Empirical Analysis of Early Stopping and Learning Rate Scheduling for Deep Learning on Resource-Constrained Hardware

Karl Satchi Navida
navidake@national-u.edu.ph
College of Computing & Information Technology
National University
Manila, Philippines

Abstract

Efficient training of deep neural networks on resource-constrained consumer hardware presents unique challenges that differ fundamentally from datacenter environments. While most optimization research targets high-performance computing infrastructure, practitioners with limited resources face hardware constraints that can alter the effectiveness of traditional optimization strategies. This paper provides the first systematic empirical analysis of how training optimizations behave under consumer hardware constraints, specifically examining the interactions between early stopping criteria and learning rate adaptation strategies. We evaluate these techniques across three representative consumer platforms: CPU-only systems, entry-level GPUs, and free-tier cloud environments, using CIFAR-10 as our benchmark task. Our findings shows that optimization strategies effective on high-end hardware do not necessarily maintain their relative performance under resource constraints, and we provide practical guidelines for practitioners operating under strict computational budgets. The study introduces resource-normalized performance metrics and establishes reproducible evaluation protocols for assessing training optimizations on consumer hardware.

CCS Concepts

• **Computing methodologies → Supervised learning**; **Neural networks**; *Machine learning algorithms.*

Keywords

Early Stopping, Learning Rate Scheduling, Resource-Constrained Training, Consumer Hardware, Deep Learning Optimization

1 Introduction

1.1 Background of the Study

The proliferation of deep learning applications has created a significant disparity between research conducted on high-performance computing infrastructure and the reality faced by practitioners with limited resources. While most optimization research targets datacenter environments with abundant computational resources, a substantial portion of the deep learning community operates on consumer-grade hardware with strict constraints on memory, processing power, and energy consumption.

Current optimization literature predominantly focuses on scaling to larger models and datasets, often overlooking the unique challenges present in resource-constrained environments. These challenges include thermal throttling, limited memory bandwidth, and the need to balance training time with energy consumption—factors that can fundamentally alter the effectiveness of traditional optimization strategies.

The assumption of unlimited computational resources pervades much of the optimization literature, leading to techniques that may not translate effectively to environments where every computational cycle and memory access carries a direct cost. This disconnect between research assumptions and practical constraints creates a gap that limits the applicability of state-of-the-art optimization techniques for a large portion of the deep learning community.

1.2 Statement of the Problem

This study addresses a specific gap in understanding how training optimizations behave under hardware constraints that are fundamentally different from those assumed in most prior work. The central hypothesis is that resource constraints introduce interaction effects between optimization strategies and hardware characteristics that are not captured by traditional evaluation methodologies.

Specifically, we investigate the following research questions:

1.2.1 *Primary Research Question.*

: How do the interactions between early stopping criteria and learning rate adaptation strategies change when training is constrained by limited computational resources, memory bandwidth, and thermal management?

1.2.2 *Secondary Questions.*

:

1. Do optimization strategies that perform well on high-end hardware maintain their relative effectiveness on consumer devices?
2. How do hardware-specific constraints (thermal throttling, memory limitations) influence the optimal choice of hyperparameters?
3. What practical guidelines can be derived for practitioners operating under strict resource budgets?
4. Can resource-normalized metrics provide better guidance for optimization choices in constrained environments?

These questions address the fundamental challenge of translating optimization research from idealized high-performance environments to the practical constraints faced by individual researchers, students, and practitioners in resource-limited settings.

1.3 Significance of the Study

Our work makes the following specific contributions to the field of resource-aware deep learning optimization:

1.3.1 *Empirical Characterization.*

: We provide the first systematic analysis of how training optimization strategies interact with hardware-imposed constraints on three distinct consumer platforms. We evaluate these techniques across three representative consumer platforms: CPU-only systems, entry-level GPUs, and free-tier cloud environments, using CIFAR-10 [9] as our benchmark task. This characterization reveals performance patterns that differ significantly from those observed in high-performance environments.

1.3.2 *Resource-Aware Guidelines.*
: We develop practical recommendations that explicitly account for hardware limitations, moving beyond generic optimization advice to provide actionable guidance for specific resource constraints.

1.3.3 *Efficiency Metrics.*
: We introduce resource-normalized performance metrics that better capture the trade-offs relevant to resource-constrained practitioners, including accuracy-per-watt and convergence-per-peso (Philippine Peso or PHP) measures.

1.3.4 *Rigorous Evaluation Protocol and Detailed Resource Profiling.*
: We establish a rigorous and transparent evaluation protocol for assessing training optimizations on consumer hardware, including detailed profiling of resource utilization, power consumption, and thermal behavior. This methodology provides a foundational framework for future empirical investigations into resource-aware deep learning.

1.3.5 *Interaction Analysis.*
: We demonstrate that optimization strategies do not combine linearly under resource constraints, revealing synergistic and antagonistic effects that have practical implications for hyperparameter selection.

1.4 Scope and Limitations

This study focuses specifically on image classification tasks using convolutional neural networks, trained on consumer hardware representative of typical academic and individual research settings. We examine two primary optimization strategies—early stopping and learning rate scheduling—chosen for their widespread applicability, implementation simplicity, and potential for significant impact under resource constraints.

The hardware platforms studied represent three common scenarios: CPU-only training on modern laptops, entry-level discrete GPU training, and free-tier cloud GPU access. These platforms capture the majority of resource-constrained training scenarios while maintaining experimental control and reproducibility.

Our analysis is limited to training-phase optimizations and does not address inference optimization, model architecture design, or distributed training scenarios. The findings may not generalize to other domains such as natural language processing or fundamentally different architectures such as transformers, though the methodology could be extended to these areas in future work.

The study also does not address data loading optimization, storage constraints, or network bandwidth limitations, focusing instead on computational and memory constraints during the training process itself.

2 Background and Related Work

2.1 Training Optimization Under Resource Constraints

The conventional approach to deep learning optimization assumes abundant computational resources, enabling extensive hyperparameter search and long training runs. However, resource-constrained environments introduce qualitatively different challenges that are often overlooked in mainstream optimization research.

Thermal Management and Performance Variability: Consumer hardware typically lacks the sophisticated cooling systems found in datacenter environments. Sustained high utilization can trigger thermal throttling, dynamically reducing clock speeds and computational throughput [8, 13]. This phenomenon can fundamentally alter training dynamics, making some optimization strategies counterproductive.

Unlike datacenter GPUs that maintain consistent performance under thermal management systems, consumer GPUs may experience performance degradation of 20-40% during extended training sessions [12]. This variability affects the reproducibility of optimization strategies and suggests that techniques designed for consistent hardware performance may require adaptation.

Memory Hierarchy Effects: Consumer GPUs often have limited VRAM and slower memory interfaces compared to professional accelerators. This constraint affects optimal batch sizes, gradient accumulation strategies, and the feasibility of certain optimization techniques that rely on maintaining large buffers or historical information [20].

The memory bandwidth limitations of consumer hardware can create bottlenecks that shift the optimal balance between computation and memory operations. Optimization strategies that assume high memory bandwidth may become counterproductive when memory access becomes the limiting factor.

Energy Constraints and Cost Considerations: Unlike datacenter environments where energy costs are amortized across many users, individual practitioners must consider energy consumption as a direct cost. This consideration changes the optimization objective from pure performance to efficiency-aware metrics that account for power consumption and training time trade-offs [16].

2.2 Early Stopping in Practice

Early stopping is a fundamental regularization technique that prevents overfitting by halting training when validation performance plateaus. While the theoretical foundations are well-established [14], practical implementation involves numerous design choices that can significantly impact effectiveness under resource constraints.

Validation-Based Stopping Criteria: Traditional early stopping relies on monitoring validation loss or accuracy, requiring practitioners to set aside validation data and computational resources for periodic evaluation [1]. In resource-constrained settings, both the data allocation and computational overhead of validation can represent significant costs.

Recent work by Mahsereci et al. [11] introduced validation-free early stopping using gradient statistics, which is particularly relevant for resource-constrained settings. However, their approach has not been evaluated on consumer hardware where computational constraints may affect the reliability of gradient-based stopping criteria.

Patience and Restoration Strategies: The choice of patience parameter (number of epochs to wait before stopping) and model restoration strategy (whether to restore the best weights) significantly influence both final model quality and resource utilization. Longer patience periods provide more opportunities for recovery from local minima but consume additional resources that may be critical under constraints [2].

Gap in Current Understanding: Most early stopping research evaluates stopping criteria in isolation, without considering their interaction with other optimization components or hardware constraints. The optimal patience parameter on a high-performance GPU may be suboptimal on consumer hardware where training dynamics are affected by thermal throttling and variable performance.

2.3 Learning Rate Scheduling Strategies

Learning rate adaptation has evolved from simple step decay to sophisticated adaptive methods, but the choice of scheduling strategy interacts complexly with hardware characteristics in ways that are poorly understood.

Cyclical Learning Rates: Smith's cyclical learning rates [15] demonstrated significant improvements on standard benchmarks, but these results were obtained on high-performance hardware with consistent computational throughput. Consumer hardware with variable performance due to thermal management may require different cycle lengths or amplitude choices.

The effectiveness of cyclical schedules depends on the assumption that training dynamics remain consistent across cycles. Hardware-induced performance variability may disrupt this assumption, suggesting the need for adaptive cycle parameters that account for actual rather than nominal computational capacity.

Adaptive Methods and Memory Overhead: Modern adaptive optimizers such as Adam [7] and RMSprop [18] maintain per-parameter learning rate information, increasing memory requirements substantially. On memory-constrained hardware, this overhead may offset the convergence benefits, suggesting different trade-offs than those observed in unconstrained environments.

The memory requirements of adaptive optimizers scale with model size, creating a tension between optimization effectiveness and resource utilization that is particularly acute on consumer hardware with limited VRAM.

Warm Restarts and Resource Budgets: Loshchilov and Hutter's warm restarts [10] showed promising results by periodically resetting the learning rate and allowing the optimization to escape local minima. However, the optimal restart schedule may depend critically on the available computational budget.

A restart schedule designed for unlimited training time may be suboptimal when training must complete within a fixed time window or energy budget, suggesting the need for budget-aware scheduling strategies.

2.4 Hardware-Aware Optimization

The computer systems community has produced substantial work on hardware-aware optimization, but this research has largely focused on inference rather than training, and on mobile/embedded devices rather than consumer desktop hardware.

Architectural Adaptations: Work on efficient architectures such as MobileNets [5], SqueezeNet [6], and EfficientNet [17] focuses on designing models that are inherently efficient. However, these approaches address model architecture rather than training procedures, and do not consider how training strategies should be adapted for the target hardware.

Compiler and System-Level Optimizations: Frameworks such as TVM [3] and TensorRT [19] optimize individual operations and kernel execution but do not address higher-level training strategies or their interaction with hardware constraints. These optimizations are orthogonal to our focus on algorithmic training strategies.

Energy-Aware Training: Recent work has begun to address energy consumption in training [4, 16], but primarily from an environmental perspective focusing on large-scale training rather than as a practical constraint for individual researchers with limited budgets.

2.5 Research Gap and Positioning

Despite extensive research on both training optimization and hardware-efficient deep learning, there exists a significant gap in understanding how these domains interact in practical resource-constrained scenarios.

Evaluation Methodology Gap: Most optimization research uses high-end hardware and focuses on final accuracy or convergence speed, ignoring resource utilization metrics that are critical for constrained environments. Standard benchmarking practices do not capture the trade-offs between optimization effectiveness and resource consumption.

Interaction Effects: Training optimizations are typically evaluated in isolation, but their effectiveness may be fundamentally altered by hardware constraints such as thermal throttling or memory limitations. The assumption of independent effects may not hold under resource constraints.

Practical Guidelines: Existing work provides either theoretical analysis or high-level architectural recommendations, but lacks practical guidance for practitioners with specific hardware constraints and budget limitations.

Reproducibility and Standardization: The lack of standardized evaluation protocols for resource-constrained training makes it difficult to compare results across studies or translate research findings to practical applications.

Our work addresses these gaps by providing a systematic empirical analysis of training optimization strategies specifically designed for and evaluated on consumer hardware, with explicit consideration of resource constraints and practical applicability. This focus on the intersection of optimization effectiveness and resource efficiency represents a novel contribution to the field.

References

[1] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. *Neural networks: Tricks of the trade* (2012), 437–478. doi:10.1007/978-3-642-35289-8_26

[2] Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems* 13 (2001).

[3] Tianqi Chen, Lianmin Zheng, Eddie Yan, Ziheng Jiang, Thierry Moreau, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. 2018. TVM: An automated end-to-end optimizing compiler for deep learning. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*. 578–594.

[4] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research* 21, 248 (2020), 1–43.

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[6] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. In *arXiv preprint arXiv:1602.07360*.

[7] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[8] Jonathan Koomey, Stephen Berard, Marla Sanchez, and Henry Wong. 2011. Web extra appendix: implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing* 33, 3 (2011), 46–54.

[9] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[10] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).

[11] Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. 2017. Early stopping without a validation set. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2238–2247.

[12] Sparsh Mittal. 2014. A survey of techniques for improving energy efficiency in embedded computing systems. *Journal of Systems and Software* 95 (2014), 208–237. doi:10.1016/j.jss.2014.05.021

[13] Massoud Pedram and Inkwon Hwang. 2012. Energy-efficient datacenters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 10 (2012), 1465–1484. doi:10.1109/TCAD.2012.2212898

[14] Lutz Prechelt. 1998. Early stopping-but when? *Neural Networks: Tricks of the trade* (1998), 55–69. doi:10.1007/978-3-642-35289-8_5

[15] Leslie N Smith. 2017. Cyclical learning rates for training neural networks. *2017 IEEE winter conference on applications of computer vision (WACV)* (2017), 464–472. doi:10.1109/WACV.2017.58

[16] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), 3645–3650. doi:10.18653/v1/P19-1355

[17] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[18] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[19] Han Vanholder. 2016. Efficient inference with tensorrt. In *GPU Technology Conference*, Vol. 1. 2.

[20] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.