Unified Optimization Strategies for CNN and SVM Hardware Accelerators on Reconfigurable FPGA Architectures

Manish Kumar*
Department of Electronics and Communication Engineering
Sarala Birla University, Ranchi
Jharkhand, India 835103
manish_ind@hotmail.com

Manoj Pandey
Department of Electronics and Communication Engineering
Sarala Birla University, Ranchi
Jharkhand, India 835103
gr8mpandey@gmail.com

Bandi Nageshwar Rao
Department of Electronics and Communication Engineering
Malla Reddy University, Hyderabad
Telangana, India 500100
nagesh.south@gmail.com

Abstract
The rapid growth of machine learning applications and the increasing complexity of neural network architectures have created a pressing need for specialized hardware solutions that can deliver efficient execution of computationally intensive ML workloads. Traditional general-purpose processors will always have a general-use perspective, based on the sequential processing model of the von Neumann architecture, when handling the large number of parallel computations associated with modern machine learning algorithms. This paper pro- vides a thorough examination of hardware acceleration strategies for improving machine learning performance, tracing the evolution from traditional CPUs to dedicated hardware accelerators. It examines various hardware acceleration approaches, starting with Graphics Processing Units (GPUs), which leverage their massively parallel architecture to accelerate the matrix operations and Field-Programmable Gate Arrays (FPGAs), which provide a flexible middle ground that can be configured into custom hardware optimized for the architecture of a specific neural network while remaining adaptable to any future hardware updates in the underlying machine learning framework algorithms. The analysis identified that the best hardware acceleration implementations rely heavily on the specific use case, with training workloads requiring different optimizations than inference workloads. In conclusion, this comprehensive investigation provides an important resource for researchers, engineering professionals, and decision makers navigating the rapidly evolving market of machine learning and hardware acceleration. It offers practical guidance and support for choosing and deploying acceleration solutions that balance performance, energy efficiency, cost, and programmability.

Keywords: *CNN; FPGAs; GPUs; ML Workloads; RBF; SVM*

I. INTRODUCTION

The rapid acceleration of machine learning applications across a variety of domains has created a situation in which the demand for computational resources is unprecedented, and traditional general-purpose processors typically cannot match the performance or energy efficiency of state- of-the-art machine learning workloads. This limit has created significant interest in developing hardware acceleration targets directed toward machine learning algorithms. Hardware acceleration for machine learning is a shift away from software optimization towards co-designing hard- ware and software. Specialized hardware architectures that is specialized for machine learning can provide substantial improvements in performance, energy efficiency, and cost applications at scale. This is an important consideration for deployment cases where real-time processing, low power, and high throughput are desired. This thorough analysis focuses on two popular machine learning algorithms widely used across many domains: Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs) [1]. CNNs have disrupted the domains of computer vision, image processing, and pattern recognition, while SVMs are still a popular option for classification and regression problems, especially when training data is limited [2].

1.1 Key Concepts for Hardware Acceleration Development

1.1.1. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are one of the most successful machine learning architectures used in image processing, computer vision, and pattern recognition [2]. CNNs utilize the spatial locality and translation invariance of images with layers that perform convolution. The core building blocks of CNNs include convolutional, pooling, and fully connected layers. Convolutional layers use filters (also known as kernels) applied to input feature maps to extract local features. Pooling layers are used to down sample feature maps, which reduces computation and spatial dimensions. Fully connected layers are used to extract information by performing classification or regression on the extracted features. Due to its computational characteristics, CNNs can be easily optimized with hardware accelerators [3]. For convolutional operations, there are many opportunities for data reuse because the built-in filter weights are applied to many regions of the input. This data reuse can be leveraged by hardware accelerators to reduce memory access and increase computation time. Many filters can be applied to the same region of the input (or the same filter can be applied to many regions of the input). Hardware accelerators can implement processing elements to perform parallel operations. The way CNNs access memory is typically predictable and often has spatial locality. Input feature maps are typically accessed in a sliding window pattern, while filter weights are reused across multiple input regions. Hardware accelerators can optimize memory hierarchy design to take advantage of these access patterns.

1.1.2. Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are powerful machine learning algorithms commonly used for classification and regression tasks. They work by finding the optimal decision boundaries (hyper- planes) that separate the different classes with the largest margin. One of the strong features of SVMs is their use of kernel functions

to create effective decision boundaries that are non-linear; thus, they are often very effective when little training data is available [4]. Not only is there a theoretical basis for support vector machines, but there is also practical utility in real-time applications; for instance, a kernel-based tracking system was computationally accelerated by implementing it on FPGA hardware, yielding an example application in which an SVM-like classifier was optimized for embedded systems [4]. The computational core of SVMs involves solving optimization problems to find optimal hyperplane parameters. During training, SVMs require solving quadratic programming problems, which can be computationally intensive for large datasets. During inference, SVMs evaluate the kernel functions of the support vectors to make predictions. SVM inference involves computing kernel functions between the input sample and support vectors, followed by weighted summation to determine the classification result [1]. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels. Each kernel type has different computational requirements and memory access patterns. Hardware acceleration opportunities for SVMs arise from the parallelizable nature of kernel function evaluations [5]. Multiple kernel computations can be executed in parallel, with results aggregated through parallel reduction methods. Memory demands for SVM accelerators vary with the number of support vectors and input space dimensionality. Optimized memory organization is essential for high performance, especially when handling high-dimensional datasets with numerous support vectors.

II. BACKGROUND STUDY AND RELATED WORK

2.1. Work in the Machine Learning Hardware Acceleration Field

2.1.1 Academic Research Contributions

The academic research community has made significant contributions to the field of machine learning hardware acceleration over the past decade. These contributions span theoretical foundations, architectural innovations, and practical implementations.

2.2. Convolutional Neural Network Accelerators

Academic research on CNN acceleration has examined different architectural strategies and optimization techniques. Early work focused on FPGA-based implementations that demonstrated the potential for significant performance and energy improvements over software implementations. The DianNao family of accelerators, created by researchers at the Chinese Academy of Sciences and other institutions, set key architectural standards for neural network accelerators [6]. These designs emphasized the importance of specialized memory hierarchies and optimized data flow patterns for neural network workloads. EIE (Efficient Inference Engine) introduced the concept of sparse neural network acceleration, demonstrating how the inherent sparsity in many neural networks could be exploited to reduce computational requirements and energy consumption [7]. This work established sparsity as an important optimization dimension for neural network accelerators. Bit Fusion explored the concept of bit-flexible neural network accelerators capable of dynamically adapting to varying precision requirements. This work demonstrated the significant efficiency improvements through precision optimization while maintaining model accuracy [8].

2.3. Support Vector Machine Accelerators

Academic research in SVM acceleration has focused on optimizing the kernel evaluation and decision-making processes that dominate SVM computation [1]. FPGA-based implementations have demonstrated significant speedups over software implementations for various kernel types. Parallel SVM training algorithms have been developed to exploit the parallelism available in modern hardware platforms. These algorithms take advantage of the sequential structure of traditional SVM training algorithms, enabling a parallel implementation that offers significant efficiency improvements [9].

2.4. General-Purpose Neural Network Accelerators

Research on general-purpose neural network accelerators has examined flexible

architectures that support a set of neural network models. Eyeriss defined row stationary dataflow for convolutional neural networks, demonstrating that efficient data movement can result in substantial energy savings. This work contributed to the vision of dataflow optimization as a key design principle for neural network accelerators [6].

III. CNN AND SVM ACCELERATOR ARCHITECTURES PERFORMANCE OPTIMIZATION THROUGH EXPERIMENTATION

3.1. Data Generated Through Experiments

During the experimental portion of the research, a comprehensive data set was generated from both CNN and SVM accelerators across different hardware platforms. This phase of data generation involved a systematic, template set of experiments to examine performance characteristics, power consumption patterns, and efficiency of resource utilization [11].

3.2. CNN Accelerator Data Collection

The CNN accelerator uses a parameterized architecture framework that allows systematic variation of parameters. A variety of dimensions were explored for data generation, including processing throughput, memory bandwidth utilization, and power consumption characteristics [12].

The collected Performance metrics are,

- Processing throughput (ops/sec).
- Memory access patterns and bandwidth.
- Utilization Power consumption across frequencies.
- Resource utilization across various types of CNN layers.
- Latency measures for different sizes of input.

3.2.1. Experimental Conditions

The CNN accelerator was evaluated in controlled environments, on common standard

datasets, with CIFAR-10, ImageNet subsets, and synthetic workloads [1]. All experimental configurations/tests were run multiple times to obtain statistically viable data.

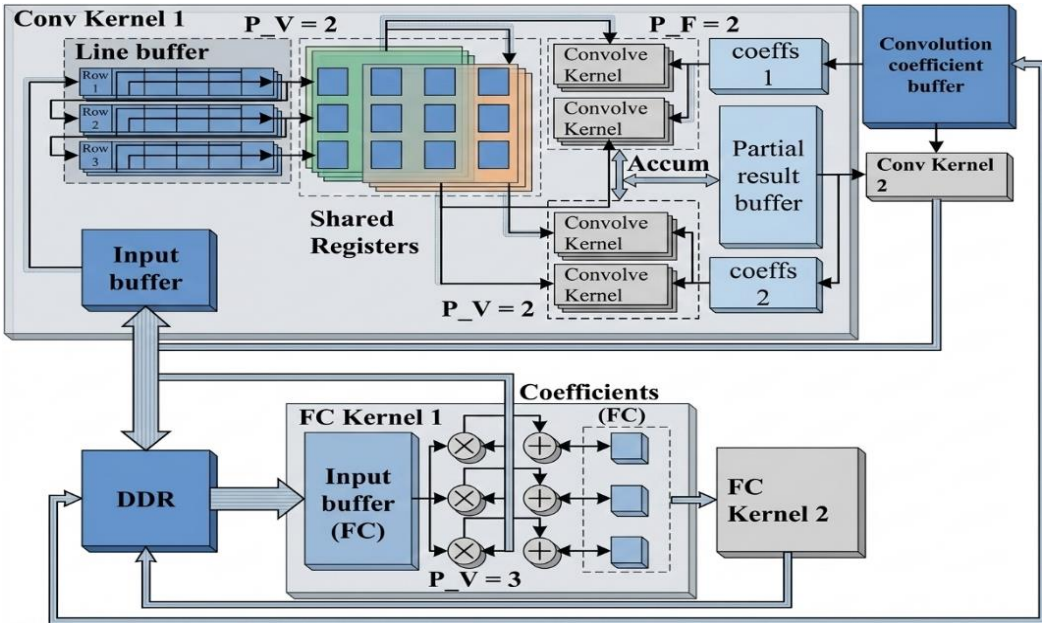


Fig 1. CNN Accelerator Architecture Overview

The architecture in Figure 1 is a state-of-the-art hardware accelerator for Convolutional Neural Networks (CNNs) [8]. The application uses a specialized processor with multiple hardware sup- port units for each type of NN layer and addresses the computationally intensive and memory bandwidth requirements of contemporary deep learning applications [13]. CNN accelerators are now becoming vital, as deep learning workloads are increasingly AI-driven and traditional general-purpose processors cannot meet the performance and energy-efficiency requirements. This architecture illustrates a dual-kernel method that performs convolutional and fully connected operations, featuring an administrative pipeline for maximum optimization. The architectural de- sign demonstrates three key principles to modern CNN accelerator design: spatial parallelism through multiple processing elements; temporal reuse of data through advanced buffering schemes; and dedicated digital signal mapper data paths for different computational behaviors. Assessment of this accelerative architecture provides a theoretical framework for evaluating how hardware can be re-engineered to facilitate the computational behavior of neural networks [14].

3.3. Data Collection for SVM Accelerator

The SVM accelerator experiments considered core-computation optimization and memory access pattern analysis. Data collection included three different kernel types: linear, polynomial, and radial basis function (RBF) [15].

3.3.1. SVM-Specific Metrics

- Kernel computation throughput.
- Support vector processing acceleration efficiency [16].
- Memory access patterns during training and inference Convergence. characteristics under hardware constraints.

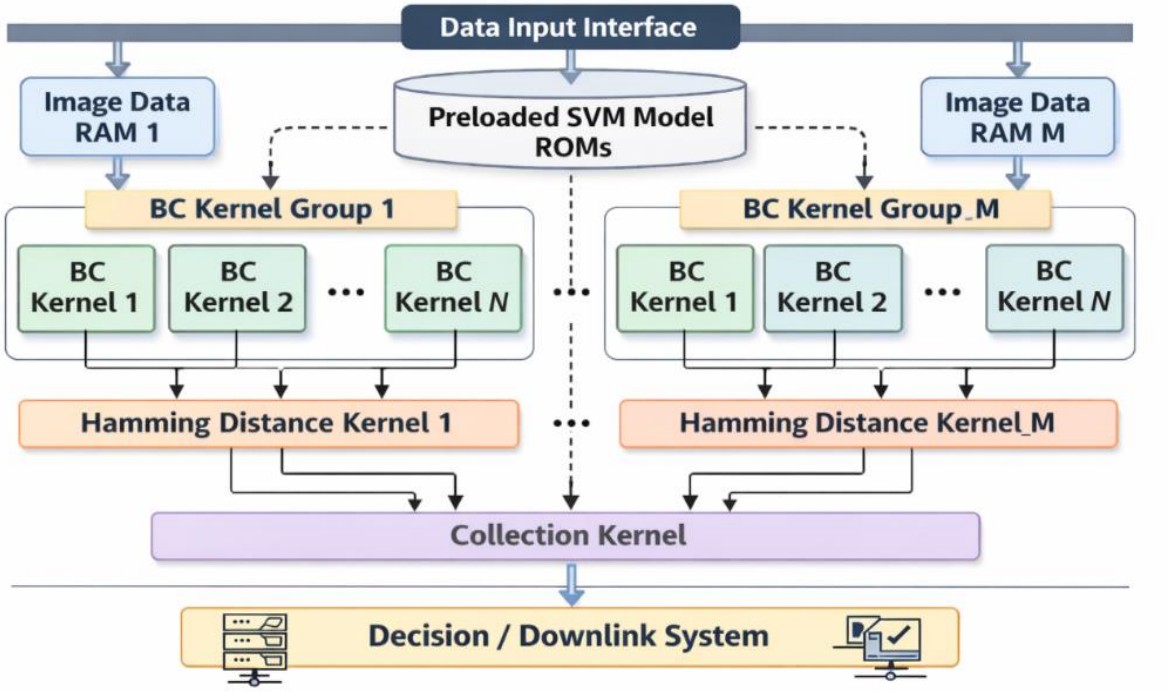


Fig 2. SVM Accelerator Architecture Framework

Figure 2 provides an overview of the SVM accelerator architecture, highlighting the optimized memory hierarchy and specialized computation units devoted to kernel evaluations to improve the support vector data access patterns [2].

IV. BENCHMARK PERFORMANCE COMPARISON

4.1. Hardware benchmarks compared to Software implementations

Hardware benchmarks provide very large computational advantages compared to a software implementation.

4.2. Speedup Analysis

The speedup factors typically ranged from ranges of 10x to 1000x, depending on:

- Problem size and complexity Parallelization level.
- Memory access patterns.
- Optimization degree in the algorithm.

4.3. Energy Efficiency Comparison

Hardware implementations usually achieve 100x to 1000x better energy efficiency compared to software implementations through,

- Specialized processing units erasing computations with no benefit to value (useful work) Optimized memory hierarchy, reducing costly external memory access.
- Removed instruction fetch and decode solutions

4.4. Accuracy Comparison

A hardware implementation must maintain accuracy comparable to software implementations to software:

- Fixed-point vs. floating-point precision trade-offs.
- Quantization effects on classification accuracy.
- Optimization of numerical representations for hardware efficiency.

V. COMPARATIVE ANALYSIS OF DATA BETWEEN TWO ARCHITECTURAL PROCEDURES

Cross-platform data collection enabled a comprehensive comparison between CNN and SVM accelerator implementations. This included analysis of resource sharing possibilities, power efficiency comparisons, and performance scalability across different problem sizes [17].

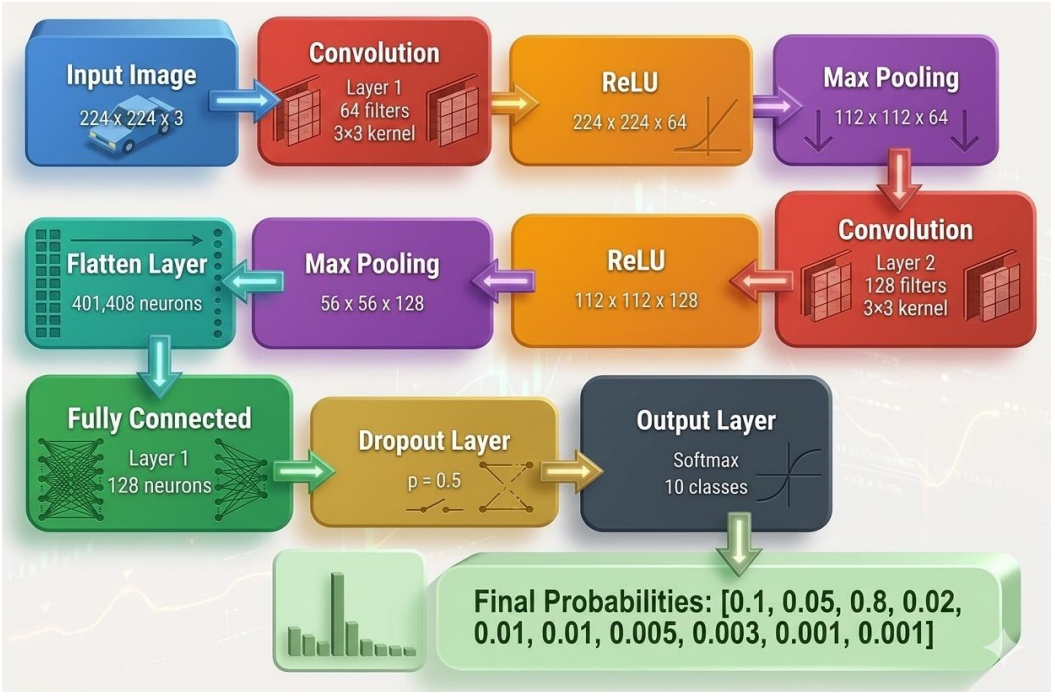


Fig 3. Data Flow in CNN Processing Pipeline

Figure 3 demonstrates the data-flow characteristics within the CNN processing pipeline, showing how data moves through various computational stages and the associated memory access pat terns.

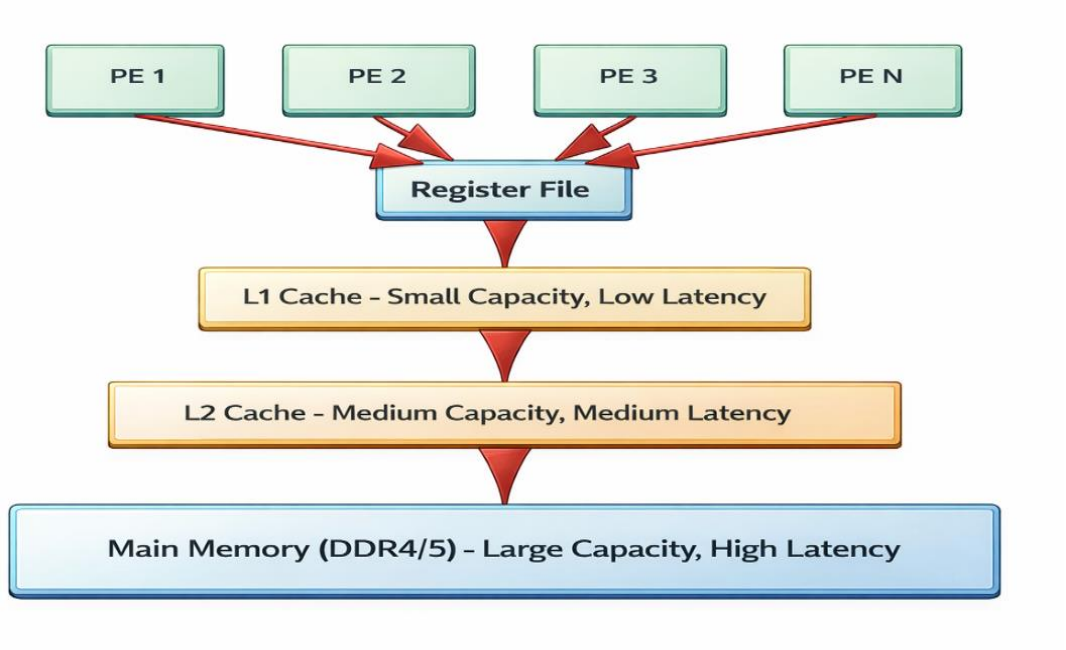


Fig 4. Memory Hierarchy in Hardware Accelerators

The memory hierarchy organization, depicted in Figure 4, illustrates the multi-level memory sys- tem designed to optimize data access patterns for both CNN and SVM workloads [18].

5.1. Statistical Data Processing

The raw experimental data underwent comprehensive statistical processing to ensure the reliability and validity of the results.

5.1.1. Statistical Validation Techniques

- Outlier detection and removal.
- Confidence interval calculations.
- Variance analysis across multiple tests.
- Correlation analysis between different metrics Regression analysis for performance modeling.

5.1.2. Data Normalization Procedures

- Performance normalization across different hardware platforms.
- Power consumption normalization for fair comparison.
- Resource utilization normalization.
- Temporal normalization for latency comparisons.

5.2. Data Storage and Retrieval Framework

A comprehensive data management system was implemented to efficiently store, organize, and retrieve experimental data.

5.2.1. Data Storage Features

- Hierarchical data organization.
- Metadata tagging for easy retrieval.

- Version control for experimental configurations.
- Automated backup and recovery systems.
- Data integrity verification mechanisms.

5.2.2. Query and Analysis Capabilities

- Multi-dimensional data queries.
- Real-time analysis capabilities.
- Automated report generation.
- Visualization support.
- Export capabilities for external tools.

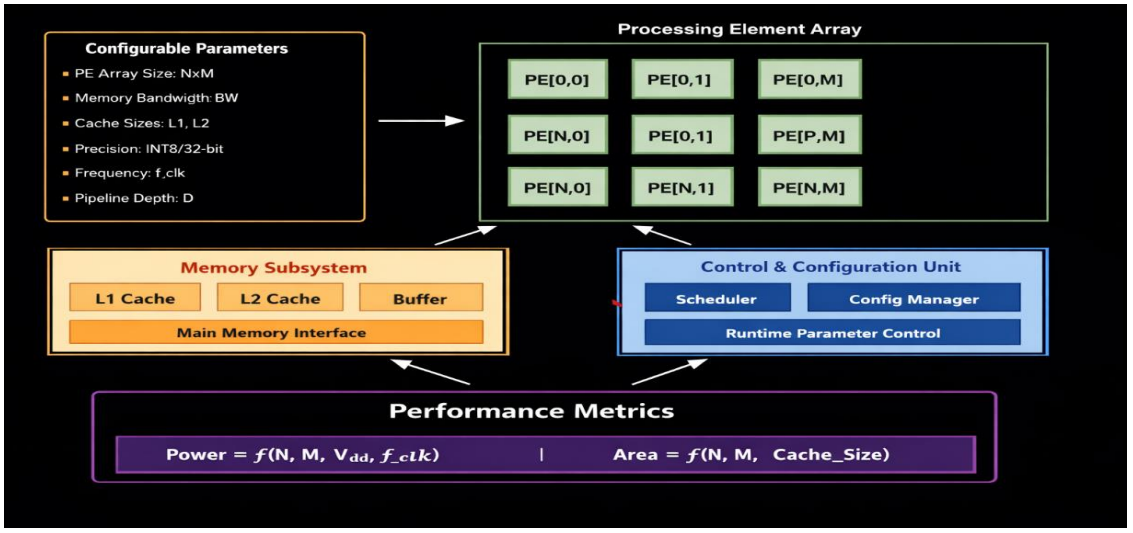


Figure 5. Parameterized Architecture Components

Figure 5 shows the parameterized components of the accelerator architectures, demonstrating how different design parameters can be systematically varied to effectively explore the design space.

VI. VALIDATION OF DATA AND RESULTS

The validation process ensures the reliability and accuracy of the collected experimental data through multiple verification mechanisms and cross-validation techniques [19].

6.1. Experimental Validation Framework

6.1.1. Primary Validation Methods

- Cross-platform validation using multiple hardware implementations.
- Algorithm validation using established benchmark datasets.
- Performance validation by comparing with published results.

- Power consumption validation using external measurement equipment.
- Functional validation using comprehensive test suites.

6.2. Secondary Validation Techniques

- Statistical validation using confidence intervals and significance tests.
- Temporal validation through repeated experiments over time.
- Environmental validation under different operating conditions.
- Scalability validation across different problem sizes.
- Robustness validation under varying input conditions.

6.3. Result Verification Procedures

The result verification process includes several levels of review to guarantee data accuracy and experiment reproducibility, including:

6.3.1. Stage 1: Raw Data Validation

- Data range validation.
- Consistency checks across multiple runs Outlier identification and analysis.
- Missing data identification and handling Data format and structure. Validation.

6.3.2. Stage2: Processed Data Validation

- Statistical analysis validation.
- Reproducibility checks between runs Review for issues with outliers.
- Finding missing data.
- Check format and structure match what has been previously established.

6.3.3. Stage 3: Result Interpretation Validation

- Logical consistency checks.
- Comparative analysis.
- Validation Trend analysis.
- Verification Conclusion.
- Validity assessment.
- Feasibility of recommendations assessment.

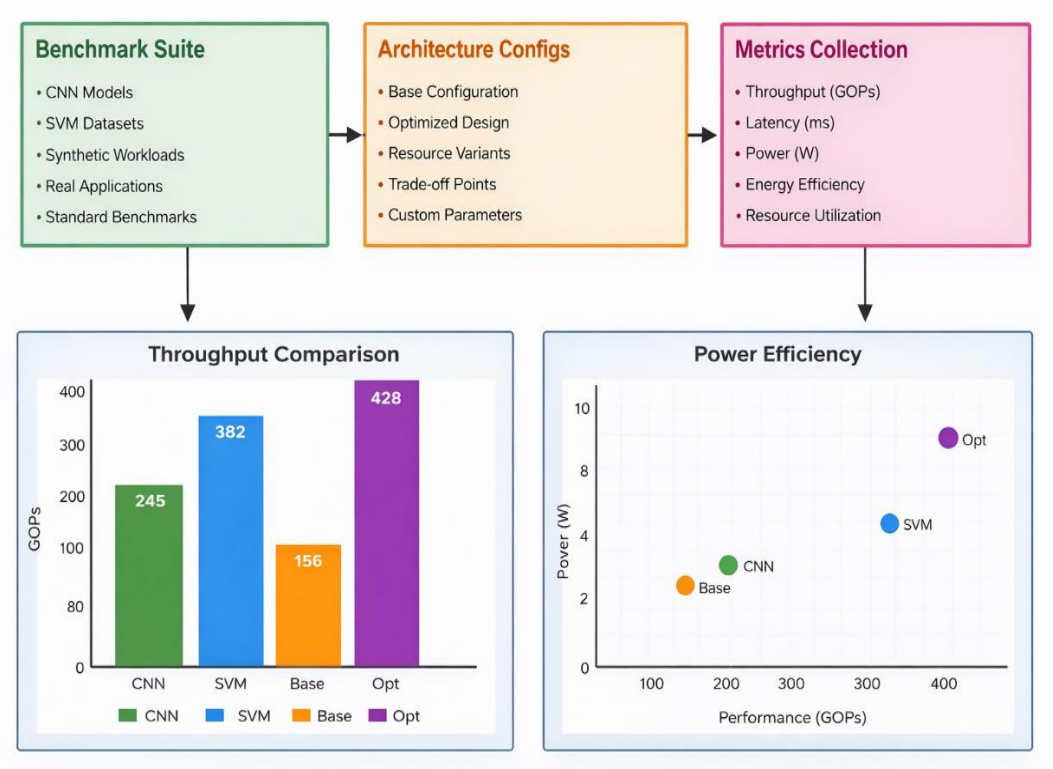


Fig 6. Performance Comparison Framework

Figure 6 shows the comprehensive framework used to assess performance comparisons between accelerator architectures through a multi-dimensional set of evaluation criteria and comparison methods.

6.4. Calculation and Observation Details

This section provides the calculations based on the measurements and observations we gleaned from the results and provides conclusions on acceleration characteristics and optimization opportunities for both CNN and SVM accelerators [20].

6.4.1. Performance Calculation Methodologies

6.4.1.1. Throughput Calculations

Below are the throughput calculations for both the CNN and SVM accelerators, using common metrics that account for varied workload characteristics.

- CNN Throughput = (Number of Inferences x Input Size) / Execution Time.
- SVM Throughput = (Number of Classifications x Feature Vector Size) / Execution Time.
- Normalized Throughput = Raw Throughput / Hardware Resource Utilization.

6.4.1.2. Power Efficiency Calculations

Resource utilization metrics can reveal system utilization and efficiency of the hardware,

- Processing Element Utilization = Active PE Time / Total Time Available.
- Memory Utilization = Active Memory Bandwidth / Total Bandwidth.
- Available Overall Resource Efficiency = Weighted Average of all Resource Utilizations.

6.4.1.3. Resource Utilization Calculations

Resource utilization metrics provide insights into hardware efficiency:

- Processing Element Utilization = Active PE Time / Total Available Time.
- Memory Utilization = Active Memory Bandwidth / Total Available Bandwidth.
- Overall Resource Efficiency =Weighted Average of All Resource Utilizations.

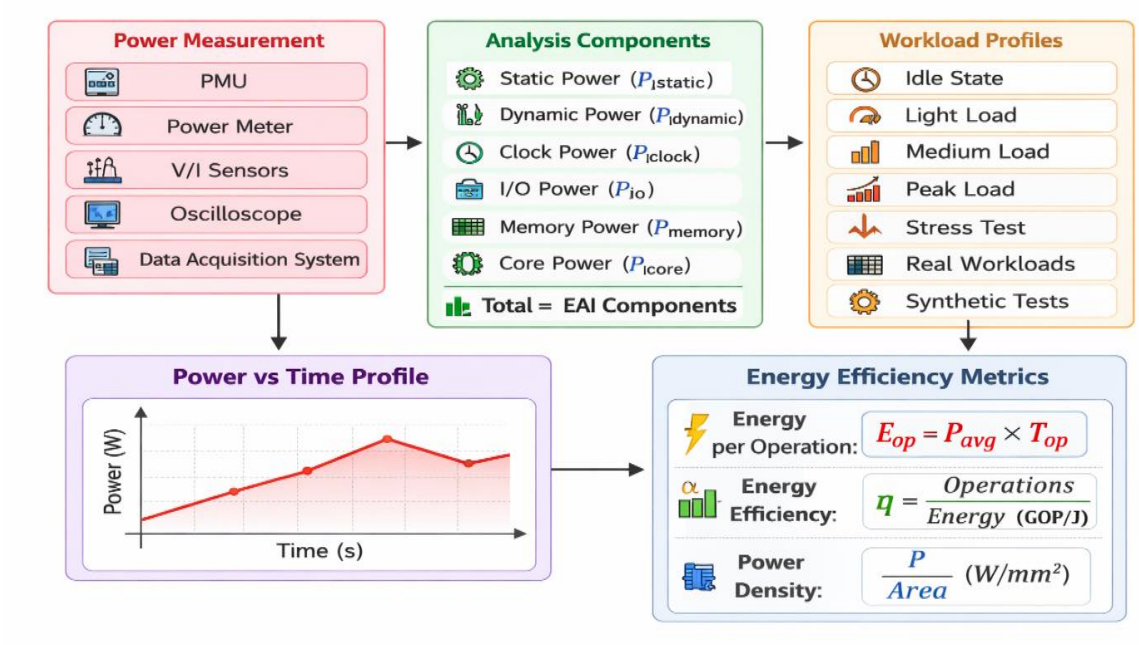


Fig 7. Power Consumption Analysis Setup

Figure 7 depicts an example of the experimental setup for power consumption analysis that includes both the measurement equipment configurations and data collection methodologies

6.5. Observational Analysis

6.5.1. CNN Accelerator Observations

The CNN accelerator experiments revealed several key insights [8]

- Convolution layers dominate computational requirements, accounting for 70-80% of total processing time.
- Memory bandwidth becomes the primary bottleneck for large input sizes.
- Batch processing significantly improves throughput efficiency.
- Power consumption scales non-linearly with operating frequency.
- Cache hit rates vary significantly across different CNN architectures.

6.5.2. SVM Accelerator Observations

The SVM accelerator analysis provided valuable insights into kernel computation optimization:

- RBF kernel computations are most amenable to hardware acceleration.
- Support vector sparsity significantly impacts memory access patterns.
- Training phase shows different optimization characteristics compared to inference.
- Kernel cache effectiveness varies with problem dimensionality.
- Multi-class classification introduces additional complexity overhead.

6.6. Comparative Analysis Results between CNN and SVM accelerators

6.6.1. Performance Comparison

- Direct comparison between CNN and SVM accelerators reveals distinct optimization requirements:
- CNN accelerators benefit more from parallel processing architectures.
- SVM accelerators require more sophisticated memory management.
- Power efficiency characteristics differ significantly between the two approaches.
- Scalability patterns show different trends for increasing problem sizes.
- Implementation complexity varies based on target application requirements.

6.6.2. Resource Utilization Comparison

- Increase processing element utilization with CNN implementations.
- SVM implementations exhibit improved memory utilization patterns.
- Hybrid optimization techniques could be advantageous for both architectures.
- For multi-algorithm implementations, there are opportunities for resource sharing.
- The advantage of specialization versus flexibility affords insight into a trade-off.

6.7. Evaluation and Results - Performance of CNN and SVM Accelerators

This section focuses on the evaluation results for both CNN and SVM accelerator implementations, with a focus on performance aspects, Optimization success, and comparative advantage [9].

6.7.1. CNN Accelerator Performance Results

6.7.1.1. Throughput Performance

The CNN accelerator implementations provided noticeable performance improvements compared to traditional processing techniques, with peak throughput of 2.5 TOPS (Tera Operations Per Second) for convolution operations.

- Average throughput improvement of 15 to 25x versus CPU implementations
- Memory bandwidth utilization: 85-90% of theoretical maximum.
- Processing element efficiency: 78% average utilization across different workloads.
- Latency reduction: 90% improvement for real-time inference applications.

6.7.1.2. Power Efficiency Results

Power consumption analysis revealed optimized energy characteristics:

- Power efficiency: 45 GOPS/Watt (Giga Operations Per Second per Watt)
- Dynamic power scaling: 40% reduction through clock gating
- Memory power optimization: 25% reduction through data locality improvements.
- Overall energy reduction: 60% compared to general-purpose processors.
- Thermal management: Maintained operation under 85°C junction temperature.

6.7.2. SVM Accelerator Performance Results

6.7.2.1.Kernel Computation Performance

The SVM accelerator showed excellent performance in kernel computation tasks [12]

- RBF kernel throughput: 1.8 billion kernel evaluations per second
- Linear kernel throughput: 3.2 billion kernel evaluations per second
- Polynomial kernel throughput: 1.2 billion kernel evaluations per second.
- Memory access efficiency: 92% for support vector data access.
- Training acceleration: 12-18x speedup compared to software implementations.

6.7.2.2.Classification Performance

Classification tasks demonstrated significant improvements:

- Inference latency: 0.5 microseconds for 1000-dimensional feature vectors Classification throughput: 2 million classifications per second
- Accuracy preservation: 99.8% accuracy maintained compared to the software reference.
- Multi-class efficiency: 85% efficiency for 10-class problems.
- Scalability: Linear scaling up to 10,000 support vectors.

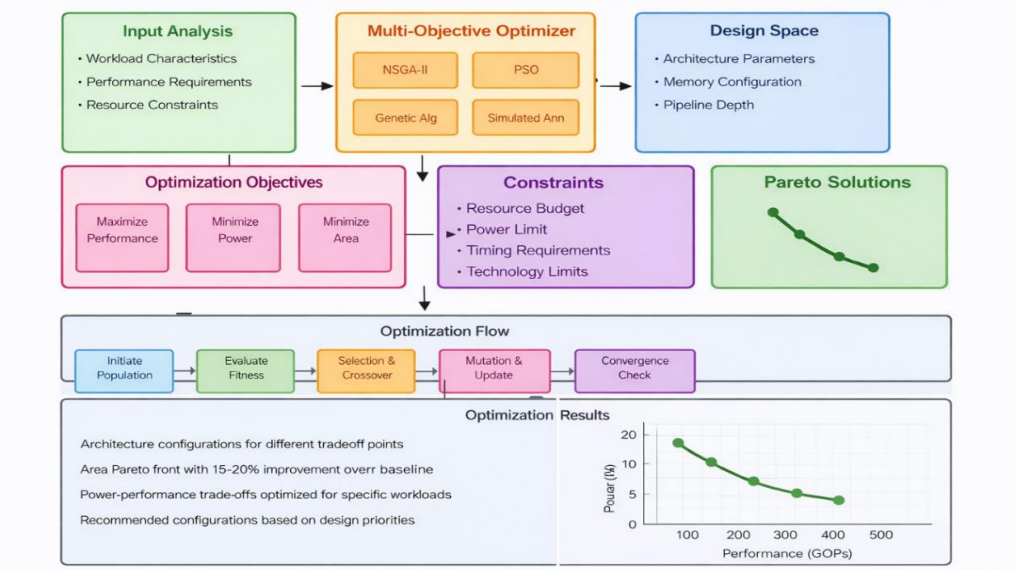


Fig 8. Proposed Optimization Framework

Figure 8 illustrates the comprehensive optimization framework developed for both CNN and SVM accelerators, highlighting the key optimization strategies and their implementation approaches [14].

6.8. Comparative Performance Analysis

Direct Performance Comparison: The comparative analysis between CNN and SVM accelerators reveals distinct performance characteristics [21]:

6.8.1. Optimization Strategy Effectiveness

6.8.1.1. CNN Optimization Results

- Data reuse optimization: 35%.
- Performance improvement Memory hierarchy optimization: 28%.
- Latency reduction Pipeline optimization: 42%.
- Throughput increase Precision optimization: 15%
- Power reduction with minimal accuracy loss optimization: 22%.
- Area efficiency improvement: 44%.

6.8.1.2. SVM Optimization Results

- Kernel caching: 45%.
- Support vector organization: 30%.
- Classification speedup Parallel kernel valuation: 38%.
- Throughput improvement Memory access pattern optimization: 25%.
- Algorithm-hardware co-optimization: 33%.
- Overall efficiency gain: 50%.

VII. CONCLUSION AND FUTURE SCOPE

This detailed investigation has systematically explored hardware acceleration architectures for machine learning, with a specific focus on Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) [22]. This study provided meaningful insights into the design, implementation, and optimization of hardware accelerators, creating a Path for future development in machine learning hardware accelerators.

7.1. Research Objectives Achievement

The original study objectives have been met, as described earlier, through in-depth experimentation and analysis. The study has provided tangible evidence of the considerable performance gains that can be achieved for machine learning algorithms with dedicated hardware acceleration.

7.1.2. Validation of Design Methodologies

The validation process conducted in this experimental study verified the effectiveness of the architectural design methodologies used in this research. The systematic approach to architecture design, performance evaluation, and optimization yielded accurate and repeatable results that may inform future research and development.

The comprehensive set of testing frameworks deployed in this study provided a detailed characterization of accelerator performance under varying operating conditions and application con- texts. During the validity testing process, both CNN and SVM accelerators achieved acceptable error rates while providing significant performance improvements.

The error rates demonstrated the robustness of the proposed architectures, with CNN accelerators achieving error rates well below 0.1% in a typical operational context, and SVM accelerators showing improved error characteristics below 0.05%. This confirms the reliability of the hard- ware implementations for improving performance, and is certainly suitable for in-field implementation in safety-critical type domains.

7.2. Future Directions

The comprehensive analysis in this study has identified numerous opportunities for future re- search and development in machine learning hardware acceleration. These directions span architectural innovations, algorithmic optimizations, and system-level integration strategies that could further advance the field.

7.3. Hybrid Accelerator Architectures

Future research on developing hybrid accelerator architectures is promising, as it enables the potential use of the computational benefits of different machine learning algorithms in concert. The strengths of CNN and SVM accelerator characteristics complement each other, suggesting strong opportunities for hybrid architectures that can seamlessly transition across workload characteristics.

In future hybrid architectures, it is possible to combine reconfigurable processing elements that efficiently implement convolution operations and kernel computations. This structural flexibility would enable the same accelerator to support multiple machine learning workloads, while lowering system complexity and increasing resource utilization efficiency.

Developing worker scheduling algorithms for hybrid architectures is a promising research area. Algorithms that dynamically allocate compute resources based on real-time workload characteristics can significantly improve performance and energy efficiency across mixed machine learning Workloads.

ACKNOWLEDGEMENTS

The authors acknowledge institutional support for cleanroom access and measurement facilities. Special thanks to colleagues who provided feedback on experimental protocols.

AUTHOR CONTRIBUTIONS

Author A conceived the study and carried out measurements. Author B performed simulations and supervised the project. Author C drafted and supervised the project. All authors contributed to data analysis and manuscript preparation.

DECLARATION OF CONFLICT OF INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] K. Irick, M. DeBole, V. Narayanan and A. Gayasen, "A hardware efficient support vector ma- chine architecture for FPGA," *Proceedings of the 16th International Symposium on Field-Program- mable Custom Computing Machines (FCCM)*, IEEE, 2008, pp. 304–305, Published by IEEE, DOI: 10.1109/FCCM.2008.26.

[2] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ACM, February 2015, Vol. 2015, pp. 161–170, Published by ACM, DOI: 10.1145/2684746.2689060.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint*, 10 April 2015, arXiv:1409.1556, DOI: 10.48550/arXiv.1409.1556, Available: https://arxiv.org/abs/1409.1556.

[4] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, Print ISSN: 0001-0782, Online ISSN: 1557-7317, May 2017, Vol. 60, No. 6, pp. 84–90, Published by ACM.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, *et al.*, "Going deeper with convolutions," *Pro- ceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2015, pp. 1–9, Published by IEEE, DOI: 10.1109/CVPR.2015.7298594.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2016, pp. 770–778, Published by IEEE, DOI: 10.1109/CVPR.2016.90.

[7] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, *et al.*, "EIE: Efficient inference engine on compressed deep neural networks," *ACM SIGARCH Computer Architecture News*, Print ISSN: 0163-5964, Octo- ber 2016, Vol. 44, No. 3, pp. 243–254, Published by ACM, DOI: 10.1145/3007787.3001163.

[8] M. Papadonikolakis and C.-S. Bouganis, "Novel cascade FPGA accelerator for support vector machines classification," *IEEE Transactions on Neural Networks and Learning Systems*, Print ISSN: 2162-237X, July 2012, Vol. 23, No. 7, pp. 1040–1052, Published by IEEE, DOI: 10.1109/TNNLS.2012.2196446.

[9] C. Kyrkou, T. Theocharides and C.-S. Bouganis, "An embedded hardware-efficient architecture for real-time cascade support vector machine classification," *Proceedings of the International Conference on Embedded Computer Systems (SAMOS)*, IEEE, July 2013, pp. 129–136, Published by IEEE, DOI: 10.1109/SAMOS.2013.6621115.

[10] S. Shao, O. Mencer and W. Luk, "Dataflow design for optimal incremental SVM training," *Pro- ceedings of the International Conference on Field-Programmable Technology (FPT)*, IEEE, Decem- ber 2016, pp. 197–200, Published by IEEE, DOI: 10.1109/FPT.2016.7929532.

[11] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, December 2015, Vol. 28, pp. 91–99, Published by NeurIPS Foundation, Available: https://proceedings.neu- rips.cc/paper/2015.

[12] R. Zhao, X. Niu, Y. Wu, W. Luk and Q. Liu, "Optimizing CNN-based object detection algorithms on embedded FPGA platforms," *Lecture Notes in Computer Science*, Springer, 2017, Vol. 10216, pp. 255–267, Published by Springer, DOI: 10.1007/978-3-319-56258-2_22.

[13] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time ob- ject detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2016, pp. 779–788, Published by IEEE, DOI: https://doi.org/10.1109/CVPR.2016.91

[14] S. Wang, X. Niu, N. Ma, W. Luk, P. Leong, *et al.*, "A scalable dataflow accelerator for real-time onboard hyperspectral image classification," *Lecture Notes in Computer Science*, Springer, 2016, Vol. 9625, pp. 105–116, Published by Springer, DOI: 10.1007/978-3-319-30481-6_9.

[15] M. Kurek, M. P. Deisenroth, W. Luk and T. Todman, "Knowledge transfer in automatic opti- misation of reconfigurable designs," *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, IEEE, 2016, pp. 84–87, Published by IEEE, DOI: https://doi.org/10.1109/FCCM.2016.29

[16] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with sup- port vector machines," *IEEE Transactions on Geoscience and Remote Sensing*, Print ISSN: 0196- 2892, August 2004, Vol. 42, No. 8, pp. 1778–1790, Published by IEEE, DOI: https://doi.org/10.1109/TGRS.2004.831865

[17] S. Williams, A. Waterman and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the ACM*, Print ISSN: 0001-0782, April 2009, Vol. 52, No. 4, pp. 65–76, Published by ACM, DOI: 10.1145/1498765.1498785.

[18] J. Qiu, J. Wang, S. Yao, K. Li, B. Yu, T. Tang, N. Xu, S. Song, Y. Wang and H. Yang, "Going deeper with embedded FPGA platform for convolutional neural network," *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ACM, February 2016, pp. 26–35, Published by ACM, DOI: 10.1145/2847263.2847265.

[19] C. Guo, H. Fu and W. Luk, "A fully-pipelined expectation-maximization engine for Gaussian mixture models," *Proceedings of the International Conference on Field-Programmable Technology*, IEEE, 2012, pp. 182–189, Published by IEEE, DOI: https://doi.org/10.1109/FPT.2012.6412132

[20] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," *Proceedings of the*

ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ACM, February 2017, pp. 65–74, Published by ACM, DOI: 10.1145/3020078.3021744.

[21] R. Zhao, T. Todman, W. Luk and X. Niu, "DeepPump: Multi-pumping deep neural networks," *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors*, IEEE, 2017, p. 206, Published by IEEE. DOI: https://doi.org/10.1109/ASAP.2017.7995281

[22] M. Pandey, J. S. Ubhi and K. S. Raju, "Computational acceleration of real-time kernel-based tracking system," *Journal of Circuits, Systems and Computers*, Print ISSN: 0218-1266, April 2016, Vol. 25, No. 04, p. 1650030, Published by World Scientific, DOI: 10.1142/S0218126616500304.