Reconfigurable Analog Neural Networks: Architecture, Design, and Performance Evaluation

Amr Elhossan*, Samuel Parent*, Shaan Suthar†, Anthony Turco*, Hydar Zartash*, and Mohamed B. Elamien*
*Department of Electrical and Computer Engineering
McMaster University, Hamilton, ON, Canada
†Department of Computing and Software Engineering
McMaster University, Hamilton, ON, Canada
Emails: {elhossaa, parens4, suthas2, turcoa2, zartashh, elamienm}@mcmaster.ca

***Abstract*—This paper presents a reconfigurable analog neural network architecture implemented with discrete components for machine learning inference tasks. We use digital potentiometers to store model parameters and digital-to-analog converters to represent input data as DC voltages. The system implements matrix multiplication through the op-amp summer configuration and incorporates ReLU activation functions using active rectifiers. The system achieves nearly the same accuracy as its software counterpart when evaluated on the MNIST validation dataset. We also measure the end-to-end propagation delay, and power consumption. Our results demonstrate the viability of application-specific analog computing for AI tasks. The work highlights the potential of reconfigurable analog hardware for edge computing applications where energy efficiency is paramount.**

I. Introduction

Edge computing has drawn significant attention due to the vast deployment of Internet-of-Thing (IoT) technology in applications that require personalized and real-time information processing [1]. Such edge devices enhanced with Artificial Intelligence (AI) demand novel energy-efficient computing paradigms beyond conventional digital accelerators. Analog computing presents an attractive solution due to its inherent advantages in power consumption and low-latency execution [2]–[5]. Furthermore, analog neural networks (ANNs) leverage physical laws such as Ohm's and Kirchhoff's laws to perform vector-matrix multiplications in place, allowing for massive parallelism and often requiring lower complexity compared to digital implementations [6]–[8].

Despite these many advantages, analog approaches face key implementation challenges such as non-linearitiess in device characteristics, mismatch, drift, and limited precision [9]–[11]. The early implementations of neuron-like analog blocks relied on custom analog memories, such as floating-gate transistors or resistive crosspoint arrays [2], [7]. More recent solutions have investigated the integration of standard CMOS-based analog neuron circuitry such as switched capacitors for the accumulation stage or single-transistor multi-level cells for weight storage [3], [12]. Moreover, analog current/voltage multiplier cells can reduce transistor overhead and enhance analog multiply-accumulate (MAC) efficiency. For example, in [13], a translinear-based four-quadrant multiplier/divider uses just four transistors to flexibly handle both current and voltage signals.

Reconfigurable analog accelerators are particularly important for edge computing, where updates to model parameters must be quickly implemented. For example, digital potentiometers can act as tunable resistors to set weights in matrix-multiplication blocks, while digital-to-analog converters (DACs) can map input data onto analog voltages [14]. Although these add overhead in terms of design area and peripheral circuitry, they also enable flexible parameter tuning to support different neural topologies and adapt to changing workloads.

In this work, we present a reconfigurable analog neural network platform using off-the-shelf components such as digital potentiometers for programmable weight storage. The architecture implements MAC operations through a summation of op-amp circuits and integrates Rectified Linear Unit (ReLU) activation functions with active rectifier modules. Our implementation focuses on demonstrating the feasibility and methodology behind analog computation by performing the widely recognized MNIST digit classification task. The use of digital potentiometers allows for dynamic weight adjustment, providing a level of flexibility previously unavailable in traditional analog implementations. Through this work, we aim to illustrate the practical advantages of analog neural networks and encourage further research into sustainable and efficient AI hardware implementations.

This paper is organized as follows. In Section II, we describe the analog hardware implementation. Section III details our software methodology. In Section IV, we present the experimental results and key performance metrics. Finally, Section VI concludes with a summary of contributions and directions for future research.

II. Physical Design Methodology

Inference in neural networks involves a series of linear transformations as neurons implement the inner product of their weights and the preceding layer outputs. Each neuron must also apply a non-linear activation function to this MAC operation and add a bias to introduce a threshold for activation [15], [16]. For the purpose of this paper ReLU activation is applied to logit of the hidden layer neurons. As such, the mathematical model for an individual neuron is given by:

$$N_{out,i} = \text{ReLU}\left(\left(\sum_{k=1}^{n} w_{ik}x_k\right) + b_i\right) \tag{1}$$

Relating the output of the neuron, N_{out}, to the inner product of neuron weights, w_i, and previous layer outputs, x_k, offset by a bias b_i.

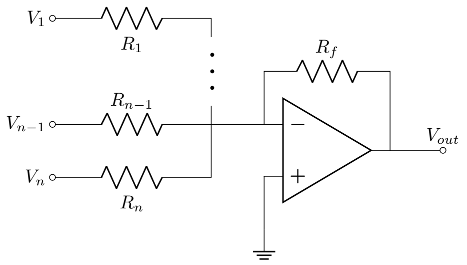


Fig. 1. Summing op-amp configuration

The summing op-amp configuration shown in figure 1 naturally implements this dot product operation, as the output voltage, V_o, assumes the form:

$$V_o = -R_f \sum_{k=1}^{n} V_k/R_k \tag{2}$$

Implementing these resistances with digital potentiometers enables programmability of weight and feedback parameters derived from an external software model. Thus, the analog system becomes reconfigurable for a variety of neural network applications. This approach also simplifies the implementation of four-quadrant multiplication. By applying the signal to one input terminal of the potentiometer and its negative counterpart to the other, both current addition and subtraction are supported. Switches are used to connect the output to sourcing or sinking terminals. Using this approach, weights are programmed to behave as positive or negative multipliers. The biases were implemented similarly, with the exception of having a static supply voltage as input.

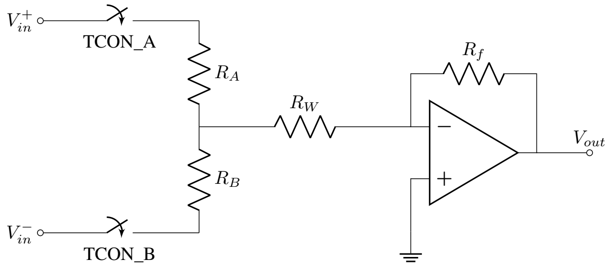


Fig. 2. Switched potentiometer enabling four-quadrant multiplication

A. Implementing the activation function

In addition to linear matrix multiplication, a non-linear activation function must be incorporated in the set of circuit operations. The ReLU activation function is widely favoured in machine learning due to its computational efficiency and ease of implementation [17]. This advantage becomes even more pronounced in the analog domain, where ReLU can be effectively realized through the use of a modified active-rectifier configuration which compliments the summing amplifier implementation.

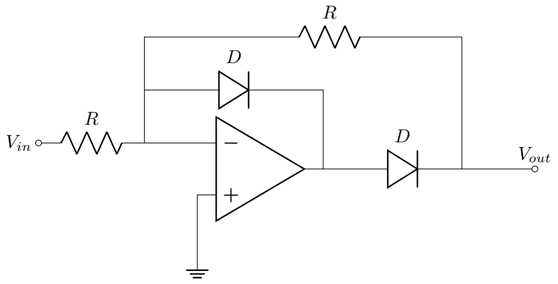


Fig. 3. Active rectifier implementation of negative ReLU

The drawback of this approach is a limitation on output swing. In addition to the output bound imposed by the op-amp supply voltage, the active rectifier is further limited on account of the diode voltage drop.

B. Feedback Resistance Optimization

Quantization error and feedback resistor misalignment pose additional challenges in this approach. It is clear from (2) that the weights are realized as: $w_k = R_f/R_k$. The feedback resistor, however, is common to every weight of a given neuron. This places a constraint on w_k due to its inversely proportional relationship to R_k. A feedback optimization algorithm is used to address this issue. Error is minimized with a priority placed on reducing error in large weights. Thus, the objective function for this problem is given by:

$$\epsilon_i = \sum_{i=1}^{n} \left| \Big(|W_{t,i}| + 1\Big)\left(\frac{R_{f,i}}{R_{k,i}} - |W_{t,i}|\right) \right| \tag{3}$$

Where the decision variables are the programmed path resistance, $R_{k,i}$, and the feedback resistance $R_{f,i}$. W_{ti} represents the desired weight to be achieved in circuitry, and n is the number of paths feeding into a given node. The algorithm implemented in this design iterates through all possible values of R_{fi} that can be implemented by the chosen potentiometers. Path weights are then analytically determined and the quantization error is calculated and incorporated in the objective function. After iterating through all possible quantization levels, the optimal feedback resistance can be selected for the chosen application.

C. Analog SoftMax and Winner-Takes-All Circuitry

SoftMax is a widely used activation function applied to the output of the final network layer. The output of SoftMax is a set of relative probabilities indicating the perceived confidence of inference results. The SoftMax function is given by:

$$\sigma_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{4}$$

Where for a given neuron of the output layer, x_i and σ_i represent the corresponding logit and output probability, respectively. In a circuit implementation, exponential operation can be achieved by driving the base terminals of matched parallel BJTs with layer output voltages. The proportional current through a given path would then be given by:

$$\sigma_i = \frac{I_s e^{\frac{V_{BE,i}}{V_T}}}{\sum_{j=1}^{n} I_s e^{\frac{V_{BE,j}}{V_T}}} \tag{5}$$

By fixing the sum of these currents to some value I_{max} as in [18] and incorporating some known resistance in series with each BJT path, the softmax output can be encoded as a measurable voltage, $V_{meas,i}$:

$$\sigma_i = \frac{V_{meas,i}}{V_{max}} \tag{6}$$

Where $V_{max} = I_{max} * R_{ref}$, and R_{ref} is the resistance in series with each BJT. The output of the SoftMax stage is decoded using a winner-takes-all topology to determine the final network output. All σ_i values are passed as inputs to op-amp comparators. Transmission gate switches are used to pass the winning output to the next comparator, until only the signal with highest activation remains. The comparator output bits are then decoded to determine the index of this signal with respect to the output layer.

III. Software Methodology

A. Model Training

Without dedicated hardware for training, it is necessary to first train on a functionally identical neural network in software. These external training results are then used to determine the appropriate weight and bias values programmed to on-chip potentiometers for a particular model.

A key factor in achieving system accuracy is the comprehensive software modeling of hardware limitations. To map a software-trained neural network onto our system, it was necessary to construct a training pipeline that considered all of the hardware constraints and nonlinearities. These include the saturation voltage of the op-amps, the voltage range of the active rectifier circuit, and the quantization of the DACs. The quantization of digital potentiometers was also considered as a post-training step (see Subsection II-B).

The model architecture was deliberately minimal to reduce complexity during analog realization: a 12-dimension input layer, a single hidden layer of 12 neurons, and a 10-dimensional output layer corresponding to the digit classes. To emulate the behavior of the active rectifier circuit, a clipped ReLU activation function was introduced after the hidden layer. Furthermore, the final output layer was passed through a clamp function with bounds [-2.75, 2.75] to emulate op-amp saturation.

L2 regularization was applied to all trainable parameters - this greatly improves the results of post-training digital potentiometer weight mapping as it constrains weight magnitudes.

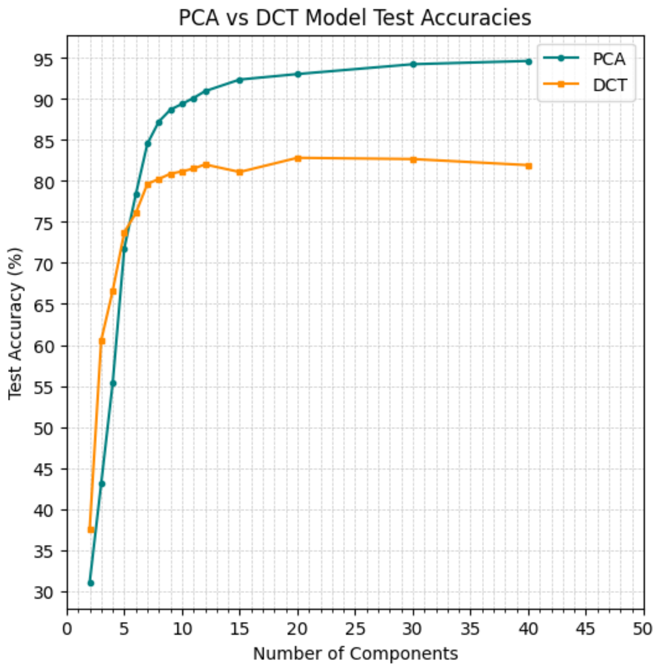


Fig. 4. PCA vs DCT Image Pre-processing for Model Performance

B. Image Processing

The MNIST dataset served as the basis for model development. A key step in enabling handwritten digit recognition on a compact neural network is to reduce the dimensionality of the input while preserving classification accuracy. Since each image is a 28×28 grayscale matrix—equivalent to 784 inputs—dimensionality reduction is essential to minimize the number of tunable parameters required for hardware implementation.

Multiple compression techniques were evaluated, with Discrete Cosine Transform (DCT) and Principal Component Analysis (PCA) yielding the most promising results [19]. As shown in Fig. 4, DCT exhibited slightly better accuracy for extremely low component counts (fewer than 6). However, PCA rapidly surpassed DCT in performance as the number of components increased. Notably, PCA achieved over 90% test accuracy using only 12 components, while DCT plateaued earlier and exhibited diminishing returns beyond 20 components.

Given its superior performance after dimensionality reduction, the PCA transform was selected for final deployment. Each input vector was transformed using the top 12 principal components, then normalized to the range [-1, 1] and quantized to 12-bit resolution over the range [-2.75, 2.75] to match the DACs output voltage range on the physical system.

IV. Results and Comparison

A. Design Configuration

The circuit topology presented was designed and fabricated as a PCB using Altium Designer software. The final implementation of a network layer is a board implementing an array of digital potentiometers, and feeding buffered inputs through

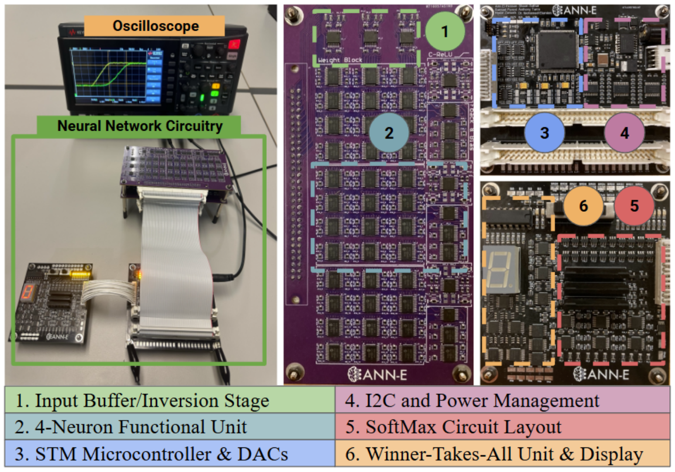
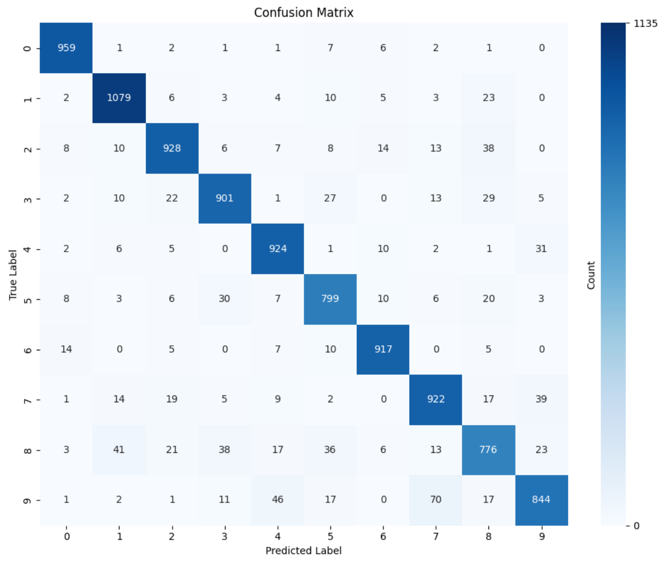

Oscilloscope
Neural Network Circuitry
1
2
3
4
5
6
1. Input Buffer/Inversion Stage
2. 4-Neuron Functional Unit
3. STM Microcontroller & DACs
4. I2C and Power Management
5. SoftMax Circuit Layout
6. Winner-Takes-All Unit & Display

Fig. 5. Inference Time Test Setup and System Overview


Confusion Matrix

True Label \ Predicted Label	0	1	2	3	4	5	6	7	8	9
0	959	1	2	1	1	7	6	2	1	0
1	2	1079	6	3	4	10	5	3	23	0
2	8	10	928	6	7	8	14	13	38	0
3	2	10	22	901	1	27	0	13	29	5
4	2	6	5	0	924	1	10	2	1	31
5	8	3	6	30	7	799	10	6	20	3
6	14	0	5	0	7	10	917	0	5	0
7	1	14	19	5	9	2	0	922	17	39
8	3	41	21	38	17	36	6	13	776	23
9	1	2	1	11	46	17	0	70	17	844

Fig. 6. Neural network confusion matrix for MNIST dataset

the terminals to achieve neuron functionality. The circuitry for ReLU is also included on these boards, and can be bypassed in manufacturing to enable use of a different activation function. A driver board hosts an STM32F767ZIT6, which drives I2C communication to the layer boards and to 12 DACs with 12 bit resolution that comprise the input layer. SoftMax is implemented as a separate PCB to display the layer output on a 7-segment display and forward the processed output back to the driver board, which manages all propagation of signals between layers. A 12V source supplies power for a 3.3V domain for the control path and a 5.5V domain for the datapath.

B. Measured Results

The accuracy of the proposed implementation was assessed by sending labeled image data through the network and comparing the output with the expected inference results. This methodology was used to develop a confusion matrix for the set of possible outputs (see Fig. 6). The overall accuracy of the system was determined to be 90.11% when run on hardware while its software counterpart yielded 90.95%. Digit-wise accuracy figures were also collected and are displayed in Table 1 below. These values closely resemble those achieved by the software model, demonstrating comparable performance for networks of similar size.

TABLE I
MEASURED ACCURACY BY INDIVIDUAL DIGIT

0	1	2	3	4
94.18%	93.04%	90.98%	90.79%	85.34%
5	**6**	**7**	**8**	**9**
89.57%	96.03%	88.71%	85.63%	87.02%

Inference time was measured by reconfiguring the board to pass a step function as input. The measured delay between the input and output steps represents the propagation time through the system. The test configuration for these measurements is shown in Fig. 5. Note that this inference time only includes the portion of the data-path which incorporates an analog implementation in circuitry. Additional delays, such as the time taken to pre-process images and configure DAC outputs, are not considered in these measurements. The inference time was measured to be 1.827μs from DAC output to layer output, and 6.114μs from DAC output to SoftMax output.

Power figures for the system were measured using an MCA1101-5-3 ACEINNA Board Mount Current Sensor. In a similar fashion, these figures encapsulate only the portion of the design powered on a 5.5V supply. This includes all analog circuitry relevant to the scope of research, such as the DACs, layer boards, and Softmax implementation. The average datapath power consumption while performing inference was measured to be 165mW.

V. Conclusion

A reconfigurable analog neural network is introduced in this work. The final design is implemented on a PCB using discrete components, and uses tunable resistive elements to realize network weights. The system is trained on the PCA-transformed MNIST dataset, and successfully performs inference to identify hand-drawn digits. Excellent overall accuracy is demonstrated, closely matching the accuracy achieved during software training. This accuracy is achieved with only 32 total network nodes in a 12-12-10 configuration. The inference time for a discrete-component implementation is reasonable, but improving this metric could be the subject of future research. This architecture demonstrates potential in prototyping applications, and edge implementations requiring smaller network sizes.

References

[1] J. Zhang and D. Tao, "Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7789–7817, 2021.

[2] M. Ronchi, S. D. Giacomo, M. Amadori, G. Borghi, M. Carminati, and C. Fiorini, "Design, implementation, and analysis of an integrated switched capacitor analog neuron for edge computing ai accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2025.

[3] S. Moriya, H. Yamamoto, S. Sato, Y. Yuminaka, Y. Horio, and J. Madrenas, "A fully analog cmos implementation of a two-variable spiking neuron in the subthreshold region and its network operation," in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–7.

[4] J. Zhu, B. Chen, Z. Yang, L. Meng, and T. T. Ye, "Analog circuit implementation of neural networks for in-sensor computing," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021, pp. 150–156.

[5] Y. C. Xiang, P. Huang, Z. Zhou, R. Z. Han, Y. N. Jiang, Q. M. Shu, Z. Q. Su, Y. B. Liu, X. Y. Liu, and J. F. Kang, "Analog deep neural network based on nor flash computing array for high speed/energy efficiency computation," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–4.

[6] Z. Zhu and L. Feng, "A review of sub-μw cmos analog computing circuits for instant 1-dimensional audio signal processing in always-on edge devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 9, pp. 4009–4018, 2024.

[7] T. P. Xiao, B. Feinberg, C. H. Bennett, V. Prabhakar, P. Saxena, V. Agrawal, S. Agarwal, and M. J. Marinella, "On the accuracy of analog neural network inference accelerators," *IEEE Circuits and Systems Magazine*, vol. 22, no. 4, pp. 26–48, 2022.

[8] Y. Du, L. Du, X. Gu, J. Du, X. S. Wang, B. Hu, M. Jiang, X. Chen, S. S. Iyer, and M.-C. F. Chang, "An analog neural network computing engine using cmos-compatible charge-trap-transistor (ctt)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1811–1819, 2019.

[9] S. Garg, J. Lou, A. Jain, Z. Guo, B. J. Shastri, and M. Nahmias, "Dynamic precision analog computing for neural networks," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 29, no. 2: Optical Computing, pp. 1–12, 2023.

[10] M. Seok, M. Yang, Z. Jiang, A. A. Lazar, and J.-S. Seo, "Cases for analog mixed signal computing integrated circuits for deep neural networks," in *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2019, pp. 1–2.

[11] J. Zurada, "Analog implementation of neural networks," *IEEE Circuits and Devices Magazine*, vol. 8, no. 5, pp. 36–41, 1992.

[12] M. D. Edwards, N. J. Sarhan, and M. Alhawari, "A cmos analog neuron circuit with a multi-level memory," in *2023 International Conference on Microelectronics (ICM)*, 2023, pp. 11–15.

[13] A. Elwakil, B. Maundy, M. B. Elamien, and L. Belostotski, "A four-quadrant current multiplier/divider cell with four transistors," *Analog Integrated Circuits and Signal Processing*, vol. 95, no. 1, pp. 173–179, 2018.

[14] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," *Frontiers in neuroscience*, vol. 10, p. 333, 2016.

[15] T. L. Fine, *Feedforward Neural Network Methodology Information Science and Statistics*. Springer Science Business Media, 2006.

[16] N. Aftabi, N. Moradi, and F. Mahroo, "Feed-forward neural networks as a mixed-integer program," *Engineering with Computers*, 2025.

[17] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," in *Neurocomputing*, vol. 503, 2022, pp. 92–108.

[18] J. Sillman, "Analog implementation of the softmax function," 2023. [Online]. Available: https://arxiv.org/abs/2305.13649

[19] D. García Moreno, A. A. Del Barrio, G. Botella, and J. Hasler, "A cluster of fpaas to recognize images using neural networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 11, pp. 3391–3395, 2021.