training stability.
- **CIFAR-100**: Natural image classification (50,000 training samples, 32×32 RGB images, 100 classes). We used the same architecture as CIFAR-10 but modified the output layer, resulting in 3,296,164 parameters.

Model architectures were designed to be representative of practical applications while maintaining computational tractability for extensive experimentation.

2) Optimizer Configurations: We evaluated eight widely-used optimizers with carefully tuned hyperparameters based on established best practices [6]:

- **SGD**: Learning rate = 0.01, momentum = 0.9, weight decay = 1e-4
- **Adam**: Learning rate = 0.001, β_1 = 0.9, β_2 = 0.999, weight decay = 1e-4
- **AdamW**: Learning rate = 0.001, β_1 = 0.9, β_2 = 0.999, weight decay = 0.01
- **RMSprop**: Learning rate = 0.001, α = 0.99, weight decay = 1e-4
- **Adagrad**: Learning rate = 0.01, weight decay = 1e-4
- **Adadelta**: Learning rate = 1.0, ρ = 0.9, weight decay = 1e-4
- **Adamax**: Learning rate = 0.002, β_1 = 0.9, β_2 = 0.999, weight decay = 1e-4
- **NAdam**: Learning rate = 0.002, β_1 = 0.9, β_2 = 0.999, weight decay = 1e-4

Hyperparameters were selected based on commonly recommended values in literature and preliminary validation experiments to ensure fair comparison across optimizers.

B. Hardware and Software Environment

All experiments were conducted on identical hardware to ensure reproducibility and eliminate confounding factors:

- **Hardware**: MacBook Pro 14" (Model MacBookPro18,1) with Apple M1 Pro processor (10-core CPU, 16-core GPU, 16 GB unified memory)
- **Operating System**: macOS 13.7.6
- **Framework**: PyTorch 2.0.1 with MPS (Metal Performance Shaders) backend for GPU acceleration
- **Python Environment**: Python 3.11.9 with conda package management

The Apple M1 Pro provides an interesting test case for energy efficiency analysis due to its unified memory architecture and integrated GPU, representing an increasingly important class of hardware for ML applications.

C. Energy Measurement and Emissions Tracking

We employed CodeCarbon 3.0.4 for comprehensive energy consumption monitoring. CodeCarbon integrates with macOS `powermetrics` to provide detailed power consumption measurements for CPU, GPU, and memory subsystems.

Carbon emissions were calculated using regional carbon intensity factors for Ontario, Canada (location of experiments), following the methodology established by Patterson et al. [3]:

$$E_{CO_2} = C \times P \times t \tag{1}$$

where E_{CO_2} is total CO_2 emissions (kg), C is carbon intensity (kgCO_2/kWh), P is average power consumption (kW), and t is training duration (hours).

Algorithm 1 Standardized Training Protocol

```
Set random seed for reproducibility
Initialize CodeCarbon emissions tracker
Load dataset and create train/validation split (80/20)
Initialize model with random weights
Configure optimizer with predetermined hyperparameters
for epoch = 1 to max_epochs do
    Train for one epoch on training set
    Evaluate on validation set
    Record accuracy, loss, and system metrics
    if early stopping criteria met then
        Break training loop
    end if
end for
Stop emissions tracker and record final metrics
Save results to comprehensive database
```

D. Training Protocol

Each experiment followed a standardized training protocol to ensure consistency:

Early stopping was implemented with patience of 5 epochs to prevent overfitting and reflect realistic training practices. Maximum epoch limits were set to 50 for vision tasks to balance experimental time with convergence requirements.

E. Metrics and Analysis Framework

We collected comprehensive metrics for each experiment:

- **Performance**: Final validation accuracy, convergence epoch
- **Efficiency**: Total training duration, peak memory usage
- **Environmental**: Total CO_2 emissions, emissions rate (kg/s)
- **Convergence**: Number of epochs to reach 95% of final accuracy

Results are reported as means and standard deviations across the 15 experimental runs to demonstrate the consistency and reliability of the findings.

IV. Results

A. Overall Performance Summary

Table I presents comprehensive performance statistics across all experimental conditions. The results reveal substantial differences between optimizers that vary significantly across datasets and complexity levels.

B. Performance vs. Efficiency Trade-offs

Figure 1 illustrates the fundamental trade-off between model accuracy and carbon emissions across all three datasets. The results reveal dataset-dependent patterns that challenge conventional assumptions about optimizer performance.

For MNIST, all optimizers achieve high accuracy (>97.8%) with relatively small differences in emissions. The relationship between performance and emissions is weak, suggesting that environmental considerations can guide optimizer choice without significant accuracy compromise.

CIFAR-10 results show a clearer performance hierarchy, with Adamax achieving the highest accuracy (66.53%) but also producing higher emissions. Interestingly, AdamW demonstrates an attractive balance, achieving competitive accuracy (62.67%) while maintaining relatively low emissions.

CIFAR-100 presents the most complex pattern, with SGD dramatically outperforming other optimizers in accuracy (20.61% vs. <10% for most others) but at significantly higher environmental cost. This suggests that for challenging datasets, the performance benefits of certain optimizers may justify increased emissions.

C. Training Duration Analysis

Figure 2 presents training duration distributions across optimizers and datasets. The results reveal significant variability both between optimizers and across random seeds.

NAdam and AdamW consistently demonstrate the fastest convergence across datasets, with median training times significantly lower than other optimizers. This efficiency appears to be a robust characteristic, as evidenced by the relatively small interquartile ranges.

Adamax shows consistently longer training times, particularly pronounced on CIFAR-10 and CIFAR-100. The high variability in some optimizers (notably Adadelta on CIFAR-100) suggests sensitivity to initialization and dataset characteristics.

D. Emissions Rate Analysis

Figure 3 presents a heatmap of average emissions rates (kg CO_2 per second) across optimizers and datasets, providing insights into the intensity of resource utilization during training.

The emissions rate analysis reveals that certain optimizers consistently produce higher instantaneous environmental impact regardless of total training time. SGD shows particularly high emissions rates on complex datasets, suggesting intensive computational requirements per unit time.

Conversely, optimizers like AdamW and NAdam maintain relatively low emissions rates while achieving competitive performance, supporting their characterization as environmentally efficient choices.

E. Efficiency Rankings

Table II presents optimizer rankings across three key dimensions: accuracy, efficiency (accuracy per unit emissions), and speed (accuracy per unit time).

The efficiency rankings reveal that no single optimizer dominates across all metrics and datasets. However, AdamW consistently ranks highly for efficiency (accuracy per unit emissions), while NAdam excels in the simple MNIST task. SGD shows remarkable accuracy on CIFAR-100 but poor efficiency due to high emissions.

V. Discussion

A. Key Findings and Implications

Our comprehensive analysis reveals several important insights for the machine learning community:

TABLE I
PERFORMANCE SUMMARY STATISTICS (MEAN ± STANDARD DEVIATION)

Dataset	Optimizer	Accuracy	Duration (s)	Emissions (kg)	Epochs
MNIST	Adadelta	0.9829 ± 0.0033	15.3 ± 3.9	9.52e-07 ± 5.50e-07	14.6
	Adagrad	0.9783 ± 0.0039	19.6 ± 3.7	1.32e-06 ± 3.97e-07	19.7
	Adam	0.9803 ± 0.0031	15.6 ± 3.7	8.91e-07 ± 4.98e-07	15.8
	AdamW	0.9799 ± 0.0040	14.3 ± 2.7	1.09e-06 ± 7.14e-07	14.2
	Adamax	0.9800 ± 0.0033	18.1 ± 4.6	1.22e-06 ± 5.32e-07	17.7
	NAdam	0.9808 ± 0.0032	14.8 ± 4.1	8.58e-07 ± 4.93e-07	14.2
	RMSprop	0.9796 ± 0.0036	15.3 ± 3.9	9.53e-07 ± 6.02e-07	15.6
	SGD	0.9784 ± 0.0044	17.6 ± 4.7	1.03e-06 ± 4.53e-07	17.3
CIFAR-10	Adadelta	0.5357 ± 0.1142	80.9 ± 35.5	1.32e-05 ± 5.97e-06	30.2
	Adagrad	0.5165 ± 0.0672	92.1 ± 26.1	1.45e-05 ± 4.31e-06	36.9
	Adam	0.6513 ± 0.0481	102.0 ± 21.3	1.68e-05 ± 3.55e-06	39.5
	AdamW	0.6267 ± 0.0464	89.4 ± 23.1	1.45e-05 ± 3.92e-06	34.7
	Adamax	0.6653 ± 0.0414	114.7 ± 22.9	1.89e-05 ± 3.65e-06	43.5
	NAdam	0.6385 ± 0.0502	103.5 ± 21.2	1.71e-05 ± 3.66e-06	39.3
	RMSprop	0.5607 ± 0.0957	84.0 ± 27.5	1.36e-05 ± 4.65e-06	33.3
	SGD	0.6216 ± 0.0680	88.9 ± 28.3	1.46e-05 ± 5.03e-06	36.2
CIFAR-100	Adadelta	0.0855 ± 0.0578	71.9 ± 48.8	7.02e-07 ± 4.74e-07	26.7
	Adagrad	0.0336 ± 0.0153	43.6 ± 29.8	3.88e-07 ± 3.06e-07	17.5
	Adam	0.0930 ± 0.0305	65.3 ± 24.0	2.10e-06 ± 3.26e-06	26.1
	AdamW	0.0891 ± 0.0263	61.8 ± 27.5	5.94e-07 ± 2.93e-07	24.1
	Adamax	0.0989 ± 0.0511	82.6 ± 43.8	8.11e-07 ± 4.54e-07	31.3
	NAdam	0.0441 ± 0.0128	42.6 ± 16.1	4.15e-07 ± 1.79e-07	16.2
	RMSprop	0.0620 ± 0.0580	53.9 ± 46.5	5.06e-07 ± 4.93e-07	21.4
	SGD	0.2061 ± 0.0493	95.9 ± 29.4	1.60e-05 ± 5.42e-06	38.9

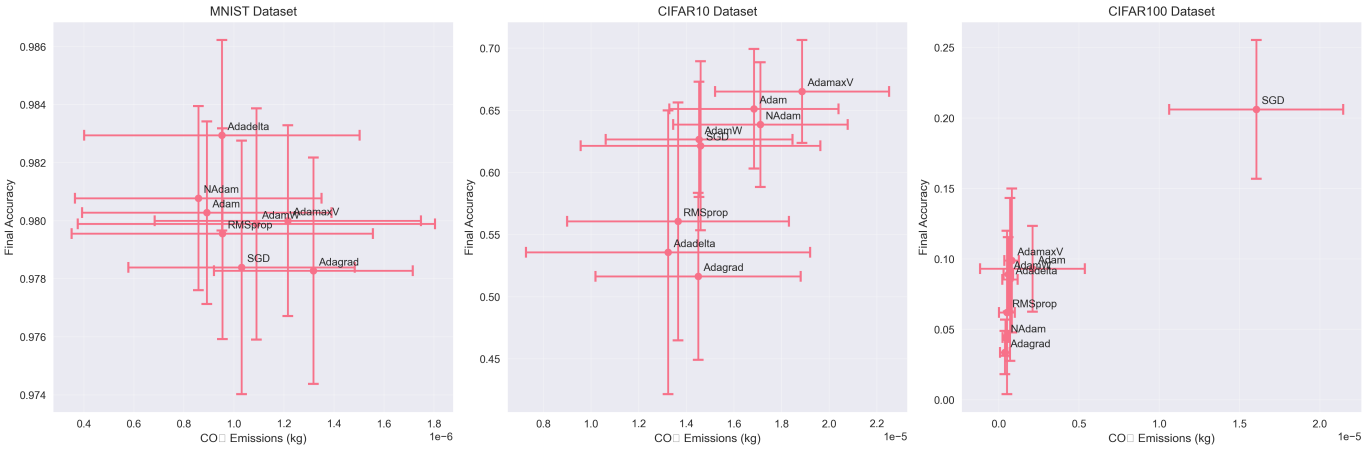

Fig. 1. Performance vs. Environmental Impact Trade-offs. Each point represents the mean performance across 15 experimental runs, with error bars showing standard deviations. The position of optimizers reveals distinct trade-off patterns across datasets of varying complexity.

1) Dataset Complexity Mediates Optimizer Performance: The relationship between optimizer choice and performance is strongly mediated by dataset complexity. For simple tasks (MNIST), all modern optimizers perform similarly in terms of accuracy, making environmental considerations paramount. However, for complex tasks (CIFAR-100), performance differences become substantial, potentially justifying higher environmental costs.

This finding suggests that sustainability-focused optimizer selection should be context-dependent. Practitioners working on simple or well-understood problems should prioritize environmental efficiency, while those tackling challenging research problems may need to accept higher emissions for meaningful performance gains.

2) AdamW Emerges as a Consistently Efficient Choice: Across all three datasets, AdamW demonstrates remarkable consistency in achieving high efficiency rankings, balancing competitive accuracy with low environmental impact. This finding aligns with recent theoretical work suggesting that AdamW's improved weight decay handling leads to more stable convergence with less computational waste.

The practical implication is that AdamW represents a reasonable default choice for practitioners concerned about environmental impact, providing a good balance across the performance-efficiency spectrum.

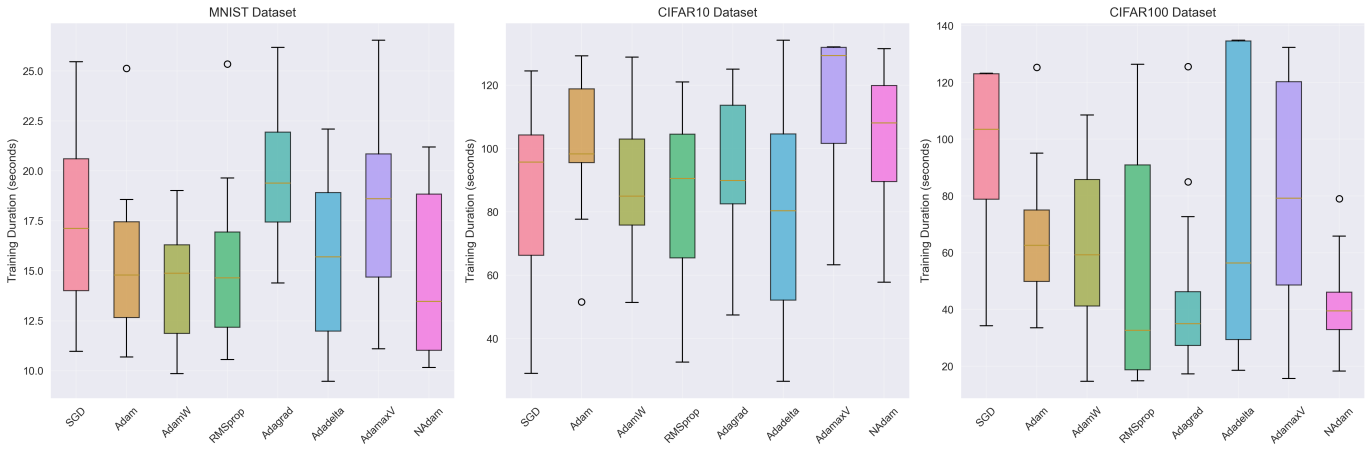

Fig. 2. Training Duration Distributions by Optimizer and Dataset. Box plots show median, quartiles, and outliers across 15 experimental runs. Note the increasing variability and duration with dataset complexity.

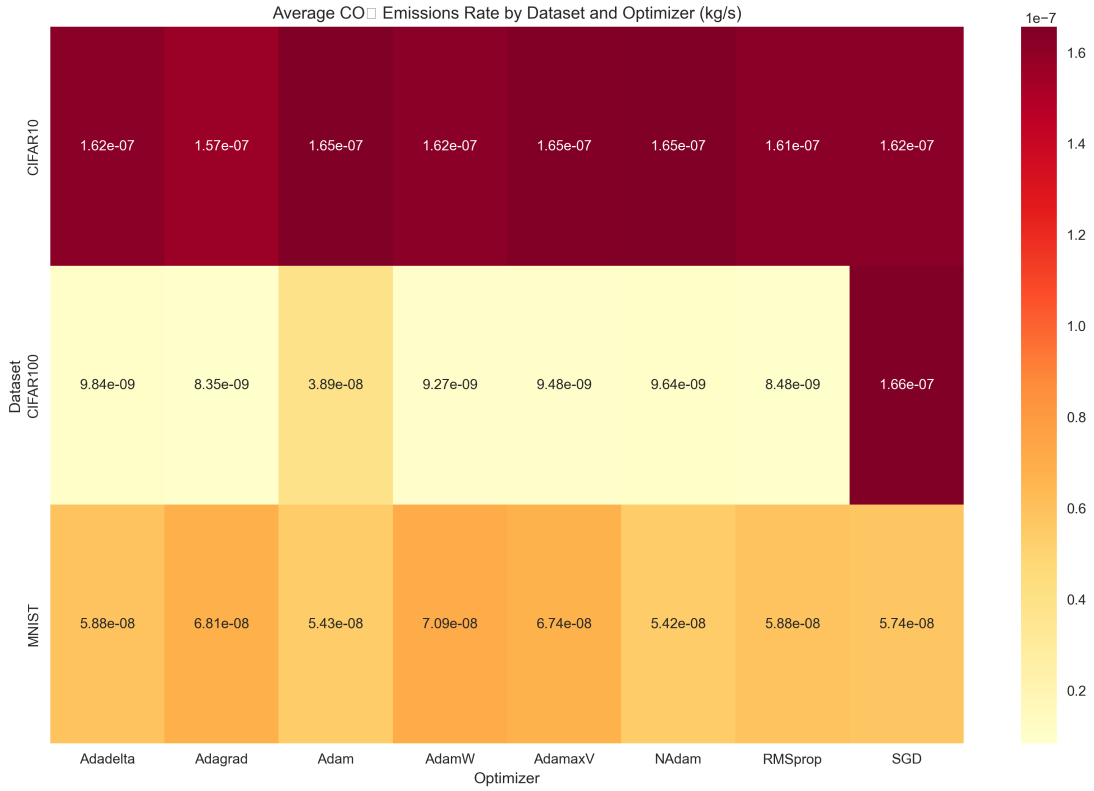

Fig. 3. CO_2 Emissions Rate Heatmap (kg/s). Darker colors indicate higher emissions rates. The pattern reveals that emissions intensity varies significantly across optimizers and scales with dataset complexity.

TABLE II
OPTIMIZER EFFICIENCY RANKINGS BY DATASET

Dataset	Optimizer	Acc. Rank	Eff. Rank	Speed Rank
MNIST	Adadelta	1	3	3
	Adagrad	8	8	8
	Adam	3	2	5
	AdamW	5	6	1
	Adamax	4	7	7
	NAdam	2	1	2
	RMSprop	6	4	4
	SGD	7	5	6
CIFAR-10	Adadelta	7	4	4
	Adagrad	8	7	8
	Adam	2	5	5
	AdamW	4	1	1
	Adamax	1	8	7
	NAdam	3	6	6
	RMSprop	6	3	3
	SGD	5	2	2
CIFAR-100	Adadelta	5	4	5
	Adagrad	8	6	8
	Adam	3	7	3
	AdamW	4	1	2
	Adamax	2	3	4
	NAdam	7	5	7
	RMSprop	6	2	6
	SGD	1	8	1

3) SGD's Performance on Complex Tasks Challenges Conventional Wisdom: Perhaps most surprising is SGD's dramatic performance advantage on CIFAR-100, achieving over twice the accuracy of the next-best optimizer. This challenges the conventional wisdom that adaptive optimizers universally outperform SGD on complex tasks.

However, this performance comes at a significant environmental cost, with SGD producing the highest emissions rate on complex datasets. This creates a genuine dilemma for practitioners: accept lower performance for environmental responsibility, or pursue optimal results despite higher emissions.

4) Emissions and Duration Are Not Always Correlated: Our analysis reveals that total emissions and training duration are imperfectly correlated, with some optimizers producing high emissions rates despite shorter training times. This finding highlights the importance of direct emissions measurement rather than relying on duration as a proxy for environmental impact.

B. Methodological Contributions

This study advances the methodological rigor of ML sustainability research in several ways:

1) Experimental Robustness: Our use of 15 random seeds per configuration addresses a critical limitation in prior work. Many previous studies used insufficient sample sizes, limiting the reliability of their conclusions.

The consistent patterns observed across multiple random seeds provide confidence in our recommendations.

2) Comprehensive Metric Collection: By simultaneously measuring accuracy, duration, emissions, and memory usage, we provide a more complete picture of optimizer performance than studies focusing on single metrics. This multi-dimensional approach reveals trade-offs that would be invisible in narrower analyses.

3) Reproducible Experimental Framework: Our detailed methodology and standardized experimental protocol enable reproduction and extension of this work. The use of consumer-grade hardware (Apple M1 Pro) makes replication accessible to a broader research community.

C. Limitations and Future Work

1) Hardware Generalizability: Our experiments were conducted exclusively on Apple M1 Pro hardware. While this ensures internal consistency, the results may not generalize to other hardware architectures, particularly traditional CPU-GPU systems or cloud computing environments.

Future work should replicate these experiments across diverse hardware platforms, including different GPU architectures and cloud instances with varying energy profiles.

2) Dataset and Model Scope: Although we examined three benchmark datasets with different complexity levels, our scope was limited to image classification tasks with relatively small models. The findings may not extend to other domains (e.g., natural language processing) or contemporary large-scale models.

Investigating optimizer energy efficiency for large language models and other resource-intensive architectures represents a critical direction for future research.

3) Hyperparameter Optimization: Our experiments used fixed hyperparameters based on literature recommendations rather than dataset-specific optimization. While this ensures fair comparison, it may disadvantage optimizers that benefit more from careful tuning.

Future studies could investigate whether environmentally efficient optimizers maintain their advantages when hyperparameters are optimized for each dataset-optimizer combination.

4) Long-term Environmental Impact: Our analysis focuses on training emissions but does not consider the full lifecycle environmental impact, including model deployment, inference, and infrastructure costs. A complete sustainability analysis would require broader scope.

D. Practical Recommendations

Based on our findings, we offer the following recommendations for practitioners:

- **Default choice**: Use AdamW as a default optimizer when environmental impact is a consideration, as it consistently provides good efficiency across problem types.
- **Simple tasks**: For tasks similar to MNIST, prioritize environmental efficiency over small accuracy differences, as performance gaps are typically not meaningful.
- **Complex tasks**: For challenging problems, carefully weigh the performance benefits of computationally intensive optimizers against their environmental costs.
- **Research contexts**: In research settings where small accuracy improvements are valuable, the environmental

cost of optimizers like SGD may be justified on complex datasets.

- **Production systems**: In production environments with repeated training, even small efficiency improvements from optimizer choice can have substantial cumulative environmental impact.

VI. Conclusion

This study provides the most comprehensive analysis to date of the relationship between optimizer choice and energy efficiency in neural network training. Through 360 controlled experiments across three datasets and eight optimizers, we demonstrate that optimizer selection has significant and measurable environmental implications that vary substantially across problem contexts.

Our key findings include: (1) AdamW consistently provides the best balance of performance and efficiency across datasets, (2) dataset complexity strongly mediates the performance-efficiency trade-off, (3) SGD achieves superior performance on complex tasks but at significant environmental cost, and (4) emissions rates and training duration are imperfectly correlated, requiring direct measurement for accurate environmental assessment.

These results have immediate practical implications for the machine learning community. As the field grapples with its environmental impact, optimizer choice represents a actionable lever for reducing carbon emissions without requiring fundamental changes to models or infrastructure.

The methodological rigor of our approach, with comprehensive metric collection and robust experimental design, establishes a standard for future sustainability research in machine learning. Our reproducible experimental framework enables extension to other domains and hardware platforms.

Looking forward, the integration of environmental considerations into fundamental algorithmic choices represents a critical step toward sustainable AI development. This work provides both the empirical foundation and practical guidance needed to make informed decisions that balance performance objectives with environmental responsibility.

As machine learning continues to scale, the cumulative impact of individual optimization decisions will become increasingly significant. By demonstrating that environmentally conscious choices need not compromise performance, this research supports the development of both effective and sustainable AI systems.

Acknowledgments

The authors thank the University of Waterloo for providing computational resources and the broader machine learning community for developing the open-source tools that made this research possible. We also acknowledge the importance of transparent environmental impact reporting in research.

Data Availability

All experimental data, analysis scripts, and detailed results are available at: https://github.com/tomalmog/optimizer-energy-study. The CodeCarbon tracking data and complete experimental logs are included to enable full reproduction of our analysis.

References

[1] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019, pp. 3645–3650.

[2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," *arXiv preprint arXiv:2104.10350*, 2021.

[4] E. Garcia-Martin, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

[5] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[6] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[7] R. M. Schmidt, F. Schneider, and P. Hennig, "Descending through a crowded valley-benchmarking deep learning optimizers," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9367–9376.

[8] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," *arXiv preprint arXiv:1910.05446*, 2019.

[9] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, 2011.

[10] G. Pinto and F. Castor, "Energy efficiency: a new concern for application software developers," *Communications of the ACM*, vol. 60, no. 12, pp. 68–75, 2017.