IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 September 2024, accepted 11 October 2024, date of publication 16 October 2024, date of current version 30 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3482110

RESEARCH ARTICLE

Optimizing Memristor-Based Synaptic Devices for Enhanced Energy Efficiency and Accuracy in Neuromorphic Machine Learning

BAKI GÖKGÖZ[1], TOLGA AYDIN[2], AND FATIH GÜL[3], (Member, IEEE)
[1]Department of Computer Technologies, Torul Vocational School, Gümüşhane University, 29100 Gümüşhane, Türkiye
[2]Faculty of Engineering, Computer Engineering Department, Atatürk University, 25030 Erzurum, Türkiye
[3]Faculty of Engineering and Architecture, Electrical and Electronics Engineering Department, Recep Tayyip Erdoğan University, 53100 Rize, Türkiye

Corresponding author: Fatih Gül (fatih.gul@erdogan.edu.tr)

This work was supported by the Scientific Research Projects (BAP) Unit of Atatürk University, Erzurum, Türkiye, under Grant FDK-2022-9895.

ABSTRACT The traditional Von Neumann computing architecture, which necessitates data transfer between external memory and the processor, incurs significant energy and time costs when running deep learning (DL) and machine learning (ML) architectures. The primary issue with the energy and time efficiency of this architecture stems from the frequent and intensive data transfers between memory and the processor. Therefore, memristive synaptic devices are utilized to overcome this energy and time inefficiency while performing cognitive tasks. The fundamental working principle of memristive devices is to reduce the need for data transfer by combining memory and processing in the same location, thereby significantly decreasing both energy consumption and the time required for operations. However, to achieve the desired level of efficiency in terms of energy and time consumption from neuromorphic systems, the performance of these systems needs to be further improved with respect to accuracy and test error rates for classification applications. Achieving high accuracy performance in such deep learning or machine learning models necessitates optimization processes not only at the hardware level but also at the algorithmic level. In this context, this paper presents a comprehensive examination and comparison of the frequently used SGD and its momentum variants for deep learning and machine learning applications in memristor-based neuromorphic computing systems. The study thoroughly investigates the performance of critical metrics such as the learning properties, energy efficiency, and accuracy rates of the nano-scale titanium dioxide (TiO_2) based synaptic device. The experimental results for the MNIST dataset showed AdaDelta 89.48%, AdaGrad 79.00%, Adam 79.13%, AdaMax 79.68%, Momentum 88.55%, Nadam 81.20%, RMSprop 84.91% and SGD 89.47% accuracy. The experimental results for the CIFAR dataset showed AdaDelta 90.51%, AdaGrad 82.08%, Adam 83.10%, AdaMax 81.76%, Momentum 91.25%, Nadam 82.45%, RMSprop 88.11% and SGD 90.21% accuracy.

INDEX TERMS Deep learning, machine learning, memristors, neuromorphic computing, optimization algorithms, synapses.

I. INTRODUCTION

Today, machine learning and deep learning are rapidly becoming widespread and finding various application areas [1]. These methods, which began in the academic world, have quickly grown to become significant in both academic and industrial applications. They are also indispensable components of data science [2]. Thanks to advancements in computer technologies, they have gained an important place in artificial intelligence fields [3]. However, the traditional Von-Neumann architecture falls short in methods inspired by biological neural networks [4]. This inadequacy is due

The associate editor coordinating the review of this manuscript and approving it for publication was Berdakh Abibullaev.

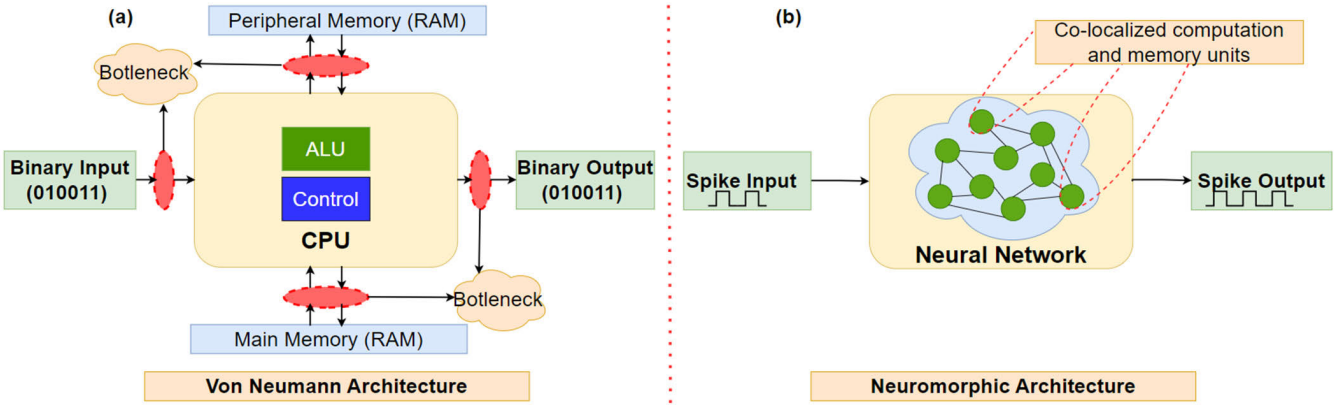

FIGURE 1. (a) and (b) Comparison between the von Neumann architecture and neuromorphic architecture. These two architectures fundamentally differ in their operation, organization, programming, communication, and timing, as shown here.

to the fact that information processing structures are quite different from biological neural systems. Fig. 1(a) shows that in sequential processing, computation and memory are separate, while Fig. 1(b) shows that in parallel processing, computation and memory are integrated and processes occur asynchronously. For example, while supercomputers use 1 MW of power, the human brain consumes only 10 W [5]. Classical systems have made machine learning fundamental in many areas. Machine learning's success is driven by optimization techniques. These algorithms learn parameters from data through optimization model [2]. Modern processors offer advanced learning capabilities [6]. However, traditional methods are energy-intensive and costly, making hardware-based neural networks important [7]. Faster, more efficient optimization methods enable machine learning on larger datasets [8]. These methods enhance performance, leading to more accurate predictions and faster processing.

In conclusion, machine learning and deep learning algorithms have advanced significantly with increasing processing power and datasets [9]. However, high computational power can create efficiency and performance issues. To overcome these problems, hardware-based neuromorphic architectures and memristive devices are needed [10]. Research has shown that memristive devices are efficient for brain-based computing [11], [12], [13].

II. RELATED STUDIES

Information processing hardware is crucial for the impact of AI in our daily lives. However, despite advancements, the energy consumption of hardware running deep neural networks remains much higher than that of the biological brain [14], [15]. Perceptrons, being single-layer structures, have limited functionality. To overcome this limitation, multilayer perceptrons (MLPs) have been developed. As shown in Figure 2(a), MLPs are feed-forward neural networks that map input vectors to output vectors and are effective for complex tasks requiring higher accuracy [16], [17]. Figure 2(b) shows the flow diagram of the MLP, which is mathematically represented in Figure 2(a). In Figure 3, the MLP shown in Figure 2(a) is presented as a physical analogy with electrical circuit components. The input layer processes the inputs received from the dataset, and these signals pass through circuit elements such as resistors and amplifiers via the hidden layers. The processed signals that reach the output layer provide the final results. The circuit elements undertake the tasks of signal processing and transformation. These algorithms are inefficient with the von Neumann architecture, which is not suitable for parallelism and continuous information exchange [18], [19], [20]. Hardware-based neural networks or neuromorphic processors mimic neurons and enhance performance [21], [22]. Neuromorphic systems process in artificial synapses, where the signal transmission coefficient acts as the "weight" [23]. Fig. 4(a) shows a biological neuron, and Fig. 4(b) illustrates an artificial neuron and their principles. In the field of neuromorphic computing, the need for fast data processing, parallel processing capacity, and energy-efficient hardware has reached its peak. Considering this, a study [24] proposed the use of Spiking Neural Networks (SNNs) based on CMOS nanoscale memristive neuronal applications to meet all computation and processing needs [25]. This approach can enable neural networks to work more efficiently and similarly to the biological brain. Oh et al. designed a compact and energy-efficient Mott activation neuron using vanadium dioxide (VO2) [26]. This neuron has been effectively integrated with a conductive bridge random access memory (CBRAM) crossbar array. Another study illustrated the on-site training of a five-layer convolutional neural network utilizing a combination of transistors and memristor arrays to adapt to the MNIST and CIFAR-10 datasets non-ideal characteristics for classification [27]. Another study [28] introduced a simple, single-layer, and nanoscale *TiO2*-based artificial synaptic device aimed at increasing energy efficiency and parallel processing capacity for neuromorphic computing applications. Additionally, the principles of operation and performance analyses of optimization algorithms in the deep learning and machine learning literature are

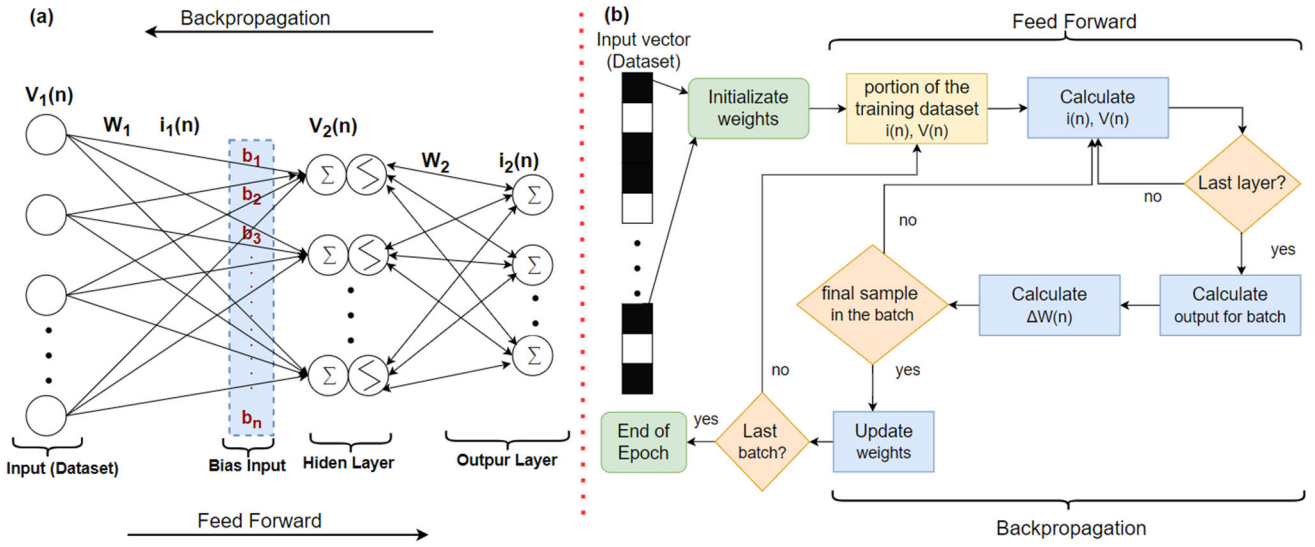

FIGURE 2. (a) Mathematical model of MLP. (b) Illustration of flow diagram explaining MLP in detail.

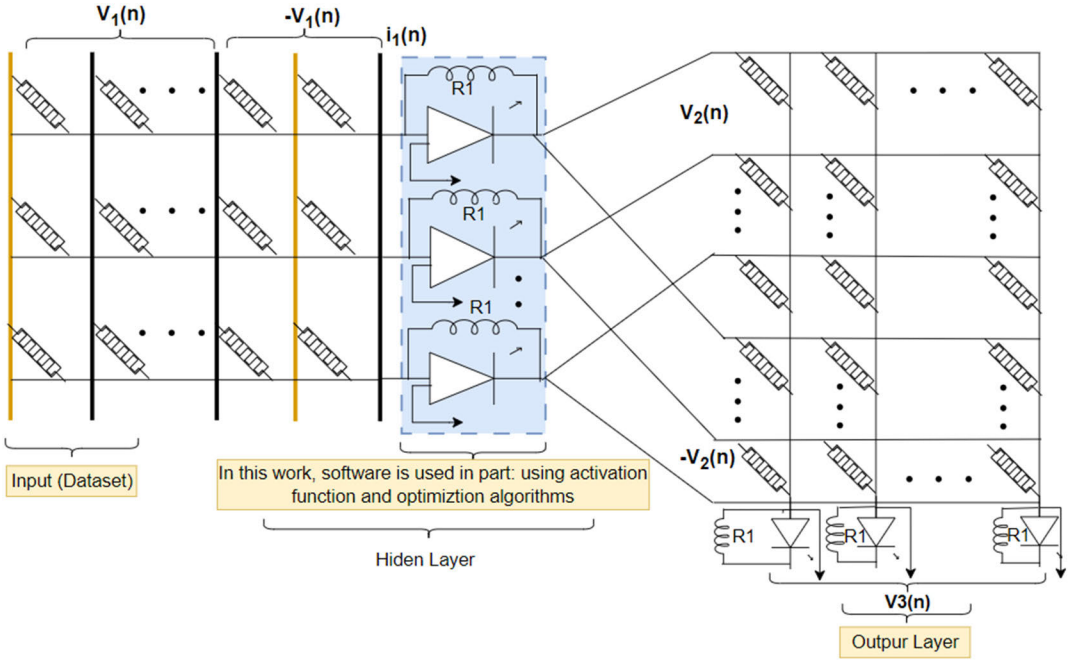

FIGURE 3. Schematic representation of the Memristor-based implementation of the MLP.

thoroughly examined [29]. A survey of numerical optimization algorithms within the scope of machine learning applications is provided [30]. Similarly, a general summary of gradient optimization algorithms is provided [31]. Another study [32] achieved 73% accuracy in neuromorphic systems using a Ag:a-Si material-based memristive synaptic device. In another study [33], 84% accuracy was obtained in pattern recognition tasks on the MNIST dataset using HfZrOx-based synaptic devices. Another study [34] proposed a sign-based Stochastic Gradient Descent (SGD) algorithm using a parallel update approach to accelerate the learning process in synaptic units. This study examines the use and performance analysis of SGD and its variants in a neural network based on TiO2 synaptic devices. This performance analysis was simulated with NeuroSim [35] and applied to neural network model based on a nanoscale memristive artificial synaptic device with some modifications.

AI chips are advanced silicon processors specifically designed for machine learning tasks [36]. AI significantly contributes to mitigating or eliminating risks to human life across various industries [37]. As the amount of data continues to grow exponentially, the demand for more efficient systems to tackle complex computational and mathematical challenges becomes increasingly clear. In

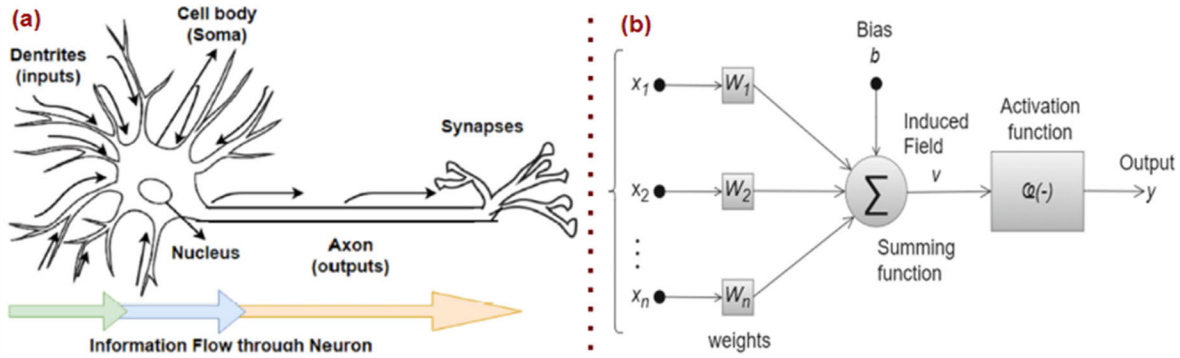

FIGURE 4. (a) Biological Neuron-information flow through and (b) Neuron artificial neuron.

TABLE 1. Comparison of CMOS neurons with Memristor-based neurons.

Reference	Device Type/Model	Configuration	Energy/Spike
[36]	CMOS	18–20 transistor	0.3–1.5 µW, 2850 pJ/spike
[37]	CMOS	90 transistor	163.4 µW
[38]	CMOS	16 transistor	40.2 pW, 0.4 pJ/spike
[39]	CMOS	9 transistor	4 fJ/spike
[40]	CMOS	14 transistor	4.3 pJ/spike
[41]	CMOS	1 transistor emulator + 3 transistor	60–110 µW
[42]	CMOS + Memristor	Memristor emulator (8 transistor)	14 fJ–1.4 pJ/spike
[43]	CMOS + Memristor	1memristor+1magnetic coupling+CMOS circuit	3.3 µW, 150 pJ/junction
[44]	Memristor	~6 x 4	~5 (nJ/spike)
[45]	Memristor	5 x 5	~700 (nJ/spike)
[46]	Memristor	<10 × 10	16 (fJ/spike)
[47]	Memristor	<10 × 10	16 (pJ/spike)

response, leading technology companies have focused their efforts on advancing AI chips and expanding their practical applications.

Various studies have focused on the power usage issues of CMOS neurons. Table 1 compares the power consumption of CMOS and memristor-based neurons, highlighting their energy efficiency performance.

III. CLASSIFICATION OF OPTIMIZATION ALGORITHMS

Optimization algorithms are classified based on objectives such as accuracy, speed, and generalization performance [48]. Accuracy refers to the model's ability to make correct predictions; speed denotes the processing time; and generalization performance indicates the model's adaptability to new data. Generalization performance is critical in optimization algorithms, and the ability to easily update parameters makes these algorithms more practical. Additionally, energy consumption performance on large datasets is an important factor. Different components require various optimization techniques and strategies. In this study, Gradient Descent (GD) and its variations have been used in machine learning and deep learning applications, and these algorithms are briefly explained in the following sections. This study compares eight optimization methods in the field of neuromorphic computing using a TiO_2 synaptic-based device.

A. GRADIENT DESCENT AND VARIANTS

Training artificial neural networks is an optimization process aimed at determining the network parameters that minimize the loss function. These learning models require large datasets and numerous model parameters to be adjusted. Finding the most suitable method for processing large datasets presents a significant challenge. Gradient descent is a crucial method for reducing loss functions by regularly updating the model parameters in the opposite direction of the gradient [48]. Various versions of gradient descent offer different algorithmic strategies to address the challenges encountered during the optimization process. There are also many models used in machine learning. SGD [50] is a model-based method that creates a model by optimising the parameters and makes future predictions with this model. It provides a fast and effective optimisation in larger data sets. kNN [51], on the other hand, does not learn a model, but makes predictions directly on the training data. It is a simple and understandable method in itself, but its slowness in the test phase and memory usage may be a problem in large data sets. For example, kNN method was used in this study

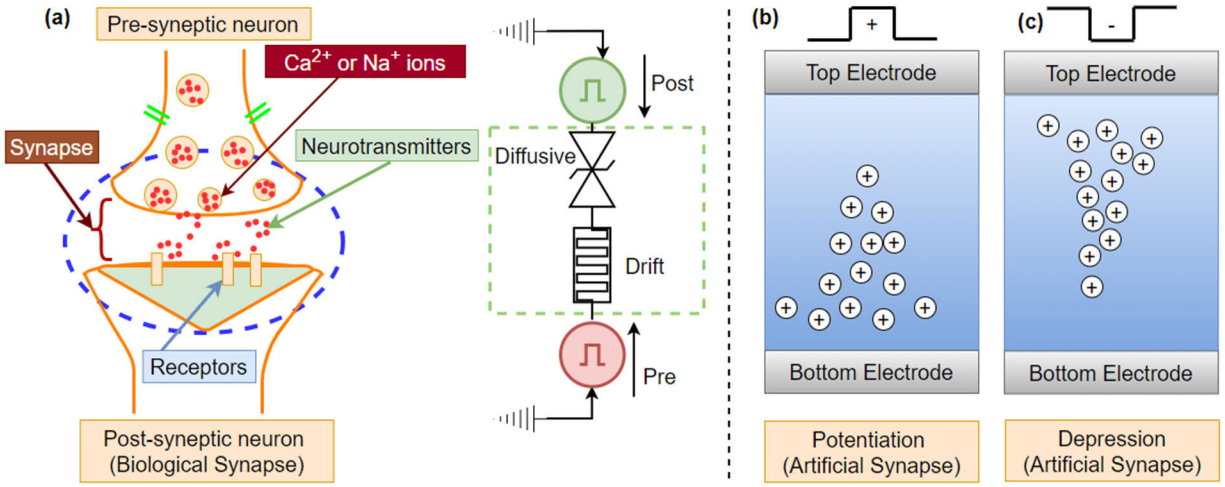

FIGURE 5. Schematic representation of biological and artificial synapses. Just like a biological synapse with neurotransmitters and receptors, PRE and POST spikes are utilized here as well.

[49]. Therefore, SGD method is a more suitable model for our study. Below is a summary of gradient descent and its variations.

1) STOCHASTIC GRADIENT DESCENT (SGD)

SGD [52] has been frequently recommended for solving machine learning and deep learning optimization problems because it is simple, understandable, and effective in AI models [53]. The mathematical equation for SGD is shown in Equation 1.

$$\theta_{t+1} = \theta_t - \alpha \nabla F_i(\theta_t) \tag{1}$$

In Equation 1 above; θ_t is the weight vector at step t. α is the learning rate, determining how much we will move at each step. $F_i(\theta_t)$ is the gradient vector, the gradient of the loss function for the randomly selected subset i. In this equation, a random subset of data is selected in every iteration, and the gradient is computed for the examples in this subset. This gradient is subtracted from the current value of the weight vector to update the weight vector.

2) ADAM (ADAPTIVE MOMENT ESTIMATION)

Adam is a widely used optimization algorithm in machine learning and deep learning. It determines adaptive learning rates using estimates of the first and second moments of the gradients. This algorithm combines the principles of both Momentum and RMSprop optimization algorithms [54]. The set of mathematical equations for the Adam optimization algorithm is shown sequentially in Equations (2), (3), (4), (5), and (6).

$$m_{t+1} = \beta_1 m_t + (1-\beta_1) g_t \tag{2}$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2) g_t^2 \tag{3}$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_2^{t+1}} \tag{4}$$

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1-\beta_2^{t+1}} \tag{5}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{m_{t+1}}{\sqrt{\hat{v}_{t+1}} + \in} \tag{6}$$

The parameters in the mathematical equations for the Adam optimizer are as follows: m_t and v_t denotes the estimates of the first and second moments, respectively. g_t is the gradient vector. β_1 and β_2 are parameters. α is the learning rate. $\in$ is a very small number used to prevent division by zero errors. These equations update moment estimates and adjust weights, allowing Adam to handle diverse problems and automatically adjust the learning rate.

3) RMSPROP (ROOT MEAN SQUARE PROPAGATION)

RMSprop [55] is an algorithm that modifies AdaGrad. RMSprop was developed to address the problem of diminishing learning rates encountered in AdaGrad. This optimization technique uses the running average of the squared gradients to adaptively adjust the individual parameter learning rates. The mathematical equation for the RMSprop optimization algorithm is presented in Equation 7.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \in} \odot g_t \tag{7}$$

The parameters in the mathematical equation of the RMSprop optimization algorithm are as follows: θ_t: model parameters at step t, g_t: gradient at step t, η: learning rate, v_t: exponentially weighted moving average of the squared gradients, $\in$: a very small number to prevent division by zero errors, and $\odot$: element-wise multiplication. In this equation, a separate learning rate is calculated for each parameter. The running average of the squares of the old gradients is updated with the square of the current gradient. This dynamically adjusts the learning rates based on the magnitudes of the gradients.

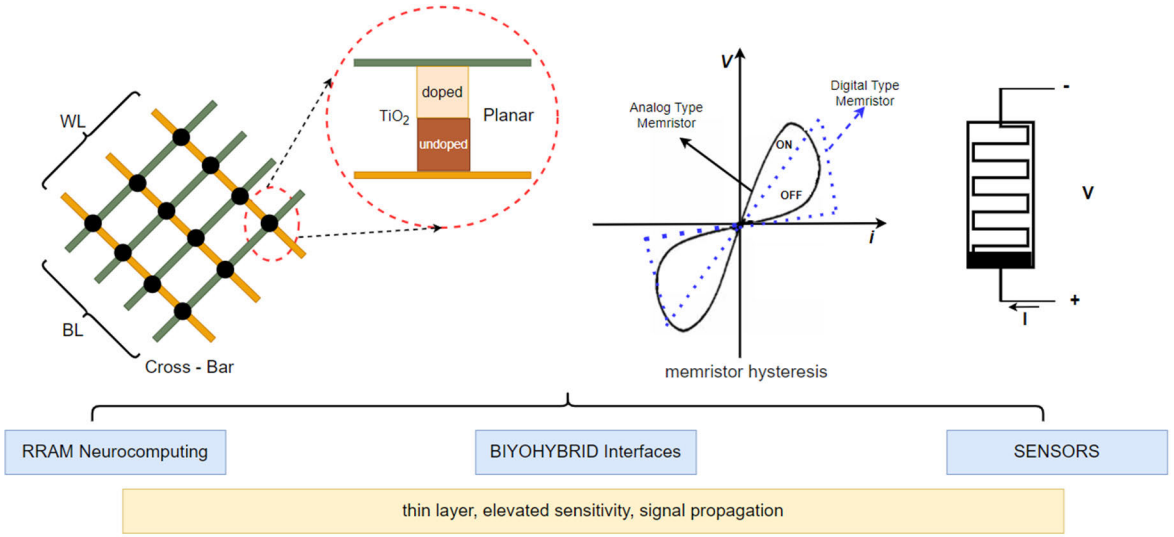

FIGURE 6. Technological advancements and applications of TiO2-based Memristive devices.

4) ADAGRAD (ADAPTIVE GRADIENT ALGORITHM)

AdaGrad optimizes the learning rates of individual parameters based on their recent gradients [56]. Parameters with large derivatives experience a rapid decrease in learning rate, while those with small derivatives decrease less significantly. This is done by using past squared values of the gradients. It is particularly useful for sparse data or data with features on different scales. The mathematical equation for the AdaGrad optimization algorithm is shown in Equation 8.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{8}$$

The parameters in the mathematical equations for the AdaGrad optimizer are as follows: $\theta_{t+1,i}$: parameter i at step t+1, $\theta_{t,i}$: parameter i at step t, η: learning rate, $G_{t,ii}$: sum of the squares of the previous gradients for parameter i at step t, $g_{t,i}$: gradient for parameter i at step t, and ϵ: a very small number to prevent division by zero errors. In the AdaGrad equation, a separate learning rate is calculated for each parameter. The sum of the squares of the past gradients is updated with the current gradient. Thus, the learning rate increases for infrequently updated parameters, while it decreases for frequently updated ones.

5) ADADELTA

AdaDelta is an adaptation of AdaGrad developed to address the problem of diminishing learning rates [57]. This algorithm eliminates the need for continuously decreasing learning rates and selecting a global learning rate. AdaDelta computes the moving average of squared gradients by utilizing a sliding window of past gradients, giving equal importance to all previous gradients while maintaining a constant window size. The mathematical equations for the AdaDelta optimization algorithm are shown in Equations (9), (10), (11), (12), and (13).

$$\mathrm{RMS}[\mathrm{E}\left[g^2\right]]_t = \sqrt{\mathrm{E}[g^2]_t + \epsilon} \tag{9}$$

$$\mathrm{RMS}[\Delta x^2]_{t-1} = \sqrt{\mathrm{E}[\Delta x^2]_{t-1} + \epsilon} \tag{10}$$

$$\mathrm{update} = -\frac{\mathrm{RMS}[\Delta x]_{t-1}}{\mathrm{RMS}[\mathrm{E}\left[g^2\right]]_t} g_t \tag{11}$$

$$\Delta x_t = \rho \Delta x_{t-1} + (1-\rho)\mathrm{update}^2 \tag{12}$$

$$x_{t+1} = x_t + \mathrm{update} \tag{13}$$

The parameters used in the mathematical equations of the AdaDelta optimization algorithm and their functions and meanings are as follows: $E[g^2]_t$: the anticipated value of the gradient squared at step t, $E[\Delta x^2]_{t-1}$: the expected value of the square of the update amount at step $t-1$, g_t: the gradient value at step t, ρ: a momentum term, ϵ: a very small number to prevent division by zero errors, x_t: the parameter values at step t, and Δx_t: the update amounts at step t. In these equations, an individual learning rate is computed for each parameter. It is updated with the squares of the previous and current gradients. The learning rate increases for infrequently updated parameters, while it decreases for frequently updated ones.

6) NADAM (NESTEROV-ACCELERATED ADAPTIVE MOMENT ESTIMATION)

Nadam adds Nesterov momentum to Adam's update rule. This method calculates the gradient by considering a future position in the direction of the momentum. Thus, it aims to increase convergence speed and improve model quality. The mathematical equations for the Nadam optimization algorithm are shown in Equations (14), (15), (16), (17), and (18).

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{14}$$

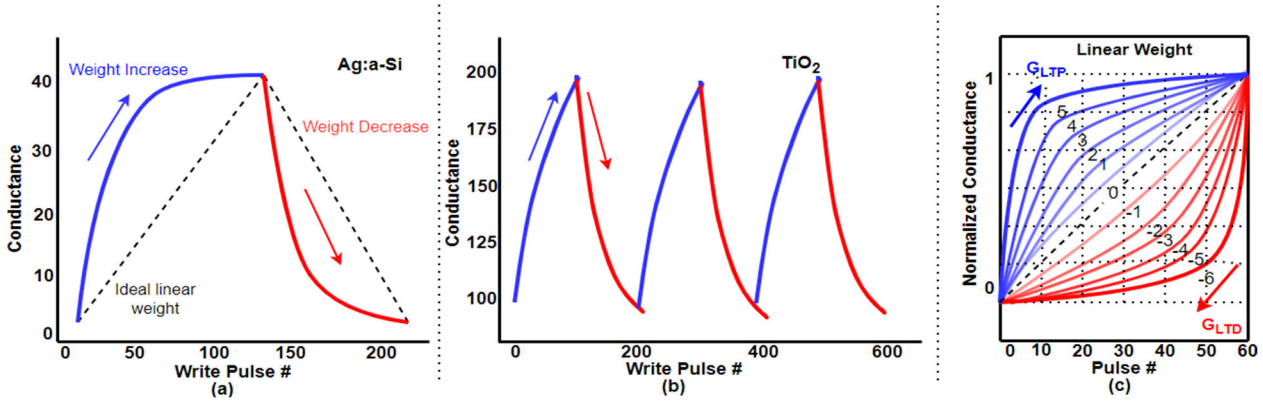

FIGURE 7. Nonlinear weight update for Ag:a-Si, (b) Nonlinear weight update for TiO2, (c) Schematic representation of the analog eNVM device behavioral model with nonlinear weight updates labeled from −6 to 6.

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{15}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_2^t} \tag{16}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{17}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \in}(\beta_1 \hat{m}_t + (1-\beta_1) g_t \tag{18}$$

The parameters in the mathematical equations of the Nadam optimization algorithm are as follows: m_t and v_t: the estimates of the first and second moments at step t, respectively. g_t: the gradient vector. β_1 and β_2: exponential decay rates for the moving averages. η: the learning rate. $\hat{m}_t$ and $\hat{v}_t$: the corrected estimates of the first and second moments. $\in$: a very small number used to prevent division by zero errors. Nadam combines Nesterov Momentum and the Adam algorithms to enhance the concept of previous momentum, thereby reducing the risk of getting stuck in local minima.

7) ADAMAX

AdaMax, developed as a variant of Adam, aims to improve convergence stability by using the infinity norm [58]. In this approach, the L-infinity norm of the exponential moving averages of the gradients is computed and employed instead of the gradient's infinity norm in the Adam algorithm. The mathematical equations for the AdaMax optimization algorithm are shown in Equations (19), (20), and (21).

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{19}$$

$$u_t = \max(\beta_2 u_{t-1}, \|g_t\|_\infty \tag{20}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{1-\beta_1^t} \frac{m_t}{u_t + \in} \tag{21}$$

The parameters in the mathematical equations of the AdaMax optimization algorithm are as follows: θ_t: model parameters at step t. g_t: gradient at step t. η: learning rate. β_1 and β_2: two parameters representing exponential decay rates for the moving averages. m_t: estimate of the first moment at step t. u_t: maximum value selected from the first moment estimate and the infinity norm (maximum absolute value) at step t. $\in$: a very small number used to prevent division by zero errors. AdaMax aims to provide more stable performance by using the infinity norm instead of the second moment, similar to the Adam optimization algorithm. This can lead to better results, especially in deep networks.

8) MOMENTUM

The Momentum method [31] is similar to the concept of physical momentum. The aim is to provide faster movement in the direction of optimization by accounting for the velocity of previous steps with a momentum term. This approach, the information obtained from previous gradients is preserved with a 'momentum' effect, allowing for faster updates each time. The mathematical equations for the Momentum optimization algorithm are shown in Equations (22) and (23).

$$v_{t+1} = \beta_1 v_t + \alpha g_t \tag{22}$$

$$\theta_{t+1} = \beta_1 v_t + \alpha g_t \tag{23}$$

The parameters in the mathematical equations of the Momentum optimization algorithm are as follows: v_{t+1}: represents the momentum and is a weighted average of past gradients. θ_t: model parameters at step t. g_t: gradient at step t. α: learning rate. β_1: it is a hyperparameter that controls the momentum.

IV. DEVICE PROPERTIES AND APPLICATION EXPERIMENTS

A. MATERIAL AND DEVICE PROPERTIES

Neuromorphic computing systems based on memristors are more energy-efficient than software-based neural networks [1]. Memristors, acting as natural analog resistors, perform computations within memory, offering an alternative to the von Neumann architecture. Simulations with NeuroSim are needed to predict and optimize large-scale memristor designs. Binary oxides are crucial for electronic memristors due to their simple production and compatibility with CMOS technology [59]. These memristors are easy to produce, low-cost, and integrate well with existing semiconductor

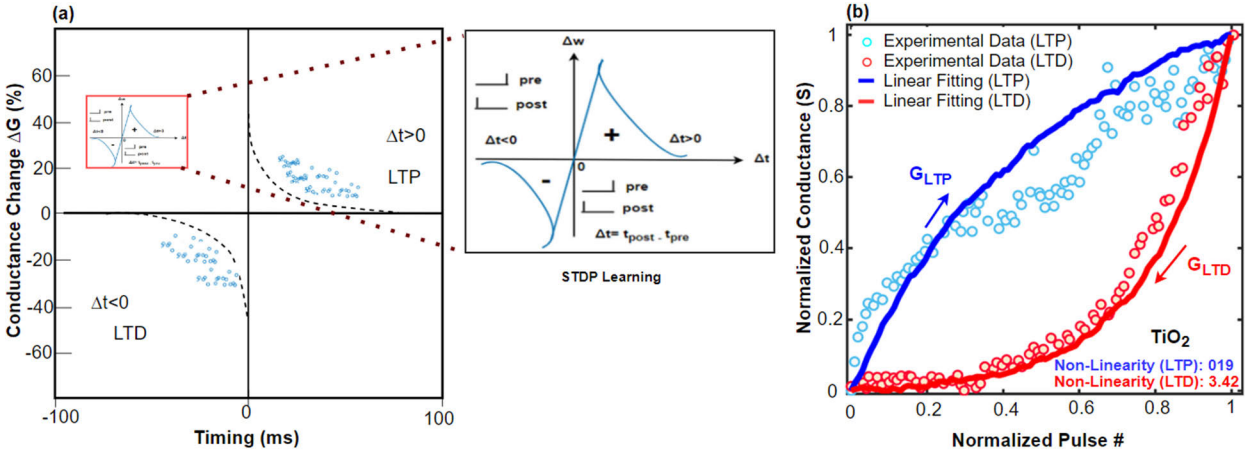

FIGURE 8. Spike-Timing-Dependent Plasticity (STDP) properties of the TiO2-Based synaptic device. (b) The graph illustrating the relationship between normalized pulse count and normalized conductance for TiO2 weight adjustment data.

technologies. Oxides like Hf, Zn, V, Ni, and Ti [60] are used in synaptic nano devices for their gradual resistance switching. This synapse-like behavior makes memristors ideal for neuromorphic computing and neural network applications. There is significant interest in developing nano-electronic devices that mimic synaptic functions [61]. TiO_2 is promising for neuromorphic computing due to its prominence in CMOS manufacturing. TiO_2-based synaptic devices effortlessly incorporate into crossbar architectures for in-memory computation [62]. Fig. 5(a) details the biological synapse counterparts and switching behavior of the TiO_2 memristor. Artificial synapse creation relies on the electrochemical reactions of nanoparticles situated between bipolar electrodes. Voltage application causes ion and atom migration, providing bipolar switching [63]. Fig. 5(c) shows voltage causing nanoparticles to grow, increasing current and transitioning the memristor to a low-resistance state. When voltage is withdrawn, as shown in Fig. 5(b), nanoparticles shrink, reducing current and transitioning the memristor to a high-resistance state. TiO_2-based memristor devices can dynamically change resistance and mimic the functionality of biological synapses. These devices hold great potential for advanced technological applications such as neuromorphic engineering and artificial neural networks.

B. CHARACTERISTICS OF TiO2 SYNAPTIC DEVICE

The development of TiO_2 memristors is linked to high-tech applications like RRAM, biohybrid systems, and sensors. These applications are shown in Fig. 6, reflecting the potential of TiO_2 memristors. TiO_x is among the earliest materials studied for RRAM applications. However, memristors are generally used as resistive switching memory devices that change between ON or OFF states [64]. Memristor-based RRAM devices store data by switching between two states and remain accessible when needed, even when power is off [65]. Some nano-scale metal oxide memristors allow fine adjustments in resistance, enabling precise data processing and storage. This makes them ideal for advanced neuromorphic computing and innovative memory applications.

Theoretically, synaptic weight changes should linearly correlate with the number of write pulses, but real-world devices deviate from this behavior. Conductivity changes rapidly during initial stages of LTP and LTD, gradually reaching saturation. Fig. 7 ((a), (b), and (c)) illustrates these deviations, which affect the accuracy and efficiency of neuromorphic systems. This necessitates detailed characterization and optimization. To model nonlinear weight updates, a device model was developed. MLP + NeuroSim models power usage, training delay, and spatial requirements of neuromorphic systems with analog eNVM synapses, such as the TiO_2 device [66]. Conductance change is related to write pulses (P) and is expressed by the following equations:

$$G_{LTP} = B\left(1 - e^{\left(-\frac{P}{A}\right)}\right) + G_{min} \tag{24}$$

$$G_{LTP} = -B\left(1 - e^{\left(-\frac{P\text{-}P_{max}}{A}\right)}\right) + G_{max} \tag{25}$$

$$B = (G_{max} - G_{min})/(1 - e^{-P_{max}/A}) \tag{26}$$

G_{LTP} and G_{LTD} represent the conductance values for the *LTP* and *LTD* processes, respectively. G_{max} , G_{min}, and P_{max} are values directly derived from experimental results and signify the peak conductance, the lowest conductance, and the maximum pulse count needed for the device to transition between the highest and lowest conductance states, respectively. The parameter A controls the nonlinear nature of the weight update behavior and can be either positive (blue) or negative (red). In Fig. 7(a) and Fig. 7(b), the magnitudes of the A values for *LTP* and *LTD* are the same, but their signs are different. B is defined as a function of A and is used to fit the functions within the scope of G_{max} , G_{min}, and P_{max}. Using Equation ((24), (25) and (26)), various non-linear weight augmentation (blue) and weight reduction (red) behaviors can be obtained by adjusting the value of A, as shown in Fig. 7 (c). Each nonlinear curve is labeled

with nonlinearity values ranging from +6 to −6 [35]. Upon examining Equation (24) and (25), it can be proven that they are equivalent except for the sign of A. Therefore, only Equation (24) will be used for both nonlinear LTP and LTD weight updates. Unlike Fig. 8 ((a) and (b)), all LTD curves are mirrored and shifted horizontally to ensure the curves start from pulse number 0 for simpler formulation.

In biological neural networks, synaptic weight is modulated by STDP, measured by cell conductance [67]. These mimics learning in artificial neural networks. The TiO_2 device's synaptic weight change is determined by conductance changes from spike timing pulses, leading to potentiation or depression based on pre- and post-synaptic spike timing. TiO_2-based memristive devices can mimic biological synapses' functionality. The amount and direction of synaptic weight change relate to the relative conductance change between spike timings, reflecting STDP principles and neural network learning mechanisms. Equation (27) defines the relative conductance change mathematically as follows:

$$\Delta G = (G_{after} - G_{before})/G_{before} \tag{27}$$

According to this equation, Δ G represents the synaptic weight change, while, $\boldsymbol{G_{after}}$ and $\boldsymbol{G_{before}}$ represent the conductance values following and preceding the activation of the pre-post spikes, in that order [68]. This ratio determines synaptic modulation effectiveness and direction. Fig. 8(a) presents the spike-timing dependent plasticity (STDP) characteristics of the TiO_2-based synaptic device, illustrating how the timing between pre- and post-synaptic spikes influences changes in synaptic conductance. The STDP mechanism plays a key role in modulating synaptic strength, either reinforcing or weakening connections. TiO_2 memristors show promise in emulating the behavior of biological synapses and their applicability in neuromorphic systems. Fig. 8(b) depicts the device's performance across various timing intervals and demonstrates the dynamics of synaptic weight modification.

MLP + NeuroSim is widely used for comparing online learning processes [66]. It simulates metrics like power consumption, training latency, and area utilization of neuromorphic hardware with TiO_2-based analog eNVM synapses. Normalization parameters were determined by re-arranging and normalizing experimental weight data. LTP and LTD data were mirrored and fitted using the NeuroSim nonlinear-fitting script, yielding nonlinearity values of 0.19 and 3.42, respectively. These values were used in the simulator, and the results are shown in Fig. 8(b). These simulations help optimize the performance of TiO_2-based neuromorphic hardware.

C. DATASETS AND EXPERIMENTAL CONFIGURATIONS

The performance of the optimisation algorithms was evaluated on datasets called MNIST [69] and CIFAR-10 [70]. The MNIST dataset is a widely known and used reference dataset developed for the purpose of recognizing handwritten digits. The dataset includes 60,000 training samples and 10,000 test samples. Training samples develop the model, while test samples evaluate its performance. The MNIST dataset is standard for recognizing handwritten digits, ideal for comparing optimization algorithms. A new dataset of computer-generated characters was created by digitizing text images into 28 × 28 pixels, shown in Fig. 9(a) and Fig. 9(b). Input vectors for the neural network were constructed by combining perpendicular and parallel vectors of the images, resulting in a single input vector with 56 features for both computer-generated and handwritten characters. This provides the information richness needed for effective recognition. CIFAR-10 is an image dataset widely used in machine learning and computer vision research, consisting of small colour images categorized into 10 different classes. The dataset contains 60,000 32 × 32 colour (RGB) images. These images are divided into two main groups: train (training) and test: 50.000 training images and 10.000 test images.

D. PROPOSED APPROACH

This study presents the implementation of neural networks in hardware using TiO_2-based nano-synaptic devices and evaluates this method in terms of accuracy, area usage, and energy consumption by utilizing various optimization algorithms. The proposed hardware-based model possesses features recognition and classification capabilities. This application maps inputs to outputs using feedforward (FF) and backpropagation (BP) processes to recognize computer-generated and handwritten digits. Fig. 10 shows the overall structure. The training process consists of two components: FF and BP. In FF, input data enters the input layer, passes through hidden layers via weighted sums and activation functions, and reaches the output layer [71]. The output is compared with the correct label to calculate the prediction error. In BP, this error propagates backward, adjusting weights to minimize the error using Adadelta, AdaGrad, Adam, AdaMax, Momentum, Nadam, RMSprop, and SGD optimization methods. Unlike traditional gradient descent, BP updates weights for individual images immediately after FF. The testing process involves only FF phase, using learned weights to make predictions on new data, evaluating the model's performance and generalization ability.

E. IMPLEMENTATION OF MEMRISTOR-BASED NEURAL NETWORK USING TiO2 SYNAPTIC-BASED DEVICE

Recent studies highlight memristive synaptic devices in hardware-based neural networks for better brain function mimicry and compact, scalable structures [13], [72]. These devices improve performance in digit recognition and image classification using advanced algorithms and high-quality datasets, offering reduced power consumption and increased processing speed, ideal for AI and machine learning innovations. This study focuses on optimizing speed, energy efficiency, and accuracy using various algorithms. Address-

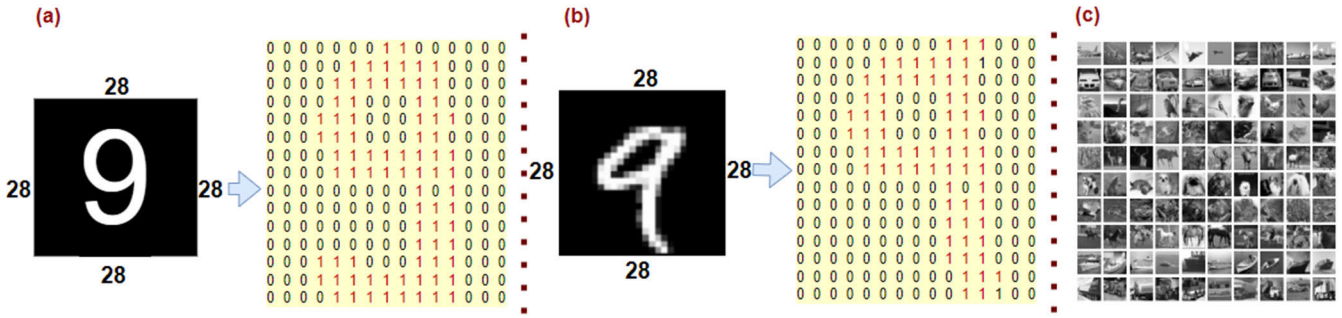

FIGURE 9. (a) Binarisation of a computer written digit (b) Binarisation of a hand written digit (c) CIFAR-10 dataset.

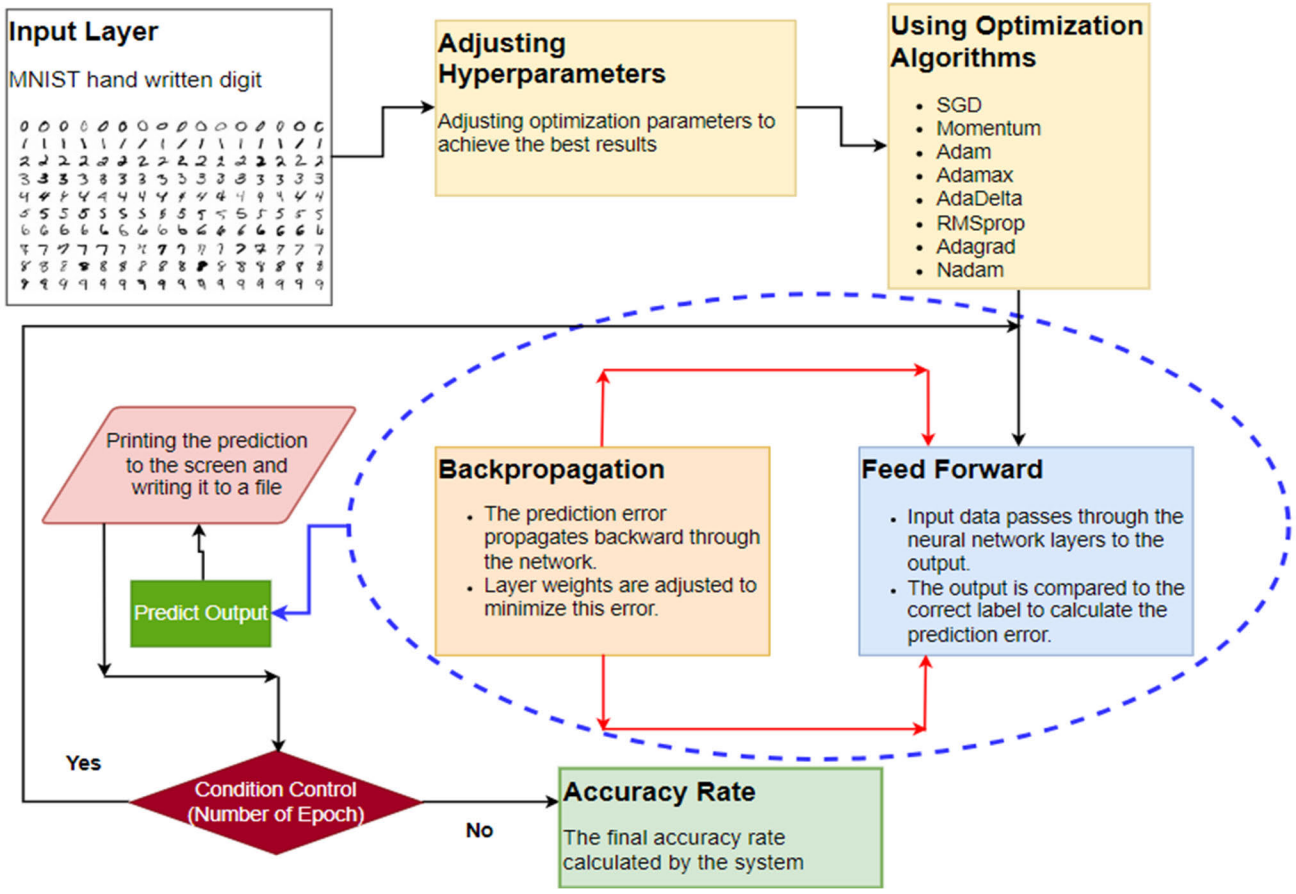

FIGURE 10. The overall structure of the developed application.

ing the gap in *TiO*2-based nano-synaptic devices' properties and learning performances, this study details a neuro-inspired hardware solution for neural networks. The *TiO*2-based device, using feedforward (FF) and backpropagation (BP) methods, enhances efficiency and performance through its memristive properties. A simple two-layer multilayer perceptron (MLP) neural network was designed to compare performance and serve as a reference. As shown in Fig. 11(a), the network comprises input, hidden, and output layers, with each neuron in one layer being fully connected to all neurons in the subsequent layer, enhancing the network's ability to learn complex data patterns. This structure offers high flexibility and computational power, with connections represented by weighted synapses. The matrix representing the connection weights between the input and hidden layers are referred to as W_{IH}, while the matrix representing the weights of the connections between the hidden and output layers is referred to as W_{HO}. The MNIST handwritten digits dataset, with images resized to 28x28 pixels, was used as input data. The network's default topology includes an input layer with 784 neurons, a hidden layer with 128 neurons, and an output layer with 10 neurons, corresponding to

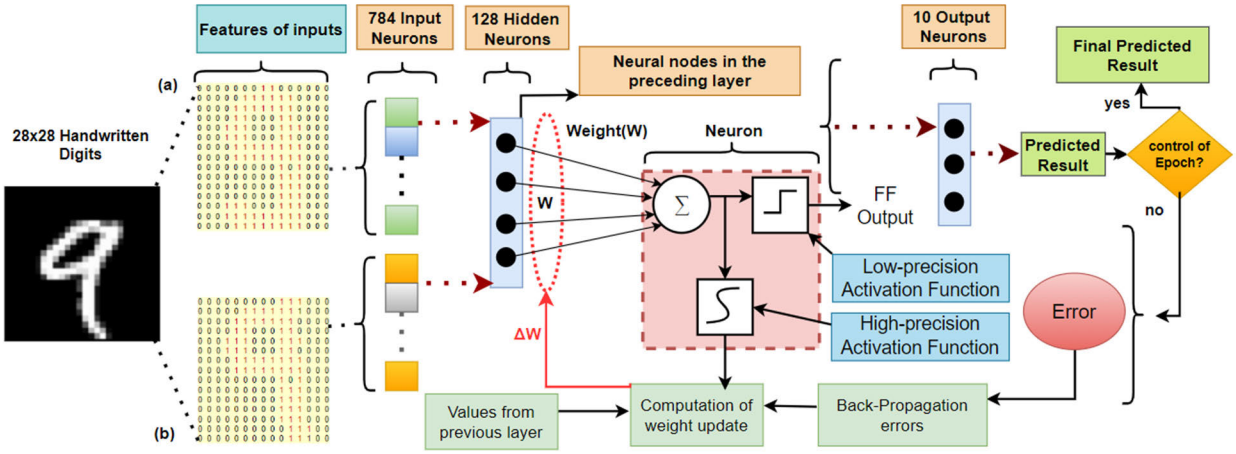

FIGURE 11. MLP neurol network. The architecture of the employed BPNN and its neurons. (a) Binary matrix (28)-by-28) representing a computer-generated digit and (b) handwritten digit.

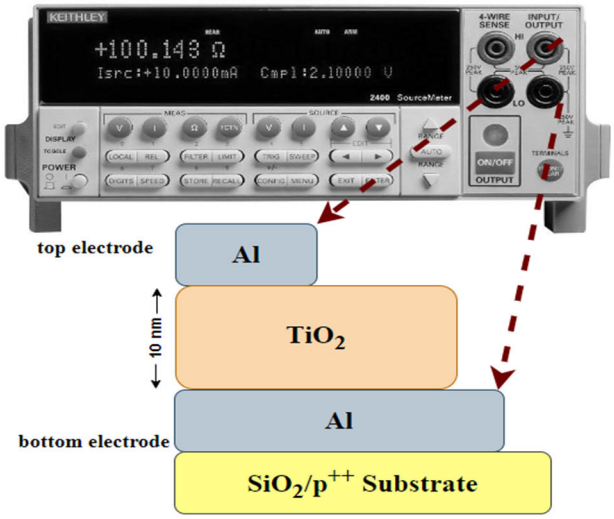

FIGURE 12. SMU/Pulse Source. The layout of the TiO2 synaptic device and experimental setups.

the 10 digits classes. This setup enables the network to process each image and determine the appropriate digit class. Changes to these parameters may require new optimizations. Fig. 11(b) illustrates a neuron node that computes the weighted sum from incoming synapses and applies a 1-bit low-precision activation function for offline classification. However, higher precision is necessary for weight updates during backpropagation to correct small errors.

Optimization methods were applied to the designed *TiO*2 synaptic device [28]. The synaptic weight change results, which occurred during the learning process on this device, were collected using a specially designed probe station. The synaptic analyses of the *TiO*2 device were finalized using specially tailored software. The primary energy consumption in memristive eNVM synaptic cores is primarily due to static power utilization, as opposed to the dynamic power consumption seen in conventional memory types. Current flow through the memristive synapses contributes to the energy expenditure. Total energy use is the sum of the energy consumed by both synaptic nuclei and peripheral circuits. The peripheral circuit energy consumption was calculated using the predictive technology model (PTM) for 32 nm node technology [73]. The energy consumption for the neural network application on the computer was measured using a system with i7-10750H CPU (2.90 GHz) and 8 GB RAM. The energy analysis followed the guidelines given in [74] and the DeLight [75] tool shown in Fig. 12 was used to estimate the energy usage of the neural network application on the computer.

Fig. 13 illustrates the circuit block diagram for the hardware-based implementation of the two-layer MLP neural network. In this implementation, the weighted sum calculations are performed through synaptic cores. However, the weights utilized in a standard synaptic array can only take non-negative values, with a range of $W_H = 0 \sim 1$, whereas in the neural network algorithm, both positive and negative weight values are present, i.e., $W_A = -1 \sim 1$. This situation presents certain challenges in hardware design and requires special solutions to represent negative weights. In this implementation, an algorithmic structure is used to convert the weights from the range $W_A = -1 \sim 1$ to the range $W_H = 0 \sim 1$. The weighted sum calculation in the algorithm is expressed as follows:

$$\mathrm{W_A V} = 2\,(\mathrm{W_H} - 0.5\mathrm{J})\,V = 2\mathrm{W_H V} - \mathrm{JV}) \qquad (28)$$

in this expression, V represents the input vector, and J is a matrix of the same size as $\mathrm{W_A}$ and $\mathrm{W_H}$, with all elements equal to one. In Equation (4), $\mathrm{W_H V}$ represents the result of the weighted sum obtained from the synaptic core. Therefore, $\mathrm{W_A}$ is converted from the range $(-1 \sim 1)$ to the range of $\mathrm{W_H}$ $(-1 \sim 0)$. In summary, Fig. 13 shows the architecture of a neural network. A portion of the MNIST and CIFAR-10 datasets is used as input vector and processed through synaptic nuclei ($\mathrm{W_{IH}}$ and $\mathrm{W_{HO}}$) to predict the output vector. In the intermediate layer, the weights are set by hardware control logic and processed by MSB (most

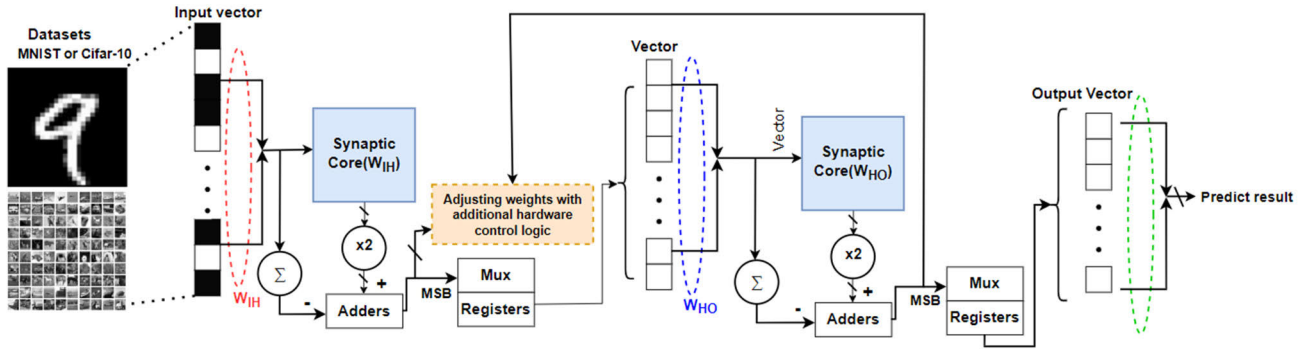

FIGURE 13. Block diagram of the circuit for implementing the 2-layer MLP network in hardware.

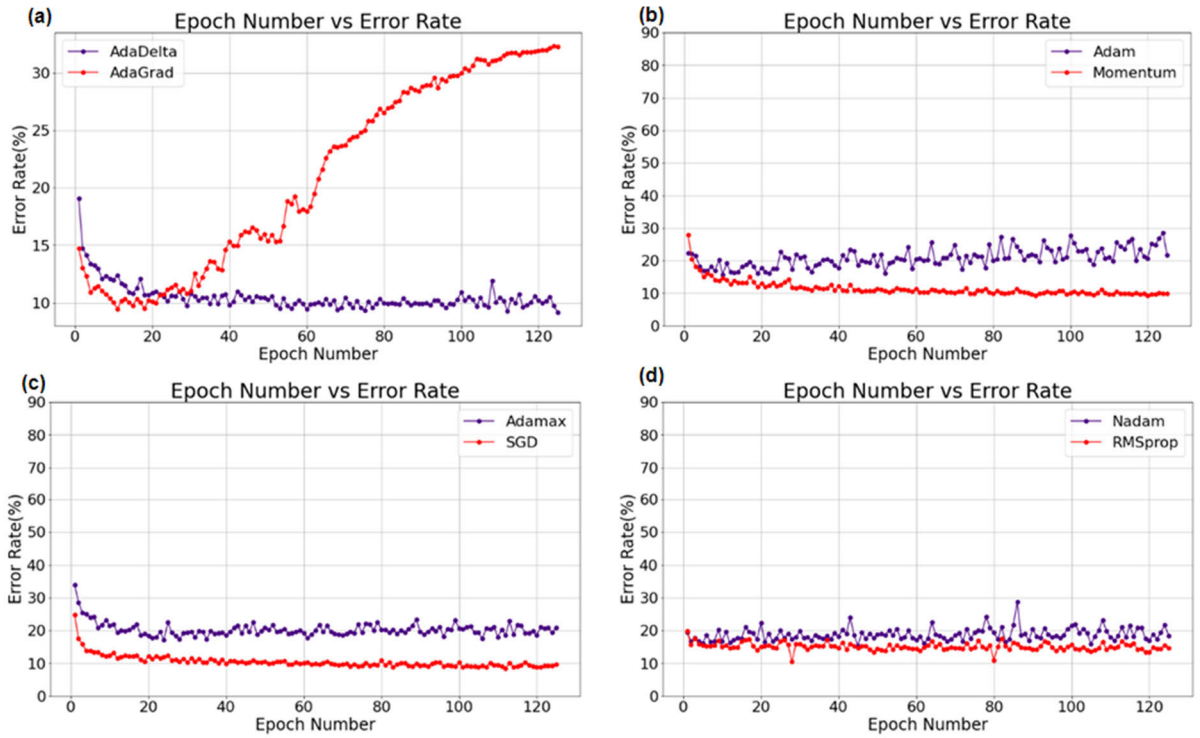

FIGURE 14. Comparative: (a) Error Rate graphs for AdaDelta-AdaGrad optimization algorithms, (b) Error rate graphs for Adam-momentum optimization algorithms, (c) Error rate graphs for Adamax-SGD optimization algorithms, and (d) Error rate graphs for Nadam-RMSprop optimization algorithms.

significant bit). As a result, the output layer provides the prediction. Fig. 13 illustrates the hardware implementation of neural networks.

V. EVALUATION METRICS AND OUTCOMES

The key performance metrics for evaluating hardware-based learning include accuracy, test error rate, precision, and specificity. Accuracy measures the proportion of correct predictions, while the test error rate indicates misclassifications. Precision evaluates the model's capability to accurately identify true positives, while specificity measures its accuracy in recognizing true negatives [76]. This study focuses on learning in optimization algorithms, starting with calculating the accuracy metric for each method.

Accuracy plays a critical role in evaluating model performance and, alongside other metrics, provides a comprehensive assessment of overall effectiveness. These metrics form the basis of this research, offering insights into model performance across datasets and facilitating comparative analyses, thus contributing to more efficient and accurate model development. Below are the mathematical equations for these metrics:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{29}$$

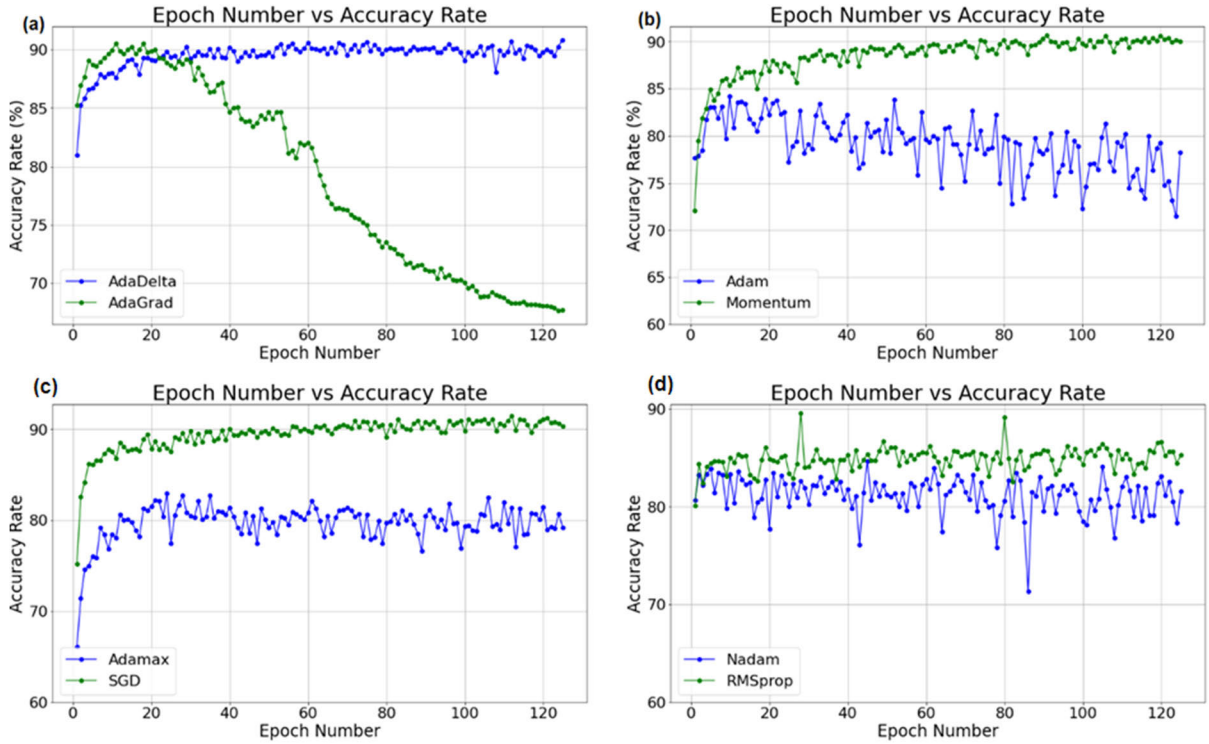

FIGURE 15. Comparative: (a) Accuracy graphs for AdaDelta-AdaGrad optimization algorithms, (b) Accuracy graphs for Adam-momentum optimization algorithms, (c) Accuracy graphs for Adamax-SGD optimization algorithms, and (d) Accuracy graphs for Nadam-RMSprop optimization algorithms.

$$\text{Error Rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{30}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{31}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{32}$$

the terms used in calculating these metrics are defined as follows: TP (true positive) refers to the number of cases where handwritten digits are correctly recognized. FP (false positive) indicates the number of cases where digits are incorrectly recognized as correct. TN (true negative) denotes the number of cases where incorrect digits are correctly identified as incorrect. FN (false negative) represents the number of cases where digits that should have been recognized as correct are incorrectly identified. These terms are used to evaluate the model's prediction performance in detail, each having its unique meaning and importance.

VI. RESULTS AND DISCUSSION

This paper employed a neural network (NN) model utilizing TiO_2 synaptic devices for handwritten digit recognition. The model was trained and tested on the MNIST dataset, which comprises 60,000 training samples and 10,000 test samples of digits ranging from 0 to 9. Besides the MNIST dataset, a neural network (NN) model using TiO_2 synaptic devices for image classification is used in this paper. The model was trained and tested on the CIFAR-10 dataset which includes 50,000 training samples and 10,000 test samples across 10 classes. The circuit-level performance was analyzed using Neurosim, evaluating metrics such as energy consumption, latency, and area requirements. Various optimization methods were applied to the TiO_2-based machine learning model, including Adadelta, AdaGrad, Adam, AdaMax, Momentum, Nadam, RMSprop, and SGD. Each method updates weights differently to improve accuracy. Experimental results showed accuracy rates of 89.48% (Adadelta), 79.00% (AdaGrad), 79.13% (Adam), 79.68% (AdaMax), 88.55% (Momentum), 81.20% (Nadam), 84.91% (RMSprop), and 89.47% (SGD). Fig. 16(a) and Fig. 16(b) presents these accuracy rates and error rates graphs. These findings emphasize the critical impact of optimization methods on model performance and the importance of selecting the right optimization strategies for training neural networks. Fig. 14(a) compares AdaDelta and AdaGrad, Fig. 14(b) compares Adam and Momentum, Fig. 14(c) compares AdaMax and SGD, and Fig. 14(d) compares Nadam and RMSprop in terms of accuracy and epoch numbers, grouped in pairs. As can be seen from the graphs, the epoch numbers are same, but the accuracy values show significant differences. The data presented graphically include accuracy and epoch values obtained using optimization algorithms. It was observed that the Adadelta optimization algorithm achieved higher accuracy values compared to the other algorithms. These findings

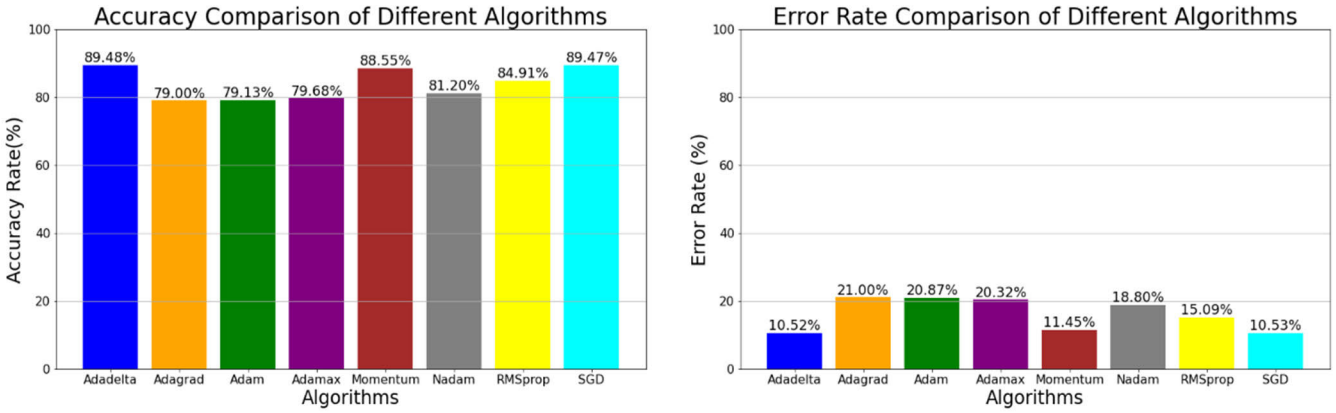

FIGURE 16. This figure compares the accuracy rates of different optimization algorithms. AdaDelta (89.48%), SGD (89.47%), and Momentum (88.55%) have the highest accuracy, while AdaGrad (79.00%) shows the lowest accuracy.

TABLE 2. Comparative performance results of various optimization models with MNIST dataset.

Optimization Name	AdaDelta	AdaGrad	Adam	AdaMax	Momentum	Nadam	SGD	RMSprop	Traditional Computer
Energy Consumption (J)	0.03049	0.01125	0.24407	0.01153	0.14311	0.1126	0.02762	0.16036	4.275×10^3
Accuracy (%)	89.48	79.00	79.13	79.68	88.55	81.20	89.47	84.91	96.95
Training Latency (epoch/s)	804.58 (s)	953.69 (s)	717.19 (s)	851.18 (s)	737.09 (s)	828.19 (s)	639.34 (s)	847.79 (s)	140.25 (s)

TABLE 3. Comparative accuracy performance results of various optimisaPtion models with MNIST and CIFAR dataset.

Optimization Name and Dataset	AdaDelta	AdaGrad	Adam	AdaMax	Momentum	Nadam	SGD	RMSprop	Traditional Computer
Accuracy (%) CIFAR-10	92.51	82.08	83.10	81.76	91.25	82.45	90.21	88.11	96.95
Accuracy (%) MNIST	89.48	79.00	79.13	79.68	88.55	81.20	89.47	84.91	

provide important insights into the selection of optimization algorithms to balance accuracy.

Similarly, error rate comparisons are shown in Fig. 15. Fig. 15(a) illustrates the comparison between AdaDelta and AdaGrad, while Fig. 15(b) shows the comparison betweenAdam and Momentum. Fig. 15(c) presents a comparison between AdaMax and SGD, and Fig. 15(d) depicts the comparison between Nadam and RMSprop. These figures evaluate the performance of these optimization algorithms based on error rate and epoch numbers, displayed in pairs. Experimental studies on the MNIST dataset show that the TiO_2-based nano-synaptic device's classification accuracy varies with different optimization algorithms. These findings reveal the interaction between TiO_2-based synaptic devices and optimization techniques, demonstrating the method's broad applicability in neuro-inspired hardware solutions. TiO_2-based synaptic devices were preferred due to their compatibility with existing CMOS technology and widespread use in semiconductor chip production. TiO_2 is cost-effective and suitable for sub-nanometer production, enabling higher-density, energy-efficient neuromorphic devices [77]. TiO_2-based synaptic cores, especially compatible with 10-nanometer and below technologies, promise energy efficiency and scalability. Current findings with 32-nanometer CMOS suggest even greater energy gains and area savings at 10 nanometers and below.

Thus, the learning performance results of the neuromorphic models based on synaptic devices and the optimization methods are compared. Table 3 shows the comparative accuracy performance results of various optimisation models on the MNIST and CIFAR-10 datasets. The table presents the accuracy percentages of popular optimisation algorithms such as AdaDelta, AdaGrad, Adam, AdaMax, Momentum, Nadam, SGD and RMSprop on both datasets. On the CIFAR-10 dataset, the AdaDelta algorithm performed the best with an accuracy of 92.51%, while in the MNIST dataset, the AdaDelta algorithm achieved the highest accuracy of 89.48%.

VII. CONCLUSION

In conclusion, the performance of the neural network based on TiO_2 synaptic devices was thoroughly evaluated using various optimization methods on the MNIST and CIFAR datasets. Different optimization algorithms, including SGD and its variants, were tested, achieving a 90% accuracy rate. The model demonstrated robustness and generalization ability across various optimization methods. This high performance under these optimization methods highlights its adaptability and effectiveness across datasets, indicating its potential for diverse applications. This evaluation confirms the effectiveness of combining TiO_2-based synaptic devices with various optimization algorithms for neuromorphic computing and hardware-based AI applications. These findings guide future studies, contributing to the creation of more energy-efficient and precise neural network models.

REFERENCES

[1] *A Memristor-Based DNN Crossbar Array for Iterative Network Pruning and Quantization*. Accessed: Oct. 21, 2024. [Online]. Available: https://www.researchgate.net/publication/365826477_A_Memristor-Based_DNN_Crossbar_Array_for_Iterative_Network_Pruning_and_Quantization

[2] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020, doi: 10.1109/TCYB.2019.2950779.

[3] S. Kim, H.-D. Kim, and S.-J. Choi, "Impact of synaptic device variations on classification accuracy in a binarized neural network," *Sci. Rep.*, vol. 9, no. 1, p. 15237, Oct. 2019, doi: 10.1038/s41598-019-51814-5.

[4] P. Wang, E. Fan, and P. Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," *Pattern Recognit. Lett.*, vol. 141, pp. 61–67, Jan. 2021, doi: 10.1016/j.patrec.2020.07.042.

[5] J. Zhu, T. Zhang, Y. Yang, and R. Huang, "A comprehensive review on emerging artificial neuromorphic devices," *Appl. Phys. Rev.*, vol. 7, no. 1, Mar. 2020, Art. no. 011312, doi: 10.1063/1.5118217.

[6] A. Biglari and W. Tang, "A review of embedded machine learning based on hardware, application, and sensing scheme," *Sensors*, vol. 23, no. 4, p. 2131, Feb. 2023, doi: 10.3390/s23042131.

[7] G. W. Burr, R. M. Shelby, C. di Nolfo, J. W. Jang, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. Kurdi, and H. Hwang, "Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element," in *IEDM Tech. Dig.*, Dec. 2014, pp. 29, doi: 10.1109/IEDM.2014.7047135.

[8] Z. Cao, B. Sun, G. Zhou, S. Mao, S. Zhu, J. Zhang, C. Ke, Y. Zhao, and J. Shao, "Memristor-based neural networks: A bridge from device to artificial intelligence," *Nanosc. Horizons*, vol. 8, no. 6, pp. 716–745, 2023, doi: 10.1039/d2nh00536k.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[10] M. Yu, T. P. S. Xiang, K. T. N. Chu, B. Amornpaisannon, Y. Tavva, V. P. K. Miriyala, and T. E. Carlson, "A TTFS-based energy and utilization efficient neuromorphic CNN accelerator," *Frontiers Neurosci.*, vol. 17, May 2023, Art. no. 1121592, doi: 10.3389/fnins.2023.1121592.

[11] I. Vourkas and G. Ch. Sirakoulis, "Emerging memristor-based logic circuit design approaches: A review," *IEEE Circuits Syst. Mag.*, vol. 16, no. 3, pp. 15–30, 3rd Quart., 2016, doi: 10.1109/MCAS.2016.2583673.

[12] R. Ptucha, F. Petroski Such, S. Pillai, F. Brockler, V. Singh, and P. Hutkowski, "Intelligent character recognition using fully convolutional neural networks," *Pattern Recognit.*, vol. 88, pp. 604–613, Apr. 2019, doi: 10.1016/j.patcog.2018.12.017.

[13] B. Gökgöz, F. Gül, and T. Aydin, "An overview memristor based hardware accelerators for deep neural network," *Concurrency Comput., Pract. Exper.*, vol. 36, no. 9, pp. 1–22, Apr. 2024, doi: 10.1002/cpe.7997.

[14] D. S. Jeong, K. M. Kim, S. Kim, B. J. Choi, and C. S. Hwang, "Memristors for energy-efficient new computing paradigms," *Adv. Electron. Mater.*, vol. 2, no. 9, Sep. 2016, Art. no. 1600090, doi: 10.1002/aelm.201600090.

[15] Y. Huang, V. Ravichandran, W. Zhao, and Q. Xia, "Towards energy-efficient computing hardware based on memristive nanodevices," *IEEE Nanotechnol. Mag.*, vol. 17, no. 5, pp. 30–38, Oct. 2023, doi: 10.1109/MNANO.2023.3297106.

[16] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans. Neural Netw.*, vol. 1, no. 4, pp. 296–298, Dec. 1990, doi: 10.1109/72.80266.

[17] G. Thimm and E. Fiesler, "High-order and multilayer perceptron initialization," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 349–359, Mar. 1997, doi: 10.1109/72.557673.

[18] S. H. Sung, T. J. Kim, H. Shin, H. Namkung, T. H. Im, H. S. Wang, and K. J. Lee, "Memory-centric neuromorphic computing for unstructured data processing," *Nano Res.*, vol. 14, no. 9, pp. 3126–3142, Sep. 2021, doi: 10.1007/s12274-021-3452-6.

[19] J. Han, S. Yun, S. Lee, J. Yu, and Y. Choi, "A review of artificial spiking neuron devices for neural processing and sensing," *Adv. Funct. Mater.*, vol. 32, no. 33, Aug. 2022, Art. no. 2204102, doi: 10.1002/adfm.202204102.

[20] T. Zanotti, F. M. Puglisi, and P. Pavan, "Energy-efficient non-von Neumann computing architecture supporting multiple computing paradigms for logic and binarized neural networks," *J. Low Power Electron. Appl.*, vol. 11, no. 3, p. 29, Jul. 2021, doi: 10.3390/jlpea11030029.

[21] Y. Liang, L. Lu, Y. Jin, J. Xie, R. Huang, J. Zhang, and W. Lin, "An efficient hardware design for accelerating sparse CNNs with NAS-based models," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 3, pp. 597–613, Mar. 2022, doi: 10.1109/TCAD.2021.3066563.

[22] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, "Analog architectures for neural network acceleration based on non-volatile memory," *Appl. Phys. Rev.*, vol. 7, no. 3, Sep. 2020, Art. no. 031301, doi: 10.1063/1.5143815.

[23] M. Onen, B. A. Butters, E. Toomey, T. Gokmen, and K. K. Berggren, "Design and characterization of superconducting nanowire-based processors for acceleration of deep neural network training," *Nanotechnology*, vol. 31, no. 2, Jan. 2020, Art. no. 025204, doi: 10.1088/1361-6528/ab47bc.

[24] J. Gupta and D. Koppad, "A survey on memristor and CMOS based spiking neural networks," in *Proc. 2nd Int. Conf. Inventive Res. Comput. Appl. (ICIRCA)*, Jul. 2020, pp. 1052–1058, doi: 10.1109/ICIRCA48905.2020.9183111.

[25] S. Shin, K. Kim, and S. S. Kang, "Memristor macromodel and its application to neuronal spike generation," in *Proc. Eur. Conf. Circuit Theory Design (ECCTD)*, Sep. 2013, pp. 1–4, doi: 10.1109/ECCTD.2013.6662306.

[26] Y. Shi, S. Oh, Z. Huang, X. Lu, S. H. Kang, and D. Kuzum, "Performance prospects of deeply scaled spin-transfer torque magnetic random-access memory for in-memory computing," *IEEE Electron Device Lett.*, vol. 41, no. 7, pp. 1126–1129, Jul. 2020, doi: 10.1109/LED.2020.2995819.

[27] Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, J. P. Strachan, N. Ge, N. McDonald, Q. Wu, M. Hu, H. Wu, R. S. Williams, Q. Xia, and J. J. Yang, "In situ training of feed-forward and recurrent convolutional memristor networks," *Nature Mach. Intell.*, vol. 1, no. 9, pp. 434–442, Sep. 2019, doi: 10.1038/s42256-019-0089-1.

[28] F. Gul, "Nano-scale single layer TiO_2-based artificial synaptic device," *Appl. Nanoscience*, vol. 10, no. 2, pp. 611–616, Feb. 2020, doi: 10.1007/s13204-019-01179-y.

[29] D. Soydaner, "A comparison of optimization algorithms for deep learning," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 34, no. 13, Dec. 2020, Art. no. 2052013, doi: 10.1142/s0218001420520138.
[30] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, Jan. 2018, doi: 10.1137/16m1080173.
[31] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.
[32] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, Apr. 2010, doi: 10.1021/nl904092h.
[33] T. D. Vu, H.-J. Yang, V. Q. Nguyen, A.-R. Oh, and M.-S. Kim, "Multi-modal learning using convolution neural network and sparse autoencoder," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 309–312, doi: 10.1109/BIGCOMP.2017.7881683.
[34] Y. Liao, N. Deng, H. Wu, B. Gao, Q. Zhang, and H. Qian, "Weighted synapses without carry operations for RRAM-based neuromorphic systems," *Frontiers Neurosci.*, vol. 12, pp. 1–9, Mar. 2018, doi: 10.3389/fnins.2018.00167.
[35] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018, doi: 10.1109/TCAD.2018.2789723.
[36] G. Indiveri, "A low-power adaptive integrate-and-fire neuron circuit," in *Proc. Int. Symp. Circuits Syst.*, 2003, pp. 1–4, doi: 10.1109/ISCAS.2003.1206342.
[37] Y. J. Lee, J. Lee, Y. B. Kim, J. Ayers, and A. Volkovskii, "Low power real time electronic neuron VLSI design using subthreshold technique," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2004, p. 744, doi: 10.1109/ISCAS.2004.1329111.
[38] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and STDP integrated circuits," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 3, pp. 246–256, Jun. 2012, doi: 10.1109/TBCAS.2011.2174152.
[39] I. Sourikopoulos, S. Hedayat, C. Loyez, F. Danneville, V. Hoel, E. Mercier, and A. Cappy, "A 4-fJ/spike artificial neuron in 65 nm CMOS technology," *Frontiers Neurosci.*, vol. 11, p. 123, Mar. 2017, doi: 10.3389/fnins.2017.00123.
[40] J. Shamsi, K. Mohammadi, and S. B. Shokouhi, "A hardware architecture for columnar-organized memory based on CMOS neuron and memristor crossbar arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 12, pp. 2795–2805, Dec. 2018, doi: 10.1109/TVLSI.2018.2815025.
[41] Y. Babacan, F. Kaçar, and K. Gürkan, "A spiking and bursting neuron circuit based on memristor," *Neurocomputing*, vol. 203, pp. 86–91, Aug. 2016, doi: 10.1016/j.neucom.2016.03.060.
[42] V. Saxena, X. Wu, and K. Zhu, "Energy-efficient CMOS memristive synapses for mixed-signal neuromorphic system-on-a-chip," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351766.
[43] A. Mizrahi, T. Hirtzlin, A. Fukushima, H. Kubota, S. Yuasa, J. Grollier, and D. Querlioz, "Neural-like computing with populations of superparamagnetic basis functions," *Nature Commun.*, vol. 9, no. 1, p. 1533, Apr. 2018, doi: 10.1038/s41467-018-03963-w.
[44] Z. Wang, S. Joshi, S. Savel'Ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo, and H. Jiang, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electron.*, vol. 1, no. 2, pp. 137–145, Feb. 2018, doi: 10.1038/s41928-018-0023-2.
[45] X. Zhang, W. Wang, Q. Liu, X. Zhao, J. Wei, R. Cao, Z. Yao, X. Zhu, F. Zhang, H. Lv, S. Long, and M. Liu, "An artificial neuron based on a threshold switching memristor," *IEEE Electron Device Lett.*, vol. 39, no. 2, pp. 308–311, Feb. 2018, doi: 10.1109/LED.2017.2782752.
[46] Y.-F. Lu, Y. Li, H. Li, T.-Q. Wan, X. Huang, Y.-H. He, and X. Miao, "Low-power artificial neurons based on Ag/TiN/HfAlO$_x$/Pt threshold switching memristor for neuromorphic computing," *IEEE Electron Device Lett.*, vol. 41, no. 8, pp. 1245–1248, Aug. 2020, doi: 10.1109/LED.2020.3006581.
[47] M. S. Feali, "Using volatile/non-volatile memristor for emulating the short-and long-term adaptation behavior of the biological neurons," *Neurocomputing*, vol. 465, pp. 157–166, Nov. 2021, doi: 10.1016/j.neucom.2021.08.132.
[48] K. Bian and R. Priyadarshi, "Machine learning optimization techniques: A survey, classification, challenges, and future research issues," *Arch. Comput. Methods Eng.*, vol. 31, pp. 4209–4233, Mar. 2024, doi: 10.1007/s11831-024-10110-w.
[49] A. Ali, M. Hamraz, N. Gul, D. M. Khan, S. Aldahmani, and Z. Khan, "A K nearest neighbour ensemble via extended neighbourhood rule and feature subsets," *Pattern Recognit.*, vol. 142, Oct. 2023, Art. no. 109641, doi: 10.1016/j.patcog.2023.109641.
[50] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016. Accessed: Oct. 21, 2024. [Online]. Available: http://arxiv.org/abs/1609.04747
[51] N. S. Altman, "An introduction to kernel and nearest-neighbor non-parametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992, doi: 10.1080/00031305.1992.10475879.
[52] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951, doi: 10.1214/aoms/1177729586.
[53] Y. Tian, Y. Zhang, and H. Zhang, "Recent advances in stochastic gradient descent in deep learning," *Mathematics*, vol. 11, no. 3, p. 682, Jan. 2023, doi: 10.3390/math11030682.
[54] X. Liang, Z. Chen, Y. Deng, D. Liu, X. Liu, Q. Huang, and T. Arai, "Field-controlled microrobots fabricated by photopolymerization," *Cyborg Bionic Syst.*, vol. 4, p. 9, Jan. 2023, doi: 10.34133/cbsystems.0009.
[55] S. Ma, Y. Chen, S. Yang, S. Liu, L. Tang, B. Li, and Y. Li, "The autonomous pipeline navigation of a cockroach bio-robot with enhanced walking stimuli," *Cyborg Bionic Syst.*, vol. 4, p. 67, Jan. 2023, doi: 10.34133/cbsystems.0067.
[56] Z. Cai, X. Zhu, P. Gergondet, X. Chen, and Z. Yu, "A friction-driven strategy for agile steering wheel manipulation by humanoid robots," *Cyborg Bionic Syst.*, vol. 4, p. 64, Jan. 2023, doi: 10.34133/cbsystems.0064.
[57] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*.
[58] X. Long, C. Lu, Y. Su, and Y. Dai, "Machine learning framework for predicting the low cycle fatigue life of lead-free solders," *Eng. Failure Anal.*, vol. 148, Jun. 2023, Art. no. 107228, doi: 10.1016/j.engfailanal.2023.107228.
[59] L. Ye, Z. Gao, J. Fu, W. Ren, C. Yang, J. Wen, X. Wan, Q. Ren, S. Gu, X. Liu, X. Lian, and L. Wang, "Overview of memristor-based neural network design and applications," *Frontiers Phys.*, vol. 10, pp. 1–27, Jul. 2022, doi: 10.3389/fphy.2022.839243.
[60] E. Gale, "TiO_2-based memristors and ReRAM: Materials, mechanisms and models (a review)," *Semicond. Sci. Technol.*, vol. 29, no. 10, Oct. 2014, Art. no. 104004, doi: 10.1088/0268-1242/29/10/104004.
[61] D. Kuzum, "Neuro-inspired computing with resistive switching devices [guest editorial]," *IEEE Nanotechnol. Mag.*, vol. 12, no. 3, p. 4, Sep. 2018, doi: 10.1109/MNANO.2018.2849799.
[62] A. El Mesoudy, G. Lamri, R. Dawant, J. Arias-Zapata, P. Gliech, Y. Beilliard, S. Ecoffey, A. Ruediger, F. Alibart, and D. Drouin, "Fully CMOS-compatible passive TiO_2-based memristor crossbars for in-memory computing," *Microelectronic Eng.*, vol. 255, Feb. 2022, Art. no. 111706, doi: 10.1016/j.mee.2021.111706.
[63] N. Ilyas, D. Li, C. Li, X. Jiang, Y. Jiang, and W. Li, "Analog switching and artificial synaptic behavior of Ag/SiO$_x$:Ag/TiO$_x$/p^{++}-Si memristor device," *Nanosc. Res. Lett.*, vol. 15, no. 1, p. 10, Dec. 2020, doi: 10.1186/s11671-020-3249-7.
[64] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–Oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012, doi: 10.1109/JPROC.2012.2190369.
[65] Q. Wan, M. T. Sharbati, J. R. Erickson, Y. Du, and F. Xiong, "Emerging artificial synaptic devices for neuromorphic computing," *Adv. Mater. Technol.*, vol. 4, no. 4, Apr. 2019, Art. no. 1900037, doi: 10.1002/admt.201900037.
[66] Y. Luo, X. Peng, and S. Yu, "MLP+NeuroSimV3.0," in *Proc. Int. Conf. Neuromorphic Syst.*, New York, NY, USA, Jul. 2019, pp. 1–7, doi: 10.1145/3354265.3354266.
[67] M. Prezioso, F. M. Bayat, B. Hoskins, K. Likharev, and D. Strukov, "Self-adaptive spike-time-dependent plasticity of metal-oxide memristors," *Sci. Rep.*, vol. 6, no. 1, p. 21331, Feb. 2016, doi: 10.1038/srep21331.
[68] S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, and H.-S. P. Wong, "An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation," *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2729–2737, Aug. 2011, doi: 10.1109/TED.2011.2147791.
[69] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.
[70] M. B. McCrary, "Urban multicultural trauma patients," *ASHA*, vol. 34, no. 4, pp. 37–40, 1992.

[71] O. Krestinskaya, K. N. Salama, and A. P. James, "Learning in memristive neural network architectures using analog backpropagation circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 2, pp. 719–732, Feb. 2019, doi: 10.1109/TCSI.2018.2866510.
[72] C.-Y. Chen and K. Chakrabarty, "Pruning of deep neural networks for fault-tolerant memristor-based accelerators," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 889–894, doi: 10.1109/DAC18074.2021.9586269.
[73] W. Zhao and Y. Cao, "Predictive technology model for nano-CMOS design exploration," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 3, no. 1, p. 1, Apr. 2007, doi: 10.1145/1229175.1229176.
[74] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *J. Parallel Distrib. Comput.*, vol. 134, pp. 75–88, Dec. 2019, doi: 10.1016/j.jpdc.2019.07.007.
[75] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "DeLight," in *Proc. Int. Symp. Low Power Electron. Design*, New York, NY, USA, Aug. 2016, pp. 112–117, doi: 10.1145/2934583.2934599.
[76] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning," *Genetic Program. Evolvable Mach.*, vol. 19, nos. 1–2, pp. 305–307, Jun. 2018, doi: 10.1007/s10710-017-9314-z.
[77] F. Gul, "Circuit implementation of nano-scale TiO_2 memristor using only metal-oxide-semiconductor transistors," *IEEE Electron Device Lett.*, vol. 40, no. 4, pp. 643–646, Apr. 2019, doi: 10.1109/LED.2019.2899889.

BAKI GÖKGÖZ received the M.Sc. degree in computer engineering from Karadeniz Technical University, Türkiye. He is currently a Lecturer with the Department of Computer Technologies, Gümüşhane University. His research interests include computer science, algorithms and calculation theory, and artificial intelligence.

TOLGA AYDIN received the M.Sc. and Ph.D. degrees in computer engineering from Bilkent University, Türkiye. He is currently an Assistant Professor of computer engineering with Atatürk University, Türkiye. His research interests include algorithms, artificial intelligence, machine learning, and data mining.

FATIH GÜL (Member, IEEE) received the B.Sc. degree in electrical and electronics engineering from İnönü University, Malatya, Türkiye, in 2004, and the M.Sc. and Ph.D. degrees in electrical and electronics engineering from Atatürk University, Erzurum, Türkiye, in 2008 and 2017, respectively. He is currently an Associate Professor with the Department of Electrical-Electronics Engineering, Recep Tayyip Erdoğan University, Rize. His research interests include semiconductors, resistive switching memory devices, memristors, embedded IoT devices, and neuromorphic systems.