SIMD Instruction Set Support for AI Computing

Ching-Hua Chang
312581031
NYCU IAIS
Taiwan
tony70934@gmail.com

Mian-Heng Shan
312553049
Institute of Multimedia Engineering
Taiwan
a285.nycu@gmail.com

Ching Chen
313551007
Institute of Computer Science and Engineering
Taiwan
cchen.cs13@nycu.edu.tw

ABSTRACT

We present a custom SIMD instruction set architecture for accelerating CNN inference on RISC-V processors. By targeting computational bottlenecks in convolution, matrix multiplication, and pooling operations, our implementation achieves speedups of 1.53x, 2.4x, and 2.02x respectively on a 32-bit RISC-V CPU compared to serial implementations. These results demonstrate the effectiveness of SIMD optimizations for enhancing deep learning performance on embedded systems without specialized hardware accelerators.

1 INTRODUCTION

The exponential growth in deep learning applications has brought significant attention to the computational efficiency of neural network operations, particularly in resource-constrained environments. Although Convolution neural networks (CNNs) have achieved remarkable success in various computer vision tasks [3, 6], their computational demands pose significant challenges for deployment on edge devices and IoT platforms that lack dedicated GPU acceleration.

Current state-of-the-art CNN implementations primarily rely on GPU acceleration or specialized neural processing units (NPUs) to achieve optimal performance. However, these solutions are often impractical for small-scale IoT devices and embedded systems due to power constraints, cost considerations, and physical limitations. This has created a critical need for efficient CNN inference methods that can operate effectively on devices with limited computational resources.

AlexNet, a pioneering CNN architecture, serves as an excellent case study for optimization due to its widespread adoption and representative structure. Analysis reveals that certain operations within CNNs, particularly convolution operations, General Matrix Multiplication (GEMM), and Max Pooling, constitute the majority of computational overhead. Specifically, convolution operations can account for more than 90% of the total execution time in typical CNN inference pipelines.

Our work addresses these challenges by leveraging single-intruction multiple-data (SIMD) techniques, a fundamental parallel processing capability available in most modern processors. SIMD instructions offer the potential to accelerate critical CNN operations without requiring specialized hardware, making them particularly suitable for resource-constrained environments. By designing custom SIMD instruction sets specifically optimized for CNN operations, we aimed to bridge the gap between the computational demands of modern deep learning models and the capabilities of edge devices.

This research is motivated by several key factors: (1) the growing need to deploy CNN-based applications on edge devices, (2) the limitations of current hardware acceleration solutions for resource-constrained environments, and (3) the opportunity to leverage existing processor capabilities through optimized SIMD instructions. Our approach demonstrates that performance improvements can be achieved through careful hardware-software co-design, potentially enabling broader adoption of CNN-based applications in embedded and IoT systems.

The remainder of this paper is organized as follow: Section 2 presents our custom SIMD instruction set design. Section 3 and 4 explain our experimental setup and results in detail, respectively. Section 5 elaborates on a wide range of previous works related to CNN acceleration. Finally, Section 6 concludes with a discussion of implications and future directions for hardware-accelerated machine learning.

2 PROPOSED SOLUTION

We aim to design an ISA extension for AI acceleration that support SIMD instructions. CFU-Playground is a System-on-Chip (SoC) equipped with a 32-bit RISC-V CPU, a custom function unit, and a comprehensive software stack designed for executing Deep Neural Network (DNN) models. We have the capability to effortlessly develop our own ISA extensions and conduct RTL simulations for end-to-end model inference.

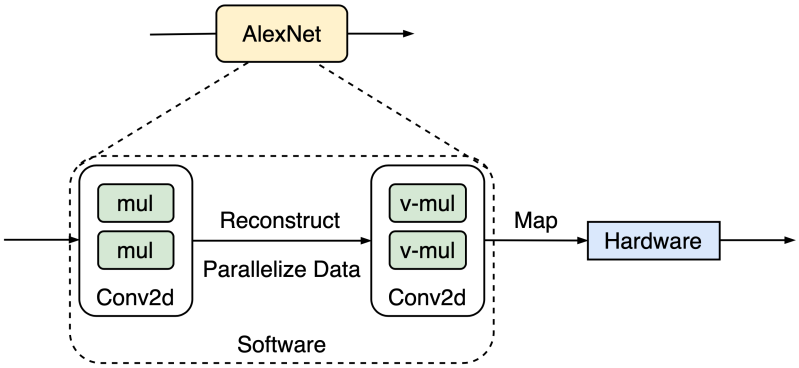

Figure 1: Architecture Overview

To enhance the computational efficiency of AlexNet through low-level optimizations, we propose designing new instructions within an Instruction Set Architecture (ISA) extension. To design these new instructions, we must first determine the specific functionalities and optimizations they should provide. Our focus is on creating Single Instruction Multiple Data (SIMD) instructions that can accelerate fundamental operations commonly used in neural network computations.

The proposed SIMD instructions include:

- **ADD**: Performs parallel addition on multiple data elements simultaneously, accelerating vector addition operations within network layers.
- **SUB**: Executes parallel subtraction across data vectors, useful for operations such as bias adjustments and error computations.
- **PMUL** (Parallel Multiply): Conducts element-wise multiplication on data vectors in parallel, essential for operations like Hadamard products in activation functions.

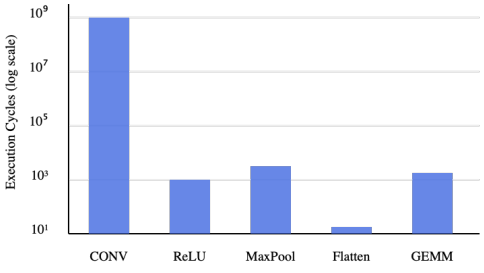

Figure 2: Execution Cycles Distribution Across AlexNet Modules

- **AMUL** (Accumulating Multiply): Performs multiplication with accumulation, effectively accelerating dot products and convolution operations by combining multiplication and addition in a single instruction.
- **QNT** (Quantization): Implements parallel quantization of data elements, reducing precision to lower bit-width representations to optimize memory usage and computational load.

As we can see in Figure 1, we aim to optimize the core computational kernels of AlexNet by reconstructing them by integrated SIMD instructions. This approach allows us to exploit data parallelism at the hardware level, reducing instruction overhead and improving execution throughput. The use of custom SIMD instructions tailored to neural network operations provides a fine-grained optimization strategy that complements higher-level software optimizations.

Our methodology involves implementing these instructions in C, leveraging intrinsic functions to map high-level code to the underlying hardware instructions. This allows for precise control over the generated machine code, ensuring that the intended performance gains are realized on the target architecture. By applying these SIMD-optimized operations to AlexNet, we expect to achieve significant acceleration in both training and inference phases, demonstrating the effectiveness of our approach in practical deep learning applications.

Hardware: Define the operation semantics, opcode, operand formats, and involved parameters for each new instruction. Create the encoding scheme for the new instructions. Define the bit fields for opcodes, data operands, and control bits needed to differentiate instructions from each others. Implement the operation functionalities.

Software: Construct SIMD functions that map to SIMD instructions in hardware, and takes multiple data at a time. For function in AI models, ex.conv2D, reconstruct it by calling SIMD functions, and parallelize the data to them.

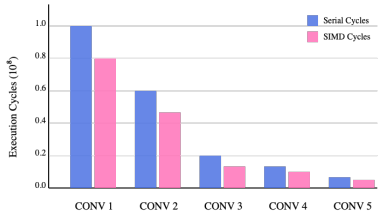

Figure 3: Execution Cycle Comparison of Serial and SIMD Convolution

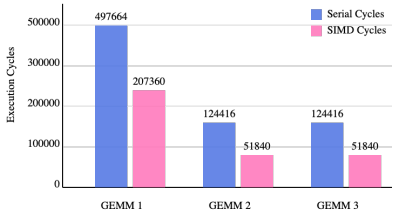

Figure 4: Execution Cycle Comparison of Serial and SIMD GEMM

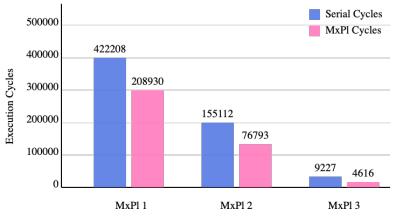

Figure 5: Execution Cycle Comparison of Serial and SIMD Max Pooling

3 EXPERIMENTAL METHODOLOGY

We evaluated the proposed SIMD instruction set by conducting controlled experiments comparing the execution cycles of serial and SIMD-optimized implementations across key computational modules. The input for these tests consisted of a single image used for inference.

The experiments were executed on the CFU (Custom Function Unit) Playground System-on-Chip (SoC) platform, featuring a 32-bit RISC-V CPU with a custom function unit designed to support the SIMD instructions. Software was implemented in C for precise mapping to hardware instructions, while hardware functionalities were developed using the Chisel3 hardware description language.

4 EXPERIMENTAL RESULTS

An in-depth analysis of AlexNet's computational modules reveals that the CONV module overwhelmingly dominates the execution cycles, as illustrated in Figure 2, with its computational time surpassing other modules by more than 100x. Following this, MaxPool and GEMM are identified as the next most computationally demanding modules, further emphasizing the need for optimization in these critical components.

To evaluate the efficacy of our proposed SIMD instruction set, we conducted a comparative analysis of the computational times for the CONV, GEMM, and MaxPool modules using both SIMD-optimized and serial implementations. The results, presented in Figures 3, 4, 5, demonstrate a significant reduction in execution cycles for all three modules when utilizing our SIMD instructions.

Moreover, Table 1 highlights the overall speedup achieved in the computational times of these modules. Specifically, the SIMD-based implementation accelerates the CONV module by 1.53x, the GEMM module by 2.4x, and the MaxPool module by 2.02x when compared to their serial counterparts. These findings underscore the transformative potential of our SIMD instruction set in substantially enhancing the computational efficiency of key modules within AlexNet, paving the way for broader applications in high-performance neural network processing.

	Serial	SIMD	SpeedUp
CONV	188M	123M	1.53
GEMM	746k	311k	2.4
MxPl	586k	290k	2.02

Table 1: Total Execution Cycle Comparison between Serial and SIMD Implementations

5 RELATED WORK

Optimizing the computational efficiency of deep learning models has been an area of significant research, especially in the context of reducing training and inference times for large-scale neural networks such as AlexNet [3]. Previous work has extensively explored methods such as model pruning [5, 8], quantization [4, 7], and hardware-specific optimizations to accelerate neural networks. These techniques focus on reducing model complexity and leveraging specialized hardware, but few approaches focus on optimizing low-level operations at the instruction set level.

The use of single-input multiple-data (SIMD) instructions to accelerate deep learning models has attracted attention in recent years. SIMD's ability to perform parallel operations on multiple data points simultaneously aligns well with the matrix operations and convolutions that dominate neural network workloads. Several studies have demonstrated the effectiveness of SIMD in accelerating various computational tasks, particularly in image processing and scientific computing.

However, its application to low-level operation optimization in neural networks, specifically AlexNet, remains underexplored [9, 10]. Unlike higher-level optimizations, our approach targets fundamental computations within AlexNet, such as matrix multiplications and convolution operations, to achieve a more granular level of performance improvement.

Our work contributes to the growing body of research on hardware-level acceleration techniques, offering a practical demonstration of SIMD's potential to improve deep learning model efficiency.

6 CONCLUSIONS

This study addresses the critical challenge of computational efficiency in deep learning models, particularly focusing on resource-constrained environments where GPU acceleration is not feasible. Through detailed analysis of AlexNet's computational profile, we identified and optimized three key bottleneck operations using a custom SIMD instruction set. Our implementation on a 32-bit RISC-V processor achieved significant performance improvements: a 1.53x speedup for convolution operations which dominate the computational workload, a 2.4x speedup for GEMM computations in fully connected layers, and a 2.02x speedup for Max Pooling operations, as shown in Table 1. Future work could explore more sophisticated SIMD instruction sets optimized for specific neural network architectures, advanced data handling mechanisms to reduce memory access overhead, and techniques for balancing computational efficiency with hardware complexity. These developments would further advance the field of hardware-accelerated machine learning, particularly for resource-constrained computing environments.

REFERENCES

[1] Boosting machine learning with tailored accelerators: Custom function units in renode.

[2] Risc-v "p" extension v0.9.

[3] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

[4] Ido Ben-Yair, Gil Ben Shalom, Moshe Eliasof, and Eran Treister. Quantized convolutional neural networks through the lens of partial differential equations. *Research in the Mathematical Sciences*, 9(4), September 2022.

[5] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017.

[8] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017.

[9] Chih-Ting Liu, Yi-Heng Wu, Yu-Sheng Lin, and Shao-Yi Chien. Computation-performance optimization of convolutional neural networks with redundant kernel removal, 2018.

[10] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks, 2016.