Proceedings of the 2nd International Conference on Mechatronics and Smart Systems
DOI: 10.54254/2755-2721/62/20240396

Accelerating handwritten number classifier with pipelined multiply-accumulate units in neural networks

Yifei Zou

Electrical Engineering and Computer Science, University of Michigan Ann Arbor, Ann Arbor, 48105, United States of America

yifeizzz@umich.edu

Abstract. This research project explores the development and assessment of a Handwritten Number Classifier based on Neural Networks, employing the widely used MNIST dataset. The primary objective of this study is to enhance computational efficiency, and to achieve this, it concentrates on integrating and contrasting non-pipelined and pipelined structures within the Multiply-Accumulate (MAC) unit. The initial phase involved conducting MATLAB simulations, which yielded promising results regarding the accuracy of weight calculations. Subsequently, Hardware Description Language (HDL) testing was carried out to further validate the classifier's performance. In the HDL testing phase, the classifier incorporating the pipelined MAC unit demonstrated a substantial 42.9% enhancement in processing speed when compared to its non-pipelined counterpart. These results highlight the potential advantages of employing pipeline processing in neural network architectures, emphasizing its effectiveness in achieving faster and more efficient image classification, particularly when dealing with extensive datasets. In conclusion, this research project not only presents valuable insights into improving the efficiency of neural network-based image classifiers but also lays the groundwork for potential future endeavors. These future directions may include adapting the classifier to handle more complex datasets and addressing emerging challenges.

Keywords: Neural Network, Number Classifier, Pipelining, Multiplier-Accumulator, Hardware Description Language.

1. Introduction

The evolution of Hardware Neural Networks represents a cutting-edge development in the intersection of computer science and engineering. With the rapid advancements in deep learning and artificial intelligence, conventional software-based approaches began to grapple with computational efficiency and power consumption challenges, especially when managing large-scale data and models. Researchers have pivoted towards hardware-specific solutions tailored for neural networks, such as FPGAs, ASICs, and Tensor Processing Units (TPUs), which focus on their speed-ups and energy efficiency in comparison to contemporary CPUs and GPUs for specific neural network tasks [1]. Hardware-accelerated neural networks can notably enhance processing speeds, reduce latency, and achieve lower energy consumption. In essence, the advent of hardware neural networks unveils new avenues for efficient, high-performance AI implementations. This study present a Neural Network-based Handwritten Number Classifier based on the MNIST Dataset, particularly emphasizing the

© 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: 10.54254/2755-2721/62/20240396

implementation of a pipelined structure within the Multiply-Accumulate (MAC) unit. The foundation of this classifier rests upon the widely acknowledged neuron model, known as the 'perceptron' [2, 3].

Two distinct designs were introduced: one integrates non-pipelined MAC units, while the other incorporates MAC units with pipelined structure, which was popularized in 1990s [4]. The MAC units are composed of fix-pointed adder and fix-pointed multipliers. To evaluate the system's performance, a weighted dataset was derived by training on the MNIST Dataset. By methodically comparing the processing speed of these classifiers on a consistent dataset derived from MINST Dataset, a fact was found that the classifier with the pipelined MAC unit demonstrates a notable improvement in efficiency, showing a 42.9% increase. Such a comparison not only underscores the potential advantages of pipeline processing in neural network architectures but also provides insights for future designs aiming at optimization and enhanced performance in similar computational tasks. This work paves the way for more streamlined and efficient neural network models in the realm of image classification.

2. Neural Network Handwritten Number Classifier Based on the MNIST Dataset

2.1. Overview of the MNIST Dataset

The MNIST dataset has established itself as a benchmark in the fields of computer vision and machine learning, specifically for the task of handwritten Number recognition. Created by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges in the 1990s, the intent behind this dataset was to advance the research in automatic handwritten number identification [2]. This dataset comprises grayscale images of handwritten numbers ranging from 0 to 9, with each image being 28x28 pixels in dimension. Specifically, MNIST consists of a training set and a test set: the training set is made up of 60,000 samples, while the test set encompasses 10,000 samples. These samples are originally derived from two distinct datasets from the National Institute of Standards and Technology (NIST) in the United States. Notably, the data in the training set were penned by U.S. high school students and employees from the Census Bureau, whereas the test set was exclusively written by high school students [5]. Based on the MNIST dataset, neural network models can be trained to achieve automatic recognition of handwritten numbers [5]. Due to the dataset's simple and clear structure, MNIST has become an ideal choice for deep learning beginners. Through studying this dataset, researchers and students can not only gain a foundational understanding of neural networks but also master key skills such as processing image data, tuning network parameters, and evaluating model performance. In conclusion, the MNIST dataset plays an indispensable role in the fields of machine learning and computer vision. It offers researchers a straightforward yet effective platform for validating new methodologies and techniques, while also serving as an accessible introduction for newcomers.

2.2. Working Principle of the Hand-Written Number Classifier

The foundational principle of the hand-written Number classifier rests on the integration of weighted data with the MNIST image dataset. Specifically, the architectural framework sources its input pixels from a 28x28 MNIST image. This image encompasses a total of 784 pixels, where each pixel represents a grayscale intensity ranging between 0 to 255. To generate an output, each grayscale intensity is multiplied by a set of pre-trained weights which illustrates the most famous and successful model of the neuron called the 'perceptron', as shown in Equation 1 [2]. In Equation 1, $\boldsymbol{w}$ represents weights and $\boldsymbol{x}$ represents grayscale intensity.

$$Sum = w_1 \times x_1 + w_2 \times x_2 + \ldots + w_n \times x_n \tag{1}$$

These weights are derived from a training set containing 60,000 samples. Neurons, or the "nodes in the neural network", play a pivotal role in this process. Each neuron accepts inputs from all the pixels and delivers a singular output. There are ten neurons in total within this layer.

The output from each neuron can be conceptualized, to an extent, as the probability that the input image signifies a specific number. Consequently, the neuron with the highest output value corresponds to the Number the input image most likely depicts. For a more lucid explanation: as an MNIST image

DOI: 10.54254/2755-2721/62/20240396

is fed into the architecture, each of its pixels is multiplied by its corresponding weight. Subsequently, all these weighted pixel values are channeled to every neuron within the neural network. Ultimately, every neuron produces an output value for the input image, symbolizing the likelihood of the image being a particular Number. A sigmoid function could be used here to transfer the outputs from Neurons to probability [6]. However, it hard to apply sigmoid function in Verilog and the focus is the speed of Neurons, so the sigmoid function was not used in the final design. The final prediction of the network is represented by the Number with the highest probability. The overall architecture is shown in Figure 1.

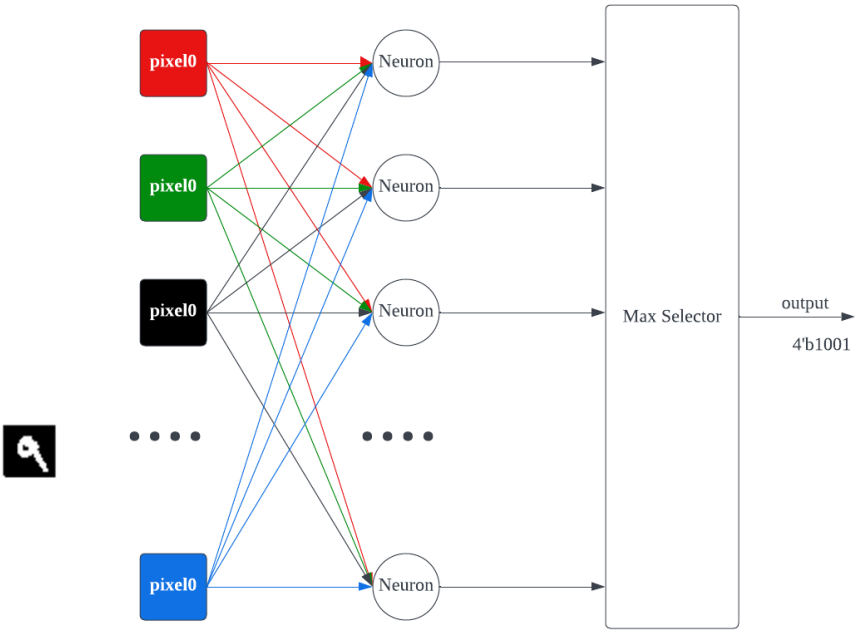


Figure 1. The architecture of the Hand Written Number Classifier [6].

2.3. Design Overview of the Image Classifier with MAC (Multiply-Accumulate) Units

The proposed design encompasses ten neurons and a maximum value selector. In the neuron design under consideration, each neuron is equipped with a MAC Units and a control unit. Inputs to each neuron consist of 783 weights and an equivalent number of pixels. A detailed exposition of each component follows:

2.3.1. MAC Units. In the construction of a high-performance classifier, the MAC units stand out as pivotal components. In pursuit of robust functionality and a wider precision range, Fixed-Point adders and Fixed-Point multipliers were the design choice for the MAC units. Such a choice, while offering precision, generally entails a demand for more hardware resources, potentially enlarging the design footprint and power consumption and impinging on the operating speed [7]. In this design, the Fixed-Point multiplier with 6 clock delay and the Fixed-Point adder with 2 clock delay were used.

To navigate these challenges and strike a balance, the design strategy involves the reduction in the number of multipliers and adders. Specifically, each MAC unit integrates only 28 multipliers. To accumulate the outputs of these 28 multipliers, 27 adders are required, as depicted in the associated Figure 2.

DOI: 10.54254/2755-2721/62/20240396

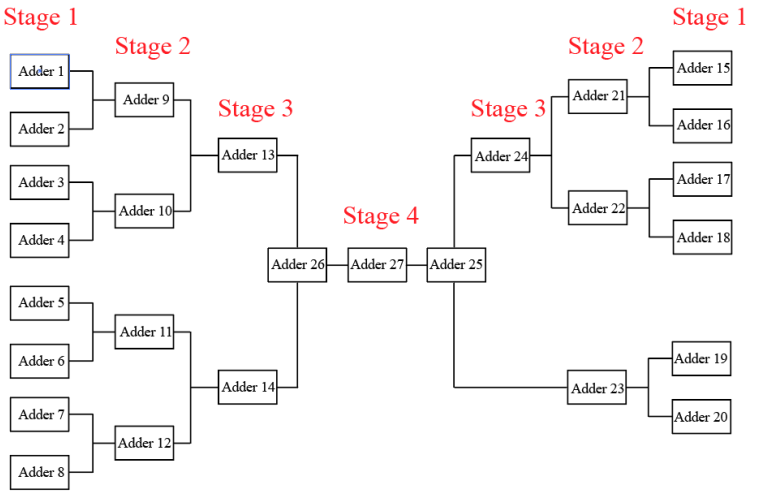


Figure 2. The Scheme of 4-stage adders (Photo/Picture credit: Original).

Collectively, the entire system incorporates 280 multipliers and 270 adders. While this approach reduces the consumption of hardware resources to some extent, in order to further enhance the system's operational speed, the introduction of a pipelined architecture for the multiply-accumulate (MAC) units has been proposed. The subsequent sections detail the implementation methodologies for both non-pipelined and pipelined architectures, with the primary distinction manifesting in the control unit.

2.3.2. Non-Pipelined MAC Units. In a non-pipelined architecture, multiplication and addition operation are sequential. Specifically, when an input data set is ushered into the MAC unit, it is first processed by the multipliers, followed by the forwarding of the results to the adders for accumulation. This progression is uninterrupted, processing a single input dataset at once [8]. As shown in Figure 3, the average time required for the MAC to complete a set of data is Multiplication Time +Addition Time.

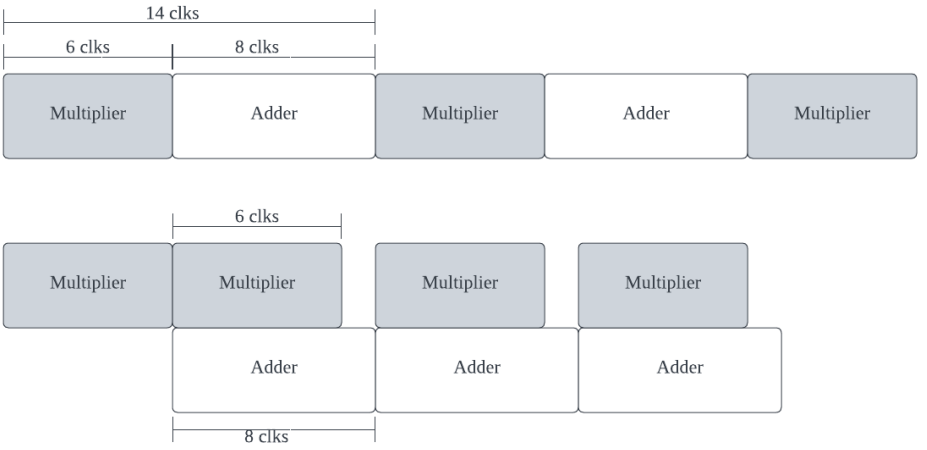


Figure 3. Timing principle of non-pipelined MAC and pipelined MAC (Photo/Picture credit: Original).

2.3.3. Pipelined MAC Units. Conversely, the pipelined architecture facilitates parallel processing of two input data sets across different stages. As the first input data set begins processing in the multiplier, a second data set can immediately queue up at the entrance of the multiplier. Upon the completion of multiplication for the first data set and its progression to the addition stage, the second set commences multiplication, allowing a third set to line up. Similarly, the adder can commence new computations immediately upon the completion of its preceding addition operations and the associated multiplication

DOI: 10.54254/2755-2721/62/20240396

tasks. This parallel processing significantly bolsters efficiency by handling two data sets concurrently. As shown in Figure 3, the average time required for the MAC to complete a set of data is MAX (Multiplication Time, Addition Time) [9].

In conclusion, through the adoption of a pipelined architecture, the design aims to elevate operational speed, while balancing hardware resource consumption, culminating in an efficient and cost-effective classifier design.

3. Simulation Verification and Analysis

3.1. Acquisition and Processing of Input Weights

Initially, the MNIST dataset was downloaded from the source: https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz. Subsequently, the dataset was converted into mnist_train.csv and mnist_test.csv formats using Python. Utilizing MATLAB tools, the weights of the model were established. The csvread function was employed to import the training and testing data from the CSV files [10]. The matrix DVec was initialized as a zero matrix, designated for storing the target outputs. Likewise, IVec was initialized as another zero matrix to store input image data, inclusive of its bias term. Throughout the training phase, a 'for' loop was adopted to iterate through all the training data. For every individual training sample, the function num2TemperatureVec was employed to convert its label into a 10-dimensional vector, which was then stored in DVec. This function essentially serves as a transformation utility, aimed at converting numerical values ranging from 0 to 9 into a 10-element one-hot encoding. Simultaneously, the input image data, along with its bias term, were saved in IVec. To compute the weight matrix WVec, the pinv function was applied to IVec for calculating its pseudo-inverse. The resultant IVecInv, when multiplied with DVec, provided the final weight matrix WVec.

3.2. Validation of Input Weights

To assess the reliability of the acquired weights, a simulation of an image classifier was constructed using MATLAB code, employing the derived weights and test data as inputs for the simulated classifier. Within MATLAB, a 'for' loop was used once more to iterate over all the test data samples. For each test sample, a bias term was initially incorporated, followed by the application of the Hardware function to compute the predicted outcome. The primary intent of this function is to ascertain the predicted number based on the provided image. This function accepts the weight matrix WVec and the input image Image as parameters and yields the anticipated number. By comparing the predicted results with the actual labels, the counts of correctly predicted samples k and incorrectly predicted samples j were updated. Ultimately, the success rate was computed and presented, derived from the ratio of correctly predicted samples k to the total predictions.

3.3. HDL Testing of the Image Classifier & Testing Result

During the evaluation phase of our Image Classifier at the Hardware Description Language (HDL) level, a set of 10 images were utilized from the MNIST dataset as depicted in Figure 4. Each of these images was systematically broken down into a structure of 784 pixels for the sake of this testing. These pixelated images, when combined with the weights derived from the MATLAB processing, served as primary dataset for the testbench simulation to test the classifier's overall performance and accuracy.

DOI: 10.54254/2755-2721/62/20240396



Figure 4. Images used for testing (Photo/Picture credit: Original).

3.3.1. Validation of Weight Integrity. The effectiveness of the weights was visualized and the classifier's accuracy with these weighted data reached an impressive 86% after running the MATLAB code. Given this high accuracy, the weighted data was selected for further classifier testing.

3.3.2. HDL Testing. Classifier Performance without Pipelined MAC Units: The image classifier passed all the test cases and exhibited a success rate of 100% as shown in Table 1. The complete execution of the classification tasks took 392 cycles.

3.3.3. HDL Testing. Classifier Performance with Pipelined MAC Units: The image classifier passed all the test cases. The corresponding Table 1 highlights that with the integration of the pipelined MAC Units, the classifier maintained a success rate of 100%. Notably, the presence of the pipelined MAC Units enhanced the efficiency, and the classifier required only 224 cycles to finalize all tasks.

Table 4. HDL Test results.

Actual Number	Estimated output Number	Clks Used pipelined MAC	Clks used non-pipelined MAC
4'b0001	4'b0001	224	393
4'b1001	4'b1001	224	393
4'b0111	4'b0111	224	393
4'b0010	4'b0010	224	393
4'b0000	4'b0000	224	393
4'b0100	4'b0100	224	393
4'b0001	4'b0001	224	393
4'b0100	4'b0100	224	393
4'b0101	4'b0101	224	393
4'b1001	4'b1001	224	393
Success Rate: 100%	Success Rate: 100%		

3.4. Comparative Analysis

Upon detailed comparison, it becomes evident that the image classifier equipped with a pipelined MAC Units showcases a significant enhancement in processing speed. When compared with the classifier that does not utilize a pipelined approach, there's a reduction in the required cycle count by approximately 42.9%. This uptick in efficiency can be attributed in part to the fact that pipeline designs inherently excel at enhancing latency efficiency. By allowing units to work in parallel, the image classifier expends less time processing an equivalent volume of data. These results clearly show that the image classifier with

DOI: 10.54254/2755-2721/62/20240396

pipelined MAC units operates faster than the classifier without such pipelined units. That explained why pipelined structure are commonly used, like [11-13].

4. Conclusion

The MNIST dataset has provided a robust platform for validating and advancing methodologies in handwritten number recognition. The Neural Network Handwritten Number Classifier, built upon neural network principles and fortified with MAC units, underscores the importance of a pipelined architecture. The comparative analysis clearly indicates that a pipelined approach improves processing speed, reducing cycle counts by approximately 42.9%. This emphasizes that the integration of optimized hardware designs can substantially augment computational efficiency in neural network-based image classifiers. While the classifier exhibits notable accuracy and efficiency, the constantly evolving landscape of machine learning and hardware design presents prospective challenges. One foreseeable challenge is adapting the classifier to more complex datasets beyond MNIST, such as the CIFAR-10 or ImageNet. These databases encompass diverse and high-resolution images, demanding further computational resources. For addressing these intricate datasets, advanced neural network models, such as CNN (Convolutional Neural Network) and Transformer, which is the base of the famous GPT series, might be integrated into the classifier. Furthermore, in the absence of synthesis and APR tools, power and area considerations haven't been factored in. Moreover, as quantum computing and neuron network chips gain traction, integrating the classifier into these architectures will be pivotal. Ensuring energy efficiency and sustainability also remains a challenge in upcoming designs. Lastly, with the rise of adversarial machine learning, ensuring the robustness of classifiers against adversarial attacks will be paramount. The hope is that future iterations of this classifier will address these challenges, setting new benchmarks in efficiency and accuracy.

References

[1] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... & Bhatia, S. (2017). In-datacenter performance analysis of a tensor processing unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1-12.

[2] LeCun, Y., Cortes, C., & Burges, C. J. (1998). The MNIST database of handwritten digits.

[3] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408

[4] Ramamoorthy, C. V., & Li, H. F. (1977). Pipeline Architecture. *ACM Computing Surveys, 9*(1), 61–102.

[5] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[6] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. University of Toronto Technical Report.

[7] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 248-255.

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008.

[9] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

[10] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Sutskever, I. (2020). Language Models are Few-Shot Learners. In Proceedings of the 34th Neural Information Processing Systems (NeurIPS 2020).

DOI: 10.54254/2755-2721/62/20240396

[11] Ottavi, M. et al. (2011). Partially Reversible Pipelined QCA Circuits: Combining Low Power With High Throughput. *IEEE Transactions on Nanotechnology*, *10*(6), 1383-1393.

[12] Lu, Z., Wang, J., Li, Z., Chen, S., & Wu, F. (2022). A Resource-Efficient Pipelined Architecture for Real-Time Semi-Global Stereo Matching. *IEEE Transactions on Circuits and Systems for Video Technology*, *32*(2), 660-673

[13] Fasfous, N., Vemparala, M.-R., Frickenstein, A., Frickenstein, L., Badawy, M., & Stechele, W. (2021). BinaryCoP: Binary Neural Network-based COVID-19 Face-Mask Wear and Positioning Predictor on Edge Devices. *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Portland, OR, USA.