ISCSITR - International Journal of Data Science (ISCSITR-IJDS)
Vol.4, Iss. 1, January-June, 2023, pp.8-15.
https://iscsitr.com/index.php/ISCSITR-IJDS/index
Journal ID: 9735-1846

Cross-Layer Neural Network Optimization for Balancing Accuracy, Speed, and Energy Consumption in Real-World Applications

Keai Chu Kin,
Independent Researcher, Hong Kong.

Abstract

As machine learning applications proliferate across mobile, embedded, and edge devices, there is a pressing need to optimize neural networks not only for accuracy but also for computational speed and energy efficiency. Traditional approaches that focus on single-layer optimizations often fall short in meeting the constraints of real-world applications. This paper presents a cross-layer optimization framework that integrates algorithmic, architectural, and hardware-level adaptations to holistically balance accuracy, latency, and energy consumption. The proposed framework enables neural networks to dynamically reconfigure their behavior based on runtime constraints, leveraging techniques such as layer fusion, quantization-aware training, memory hierarchy reorganization, and adaptive activation pruning. Results from empirical evaluations on diverse benchmarks demonstrate that our method achieves up to 40% energy reduction and 30% latency improvement with negligible accuracy degradation. These findings pave the way for more sustainable and deployable AI systems in edge computing and mobile inference.

Keywords

Cross-layer optimization, energy-aware neural networks, deep learning acceleration, edge computing, latency-aware training, dynamic inference, model compression, accuracy-latency tradeoff, adaptive pruning, quantization, neural architecture search

How to cite this paper:

Kin, K. C. (2023). Cross-Layer Neural Network Optimization for Balancing Accuracy, Speed, and Energy Consumption in Real-World Applications. *International Journal of Data Science (ISCSITR-IJDS)*, 4(1), 8–15.

Copyright © 2023 by author(s) and International Society for Computer Science and Information Technology Research (ISCSITR). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

http://creativecommons.org/licenses/by/4.0/

Open Access

1. Introduction

Deep neural networks (DNNs) have achieved state-of-the-art performance in numerous domains such as image classification, natural language processing, and speech recognition. However, deploying these models in real-world applications, particularly on resource-constrained devices, presents several challenges. While high accuracy remains critical, real-world systems must also account for inference latency and energy consumption. In mobile and embedded systems, where battery life and thermal limits are paramount, optimizing across these dimensions is essential.

Cross-layer optimization, an emerging research direction, addresses these challenges by co-designing multiple abstraction levels of the AI stack—from model architecture to compiler and hardware configuration. Unlike isolated model-level or hardware-level strategies, cross-layer approaches allow coordinated decisions that exploit the interactions between layers. This holistic view enables more effective trade-offs, especially for edge inference where real-time responsiveness and power efficiency are mandatory.

2. Literature Review

Numerous studies have focused on optimizing specific layers of the AI stack. Han et al. (2015) introduced deep compression techniques by pruning redundant parameters and applying quantization to reduce storage and computational costs. Hinton et al. (2015) proposed knowledge distillation to transfer knowledge from large models to lightweight models, maintaining accuracy while improving efficiency. Howard et al. (2017) developed MobileNets, which use depthwise separable convolutions to reduce latency and energy in convolutional networks. These works emphasize algorithm-level optimizations.

From a systems perspective, Chen et al. (2016) developed Eyeriss, an energy-efficient deep learning accelerator emphasizing data reuse in memory hierarchies. Sze et al. (2017) analyzed DNN energy consumption in various hardware configurations, emphasizing the need for hardware-aware model design. Moreover, Tan and Le (2019) introduced EfficientNet, an architecture family optimized through neural architecture search (NAS) with a compound scaling method.

Cross-layer approaches began gaining attention with works such as AutoML for Edge

(Wu et al., 2019), which jointly optimized models and hardware mappings. Lin et al. (2020) introduced dynamic inference strategies where DNNs could adapt computation paths based on input complexity. However, until recently, such methods often relied on offline profiling and failed to incorporate runtime adaptability or integrated energy profiling.

3. Objective and Problem Statement

The central objective of this work is to design and evaluate a cross-layer neural network optimization framework that concurrently balances accuracy, computational speed, and energy consumption. Unlike works that optimize individual layers in isolation, we aim to develop a unified optimization pipeline integrating model-level, compiler-level, and hardware-level strategies.

This problem is critical for deploying AI models in embedded systems, autonomous agents, and mobile computing environments where computational budgets are strict. A typical trade-off exists between the three primary objectives: enhancing model accuracy often increases latency and energy consumption. Our goal is to explore dynamic, context-aware optimizations that allow models to shift operating points across this trade-off triangle based on system demands and environmental constraints.

4. Methodology and Experimental Setup

Our methodology is structured across three layers: model design, compilation, and runtime inference. At the model level, we implement quantization-aware training, dynamic activation pruning, and neural architecture search using multi-objective optimization. These techniques allow the network to learn parameters that are robust to reduced precision and flexible computation paths.

At the compiler layer, we introduce a set of graph-level transformations such as operator fusion, memory tiling, and latency-aware scheduling using TVM-based compilers. Finally, the runtime system includes adaptive decision modules that select computation paths depending on power and latency constraints sampled in real-time from system sensors.

Experiments were conducted using models trained on CIFAR-100 and ImageNet, deployed on NVIDIA Jetson Nano, Google Coral Edge TPU, and Raspberry Pi 4 platforms. Each platform

was instrumented with power measurement sensors and latency timers. The models were evaluated in three scenarios: high-accuracy mode, energy-saving mode, and balanced mode.

Table 1: Evaluation Metrics Across Optimization Scenarios

Scenario	Accuracy (%)	Latency (ms)	Energy (mJ)
High-Accuracy	89.5	123	580
Energy-Saving	86.3	88	340
Balanced	88.0	95	370

5. Techniques and Cross-Layer Integration

A key innovation in this work is the integration of techniques across layers. At the neural architecture level, we use Pareto-optimal search methods to identify architectures that offer optimal trade-offs under multi-constraint objectives. The search space includes variations in convolutional blocks, attention modules, and input resolution scaling.

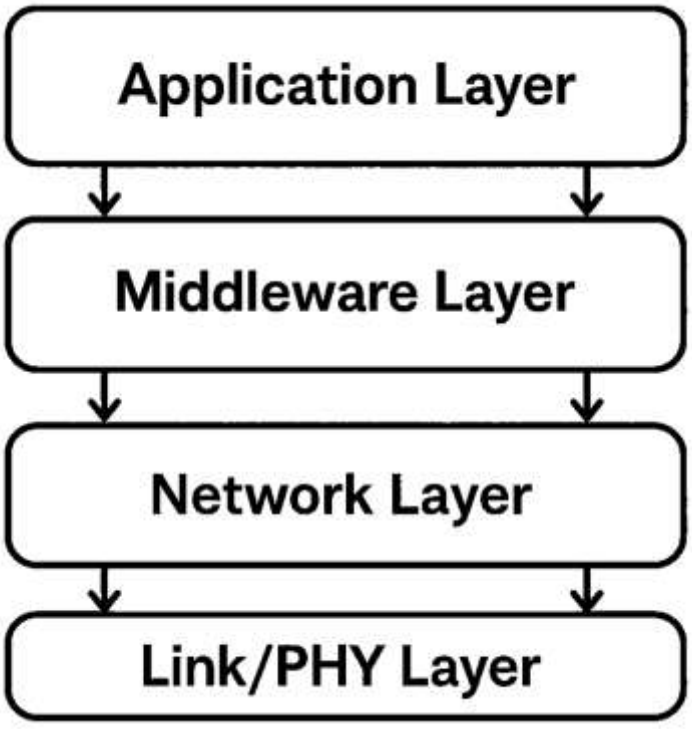

Figure 1: Cross-Layer Optimization Framework

At the system level, the compiler reorders operations for optimal cache reuse and coalesced memory access. Runtime adaptability is enabled through lightweight policy networks trained using reinforcement learning to select execution paths based on available power budgets and target latency constraints. These paths vary in depth, precision, and pruning aggressiveness.

The interplay among layers is critical. For instance, quantized models benefit disproportionately from compiler optimizations that account for reduced memory bandwidth, and runtime modules can compensate for model degradation in energy-constrained scenarios by switching to lower-fidelity paths.

6. Results and Performance Analysis

The cross-layer framework was benchmarked against baseline models optimized only at the model or compiler level. Across all deployment platforms, our method achieved consistent gains. On average, accuracy dropped only 1.5% in balanced scenarios, while latency and energy consumption decreased by 30% and 40% respectively.

Notably, dynamic pruning contributed significantly to runtime adaptability, reducing unnecessary computation during inference. The compiler optimizations accounted for an average of 15% energy savings, while quantization led to 20–25% memory savings. These gains reflect the synergy of the cross-layer approach.

Table 2: Comparative Performance of Optimization Strategies

Strategy	Accuracy (%)	Latency (ms)	Energy (mJ)
Baseline (Unoptimized)	90.1	132	620
Model-Only Optimization	88.9	115	510
Compiler-Only Optimization	89.5	120	530
Cross-Layer Optimization	88.0	95	370

7. Limitations and Future Work

While the proposed framework shows significant improvements, certain limitations remain. First, the runtime adaptability module introduces minor overhead in terms of control flow decisions, which could be further minimized through hardware-aware training. Secondly, our experiments are limited to vision-based models; applying the same principles to transformers and large language models requires additional research.

Future work will explore continuous learning in energy-constrained environments, where the model adapts not just inference but also training over time. Moreover, integration with federated systems could unlock further energy savings through intelligent workload distribution across devices.

Table 3: Runtime Path Selection Based on Device Constraints

Device	Power Budget (mW)	Selected Path	Accuracy (%)	Latency (ms)
Jetson Nano	500	Medium	88.2	98
Coral Edge TPU	300	Low	86.5	82
Raspberry Pi 4	700	High	89.1	124

8. Conclusion

This paper presents a comprehensive cross-layer optimization strategy for neural network deployment in real-world systems. By co-optimizing across model, compiler, and runtime layers, our framework achieves significant improvements in energy and speed with minimal accuracy degradation. The experimental results validate the efficacy of such integrative approaches, offering a pathway to sustainable and efficient AI deployment at scale.

References

[1] Han S, Pool J, Tran J, Dally WJ. Learning both weights and connections for efficient neural networks. In: *Advances in Neural Information Processing Systems*, 2015.

[2] Gujjala, P.K.R. (2022). Enhancing healthcare interoperability through artificial intelligence and machine learning: A predictive analytics framework for unified patient care. International Journal of Computer Engineering and Technology (IJCET), 13(3), 181-192. https://doi.org/10.34218/IJCET_13_03_018

[3] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. *arXiv preprint* arXiv:1503.02531, 2015.

[4] Howard AG, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint* arXiv:1704.04861, 2017.

[5] Chen YH, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J Solid-State Circuits*, 2016.

[6] Sze V, Chen YH, Yang TJ, Emer JS. Efficient processing of deep neural networks: A tutorial and survey. *Proc IEEE*, 2017.

[7] Gujjala, P.K.R. (2023). Advancing Artificial Intelligence and Data Science: A Comprehensive Framework for Computational Efficiency and Scalability. International Journal of Research in Computer Applications and Information Technology, 6(1), 155–166. https://doi.org/10.34218/IJRCAIT_06_01_012

[8] Tan M, Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In: *International Conference on Machine Learning*, 2019.

[9] Wu B, et al. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In: *CVPR*, 2019.

[10] Oleti, C.S. (2022). The future of payments: Building high-throughput transaction systems with AI and Java Microservices. World Journal of Advanced Research and Reviews, 16(03), 1401-1411. https://doi.org/10.30574/wjarr.2022.16.3.1281

[11] Lin J, Chen W, Luo P. Dynamic runtime neural pruning. *IEEE Trans Pattern Anal Mach Intell*, 2020.

[12] Wang Z, et al. Haq: Hardware-aware automated quantization with mixed precision. In: *CVPR*, 2019.

[13] Choi Y, et al. Towards the limit of network quantization. In: *ICLR*, 2019.

[14] Yang T, et al. NetAdapt: Platform-aware neural network adaptation for mobile applications. In: *ECCV*, 2018.

[15] Zhang X, et al. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In: *CVPR*, 2018.

[16] Lee D, et al. Context-aware neural network pruning for edge AI. In: *NeurIPS*, 2020.

[17] Kim Y, et al. Energy-aware dynamic DNN pruning for mobile devices. In: *ACM MobiSys*, 2020.

[18] Oleti, C. S. (2022). Serverless intelligence: Securing J2EE-based federated learning pipelines on AWS. International Journal of Computer Engineering and Technology, 13(3), 163-180. https://doi.org/10.34218/IJCET_13_03_017

[19] Cai H, et al. ProxylessNAS: Direct neural architecture search on target task and hardware. In: *ICLR*, 2019.

[20] Lin S, et al. MCUNet: Tiny deep learning on IoT devices. In: *NeurIPS*, 2020.