OPTIMIZATIONS IN APPLIED MACHINE LEARNING
Research Article* | *Volume 4* | *Issue 4* | *Apr 2024
Received: 2 Jan 2024 | Revised: 13 Feb 2024
Accepted: 28 Mar 2024 | Published Online: 10 Apr 2024

Accelerating Digit Classification on FPGA with Pruned Binarized Neural Networks

Syamantak Payra[1, 4], Gabriel Loke[2], Yoel Fink[3, 4], Joseph D. Steinmeyer[5, *]

[1] Dept. of Electrical Engineering, Stanford University, 94305, CA, USA

[2] Dept. of Materials Science and Engineering, Massachusetts Institute of Technology, 02139, MA, USA

[3] Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 02139, MA, USA

[4] Institute for Soldier Nanotechnologies, Massachusetts Institute of Technology, 02139, MA, USA

[5] Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 02139, MA, USA

[*]Corresponding Author, Email address: jodalyst@mit.edu

Abstract: As neural networks are increasingly deployed on mobile and distributed computing platforms, there is a need to lower latency and increase computational speed while decreasing power and memory usage. Rather than using FPGAs as accelerators in tandem with CPUs or GPUs, we directly encode individual neural network layers as combinational logic within FPGA hardware. Utilizing binarized neural networks minimizes the arithmetic computation required, shrinking latency to only the signal propagation delay. We evaluate size-optimization strategies and demonstrate network compression via weight quantization and weight-model unification, achieving 96% of the accuracy of baseline MNIST digit classification models while using only 3% of the memory. We further achieve 86% decrease in model footprint, 8mW dynamic power consumption, and <9ns latency, validating the versatility and capability of feature-strength-based pruning approaches for binarized neural networks to flexibly meet performance requirements amid application resource constraints.

Keywords: *Algorithms Implemented in Hardware, Combinational Logic, Cost/Performance, Neural Nets, Optical Character Recognition*

1. **Introduction**

In recent years, the increased utilization of artificial intelligence in everyday technologies has necessitated advances in computer architectures for both training and running machine learning models. A main limitation of neural edge-computing infrastructures is the memory- and computation-intensive nature of conventional models. Thus, it is of increasing interest to investigate reduction of computational requirements for neural networks [1]. Some recent efforts have explored the use of field- programmable gate arrays (FPGAs) as hardware accelerators for specific elements of neural network training and evaluation [2]. Logic blocks within FPGAs can be dynamically reconfigured to adjust to different computational tasks, allowing for more efficient, task-specific logic circuitry, such as accelerating image processing while reducing power consumption compared to conventional embedded system platforms [3], [4], [5], [6].

However, when applied to neural network computation, FPGAs are conventionally utilized in tandem with CPUs and GPUs [7]. Because basic neural functionalities can be replicated with transistor arrangements [8], there have been efforts to fabricate neural network-specific hardware components [9], [10]. However, the intrinsic reconfigurability of FPGAs can be further utilized to maximize functionality: dynamic memory reallocation has allowed hardware footprint minimization in various data processing tasks [11], and there has been recent interest in architectures that embed neural networks within FPGAs [12], [13], [14]. Such approaches have multiple benefits: not only can FPGA floating-point hardware accelerate neural network arithmetic [15], the large number of logic gates available in FPGA fabric allows pipelining and duplication of network segments to simultaneously perform computations on separate data [16]. Indeed, FPGAs outperform mobile platforms on machine learning benchmarks and real-time computer vision with higher efficiency [17], [18], and have been utilized for applications like particle physics experimentation to analyze collision byproducts that disappear within nanoseconds [19].

Such latency-reduction approaches are relevant to meet ever- tightening latency and performance requirements for computing needs like artificial intelligence-based services [20]. This has even led to the pursuit of alternative hardware, such as optical neural networks, to decrease latency [21]. Additional motivation for latency reduction arises from a need for data encryption for privacy: Even while leveraging parallel processing for improved throughput, performing predictions on encrypted data can require high latencies of up to 250 seconds [22], and networks that achieve 290ms latency on encrypted handwritten digits are constrained by limitations of transfer learning [23]. Recent work on secure inference has achieved 30ms latency for MNIST digit classification [24], and demand for near-instant predictions urges a search for strategies to reduce latency further.

Advancements toward making neural networks intrinsically more efficient have include compressing models by pruning the parameters that are invoked, quantizing weights, and distilling internal knowledge representations [25], [26]. Large-scale commercial approaches have demonstrated 8-bit hybrid calculations capable of similar performance as 32-bit floating point operations [27]. Some approaches with FPGA hardware optimize model design for individual accelerators [28], while others build a physical network pipeline [29]. Other biologically-inspired architectures improve energy efficiency, achieving 95% accuracy with 20ms mean latency at 0.3 watts of board power, but require large specialized hardware [30].

2. **Objectives**

Building upon the existing literature, we identify the need to explore novel approaches to hardware implementations of neural networks that can achieve high accuracy with low latency and low power consumption, maintaining the seamless user experiences of mobile and wearable platforms [31]. Techniques like weight binarization have shown promise in reducing the number of calculations required in computing results, enabling acceleration with minimal impact on accuracy [32], [33], [34], and are uniquely adaptable to implementation in hardware logic gates. By minimizing memory transfers and instruction-based computations, computational load significantly decreases and model latency can be reduced to merely the signal propagation delays through sequential binary logic.

The goal of this work was thus to propose and explore a novel FPGA architecture approach for machine learning computation on low-power platforms, demonstrating binarized neural networks and comparing the efficacy of size-optimization strategies, specifically with regard to the continuous tradeoff between algorithm compression and key performance indicators of accuracy, latency, network size, and power efficiency.

3. **Methods / Approach**

To evaluate classification capability on a well-validated computational task, strategies were developed for FPGA-based handwritten digit classification on the MNIST dataset, a longstanding benchmark for machine learning [35]. Within the original dataset, each image has a 28x28 resolution (784 pixels per image) in 8-bit grayscale (values 0 to 255). In order to process MNIST digits in binarized neural networks, the input data were converted from grayscale to binary representations. A fixed threshold of 50% grayscale (8-bit value 128/256) was used as the threshold to determine binary representation for each pixel (<128 = “0”, >128 = “1”). Each converted image thus contained 784 bits, with each pixel corresponding to background (binary “0”) or digit (binary “1”). This binary-converted dataset was used to train and evaluate multiple network implementations.

First, we created and tested a set of minimal hardware implementations (Section IV), benchmarking the potential for basic pattern-matching algorithms-in-hardware to successfully classify handwritten digits. The results from these pattern- recognition algorithms were used to determine the functional characteristics of the binarized dataset, and inform the subsequent, progressively more advanced algorithms.

Next, we created a baseline single-layer neural network (Section V) by taking a neural network architecture that is well- validated for handwritten digit detection and adapting it for direct deployment on an FPGA, then evaluating performance of the modified algorithm. We further optimized and evaluated the model with weight quantization and strategic simplifications to reduce memory usage and computational requirements.

In our final approach, we developed a binarized neural network (Section VI) that utilizes binary pixel weights rather than integer weights, decreasing integer arithmetic operations. We compare multiple pixel-weight selection and pruning strategies and evaluate their impacts on model size and accuracy. Additionally, we pursue further model compression through weight-model unification, condensing network logic to include weights in computational logic rather than in memory. We then evaluate key performance indicators and resource utilization of the pruned models, and demonstrate the ability of

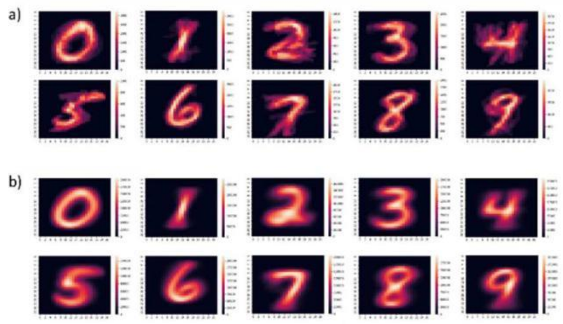


Fig. 1. Digit heatmaps generated from training dataset.

a)1000 images, b) 50,000 images. Upper row = 0-4, Lower row = 5-9

such approaches to optimize binarized neural networks.

In order to create these hardware-embedded neural networks, all algorithms were first simulated with Python 3.7.3 and TensorFlow 2.0 [36] to verify and evaluate model functionality, then implemented on a Xilinx Artix-7 FPGA development board (XC7A100T, Digilent Nexys 4 DDR) using SystemVerilog and the Xilinx Vivado HLx design suite. Additional performance metrics and hardware layout are derived from Xilinx Vivado.

4. **Minimal Hardware Implementations**

Statistical approaches to parsing handwritten digits include multinomial lassos to identify pixel predictance values and sparse principal components analysis to identify key component pixels for different handwritten digits [37]. While both of these methods require lengthy mathematical instruction sets, they offer a starting point for a minimal classification algorithm, without the need for computationally expensive convolutions.

A. Digit Predictor Pixels

Combining these approaches, we first develop a hardware implementation based on digit "predictor pixels". Fig. 1 contains heatmaps corresponding to the pixel occurrence frequency in each of ten MNIST digits, from the first 1000 and the full 50,000 training images. Different digits have primary locations in which pixels are active; the location of active pixels in a test image is utilized as a proxy for the digit contained within the image. Fig. 2a demonstrates an initial predictor-pixel matrix, constrained to pixels with only positive values. We observe that some digits are significantly over-represented; for example, "4" has far fewer active pixels than "0" or "7". To compensate for this, an activation threshold was implemented to remove low-intensity

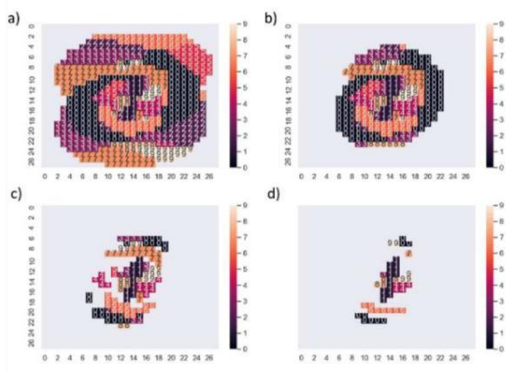


Fig. 2. Matrix of predictor pixels.

a) 0 threshold, b) 100 threshold, c) 170 threshold, d) 190 threshold.

pixels. Fig. 2b, 2c, and 2d show the predictor-pixel matrix with activation thresholds of 100, 170, and 190 (out of 28 = 256).

In this method, the final classification result was determined to be the digit (0-9) with the largest number of activated pixels, given a binarized input image. The accuracy across the test dataset was observed to be 28.07%. While 3x better than chance, the predictive power is hampered by the many cases in which the number of activated pixels within the designated regions is identical across multiple candidate digits. In other words, having a low activation threshold leads to low accuracy, as inputs are preferentially classified as the digits with more predictor pixels; increasing the activation threshold results in low discrimination capability between multiple potential classifications.

B. Limitations and Extensions

While the predictor-pixel method is appealing from a simplicity standpoint (as all inputs can be matched against one aggregate reference), we observe that this approach has limited accuracy and multiple limitations. For instance, the presence of positive-valued pixels that are common across multiple digits decreases the predictive power of those individual pixels. When one digit has a slightly higher average activation within one pixel compared to other digits, selecting a small number of predictor pixels to determine digit classifications amplifies those slight distinctions might falsely skew results toward certain digits over others.

5. **Single-Layer Neural Network**

A. Structuring, Training, and Validation

Network architecture plays a large role in determining performance and accuracy, especially with compact models or small datasets [38]. In order to establish a baseline neural network architecture for binarization and optimization, we adapt the LeNet architecture [39], which is both compact (~6-7 layers) and accurate (>90%), and has been adapted to a variety of classification tasks, such as image recognition and facial recognition [40]. To traverse the complexity-accuracy tradeoff, we pared down model parameters to evaluate performance in progressively minimal models. All models were constructed and evaluated using Python and Tensorflow, dividing the 60,000 images in the dataset into a training-testing split of 90%/10%.

Condensing the network to two convolutional layers and one fully-connected dense layer 1024 neurons wide facilitates a classification accuracy of 98.6%, while a version of the same network without the dense layer is able to achieve 97.9% accuracy. With the removal of the second convolutional layer (leaving one convolutional layer and one dense layer), we achieve 96.2% MNIST accuracy. Next, we eliminated the convolutional layer entirely, in order to form direct parallels between a conventional single-layer implementation and single-

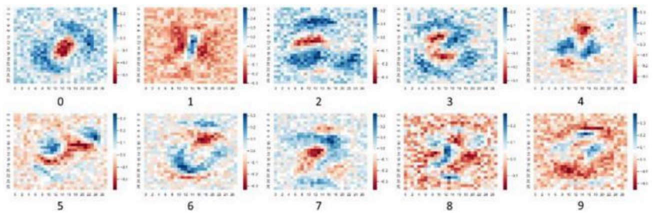


Fig. 3. Pixel weight heatmaps for each digit.

layers implemented in FPGA hardware. This model (Appendix Fig. 1) successfully achieves 91.5% accuracy, showing that even a single-layer implementation can offer significant predictive power without the need for costly convolution operations.

As shown in Fig. 3, the single-layer neural network operates on ten sets of 784 weights, one set per digit and one weight per pixel. Blue pixels indicate strong predictors of a particular digit, while red pixels imply that the presence of pixels in that region reduce the likelihood that that is the correct digit. Functioning essentially as a linear classifier, not only does this single-layer neural network provide a simple way to rapidly recognize MNIST handwritten digits with minimal calculations, this method also retains higher accuracy than convolutional neural networks (CNNs) trained on subsets of the MNIST dataset [41]. Hence, our single-layer classifier shows good performance while minimizing the computational hardware footprint.

B. Model Optimization

For implementation into FPGA fabric, we consider both the operations required to execute the model, as well as the connections between logic blocks necessary to facilitate the operations. We begin optimizing the neural network itself by distilling the architecture into a single layer, which allows us to treat the entire prediction-serving region as a single hardware module. Next, we identify two areas of optimization: the weights within the model, and the activation functions for the output.

A primary source of computational overhead during execution of neural networks is due to the mathematical operations necessary when multiplying input data by sequences of weights and summing inputs into activation functions, especially since floating-point mathematics requires additional hardware resources and clock cycles. Weight quantization can address these problems [25], [27]. We quantize the weights in our single-layer classifier from 32-bit floating point decimals to 8-bit integers, resulting in a 75% decrease in weight size from $10 \cdot 784 \cdot 32 = 250{,}880$ bits, to $10 \cdot 784 \cdot 8 = 62{,}720$ bits, with only a 0.3% decrease in accuracy (91.5% to 91.2%).

C. Implementation Architecture

The second aspect of computational overhead involves the mathematical activation functions utilized to compile results from each layer. The base neural network utilized a softmax activation function, which has been approximated in FPGA hardware [42]. However, mirroring the simplicity inherent to a single-layer slice of a model, we implement a maximum-value evaluator, which allows us to retain full accuracy while eliminating the need to instantiate additional arithmetic computation modules, minimizing hardware footprint.

Once deployed in FPGA hardware, the single-layer neural network comprises a single set of parallel pipelines. The weights for each of the ten digit nodes are stored in a set of registers, and the input image is simultaneously routed and matched against the 784x8 arrays containing the 8-bit weights. Because the input image is binary, the multiplication operation consists of a set of AND operations between each input pixel and its corresponding weight. The selected weights are added to create the output sum for a particular digit. The output sums are compared across digit nodes, and the node with maximum value is the digit output.

This single-layer, 8-bit quantized neural network (Appendix Fig. 2) achieves minimal latency, with a signal propagation delay of under 7ns per pixel, and high energy-efficiency, with a Vivado- estimated 0.007W of dynamic power consumption.

6. **Binarized Neural Network**

To pursue a more tightly-coupled neural network within FPGA hardware, we binarize the neural network, with binary weights that can be represented as transistorized logic rather than arithmetic operations. Replacing arithmetic computation with bitwise operations has been shown to improve power efficiency and computational speed, and reduce memory use and number of memory accesses required to calculate each layer within a model [33]. Direct binarization of the 8-bit quantized neural network yields almost 88% reduction in weight size; we additionally explore further reductions in size to evaluate performance of highly compact models-in-hardware.

A. *Weight Conversion and Implementation*

Previous works have selectively binarized portions of networks [43]; this work sought to characterize multiple points in the model accuracy-size space. Multiple approaches have been taken to forming and pruning binarized neural networks, such as isolating and trimming vacillatory weights that flip polarity many times near the end of model training completion

[44] or removing clusters of weights that have smaller effects on output accuracy [45]. Unique biomolecular "winner-take-all" systems have also been created for DNA pattern recognition

[46]; this can be reframed as a form of binary logic with each input corresponding to a certain "pixel" of the desired signal and the classifier as a series of logic operations reaching a deterministic outcome based on certain combinations of inputs. Based on these strategies, we binarize our quantized single- layer neural network by identifying the strongest "predictor pixels" as the top-N largest values given a number of pixels to be calculated (N) for each digit map. Each set of predictor pixels is stored in a 784-bit variable in which each pixel position is denoted with a 0 or 1 if that pixel is a designated predictor for that digit. The model iterates through each index at a rate of one input pixel per clock cycle, then tallies and compares the sums across digits to determine the final digit classification.

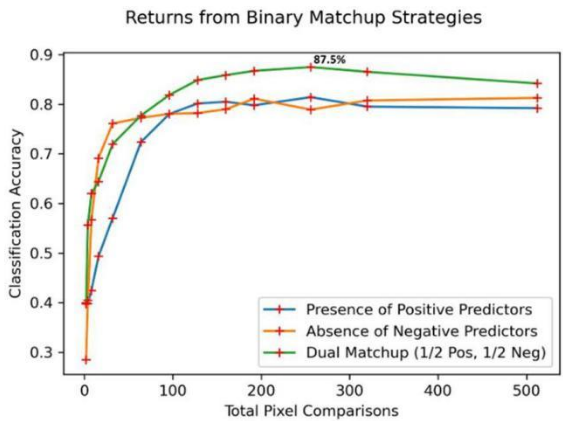


Fig. 4. Binarized Neural Network Accuracy vs. Binarization Strategy.

B. *Multi-Strategy Performance Comparison*

We further evaluate and interpret different strategies to select the top "predictor pixels" within our binarized neural network. First, it is possible to create models of varying size and accuracy by changing the number of pixels referenced in each layer (i.e. nonzero weight). Second, because weights in a trained model do not form a symmetric distribution (e.g. there are different amounts and magnitudes of positive weights vs. negative weights), there are different classification accuracies when referencing the presence of a certain number N of pixels with positive-valued weights, versus the absence of N pixels with negative-valued weights. Out of a possible 784 pixels in an image, we evaluate the accuracy performance of binary matchup strategies referencing a total number of pixels ranging from N=2 to N=512. In addition to positive and negative predictors, we also evaluate a mixed strategy, in which the pixel reference count is evenly divided between positively-weighted pixels and negatively-weighted pixels. Fig. 4 shows the characteristic curves observed with such binary matchup strategies.

We observe a few relevant details from the characteristic curves that provide insight into optimal binarization strategies. First, we note the diminishing return of referencing larger numbers of pixels; this is because the weights of lower-ranked pixels have smaller absolute values and contribute less to gains in accuracy, while still commanding computational overhead. Next, the absence of negative predictors is significantly more accurate than the presence of positive predictors in small values of N. This may be because high-valued positive predictors are more likely to be similarly placed across digits (see similarly- positioned blue regions in 0, 2, 3, 7 in Fig. 3), as opposed to more unique positioning of strong negative predictors across digits. Lastly, the predictive capability of the mixed-strategy dual matchup is consistently more accurate than either positive or negative predictors alone, with a maximum accuracy of 87.5% using N=256 reference pixels (128 positive, 128 negative).

C. Further Optimization: Weight-Model Unification

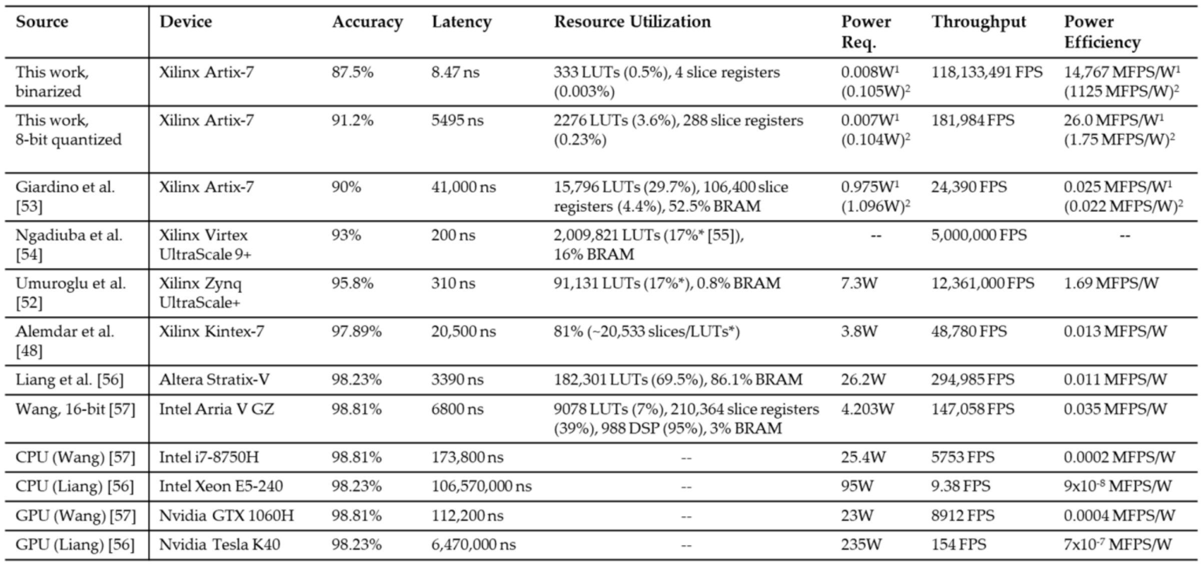
Significant improvements in computational and power efficiency can be achieved by reducing the number of memory accesses required to execute a model [47], [48]. We utilize Boolean logic minimization to consolidate weights and logic within our binarized neural network. Weights stored in memory already take the form of binary flags, which must be retrieved, multiplied with an input datum, and summed to reach a final output value. In an FPGA, such binary weights can simply be instantiated as part of the hardware algorithm logic, translating binary flags into AND operations through which input data pass and are filtered before summing. We utilize this to create an optimized representation in FPGA hardware (Appendix Fig. 3), that utilizes transistorized logic without having to reference separate memory registers for each operation.

7. **Results and Discussion**

A. Latency Minimization and Resource Efficiency

Our 8-bit quantized single-layer neural network dedicates one clock cycle to sum each pixel, reaching a final result in 785 cycles. The condensed architecture allows the use of clock cycle periods as low as 7ns, allowing a result to be reached in only 5495ns (under 6 microseconds), while retaining an accuracy of 91.2% - only 0.3% less than the 91.5% accuracy of the reference network with 32-bit floating-point integers. Our binarized neural network, however, is implemented entirely in combinational logic, and yields final determinations in under 10ns. When clocked, Vivado signal timing analysis confirms a stable result in only 8.465ns, demonstrating a near-instantaneous result. With just a 4.1% drop in accuracy, we are able to achieve over 800x faster speeds than the 8-bit quantized network and 1.5 million times faster than other state-of-the-art systems [24].

TABLE 1: Performance Comparison



Source	Device	Accuracy	Latency	Resource Utilization	Power Req.	Throughput	Power Efficiency
This work, binarized	Xilinx Artix-7	87.5%	8.47 ns	333 LUTs (0.5%), 4 slice registers (0.003%)	$0.008W^1$ $(0.105W)^2$	118,133,491 FPS	14,767 $MFPS/W^1$ $(1125\ MFPS/W)^2$
This work, 8-bit quantized	Xilinx Artix-7	91.2%	5495 ns	2276 LUTs (3.6%), 288 slice registers (0.23%)	$0.007W^1$ $(0.104W)^2$	181,984 FPS	26.0 $MFPS/W^1$ $(1.75\ MFPS/W)^2$
Giardino et al. [53]	Xilinx Artix-7	90%	41,000 ns	15,796 LUTs (29.7%), 106,400 slice registers (4.4%), 52.5% BRAM	$0.975W^1$ $(1.096W)^2$	24,390 FPS	0.025 $MFPS/W^1$ $(0.022\ MFPS/W)^2$
Ngadiuba et al. [54]	Xilinx Virtex UltraScale 9+	93%	200 ns	2,009,821 LUTs (17%* [55]), 16% BRAM	--	5,000,000 FPS	--
Umuroglu et al. [52]	Xilinx Zynq UltraScale+	95.8%	310 ns	91,131 LUTs (17%*), 0.8% BRAM	7.3W	12,361,000 FPS	1.69 MFPS/W
Alemdar et al. [48]	Xilinx Kintex-7	97.89%	20,500 ns	81% (~20,533 slices/LUTs*)	3.8W	48,780 FPS	0.013 MFPS/W
Liang et al. [56]	Altera Stratix-V	98.23%	3390 ns	182,301 LUTs (69.5%), 86.1% BRAM	26.2W	294,985 FPS	0.011 MFPS/W
Wang, 16-bit [57]	Intel Arria V GZ	98.81%	6800 ns	9078 LUTs (7%), 210,364 slice registers (39%), 988 DSP (95%), 3% BRAM	4.203W	147,058 FPS	0.035 MFPS/W
CPU (Wang) [57]	Intel i7-8750H	98.81%	173,800 ns	--	25.4W	5753 FPS	0.0002 MFPS/W
CPU (Liang) [56]	Intel Xeon E5-240	98.23%	106,570,000 ns	--	95W	9.38 FPS	$9x10^{-8}$ MFPS/W
GPU (Wang) [57]	Nvidia GTX 1060H	98.81%	112,200 ns	--	23W	8912 FPS	0.0004 MFPS/W
GPU (Liang) [56]	Nvidia Tesla K40	98.23%	6,470,000 ns	--	235W	154 FPS	$7x10^{-7}$ MFPS/W

*-- indicates values were not available. * indicates values are estimated based on provided metrics. [1] indicates values calculated using the dynamic power consumption of the implemented algorithm; [2] indicates values calculated using the sum of both dynamic power consumption + static chip power consumption.*

Further, both implementations are extremely compact hardware representations of neural networks. The quantized network uses a total of 2276 slice lookup tables (LUTs) (3.6% of the 63400 available on this FPGA), 288 slice registers (0.23%), and 690 slices (4.4%), and has a dynamic power consumption of only 0.007 W. The binarized network uses a total of 333 LUTs (0.53%), 4 slice registers (0.003%), and 101 slices (0.64%), with a dynamic power consumption of only 0.008 W. This extremely minimal resource utilization represents multiple orders of magnitude of space savings and resource conservation compared to conventional networks that may take up the majority of an FPGA fabric [49].

Prior researchers have observed that while binary logic can improve latency, unoptimized representations can exponentially increase model complexity [46], [50], or require additional weights and activations to reach similar accuracies [51]. Our model performance demonstrates that feature-strength-based pruning allows for effective retention of significant contributors to accuracy, and implementation on FPGA allows significant gains in speed and compression of models while minimizing power consumption and on-chip resource utilization.

B. Functional Comparison

To contextualize the performance of our models, we compare performance against benchmark performance and recent research. Conventional MNIST classification systems often take significantly longer times per image; some convolutional neural networks (CNNs) require as long as 7-12 seconds per image [52]. With <9ns latency, our methods are over 109x faster than such algorithms. Even specialized FPGA platforms emulating spiking neural networks only achieve 20ms latencies for MNIST digits [30]; our system is over six orders of magnitude faster.

In the high-performance regime, this work exhibits superb speed and energy efficiency, achieving low-latency and low-power objectives critical to real-time applications.

Table 1 compares our quantized and binarized networks with other state-of-the-art MNIST classification implementations. We explore the tradeoff between accuracy and performance by targeting 90% of the accuracy of comparison algorithms, but with <10% of the latency and using only <10% of the on-chip resources. The impact of this is high throughout and ultra-low power consumption, resulting in 3-4 orders of magnitude greater power efficiency than the next most efficient alternative [53].

Our minimal implementations far exceed the efficiency of prior literature in terms of dynamic power consumption. When considering total power consumption, our 8-bit quantized network offers a slight (~4%) power efficiency improvement over Umuroglu et al. [53], and our binarized network offers more than 500x greater power efficiency, while also achieving nearly 40x faster latency. We also observe that total power efficiency would further increase if FPGA fabric utilization was increased with multiple instances running in parallel for greater throughput, as the static chip power consumption overhead currently accounts for 92-93% of the total power consumption.

Notably, our system yields significantly better overall results than the most recent implementations of MNIST-classifying convolutional neural networks on the same FPGA platform. Our quantized implementation achieves greater accuracy than Giardino et al. [54] with more than a 90% decrease in resource utilization, 99.2% decrease in dynamic power consumption, and 94% reduction in latency (>15x acceleration), while our binarized approach exhibits 590,680x greater dynamic power efficiency and 45,000x greater total power efficiency (frames per second per watt). This further supports the assessment of this approach as a novel contribution enabling low-latency artificial intelligence in hardware and significantly improving hardware performance compared to state-of-the-art research.

C. Cost/Performance Analysis and Applications

The pursuit of higher-accuracy machine-learning models coupled with the conventional intuition that model acceleration is not worth decreases in accuracy, has yielded an underexplored performance envelope for networks with slightly reduced accuracy but far lower latency. Prior approaches have seen accuracy decrease under compression (e.g. 98.81% at 16-bit, to 95.53% at 6-bit, to 43.30% at 5-bit) [58], but our methods allow for accuracy retention even with significant compression from 7840 8-bit weights (91.2% accuracy), to 2560 binary yes/no decisions (87.5% accuracy), demonstrating that the performance envelope can be successfully expanded to ultra-low-latency and ultra-low-power implementations, without sacrificing accuracy: 4% lower accuracy here enables 99.85% lower latency (650x acceleration) and 86% lower hardware resource utilization.

This low resource utilization also enables versatility in applications. For instance, this architecture could be scaled up to facilitate the implementation of multi-layered networks all within a single FPGA, as opposed to having to spread networks out between multiple devices [29]. Additionally, strategic design pipelining could be utilized to increase throughput for individual networks, by allowing multiple operations to be conducted in parallel. Latency increases can be offset by higher clock speeds due to decreased module depth / propagation distance, and the flexibility from small modules facilitates deployment on integrated circuits conducting other operations, like leveraging unused sections of registers occupied by other algorithms to minimize additional on-chip resource utilization.

The low power requirements and resource utilization of our methods also make such strategies well-suited for ubiquitous computation and smaller form factors, and opens new avenues for cost-effective advanced computation on inexpensive chips. The use of individual, compact networks for specialized tasks can improve efficiency and safety in electromechanical systems that must make safety-critical decisions in fractions of a second [59]. By detecting potentially hazardous scenarios in near-real- time, safety equipment can be primed before a full-confidence determination is reached, improving reaction time and resultant safety. Since these designs are so compact, basic neural networks could feasibly and economically be deployed on low- quality silicon, low-speed processors, or on small ASIC die areas. With the energy stored in a single alkaline AA battery [60], our binarized FPGA network could continuously classify images for 20 days. Such efficiency is key for long-duration deployments for biosensors [61], on-body electronics [62], and brain-computer interfaces for prosthetics [63] or for decoding and digitizing of mental handwriting in paralyzed patients [64].

8. **Conclusion**

As the prevalence and role of neural networks in mobile and edge computing continues to increase, there is a growing drive to lower latency and increase throughput while decreasing power and resource utilization. Wielgosz and Karwatowski's review of FPGA latency optimization concludes that "in some application domains, such as…anomaly detection, the response time of the system is more critical to ensure quality of service than the quality of the answer" [65]. Indeed, Sze et al.'s survey of machine learning hardware notes that "the key metrics for embedded machine learning are accuracy, energy consumption, throughput/latency, and cost" [66]. By encoding neural network layers as combinational logic within FPGA hardware, we minimize expensive memory access operations and arithmetic computation, shrinking latency to only the signal propagation delay through FPGA fabric. We implement and compare size-optimization strategies and demonstrate network compression via weight quantization and weight-model unification, achieving up to 96% of the accuracy of baseline MNIST digit classification using only 3% of the memory. We further achieve an 86% decrease in model footprint, 8mW power consumption, and ultra-low <9ns latency, validating the versatility and capability of feature-strength-based pruning approaches for binarized neural networks to flexibly meet performance requirements depending on application resource constraints.

Not only does this work have critical implications in a variety of use cases where low latency and low power usage are crucial, it also demonstrates a significant advancement in terms of strategies for neural network construction and complex input classification leveraging FPGA logic. Low-latency, resource- efficient neural network computation is critical for high-performance edge computing, moving beyond mobile devices and wearables to on-body electronics and ubiquitous computing ecosystems in which these resource constraints are key [31]. Our architecture and method of compressing and implementing binarized networks can also be extended and applied to more complex tasks ranging from process control and safety measures to human-computer interfaces and biomedical devices.

Acknowledgment

The authors acknowledge the Xilinx University Program for facilitating educational access to FPGA development tools. S.P. thanks the course staff in MIT's 6.111 Digital Systems Laboratory for their support and feedback on this work.

Funding

National Science Foundation under Grant No. 1828993.

Institutional Reviewer Board Statement

Not applicable

Informed Consent Statement

Not applicable

Data Availability Statement

Not applicable

Conflict of Interest

The authors declare no conflict of interest.

References

[1] K. He and J. Sun, "Convolutional Neural Networks at Constrained Time Cost," ArXiv14121710 Cs, Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.1710

[2] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based Accelerators of Deep Learning Networks for Learning and Classification: A Review," IEEE Access, vol. 7, pp. 7823–7859, 2019, doi: 10.1109/ACCESS.2018.2890150.

[3] D. G. Bailey, "Image Processing Using FPGAs," J. Imaging, vol. 5, no. 5, p. 53, May 2019, doi: 10.3390/jimaging5050053.

[4] V. Secrieru, S. Zaporojan, and V. Dorogan, "A COST- PERFORMANCE ANALYSIS OF EMBEDDED SYSTEMS FOR LOW AND MEDIUM-VOLUMES APPLICATIONS," Technical

[5] University of Moldova, Chişinău, Moldova, Jan. 2012. [Online].

[6] Available: http://repository.utm.md/handle/5014/741

[7] F. Siddiqui et al., "FPGA-Based Processor Acceleration for Image Processing Applications," J. Imaging, vol. 5, no. 1, p. 16, Jan. 2019, doi: 10.3390/jimaging5010016.

[8] H. Qi, O. Ayorinde, and B. H. Calhoun, "An Ultra-Low-Power FPGA for IoT Applications," presented at the 2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), Burlingame, CA, USA, Oct. 2017. doi: 10.1109/S3S.2017.8308753.

[9] C. Lammie, W. Xiang, and M. R. Azghadi, "Accelerating Deterministic and Stochastic Binarized Neural Networks on FPGAs Using OpenCL," ArXiv190506105 Cs Stat, May 2019, doi: 10.1109/MWSCAS2019.1158.

[10] N. Farha, Ann Louisa Paul J, Naadiya Kousar L S, Devika S, and Ruckmani Divakaran, "Design and Implementation of Logic Gates and Adder Circuits on FPGA Using ANN," Int. J. Res. Appl. Sci. Eng. Technol. IJRASET, vol. 4, no. 5, pp. 623–629, May 2016.

[11] Y.-H. Chen, "Architecture Design for Highly Flexible and Energy- Efficient Deep Neural Network Accelerators," Massachusetts Institute of Technology, 2018.

[12] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," ArXiv180707928 Cs, May 2019, [Online]. Available: http://arxiv.org/abs/1807.07928

[13] P. Garcia, D. Bhowmik, R. Stewart, G. Michaelson, and A. Wallace, “Optimized Memory Allocation and Power Minimization for FPGA- Based Image Processing,” J. Imaging, vol. 5, no. 1, p. 7, Jan. 2019, doi: 10.3390/jimaging5010007.

[14] A. R. Ormondi and J. C. Rajapakse, Eds., FPGA implementations of neural networks. Dordrecht, The Netherlands: Springer, 2006.

[15] C. Kalbande and A. Bavaskar, “Implementation of FPGA-Based General Purpose Artificial Neural Network,” ITSI Trans. Electr. Electron. Eng., vol. 1, no. 3, pp. 99–103, 2013.

[16] Y. Hao, “A General Neural Network Hardware Architecture on FPGA,” ArXiv171105860 Cs, Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.05860

[17] G.-M. Lozito, A. Laudani, F. Riganti Fulginei, and A. Salvini, “FPGA Implementations of Feed Forward Neural Network by using Floating Point Hardware Accelerators,” Adv. Electr. Electron. Eng., vol. 12, no. 1, pp. 30–39, Mar. 2014, doi: 10.15598/aeee.v12i1.831.

[18] R. Gadea, J. Cerda, F. Ballester, and A. Macholi, “Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation,” in Proceedings 13th International Symposium on System Synthesis, Madrid, Spain, 2000, pp. 225–230. doi: 10.1109/ISSS.2000.874054.

[19] G. D. S. Korol, “An FPGA Implementation for Convolutional Neural Network,” Pontifical Catholic University of Rio Grande Do Sul, Porto Alegre, 2019.

[20] C. Farabet, C. Poulet, and Y. LeCun, “An FPGA-based stream processor for embedded real-time vision with Convolutional Networks,” in 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, Kyoto, Sep. 2009, pp. 878–885. doi: 10.1109/ICCVW.2009.5457611.

[21] J. Duarte et al., “Fast inference of deep neural networks in FPGAs for particle physics,” J. Instrum., vol. 13, no. 07, pp. P07027–P07027, Jul. 2018, doi: 10.1088/1748-0221/13/07/P07027.

[22] D. Crankshaw, “The Design and Implementation of Low-Latency Prediction Serving Systems,” University of California at Berkeley, UCB/EECS-2019-171, Dec. 2019. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019- 171.html

[23] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. Emer, and D. Englund, “Freely scalable and reconfigurable optical hardware for deep learning,” ArXiv200613926 Cs, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.13926

[24] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy,” in Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016, vol. 48, p. 10.

[25] A. Brutzkus, O. Elisha, and R. Gilad-Bachrach, “Low Latency Privacy Preserving Inference,” in Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, 2019, vol. 97, p. 10.

[26] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, “GAZELLE: A Low Latency Framework for Secure Neural Network Inference,” p. 17, Jan. 2018.

[27] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A Survey of Model Compression and Acceleration for Deep Neural Networks,” ArXiv171009282 Cs, Jun. 2020, [Online]. Available: http://arxiv.org/abs/1710.09282

[28] W. Niu et al., “GRIM: A General, Real-Time Deep Learning Inference Framework for Mobile Devices based on Fine-Grained Structured Weight Sparsity,” IEEE Trans. Pattern Anal. Mach. Intell., pp. 1–1, 2021, doi: 10.1109/TPAMI.2021.3089687.

[29] J. Johnson, “Rethinking floating point for deep learning,” in 32nd Conference on Neural Information Processing Systems, Montreal, Canada, 2018, p. 8.

[30] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural

[31] Networks,” in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey California USA, Feb. 2015, pp. 161–170. doi: 10.1145/2684746.2689060.

[32] C. Zhang, D. Wu, J. Sun, G. Sun, G. Luo, and J. Cong, “Energy- Efficient CNN Implementation on a Deeply Pipelined FPGA Cluster,” in Proceedings of the 2016 International Symposium on Low Power

Electronics and Design, San Francisco Airport CA USA, Aug. 2016, pp. 326–331. doi: 10.1145/2934583.2934644.

[33] E. Stromatias, D. Neil, F. Galluppi, M. Pfeiffer, S.-C. Liu, and S. Furber, "Scalable energy-efficient, low-latency implementations of trained spiking Deep Belief Networks on SpiNNaker," in 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, Jul. 2015, pp. 1–8. doi: 10.1109/IJCNN.2015.7280625.

[34] P. Corcoran and S. K. Datta, "Mobile-Edge Computing and the Internet of Things for Consumers: Extending cloud computing and services to the edge of the network," IEEE Consum. Electron. Mag., vol. 5, no. 4,

[35] pp. 73–74, Oct. 2016, doi: 10.1109/MCE.2016.2590099.

[36] J. Lyu and S. Sheen, "A Channel-Pruned and Weight-Binarized Convolutional Neural Network for Keyword Spotting," ArXiv190905623 Cs Stat, Sep. 2019, [Online]. Available: http://arxiv.org/abs/1909.05623

[37] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks," in 30th Conference on Neural Information Processing Systems, Barcelona, Spain, 2016, p. 9.

[38] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in Computer Vision – ECCV 2016, vol. 9908, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 525–542. doi: 10.1007/978-3-319-46493-0_32.

[39] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST Database of Handwritten Digits." 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[40] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," in OSDI'16, Savannah, GA, USA, Mar. 2016, pp. 265–283. doi: doi:10.5555/3026877.3026899.

[41] T. Hastie, T. Robert, and M. Wainwright, Statistical Learning with Sparsity, 1st ed. New York: Taylor & Francis Group, 2016. [Online].

[42] Available: https://doi.org/10.1201/b18401

[43] R. N. D'Souza, P.-Y. Huang, and F.-C. Yeh, "Structural Analysis and Optimization of Convolutional Neural Networks with a Small Sample Size," Sci. Rep., vol. 10, no. 1, p. 834, Dec. 2020, doi: 10.1038/s41598-

[44] 020-57866-2.

[45] Y. LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Comput., vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.

[46] G. Wang and J. Gong, "Facial Expression Recognition Based on Improved LeNet-5 CNN," in 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, Jun. 2019, pp. 5655–5660. doi: 10.1109/CCDC.2019.8832535.

[47] S. Xiong, X. Chen, and H. Zhang, "Deep Learning-Based Multifunctional End-to-End Model for Optical Character Classification and Denoising," Journal of Computational Methods in Engineering Applications, pp. 1–13, Nov. 2023, doi: 10.62836/jcmea.v3i1.030103.

[48] I. Kouretas and V. Paliouras, "Hardware Implementation of a Softmax- Like Function for Deep Learning," Technologies, vol. 8, no. 3, p. 46, Aug. 2020, doi: 10.3390/technologies8030046.

[49] L. Guerra, B. Zhuang, I. Reid, and T. Drummond, "Automatic Pruning for Quantized Neural Networks," ArXiv200200523 Cs, Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.00523

[50] Y. Li and F. Ren, "BNN Pruning: Pruning Binary Neural Network Guided by Weight Flipping Frequency," in 2020 21st International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, Mar. 2020, pp. 306–311. doi: 10.1109/ISQED48828.2020.9136977.

[51] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," presented at the International Conference on Learning Representations (ICLR), Mar. 2017. [Online].

[52] Available: http://arxiv.org/abs/1608.08710

[53] K. M. Cherry and L. Qian, "Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks," Nature, vol. 559, no. 7714, pp. 370–376, Jul. 2018, doi: 10.1038/s41586-018-0289-6.

[54] M. Nazemi, G. Pasandi, and M. Pedram, “NullaNet: Training Deep Neural Networks for Reduced-Memory-Access Inference,” ArXiv180708716 Cs Stat, Aug. 2018, [Online]. Available: http://arxiv.org/abs/1807.08716

[55] M. Kim and P. Smaragdis, “Bitwise Neural Networks,” presented at the International Conference on Machine Learning (ICML) Workshop on Resource-Efficient Machine Learning, Lille, France, Jan. 2016. [Online]. Available: http://arxiv.org/abs/1601.06071

[56] H. Alemdar, V. Leroy, A. Prost-Boucle, and F. Petrot, “Ternary neural networks for resource-efficient AI applications,” in 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, May 2017, pp. 2547–2554. doi: 10.1109/IJCNN.2017.7966166.

[57] M. Nazemi, G. Pasandi, and M. Pedram, “Energy-efficient, low-latency realization of neural networks through boolean logic minimization,” in Proceedings of the 24th Asia and South Pacific Design Automation Conference, Tokyo Japan, Jan. 2019, pp. 274–279. doi: 10.1145/3287624.3287722.

[58] X. Lin, C. Zhao, and W. Pan, “Towards Accurate Binary Convolutional Neural Network,” p. 9.

[59] A. Palvanov and Y. I. Cho, “Comparisons of Deep Learning Algorithms for MNIST in Real-Time Environment,” Int. J. FUZZY Log. Intell. Syst., vol. 18, no. 2, pp. 126–134, Jun. 2018, doi: 10.5391/IJFIS.2018.18.2.126.

[60] Y. Umuroglu et al., “FINN: A Framework for Fast, Scalable Binarized Neural Network Inference,” Proc. 2017 ACMSIGDA Int. Symp. Field- Program. Gate Arrays, pp. 65–74, Feb. 2017, doi: 10.1145/3020078.3021744.

[61] D. Giardino, M. Matta, F. Silvestri, S. Spanò, and V. Trobiani, “FPGA Implementation of Hand-written Number Recognition Based on CNN,”

[62] Int. J. Adv. Sci. Eng. Inf. Technol., vol. 9, no. 1, p. 167, Feb. 2019, doi: 10.18517/ijaseit.9.1.6948.

[63] J. Ngadiuba et al., “Compressing deep neural networks on FPGAs to binary and ternary precision with hls4ml,” Mach. Learn. Sci. Technol., vol. 2, no. 1, p. 015001, Dec. 2020, doi: 10.1088/2632-2153/aba042.

[64] R. B. Kent and M. S. Pattichis, “Design, Implementation, and Analysis of High-Speed Single-Stage N-Sorters and N-Filters,” IEEE Access, vol. 9, pp. 2576–2591, 2021, doi: 10.1109/ACCESS.2020.3047594.

[65] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, “FP-BNN: Binarized neural network on FPGA,” Neurocomputing, vol. 275, pp. 1072–1086, Jan. 2018, doi: 10.1016/j.neucom.2017.09.046.

[66] Z. Wang, “A Digits-Recognition Convolutional Neural Network on FPGA,” p. 45.

[67] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and Decision- Making for Autonomous Vehicles,” Annu. Rev. Control Robot. Auton. Syst., vol. 1, no. 1, pp. 187–210, May 2018, doi: 10.1146/annurev- control-060117-105157.

[68] M. Krause, “How Many AA Batteries Would it Take to Power a Mercedes?,” NSF Center for Sustainable Nanotechnology, Apr. 29, 2016. https://sustainable-nano.com/2016/04/29/aa-batteries-mercedes/

[69] S. Payra, I. Wicaksono, J. Cherston, C. Honnet, V. Sumini, and J. A. Paradiso, “Feeling Through Spacesuits: Application of Space-Resilient E-Textiles to Enable Haptic Feedback on Pressurized Extravehicular Suits,” in 2021 IEEE Aerospace Conference (50100), Big Sky, MT, USA, Mar. 2021, pp. 1–12. doi: 10.1109/AERO50100.2021.9438515.

[70] G. Loke et al., “Digital electronics in fibres enable fabric-based machine-learning inference,” Nat. Commun., vol. 12, no. 1, p. 3317, Dec. 2021, doi: 10.1038/s41467-021-23628-5.

[71] S. Payra, G. Loke, and Y. Fink, “Enabling Adaptive Robot-Environment Interaction and Context-Aware Artificial Somatosensory Reflexes through Sensor-Embedded Fibers,” presented at the 2020 IEEE Undergraduate Research Technology Conference, Massachusetts Institute of Technology, Oct. 2020. [Online]. Available: https://www.rle.mit.edu/wp-content/uploads/2020/10/PA20-0081_URTC_Updated.pdf

[72] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K.

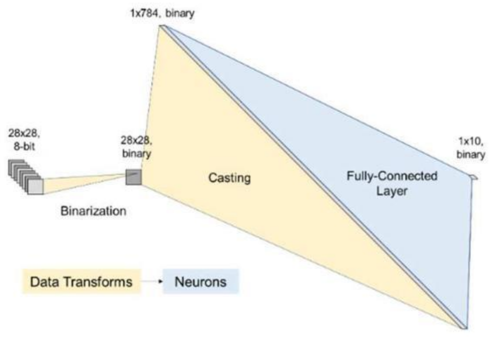
[73] V. Shenoy, “High-performance brain-to-text communication via handwriting,” Nature, vol. 593, no. 7858, pp. 249–254, May 2021, doi: 10.1038/s41586-021-03506-2.

[74] M. Wielgosz and M. Karwatowski, “Mapping Neural Networks to FPGA-Based IoT Devices for Ultra-Low Latency Processing,” Sensors, vol. 19, no. 13, p. 2981, Jul. 2019, doi: 10.3390/s19132981.

[75] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, “Hardware for Machine Learning: Challenges and Opportunities,” 2017 IEEE Cust. Integr. Circuits Conf. CICC, pp. 1–8, Apr. 2017, doi: 10.1109/CICC.2017.7993626.

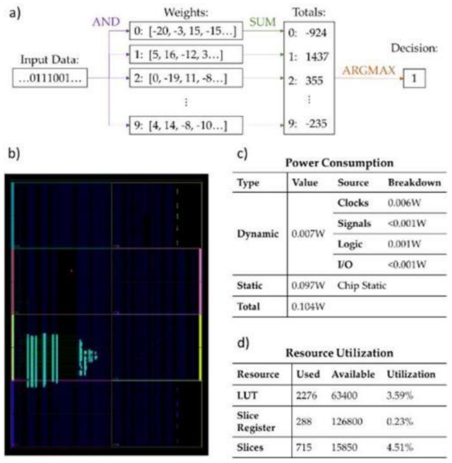
Appendices

Appendix Figure 1:



Appendix Fig. 1. Single-layer neural network model architecture. Preliminary

data transforms (yellow) include binarization of MNIST digits and casting into a linear pixel array, then a single fully-connected layer (blue) determines image classification out of ten potential digits.



Power Consumption

Type	Value	Source	Breakdown
Dynamic	0.007W	Clocks	0.006W
		Signals	<0.001W
		Logic	0.001W
		I/O	<0.001W
Static	0.097W	Chip Static	
Total	0.104W		

Resource Utilization

Resource	Used	Available	Utilization
LUT	2276	63400	3.59%
Slice Register	288	126800	0.23%
Slices	715	15850	4.51%

Appendix Figure 2:

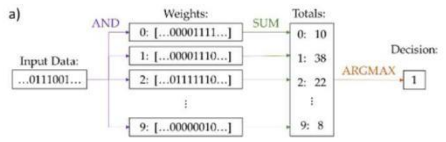
Appendix Fig. 2. Implemented 8-bit Quantized Neural Network.

a) Network structure, b) FPGA die floorplan coverage,

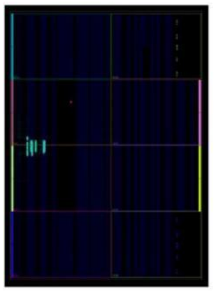
c) simulated power consumption, d) module resource utilization

Appendix Figure 3:

a)



b)



c)

Power Consumption

Type	Value	Source	Breakdown
Dynamic	0.008W	Clocks	N/A
		Signals	0.003W
		Logic	0.003W
		I/O	0.002W
Static	0.097W	Chip Static	
Total	0.105W		

d)

Resource Utilization

Resource	Used	Available	Utilization
LUT	333	63400	0.53%
Slice Register	4	126800	0.003%
Slices	101	15850	0.64%

Appendix Fig. 3. Implemented Binarized Neural Network

a) Network structure, b) FPGA die floorplan coverage,

c) simulated power consumption, d) module resource utilization