Proceedings of the 2nd International Conference on Mechatronics and Smart Systems
DOI: 10.54254/2755-2721/62/20240406

A hardware-implemented neural network for the analysis of handwritten numerals

Zhuotong Li

Henry Samueli School of Engineering, University of California, Los Angeles, 10505, United States

zli2866@ucla.edu

Abstract. This study presents a specialized hardware-accelerated neural network tailored for the recognition of handwritten digits in 28x28 pixel grayscale images. Employing the perceptron model, our single-layer neural network is composed of 10 neurons, each handling inputs from all pixels to generate an output. The digit recognition is determined by the neuron with the highest output value. Implemented in synthesizable Verilog, the design complies with a constraint of 350 multipliers. To achieve this, this paper employs a combination of parallel processing and pipelining, breaking down the 785 multiplications needed for each digit into 8 stages, simultaneously processing 98 data points per clock cycle. In testbench evaluations, the final design exhibits impressive performance, successfully recognizing the majority of the provided images, and attaining a remarkable 99% accuracy rate, all with a minimal delay of just 115 clocks. This accomplishment is achieved using only 99 multipliers and 107 adders, showcasing the efficiency and effectiveness of our hardware-accelerated neural network for handwritten digit recognition.

Keywords: Neural Network, Hardware Acceleration, Handwritten Digits, Machine Learning, Perceptron.

1. Introduction

In the contemporary computational landscape, hardware acceleration has emerged as a pivotal technique, optimizing specific tasks like graphics rendering, video playback, and augmented reality through specialized hardware components rather than conventional software-based methods on CPUs [1]. This study introduces a novel application of this paradigm: a circuit designed with adders and multipliers to facilitate a perceptron neural network's operation, specifically tailored for the recognition of handwritten digits [2]. The implications of this advancement are profound, offering potential real-world applications in sectors such as education and banking, where handwritten digit recognition can significantly enhance efficiency and accuracy, bridging the divide between traditional methods and the burgeoning digital frontier [3].

This research centers on a pioneering method to identify digits from the widely acknowledged Modified National Institute of Standards and Technology database (MNIST) [4]. This database comprises a collection of 28x28 pixel images representing handwritten digits. To interpret these images, we utilize the perceptron neural network, a machine learning algorithm. The algorithm processes each image by assessing the grayscale value of every pixel. A distinct formula is subsequently applied to these

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/62/20240406

pixels, iterated ten times with varying weights. The resultant values are then normalized to a range between 0 and 1. These values serve as scores, indicating the likelihood of each image corresponding to a digit from 0 to 9. A Max Selector mechanism is employed to identify the digit with the highest score, which is then recognized as the handwritten digit depicted in the image.

It's worth noting that for this study, the algorithm is pre-trained. This implies that the weights utilized are predetermined and operational. Our primary objective is to engineer hardware capable of executing the aforementioned calculations, bypassing the need for conventional CPUs. Such an approach is anticipated to significantly elevate both efficiency and accuracy.

2. Algorithm

In the present research, the objective is to design an image classification system capable of interpreting hand-written digits. This system leverages a pre-trained weight matrix to process an input image and subsequently deduces the numeric representation of the handwritten digit. A schematic representation of this classifier can be observed in Figure 1.

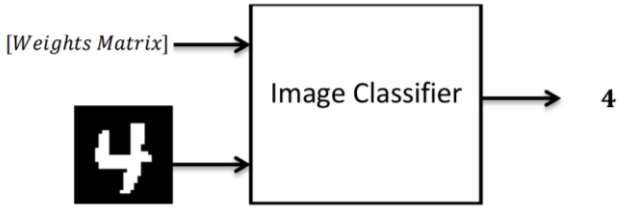

Figure 1. Schematic diagram illustrating the classifier's architecture (Photo/Picture credit: Original).

The system ingests a pre-trained weight matrix and an image, subsequently producing a numerical output corresponding to the digit depicted in the image.

To realize this objective, we employed the Perceptron Neural Network, a machine learning algorithm inspired by the neural architectures observed in biological entities. Within this algorithm, individual neurons function as computational units, each characterized by a distinct mathematical formula. These units process multiple inputs to yield a singular output value. Specifically, for our application, we implemented 10 distinct neurons, each estimating the likelihood of the input image representing a digit ranging from 0 to 9. The images from the MNIST database are all black-and-white and pixel-based, with each pixel having a grayscale value ranging from 0 to 255 [5]. Given the focus on 28x28 pixel images, each image encompasses 784 grayscale values. Plus the bias, each of the 10 neurons will need to process 785 inputs [6].

Each neuron employs a consistent computational formula, albeit with varying weights. Figure 2 delineates the architecture of an individual neuron [7].

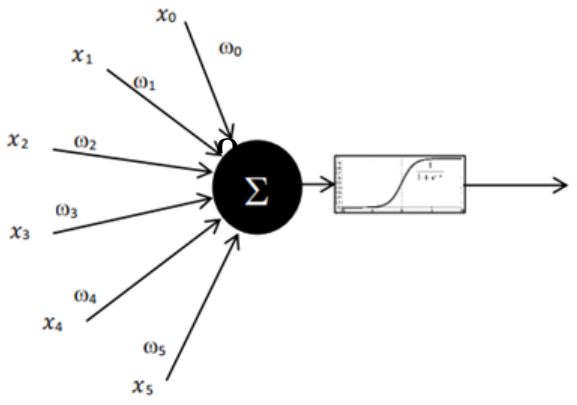

Figure 2. Neuronal architecture [7].

DOI: 10.54254/2755-2721/62/20240406

Every neuron processes 785 inputs, each input being multiplied by a corresponding weight. The aggregated sum of these products yields a final score, which is subsequently normalized using the sigmoid function [8].

As inferred from Figure 2, neurons function akin to computational processors, executing weighted calculations. The underlying mathematical formula governing these calculations is presented in Formula 1 as shown below.

$$Out_n = \sum_{m=0}^{784} \omega_{n,m} \cdot Pixel_m = \omega_{n,0} \cdot Pixel_0 + \omega_{n,1} \cdot Pixel_1 + \cdots + \omega_{n,784} \cdot Pixel_{784} \quad (1)$$

In the aforementioned formula, Out_n denotes the output of the nth neuron. The index n spans from 0 to 9, corresponding to each individual digit. $Pixel_m$ signifies the grayscale intensity of the mth pixel among the total 784 pixels and the bias. The term $\omega_{n,m}$ designates the weight associated with the mth pixel for the nth neuron.

Conclusively, the resultant output Out_n undergoes normalization via a sigmoid function, as delineated in Formula 2.

$$Out_{n-normalized} = \frac{1}{1 + e^{-Out_n}} \quad (2)$$

This formula serves to mitigate the influence of outlier output values, ensuring that the resultant output remains within a defined and interpretable range [9].

The final number we get from each neuron essentially quantifies the probability that the input image represents a specific digit. The digit associated with the neuron yielding the highest output is deemed to be the digit depicted in the input. A mechanism termed the 'Max Selector', is integrated at the final stage to identify this peak output and relay the corresponding digit. For a clearer perspective, let's envisage a scenario where the neuron outputs, after undergoing processes like weight multiplication and sigmoid normalization, are as follows:

Final Neuron Outputs = [0.1, 0.02, 0.004, 0.098, 0.81, 0.074, 0.115, 0.007, 0.0088, 0.03]

In this array, each value corresponds to the likelihood of the input image representing the respective digit. The value 0.81, corresponding to the fourth position, is the most prominent. This indicates that the digit depicted in the input image is most likely a '4'. The Max Selector, upon analyzing these output values, converts this confidence into a binary format, resulting in 3b'100, which equates to the decimal notation 1d'4. The general structure of our algorithm is shown in Figure 3 [10].

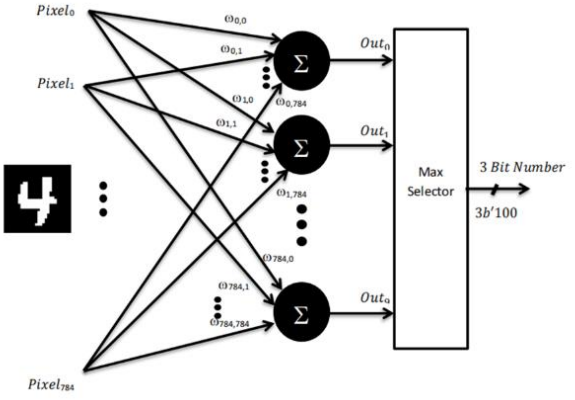

Figure 3. Schematic representation of the Perceptron Neural Network algorithm [10].

The architecture comprises 10 distinct neurons, each processing 784 grayscale values and a bias value as inputs to produce a singular output value. Subsequently, a 'Max Selector' mechanism selects the neuron with the highest output, determining the numeric representation of the handwritten digit.

DOI: 10.54254/2755-2721/62/20240406

3. Implementation

To transition the described algorithm from a conceptual framework to a tangible hardware implementation, it is imperative to decompose the intricate algorithmic processes into fundamental computational tasks. As elucidated in the preceding chapter, the crux of the algorithm is encapsulated in Formula 1, which predominantly involves multiplication and addition operations. To this end, we incorporated specialized pipelined fixed-point adders and multipliers. A comprehensive depiction of the architecture of these computational elements is presented in Figure 4.

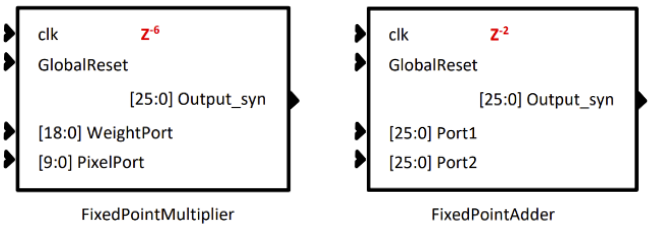

Figure 4. Architectural overview of the pipelined fixed-point adder and multiplier (Photo/Picture credit: Original).

It is noteworthy that these computational blocks are equipped with a synchronous active-high reset (termed "GlobalReset") and operate on the positive edge of the active clock (denoted as "clk"). The annotations "Z-6" and "Z-2" in red signify the latency of the multiplier and adder, amounting to 6 and 2 clock cycles respectively, indicative of their pipelined nature.

Given that Formula 1 embodies a summation function encompassing 785 multiplication and 784 addition operations, and considering the presence of 10 neurons, the computational demand escalates to 7,850 multiplications and 7,849 additions for each input image. A direct implementation would necessitate the deployment of 7,850 multipliers and 7,849 adders—a manifestly impractical proposition. Consequently, it becomes paramount to devise a strategy to judiciously minimize the requisite number of adders and multipliers.

In the scope of this project, we adopted parallel processing and pipelining as strategic methodologies to address this problem.

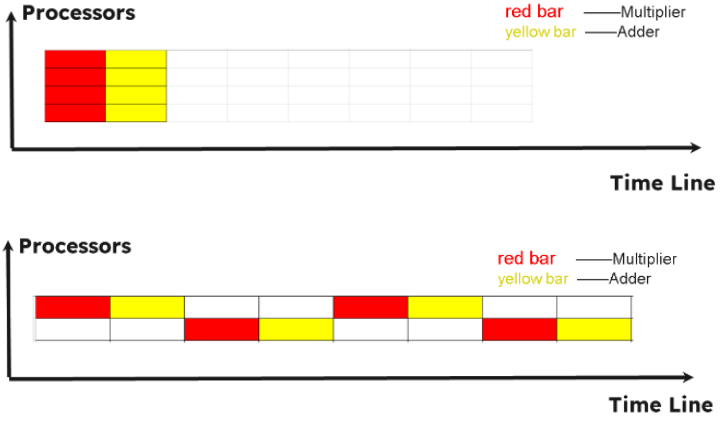

Figure 5. Two primary computational strategies (Photo/Picture credit: Original).

A comparative illustration of the overarching structure of the implementation, contrasting the approaches without and with the integration of parallel processing and pipelining is shown in Figure 5. The initial approach, as previously discussed, entails executing all 7,850 multiplications and 7,849 additions simultaneously, necessitating an equivalent number of multipliers and adders. While this method boasts unparalleled speed, completing calculations within a single clock cycle, the sheer volume

DOI: 10.54254/2755-2721/62/20240406

of required processors renders it infeasible. Consequently, our decision gravitated towards the alternative approach, which, albeit slower, significantly curtails the processor count. As depicted in Figure 5, rather than condensing all computations within a singular clock cycle, this method distributes tasks across multiple cycles, with each processor handling a segment of the entire algorithm within a given cycle.

Specifically, for this project, our design choice was to execute 98 multiplications within a single clock cycle. Given the 785 multiplications requisite for each neuron, after 8 clock cycles, 1 solitary multiplication—pertaining to the bias and its weight—remains, which is subsequently processed. This configuration demands a total of 99 multipliers for algorithmic execution.

For the addition operations, we adopted a unique method, which we refer to as the "98-49-24-12-6-3-2-1" approach. Upon completion of the multiplications, we are left with 98 resultant values that need summation. These 98 values are divided into two groups, and each corresponding value from the two groups is added together. This step requires 49 adders, producing 49 summed values. From these, one value is set aside, leaving 48 values. These 48 values are then again split into two groups and added in a pairwise manner, necessitating 24 adders. This process of halving and summing continues, reducing the number of values to 12, then 6, etc. At each stage with an odd number of values, one value is reserved and set aside. This iterative process continues until we are left with a single value.

Subsequently, the two values that were previously set aside are added back to this final value, a step that requires 2 adders. Given that each neuron processes 8 sets of these 98 values, an additional 7 adders are needed to sum up the final results for each neuron. The last step involves adding the bias to the final value, which requires 1 more adder. In total, the algorithm necessitates 107 adders: 49 for the initial step, 24 for the next, followed by 12, 6, 3, 2, 1, 2 for the reserved values, 7 for the neuron sets, and 1 for the bias.

4. Results and discussion

4.1. Results

As delineated in Table 1, a stark contrast is evident between the initial computational requirements and the optimized implementation. Contrary to the initial demand for 7,849 adders and 7,850 multipliers, our refined algorithm necessitates a mere 107 adders and 99 multipliers. This substantial reduction translates to significant savings in power consumption and spatial requirements for the final PCB. The optimization achieved amounts to a 98.64% reduction in the number of adders and a 98.74% reduction in the number of multipliers. Such metrics underscore the efficacy of our algorithm in markedly diminishing the computational resources required for its execution.

Table 1. Number of processors used in the algorithm.

Logical Blocks	Before	Now	Optimization
Adders	7849	107	98.64%
Multipliers	7850	99	98.74%
Power & Area	Tremendous	Acceptable	

Upon executing the algorithm on a simplified testbench comprising a set of 10 images, we attained a commendable success rate of 100% with a mere delay of 115 clock cycles, as illustrated in Figure 6. Subsequently, we subjected our model to a more rigorous evaluation using an expansive testbench derived from the MNIST dataset, encompassing 1,000 images. The model demonstrated an exemplary average success rate of 99.7%, signifying a substantial advancement in the domain.

4.2. Discussion

This research endeavor culminated in the development of a hardware-accelerated machine learning algorithm proficient in recognizing handwritten digits with an impressive accuracy exceeding 99%. The implications of such an algorithm extend far beyond mere academic interest, holding substantial societal value. When judiciously deployed, the algorithm has the potential to revolutionize several sectors: 1.

DOI: 10.54254/2755-2721/62/20240406

Banking: The algorithm can facilitate the conversion of handwritten checks into electronic records, heralding a significant leap in the operational efficiency of the banking sector. 2. Education: It can be instrumental in the digitization and intelligent management of educational processes, enabling automated grading of handwritten examinations. 3. Data Management: The algorithm can expedite the collection and organization of information from paper sources. 4. Mobile Computing: Enhancing the efficiency of handwriting recognition on mobile platforms, the algorithm can mitigate misrecognitions and typographical errors, offering a seamless user experience.

Looking ahead, our immediate objectives encompass rigorous testing of our model on an expansive dataset to refine its accuracy further. Additionally, this paper intends to actualize our model by synthesizing it onto a physical Printed Circuit Board (PCB), providing tangible insights into its real-world performance and facilitating iterative refinements. A pivotal enhancement under consideration is the integration of pipelining techniques. In the existing model, within each 98-value cluster, adders remain dormant until multipliers complete their computations, and subsequent clusters are queued until the preceding one is fully processed. By contrast, a pipelined approach would allow for concurrent processing: as the first cluster undergoes addition operations, multipliers can simultaneously commence computations for the subsequent cluster. This cascading mechanism promises to bolster computational efficiency and minimize latency.

5. Conclusion

In summary, this research represents a pioneering effort in the utilization of perceptron neural networks for the hardware-accelerated recognition of handwritten digits. Focused on the development of a Hardware Neural Network tailored for the classification of 28x28 pixel grayscale images, our single-layer architecture comprising 10 neurons was meticulously crafted to adhere to synthesizable Verilog code standards, with a primary emphasis on optimizing the {Energy x Area} product. Remarkably, this feat was accomplished with a minimalistic hardware footprint, utilizing only 99 multipliers and 107 adders.

Through the strategic integration of parallel processing and pipelining techniques, we efficiently segmented and processed calculations, resulting in a mere 115 clocks of delay. This innovative approach not only met the practical requirements but also showcased the immense potential of hardware-based neural networks in the domain of digit recognition. This research sets the stage for further advancements in the field of hardware-accelerated neural networks and opens up new possibilities for efficient and effective image recognition systems.

References

[1] Viet Huynh T. FPGA-based acceleration for convolutional neural networks on PYNQ-Z2. International Journal Of Computing and Digital System, 2021.

[2] Si J, Harris S L, Yfantis E. Neural networks on an FPGA and hardware-friendly activation functions. Journal of Computer and Communications, 2020, 8(12): 251.

[3] Huynh T V. Evaluation of artificial neural network architectures for pattern recognition on fpga. International Journal of Computing and Digital Systems, 2017, 6(03): 133-138.

[4] Du Y, Du L, Gu X, et al. An analog neural network computing engine using CMOS-compatible charge-trap-transistor (CTT). IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 38(10): 1811-1819.

[5] Ali S, Shaukat Z, Azeem M, et al. An efficient and improved scheme for handwritten digit recognition based on convolutional neural network. SN Applied Sciences, 2019, 1: 1-9.

[6] Kulkarni S R, Alexiades J M, Rajendran B. Learning and real-time classification of hand-written digits with spiking neural networks. 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2017: 128-131.

[7] Liu Y, Sun Y, Xue B, et al. A survey on evolutionary neural architecture search. IEEE transactions on neural networks and learning systems, 2021.

DOI: 10.54254/2755-2721/62/20240406

[8] Faghihi F, Alashwal H, Moustafa A A. A synaptic pruning-based spiking neural network for hand-written digits classification. Frontiers in Artificial Intelligence, 2022, 5: 680165.
[9] Faghihi F, Alashwal H, Moustafa A A. Explainable Feature Extraction Using a Neural Network with non-Synaptic Memory for Hand-Written Digit Classification. bioRxiv, 2021: 2021.04. 06.438597.
[10] Abdar M, Yen N Y, Hung J C S. Improving the diagnosis of liver disease using multilayer perceptron neural network and boosted decision trees. Journal of Medical and Biological Engineering, 2018, 38: 953-965.