Low-Cost Real-Time Energy-Efficient Edge Image Classifier Architecture based on Reservoir Computing with Cellular Automata

Jawdat Andraous*, Tom E. H. Glover†, Ali Muhtaroglu ‡§
* ACIT Robotics and Control MS Program † Dept. of Computer Science
‡ Dept. of Mechanical, Electrical and Chemical Engineering
§ Advanced Health Intelligence and Brain-Inspired Technologies (ADEPT)
Faculty of Technology, Art and Design (TKD)
Oslo Metropolitan University, PO Box 4, St. Olavs plass, 0130 Oslo, Norway

***Abstract*—This paper introduces a real-time, edge-oriented image classification architecture utilizing Reservoir Computing with Elementary Cellular Automata (ReCA) designed for resource-constrained environments. By integrating single-threshold binary inputs, column-wise evolution, and spatial downsampling with a lightweight, 6-bit spiking-inspired readout, the proposed pipeline drastically minimizes memory footprint and logic complexity. Architectural optimizations across the hardware modules yield a 2–4$\times$ reduction in resource overhead relative to existing baselines. Furthermore, transitioning from offline software-based classification to an online, perceptron-based winner-takes-all (WTA) neural network with 6-bit integer weights enhances energy efficiency significantly, with potential of orders of magnitude improvement. Validation through an in-house hardware emulator on the MNIST dataset demonstrates that substantial efficiency gains are achieved due to complexity reduction in learning engine at a small accuracy penalty of approximately 2.5%. The results confirm that integer-based ReCA systems with online learning provide a scalable, battery-efficient foundation for real-time adaptive inference at the edge.**

***Index Terms*—reservoir, cellular automata, edge computing, image classification, spiking neural networks, online learning, low-cost, energy-efficient design**

I. Introduction

Pattern recognition at the edge faces strict energy and latency limits, making conventional convolutional neural networks (CNNs) impractical due to heavy compute and data-movement [1]. Neuromorphic, event-driven parallelism preserves accuracy while lowering power in dynamic settings [2]. By exploiting sparsity, temporal coding, and on-device adaptation, these architectures reduce model size and frequent off-chip memory access [3], making energy-efficient neuromorphic hardware vital for scalable edge intelligence [2].

Reservoir computing (RC) maps inputs into high-dimensional dynamical reservoirs with simple readouts [4]. Fixing reservoir dynamics and training only the readout avoids backpropagation through time and reduces learning complexity [5], [6]. While Echo State Networks (ESNs) and Liquid State Machines (LSMs) process temporal structure well, their dense connectivity limits hardware efficiency. In contrast, RC with elementary Cellular Automata (ReCA) uses discrete, local, massively parallel dynamics for hardware-friendly computation [7]–[9]. Elementary Cellular Automata (ECA) enable efficient digital implementations via simple logic while providing rich transients for pattern recognition [10]–[12]. Studies show ReCA achieves competitive MNIST accuracy with substantially lower power and latency than conventional neural networks.

Despite promise [9], [10], [13], edge deployment faces obstacles: ReCA is highly sensitive to ECA rule choice and input encoding, yet their relative impact remains unclear due to limited systematic ablations [14], [15]. Designs often prioritize accuracy over power, leaving hardware savings unrealized. Heavy pre-processing and offline training hinder real-time use [9], [10]. Convolution-free CA shows progress, but rigorous benchmarking is needed to quantify accuracy–complexity trade-offs for edge hardware [11], [13], [15].

This work optimizes ReCA for image classification under tight resource budgets. We examine efficient CA propagation, aggressive quantization, and lightweight learning to achieve real-time, energy-efficient inference with competitive accuracy. We provide quantified cost and power projections from hardware-emulator simulations. The following optimization techniques are analyzed:

(i) Image downsampling;
(ii) optimization of CA propagation and temporal depth,
(iii) feature downsampling in post-CA processing,
(iv) lightweight perceptron-based winner-takes-all (WTA) neural network (PWTANN) hardware architecture.

The remainder of this paper is organized as follows: Section II discusses image classification using ReCA within the context of recent literature. Section III details

the proposed architectural enhancements aimed at reducing cost and power dissipation. Section IV provides analytical results from an in-house Python emulator to quantify trade-offs on the MNIST benchmark, followed by concluding remarks in Section V.

II. Image Classification using ReCA

A. ReCA Systems for Reservoir Computing

ECA are 1D arrays of binary cells updated in discrete time by local rules using a cell and its two neighbors. In ReCA, rule choice shapes reservoir dynamics and strongly affects performance. Studies often identify Rule 90 as a top candidate [8], [10], [13], [14], which computes each cell's next state as the XOR of its left and right neighbors, enabling efficient digital implementations (Fig. 1).This elementary local processing makes ReCA well suited to FPGA hardware, where parallelism and low power are critical [10].

In a common ReCA architecture, input data are encoded into the initial state of the CA and iterated for a fixed number of time steps. The resulting CA states across iterations are concatenated or pooled to form a feature vector, which is then passed to a linear classifier for training and inference [14], [15]. Because the ECA dynamics remain fixed, the computational burden is confined primarily to the output layer, facilitating real-time operation at the edge. The general structure of a ReCA system used for reservoir computing is shown in Fig 2.

B. Application to Image Classification

ReCA leverages parallel, local dynamics and hardware efficiency for pattern recognition [7], [9], [10]. Pixels are mapped to cellular states, and the spatiotemporal evolution forms a high-dimensional reservoir that supports simple linear readouts with low training cost [7], [9], [10].

CNN layers are replaced with convolution-free CA-based feature reasoning, which maintains competitive MNIST and CIFAR-10 performance, achieving $>$ 90% accuracy while cutting computation by $<$ 90% in selected settings [11], [13]. Combining ECA with simple classifiers further improves recognition at lower power dissipation, supporting real-time edge inference [9], [10].

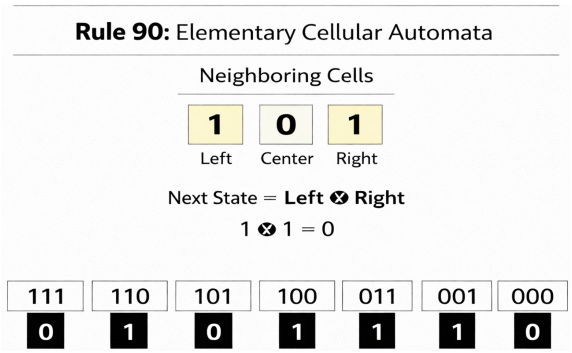

Fig. 1: Elementary CA Rule 90 operation.

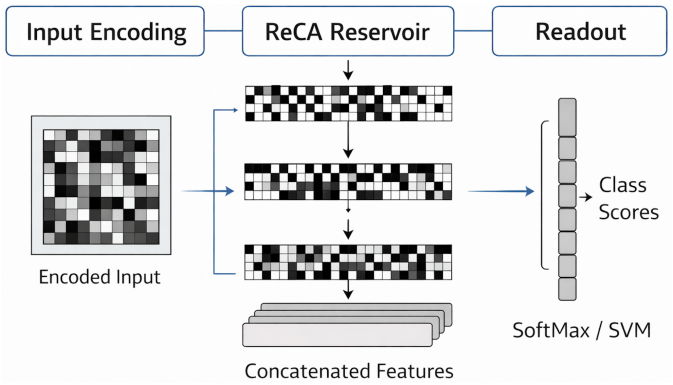

Fig. 2: ReCA system for reservoir computing.

C. Off-line Training using SoftMax Regression

A fundamental characteristic of ReCA-based learning systems is the separation of fixed reservoir dynamics from a trained readout. This avoids backpropagation and reduces training complexity compared to fully trainable networks. Offline training collects labeled reservoir outputs to optimize SoftMax or linear weights, which are then fixed for inference (Fig. 3) [9], [10].

Hardware-oriented CA reservoirs further support resource-constrained environments using offline-trained readouts in [12]. Offline SoftMax-based training provides a practical balance between training efficiency and deployment simplicity [9], [10], [13]. The reference implementation used as the starting point in this work [10] is one such hybrid approach for edge applications, where training is done offline using Adam stochastic optimization and weights are loaded to the neural network in hardware for inferencing only.

D. Low-Cost, Real-Time, Low-Energy Challenges

Despite the advantages of ReCA-based image classification, real-time edge deployment faces challenges in cost, latency, and energy. The prevailing offline-training paradigm decouples learning from inference but incurs

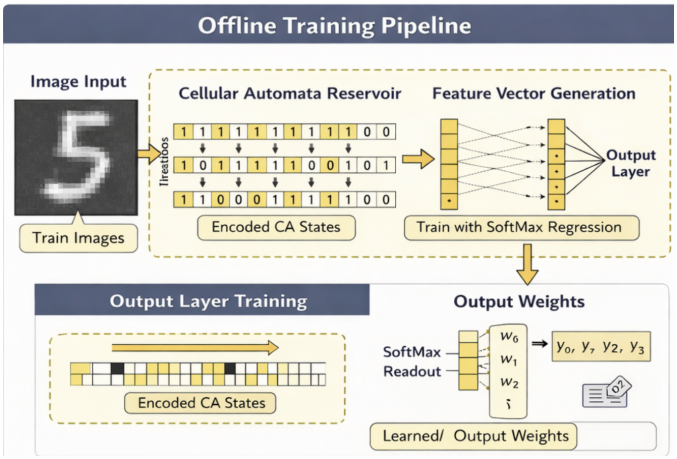

Fig. 3: Typical offline software training pipeline for SoftMax regression in ReCA systems [10].

delays for data collection, feature extraction, and retraining, limiting adaptability in dynamic environments [9], [10]. State-of-the-art ReCA further imposes computational overhead from repeated CA iterations to build high-dimensional reservoirs, increasing real-time processing latency and memory bandwidth demands; large lattices and deep temporal evolution can degrade throughput despite simple, parallelizable updates [10]. Hardware resources are also constrained by storage for intermediate CA states, feature buffers, and readout weights, with memory-centric designs emphasizing efficient data structures to manage footprint and energy [10], [12]. Architectural complexity can rise with additional preprocessing and multi-stage pipelines, including CA–neural hybrids that improve accuracy but add compute and memory costs, reducing the low-latency and low-power benefits sought for edge systems [9], [13].

These limitations motivate optimized ReCA that reduces hardware and associated power dissipation, iteration overhead and memory traffic, while preserving acceptable accuracy.

III. Real-Time Edge Architecture

As depicted in Fig. 4, the pipeline comprises image preprocessing, column-wise cellular automata propagation, spatial downsampling, and a lightweight classification stage with real-time learning, which will all be further detailed in the next subsections.

A. Image preprocessing (Resolution Downsampling)

As shown in Fig. 5, the architecture applies binary preprocessing before ReCA propagation. The reference system decomposes a 28×28 grayscale image into multiple bit-planes for multi-layer CA initialization, increasing memory use and data movement [10]. The proposed method performs single-bit encoding via a fixed global threshold as:

$$B(x,y) = \begin{cases} 1, & I(x,y) > 128 \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

This balances foreground–background separation, eliminates bit-plane stacking, reduces bandwidth requirements, and aligns with Boolean CA rules. Fixing the input to 28×28 (MNIST) allows the binarized image to directly initialize the automata, ensuring deterministic mapping and fixed-dimensional feature generation.

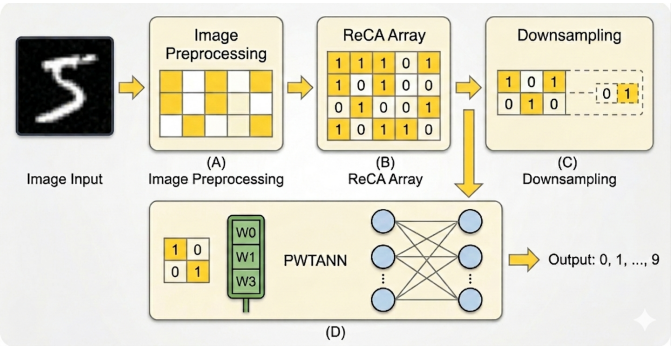

Fig. 4: Overview of the proposed ReCA edge architecture: (A) Preprocessing, (B) column-wise CA propagation, (C) feature downsampling, and (D) a lightweight real-time classifier.

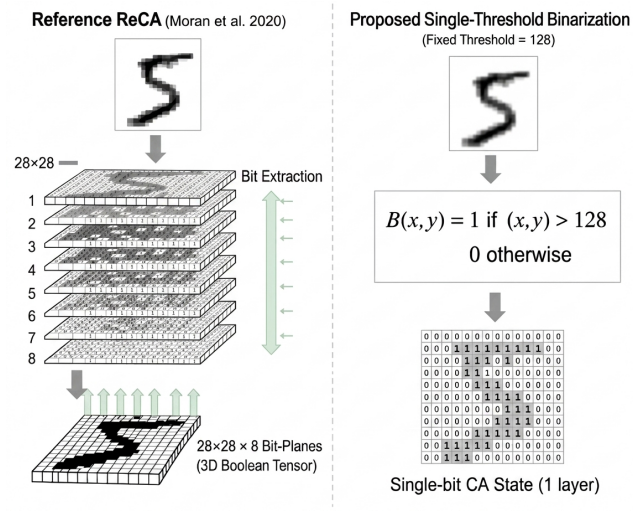

Fig. 5: Comparison of multi-bit ReCA initialization [10] (left) versus the proposed single-threshold binary mapping (right).

B. ReCA Array

Unlike the reference ReCA, which evolves cellular automata along both rows and columns with XOR fusion over 16 iterations [10], the proposed reservoir restricts propagation to the vertical dimension and reduces iterations to 8 with no measurable penalty, as depicted in Fig. 6. Column-wise updates apply Rule 90 independently per column with fixed zero boundaries, eliminating both horizontal propagation and merge stages to simplify the datapath. This results in reduced independent parallel column processing for scalable scheduling. For a binary CA state $x(k)$ at iteration k,

$$x_i(k+1) = x_{i-1}(k) \oplus x_{i+1}(k), \tag{2}$$

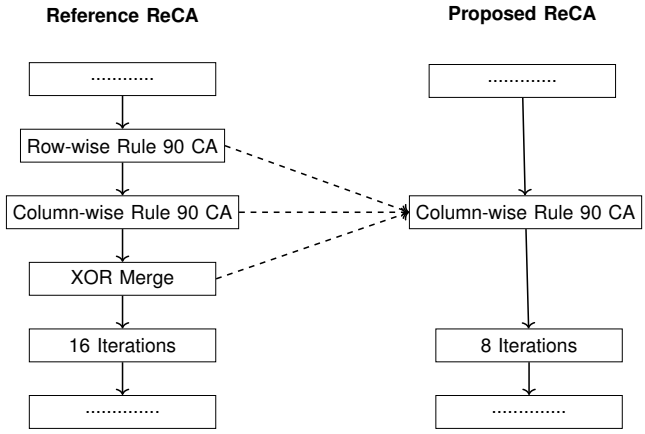

Fig. 6: Dual-direction ReCA propagation versus the proposed column-only, reduced-depth architecture.

where i represents the vertical cell index within each column. Shorter temporal depth lowers latency, buffering, and switching activity while preserving sufficient temporal mixing for discriminative features. The resulting state sequence proceeds to the downsampling stage (next section) prior to classification.

C. Feature Downsampling

After ReCA propagation, spatial downsampling is applied to reduce feature dimensionality and memory bandwidth before classification. Unlike the reference architecture, which applies pooling on higher-dimensional multi-bit reservoir states, the proposed system performs pooling directly on the single-layer binary outputs produced by the column-only ReCA array. A non-overlapping 2×2 max pooling operation with stride 2 is used to reduce each 28×28 reservoir state to a 14×14 feature map. For a reservoir state $x(k)$, the pooled output $p(k)$ is defined as:

$$p_{i,j}(k) = \max \Big\{ x_{2i,2j}(k),\ x_{2i+1,2j}(k), \\ x_{2i,2j+1}(k),\ x_{2i+1,2j+1}(k) \Big\}. \tag{3}$$

Concatenating pooled maps across eight iterations yields a 1568-bit feature vector, lowering data movement and enabling efficient real-time edge operation.

D. Perceptron-based Winner-Takes-All Neural Network

The final stage integrates a lightweight perceptron-based winner-takes-all (WTA) classifier capable of online learning (Fig. 7), replacing the offline fully connected readout used in the reference ReCA system [10]. In contrast to a true temporal spiking neural network, this classifier does *not* implement spike generation, membrane dynamics, reset behavior, or event-driven temporal processing. It is therefore more accurately described as a quantized linear classifier with binary inputs and discrete online updates, designed for hardware realization.

The classifier receives a 1568-dimensional binary feature vector extracted from the reservoir, corresponding to a 14×14 pooled representation over 8 ReCA iterations, and maps it to 10 output neurons, one for each MNIST digit class. Thus, the architecture consists of 1568-inputs fully connected to a 10-neuron output layer. Each output

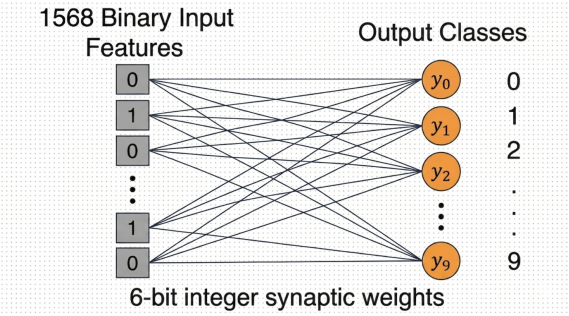

Fig. 7: Fully connected PWTANN classifier with online learning (input: 1568 feature bits; output: 10 classes) - synapses (not shown) on all connections with 6-bit integer weights.

neuron computes an integer-valued linear score from the active binary inputs:

$$s_k = \sum_{i=1}^{1568} w_{k,i} x_i + b_k, \qquad k \in \{0, \ldots, 9\} \tag{4}$$

where $x_i \in \{0, 1\}$ is the ith binary input feature, $w_{k,i}$ is the integer synaptic weight connecting input i to output class k, and b_k is the bias associated with class k. The predicted class is then obtained through a winner-takes-all decision rule:

$$\hat{y} = \arg\max_k s_k, \tag{5}$$

where y denotes the ground-truth class label and $\hat{y}$ denotes the predicted class label. Learning is performed online using a perceptron-style error-driven update with fixed step size $\eta = 1$. If $\hat{y} = y$, no update is applied. If $\hat{y} \neq y$, only the parameters associated with the correct class y and the incorrectly selected class $\hat{y}$ are modified. Specifically, the target-class weight row is incremented and the predicted-class weight row is decremented:

$$w_y \leftarrow w_y + \eta x, \tag{6}$$

$$w_{\hat{y}} \leftarrow w_{\hat{y}} - \eta x, \tag{7}$$

while all remaining class weight rows are left unchanged. Because x is binary, updates occur only at active feature positions, making the learning rule highly localized and computationally efficient. The classifier biases are adapted using the same error-driven logic:

$$b_y \leftarrow b_y + \eta, \tag{8}$$

$$b_{\hat{y}} \leftarrow b_{\hat{y}} - \eta. \tag{9}$$

Thus, when a sample is misclassified, the correct class is reinforced both by increasing its feature-aligned weights and by raising its bias, while the incorrectly winning class is discouraged by decreasing both its weights and its bias. In this formulation, the bias acts as a class-specific baseline offset, allowing each output neuron to maintain a learned preference even when few input features are active or when class scores are closely matched. This improves the robustness of single-pass online learning while preserving implementation simplicity.

To minimize hardware cost, synaptic weights and biases are stored as 6-bit integers, reduced from the 8-bit precision used in [10]. All weights are initialized to mid-range value, while the biases are assigned a common base value with a small fixed offset pattern to break symmetry across output neurons in a reproducible manner. The overall learning and inference procedure requires only integer accumulation, comparison, increment, and decrement, completely avoiding gradient computation, floating-point arithmetic, and backpropagation. This makes the classifier well suited to low-latency, real-time edge deployment while maintaining competitive accuracy, as discussed in Section IV.

IV. Results

The proposed real-time edge classification pipeline illustrated by Fig. 4 in Section III has been evaluated using Python 3 CPU-based architectural emulation. This section reports classification analyses to validate the architectural design decisions, in the order they were modeled and analyzed: input preprocessing, downsampling and iteration depth, classification complexity and precision and, ReCA propagation. Insights are also provided based on projected cost, power dissipation and energy savings. Comparisons are done with respect to the baseline architecture in [10], which uses offline elastic augmentation to add 55,000 new samples twice to MNIST dataset during the training process to increase overall classification accuracy from 91.5% to 95% (grayscale images, offline Adam optimization of weights that are quantized to 8 bits before getting loaded back to hardware network.) Elastic augmentation is ignored in this work, which focuses on real-time data streams.

A. Image Resolution Reduction

Table I summarizes the performance impact of reducing input resolution from 8-bit to 1-bit (black and white, monochrome). While there is no accuracy degradation from the baseline (91.5%) considering standard MNIST dataset without augmentation [10]), an estimated 40% reduction in hardware area and power dissipation is realized, a projection based on [16], [17]. Although latency reduction is highly dependent on the specific datapath organization, conservative estimates based on [18], [19] suggest a 10%–50% improvement in execution time. These bounds inform the energy saving projections in Table I, though empirical evidence from similar optimizations suggests these benefits may be significantly higher in practice [16], [17].

B. Sensitivity to Downsampling and Iteration Depth

To evaluate temporal depth, the number of ReCA iterations is varied while maintaining 2×2 max pooling. Fig. 8 demonstrates that accuracy improves up to eight iterations, beyond which it saturates, with some observable variance in the system due to changes in the feature vectors with different iteration count. Consequently, the proposed architecture adopts eight iterations to minimize latency and energy consumption. This optimization yields an estimated 2 times (2x) improvement in both execution time and energy consumption compared to the original 16 iterations [10].

TABLE I: Reduction for proposed 1-bit pixel input in simulated accuracy, projected cost / average power and energy (all normalized to 8-bit grayscale value of ***1*** [10]).

Input	Simulated Accuracy	Cost / Avg. Power Diss. est.	Execution Time est. range	Energy Diss. est. range
1-bit B&W	1	0.6	0.5-0.9	0.3-0.5

C. PWTANN Classifier with Reduced Weight Resolution

The correlation between weight precision and classification performance is detailed in Fig. 9. While the baseline hardware implementation employed 8-bit integer weights [10], the proposed implementation utilizes unsigned 6-bit quantization to achieve further resource optimization. In addition, the proposed architecture replaces the multi-epoch, offline softmax Adam routine with a single-pass PWTANN classifier. Comparative simulation results, summarized in Table II, indicate a nominal accuracy reduction of 2.5%. However, the hardware-native classifier eliminates gradient computations, floating-point arithmetic, and iterative training cycles. Based on contemporary literature [20], this architectural simplification is expected to yield a projected energy reduction of more than an order of magnitude.

D. ReCA propagation scheme

To evaluate spatial propagation effects, three ReCA configurations were simulated: Column-only, row-only, and combined row+column propagation of Rule 90 (with XOR merging) using 1-bit input pixels and PWTANN classifier.

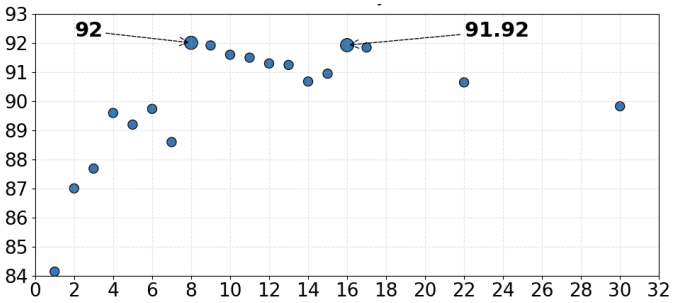

Fig. 8: Classification accuracy versus number of retained CA iterations with monochrome pixels.

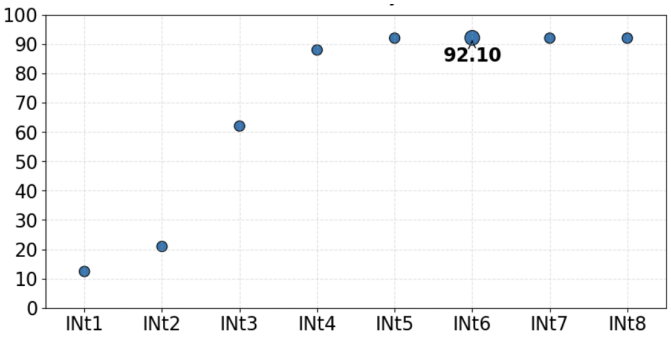

Fig. 9: Classification accuracy (%) versus weight precision (iteration count=8).

TABLE II: Classifier accuracy comparison (iteration count=8).

Classifier	Train. type	Number Precision	Accuracy (%)
Softmax (Adam)	Offline, multi-epoch	INT8	92
Proposed PWTANN	Online, single-pass	UINT6	89.5

TABLE III: Overall accuracy and per-digit accuracy spread comparison across ReCA propagation schemes (iteration count=8).

Propagation Method	Overall Accuracy (%)	Accuracy Spread (%)
Column-only (proposed)	**91.89**	**84.8–99.0**
Row-only	91.25	81.9–98.6
Row + Column + XOR	89.47	74.8–97.8

As summarized in Table III, the column-only approach achieves the best overall accuracy while streamlining the datapath by eliminating horizontal processing and XOR merging. Since the values may be within the system's numerical error, further analysis has been performed on the accuracy associated with each digit in MNIST dataset. The variance across the 10 MNIST classes in the last column further confirms vertical propagation alone provides sufficient spatiotemporal diversity for digit classification. The hardware reduction is expected to reduce the number of used XOR gates for Rule 90 to roughly one third, which has both Rule 90 computation latency and energy efficiency benefit.

V. Conclusion

The proposed edge-oriented architecture optimizes the Reservoir Cellular Automata (ReCA) framework by introducing modular reductions in hardware complexity and energy demand. By independently tuning input resolution, rule propagation, temporal depth, and feature downsampling, the online hardware modules achieve a 2–4$\times$ reduction in resource overhead compared to the baseline in [10]. Replacing the original offline software classifier with a lightweight, 6-bit perceptron-based winner-takes-all neural network (PWTANN) is expected to improve energy efficiency by orders of magnitude. Experimental validation using a hardware emulator on the MNIST dataset reveals a marginal accuracy penalty (none from structural simplifications and a nominal 2.5% from switching to PWTANN with real-time learning features), a trade-off highly viable for battery-constrained edge applications. This pipeline establishes a scalable foundation for real-time adaptive inference, with future efforts directed toward hardware realization and large-scale dataset extensions.

References

[1] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, pp. 10–19, Jan. 2022, doi: 10.1038/s43588-021-00184-y.

[2] P. Arena, M. Bucolo, A. Buscarino, L. Fortuna, and M. Frasca, "Reviewing Bioinspired Technologies for Future Trends: A Complex Systems Point of View," *Frontiers in Physics*, vol. 9, Art. no. 750090, Oct. 2021, doi: 10.3389/fphy.2021.750090.

[3] C. D. James, J. B. Aimone, N. E. Miner, C. M. Vineyard, F. H. Rothganger, K. D. Carlson, S. A. Mulder, T. J. Draelos, A. Faust, M. J. Marinella, J. H. Naegle, and S. J. Plimpton, "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications," *Biologically Inspired Cognitive Architectures*, vol. 19, pp. 49-64, Jan. 2017, doi: 10.1016/j.bica.2016.11.002.

[4] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019.

[5] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks," *GMD Report 148*, German National Research Center for Information Technology, 2001.

[6] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[7] S. Nichele and A. Molund, "Deep Reservoir Computing Using Cellular Automata," *arXiv preprint arXiv:1703.02806*, 2017.

[8] O. Yilmaz, "Reservoir computing using cellular automata," *arXiv preprint arXiv:1410.0162*, 2014.

[9] A. Morán, C. F. Frasser, and J. L. Rosselló, "Reservoir computing hardware with cellular automata," *arXiv preprint arXiv:1806.04932*, 2018.

[10] A. Morán, C. F. Frasser, M. Roca, and J. L. Rosselló, "Energy-Efficient Pattern Recognition Hardware With Elementary Cellular Automata," *IEEE Trans. Comput.*, vol. 69, no. 3, pp. 392–404, 2020, doi: 10.1109/TC.2019.2949300.

[11] N. Sawahashi and J. C. Principe, "End-to-end image classification in linear hybrid cellular automata," in *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN)*, 2024, pp. 1–6, doi: 10.1109/IJCNN60899.2024.10650865.

[12] D. Liang, M. Hashimoto, and H. Awano, "Bloomca: A memory efficient reservoir computing hardware implementation using cellular automata and ensemble bloom filter," in *Proc. 2021 Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, pp. 587–590, 2021.

[13] N. Ari, R. C. Yarnell, P. Amoruso, J. Mell, R. F. DeMara, and A. S. Wu, "FOCAL: Feature-Oriented Cellular Automata Learning for Convolution-Free Image Classification," in *Proc. IEEE 12th Int. Conf. Intelligent Systems (IS)*, 2024, pp. 1–6, doi: 10.1109/IS61756.2024.10705243.

[14] T. E. Glover, E. Osipov, and S. Nichele, "On when is Reservoir Computing with Cellular Automata Beneficial?," *arXiv preprint arXiv:2407.09501*, 2024.

[15] T. E. Glover, R. Jahren, F. Martinuzzi, P. G. Lind, and S. Nichele, "A sensitivity analysis of cellular automata and heterogeneous topology networks: partially-local cellular automata and homogeneous homogeneous random boolean networks," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 40, 2024, doi: 10.1080/17445760.2024.2396334.

[16] H. Rasheed, P. Mirtaheri and A. Muhtaroglu, "A Reduced Spiking Neural Network Architecture for Energy Efficient Context-Dependent Reinforcement Learning Tasks," *International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1-5, doi: 10.1109/ISCAS58744.2024.10558211.

[17] J. I. Okonkwo and A. Muhtaroglu, "Energy-aware bio-inspired spiking reinforcement learning system architecture for real-time autonomous edge applications," *Front. Neurosci.*, vol. 18, art. no. 1431222, Sep. 2024. doi: 10.3389/fnins.2024.1431222.

[18] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed.*, New York, NY, USA: Oxford University Press, 2010, pp. 48–51.

[19] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach, 6th ed.* Cambridge, MA, USA: Morgan Kaufmann, 2017, ch. 3.

[20] E. Eroglu, A. Bahar, H. Ulusan and A. Muhtaroglu, "Low-Energy Real-Time Self-Organizing Classifiers for Edge Computing," *37th International Conference on Microelectronics (ICM)*, pp. 104-107, 2025.