frontiers in Neuroscience

ORIGINAL RESEARCH
published: 09 December 2021
doi: 10.3389/fnins.2021.750458

Mapping the BCPNN Learning Rule to a Memristor Model

Deyu Wang[1†], Jiawei Xu[1†], Dimitrios Stathis[2], Lianhao Zhang[3], Feng Li[1], Anders Lansner[2,4], Ahmed Hemani[2*], Yu Yang[2], Pawel Herman[2] and Zhuo Zou[1*]*

[1] State Key Laboratory of ASIC and System, School of Information Science and Technology, Fudan University, Shanghai, China, [2] School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden, [3] Department of Electrical Engineering, Technical University of Denmark, Kongens Lyngby, Denmark, [4] Department of Mathematics, Stockholm University, Stockholm, Sweden

The Bayesian Confidence Propagation Neural Network (BCPNN) has been implemented in a way that allows mapping to neural and synaptic processes in the human cortex0and0has been used extensively in detailed spiking models of cortical associative memory function and recently also for machine learning applications. In conventional digital implementations of BCPNN, the von Neumann bottleneck is a major challenge with synaptic storage and access to it as the dominant cost. The memristor is a non-volatile device ideal for artificial synapses that fuses computation and storage and thus fundamentally overcomes the von Neumann bottleneck.0While the implementation of other neural networks like Spiking Neural Network (SNN) and even Convolutional Neural Network (CNN) on memristor has been studied, the implementation of BCPNN has not. In this paper, the BCPNN learning rule is mapped to a memristor model and implemented with a memristor-based architecture. The implementation of the BCPNN learning rule is a mixed-signal design with the main computation and storage happening in the analog domain.0In particular, the nonlinear dopant drift phenomenon of the memristor is exploited to simulate the exponential decay of the synaptic state variables in the BCPNN learning rule.0The consistency between the memristor-based solution and the BCPNN learning rule is simulated and verified in Matlab, with a correlation coefficient as high as 0.99. The analog circuit is designed and implemented in the SPICE simulation environment, demonstrating a good emulation effect for the BCPNN learning rule with a correlation coefficient as high as 0.98. This work focuses on demonstrating the feasibility of mapping the BCPNN learning rule to in-circuit computation in memristor. The feasibility of the memristor-based implementation is evaluated and validated in the paper, to pave the way for a more efficient BCPNN implementation, toward a real-time brain emulation engine.

Keywords: Bayesian Confidence Propagation Neural Network (BCPNN), learning rule, memristor, nonlinear dopant drift phenomenon, synaptic state update, spiking neural networks, analog neuromorphic hardware

OPEN ACCESS

Edited by:
Irem Boybat,
IBM Research-Zurich, Switzerland

Reviewed by:
Christopher Bennett,
Sandia National Laboratories, United States
Wei Wang,
Technion Israel Institute of Technology, Israel

***Correspondence:**
Anders Lansner
ala@kth.se
Ahmed Hemani
hemani@kth.se
Zhuo Zou
zhuo@fudan.edu.cn

[†] These authors have contributed equally to this work and share first authorship

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *30 July 2021*
Accepted: *11 November 2021*
Published: *09 December 2021*

Citation:
Wang D, Xu J, Stathis D, Zhang L, Li F, Lansner A, Hemani A, Yang Y, Herman P and Zou Z (2021) Mapping the BCPNN Learning Rule to a Memristor Model. Front. Neurosci. 15:750458. doi: 10.3389/fnins.2021.750458

1. INTRODUCTION

In the last decade, Artificial Neural Networks (ANNs) have made rapid and significant progress in real-world applications, demonstrating outstanding performance in a wide range of pattern recognition problems such as speech recognition (Hinton et al., 2012), image classification (Ciregan et al., 2012), and natural language processing (Yin et al., 2017). Despite the great success

and popularity of ANNs in data-driven computational paradigms, they have some limitations. First of all, most of the existing ANNs adopt supervised learning, requiring a large amount of labeled training data, which is different from the unsupervised and reward modulated learning mechanisms attributed to biological brains. Secondly, the most prominent learning algorithm used by ANNs is error back-propagation, which requires a high level of accuracy and is neither robust nor biologically plausible. Thirdly, the current mainstream ANN models do not account for the functionality underlying human cognition and inspiring artificial intelligence, like e.g., associative memory, temporal association, and reward-based trial-and-error learning. Unlike classical ANNs with non-spiking units, Spiking Neural Networks (SNNs) use the same event-based communication mechanism as the human brain where neurons communicate with spikes.

The Bayesian Confidence Propagation Neural Network (BCPNN) was originally derived from principles of Bayesian inference (Lansner and Ekeberg, 1989; Lansner and Holst, 1996) and was further developed into an architecture inspired by the modularity of the mammalian cortex with hypercolumn units (HCUs) and minicolumn units (MCUs). Later implementation within the framework of SNNs allowed mapping to neural and synaptic processes in the human cortex (Tully et al., 2014). Compared with other SNN models, BCPNN provides a compact and practical solution for the implementation of large-scale neural networks due to its modular, coarse-grained, and hierarchical architecture. Importantly, both reduced non-spiking and biologically detailed spiking realizations of BCPNN perform similar functions. They have been extensively used to model brain-like cognitive capabilities such as associative memory (Johansson and Lansner, 2007; Lundqvist et al., 2011), episodic memory (Chrysanthidis et al., 2021), and working memory (Fiebig and Lansner, 2017; Fiebig et al., 2020), which play a key role in human intelligence. In a broader perspective, we suggest that these advancements in simulating different aspects of human cognitive function within a system framework of brain-like BCPNN constitute a promising direction in the development of artificial general intelligence (AGI).

Furthermore, the local associative nature of the Bayesian-Hebbian BCPNN learning rule has also been leveraged in cortex-inspired neural networks built for pattern recognition problems in the machine learning domain. In particular, these recent developments were facilitated by the addition of a novel brain-like structural plasticity algorithm to build a hidden layer using the original synaptic trace variables of BCPNN in an unsupervised manner (Ravichandran et al., 2020, 2021a). Classification performance on the MNIST and Fashion-MNIST benchmark problems—98.6% and 88.9% on test sets, respectively (Ravichandran et al., 2021a)—is competitive with e.g., single-layer multi-layer perceptron (MLP) with backprop, restricted Boltzmann machine (RBM), and overcomplete autoencoder. The aforementioned unsupervised nature of the structural plasticity lends itself to the efficient use of unlabeled training examples, which has been exploited to perform semi-supervised learning with competitive results on MNIST for only 10–1,000 labeled training samples (Ravichandran et al., 2021b).

At present, BCPNN is usually implemented in high-performance computers, such as clusters (Johansson and Lansner, 2007), GPUs (Yang et al., 2020; Podobas et al., 2021), and ASICs (Stathis et al., 2020). However, these systems do not fully leverage the scalability of the modular BCPNN with its local learning since they are all based on the von Neumann architecture that separates computation and storage, which puts a high demand on computing and memory access. We observe that the ASIC implementation with its full customized architecture with the 3D-RAM, achieved three orders better efficiency compared to GPUs, but it is still many orders less efficient compared to a biological brain.

Besides overcoming the von Neumann bottleneck, the non-linearity of the memristor naturally mimics the behavior of synapses. This paper shows how these properties of memristors can be leveraged to implement the BCPNN learning rule. The long-term goal of this research is to realize a large-scale memristor-based BCPNN network that is 10–100x more efficient than ASICs. However, the research presented in this paper focuses on demonstrating the feasibility of mapping the BCPNN learning rule to an in-circuit memristor-based computation. Follow-up work to this paper will focus on addressing the non-idealities of memristors and the energy efficiency analysis.

The contributions of this work are as follows:

- The non-linearity of the memristor is exploited to emulate the synaptic traces in the BCPNN learning rule. On this basis, a memristor-based architecture for the BCPNN learning rule is proposed.
- The memristor-based design for the BCPNN learning rule is simulated and verified in Matlab. The consistency between the memristor-based solution and the reference model is validated, and the correlation coefficient is as high as 0.99.
- The analog circuit for the BCPNN learning rule is designed and implemented in the SPICE simulation environment. The SPICE simulation results demonstrate a good emulation effect for the BCPNN learning rule, and the correlation coefficient is as high as 0.98.

The rest of this paper is organized as follows: Section 2 introduces the background knowledge and details about BCPNN and the memristor. Section 3 shows the similarity between the BCPNN traces and the memristor non-linearity and demonstrates how to map the BCPNN learning rule to in-circuit memristor-based computation. Section 4 proposes the memristor-based architecture for the BCPNN learning rule and explains the corresponding analog circuit design. Section 5 presents the results of Matlab and SPICE-level simulations. Section 6 summarizes the paper and further discusses several aspects of this work. Finally, section 7 presents the prospects for future work.

2. PRELIMINARIES

2.1. BCPNN

2.1.1. BCPNN Overview

The BCPNN features a modular structure in terms of HCUs and MCUs, based on a generalization of the structure of the mammalian cortex, first described by Hubel and Wiesel

(Hubel and Wiesel, 1977). In models of the mammalian cortex, an HCU module has a diameter on the order of 500 μm and comprises about 100 MCUs with 50 μm diameter. Each MCU is composed of about 100 neurons, mainly excitatory pyramidal cells and one or two local inhibitory double bouquet cells (DeFelipe et al., 2006). Activity within an HCU is regulated by lateral inhibition mediated via inhibitory basket cells. In the abstract models, it takes the form of softmax that normalizes the total HCU activity (sum of the corresponding MCU activities) to 1. The number of HCUs in the human cortex has been estimated at around two million.

The BCPNN network can be represented with a $H \times M$ configuration, which means it is composed of H HCUs, and each HCU contains M MCUs. Generally, H is much larger than M. The number of HCUs H can be increased without an upper limit, while the number of MCUs M has an upper limit of about 100 based on biological data. Therefore, when it comes to the configuration of large networks, the number of H tends to be quite high. In a small network, each MCU can be fully connected to its local HCU and other HCUs, as shown in **Figure 1A**. Such full connectivity can not be employed in large networks due to the extreme cost of computation and storage. Instead, a cortex-inspired sparse patchy connection is adopted (Meli and Lansner, 2013), which greatly reduces the number of connections and yet maintains proper function.

Figure 1B presents the structure of the HCU, which is composed of 4 parts: 1) the presynaptic vector, used to store presynaptic traces Z_i, E_i and P_i; 2) the postsynaptic vector, used to store postsynaptic traces Z_j, E_j, P_j and the bias β_j; 3) the synaptic matrix, used to store synaptic traces E_{ij}, P_{ij} and the weight w_{ij}. 4) a certain number of MCUs, which integrate the incoming spiking activities and fire in a soft winner-take-all manner.

At a higher level, HCUs function like independent network modules between which spikes are transmitted. The HCU size depends on the number of incoming connections and MCUs. The biologically constrained maximum number of incoming connections and MCUs is 10,000 and 100, respectively. Consequently, in a max-size HCU, a synaptic matrix with a size of $10,000 \times 100$ is used, thus representing a million plastic synapses.

2.1.2. BCPNN Learning Rule

The BCPNN learning rule was derived from Bayes' rule while making some independent assumptions between neural activities and by transformation to log-space to achieve a proper neural activation function (Lansner and Ekeberg, 1989; Lansner and Holst, 1996; Sandberg et al., 2002). Thus, rather than being purely phenomenological as the commonly used Spike Timing Dependent Plasticity (STDP) learning rule, it was derived from the probabilistic inference. The BCPNN learning rule is in essence another kind of Hebbian learning rule in which synaptic updates are driven by co-activation between the pre- and post-synaptic neural units. It generates positive weights if the activity between neurons is positively correlated, zero weights if they are uncorrelated, and negative weights if they are anti-correlated. Besides, it has an intrinsic bias for each neural unit which reflects the prior activation and also is observed experimentally (Tully et al., 2014). The BCPNN learning rule equations estimate the activation and co-activation of network units utilizing a cascade of three exponential running averages, as shown in **Figure 2A**.

First, the incoming spikes drive pre- and post-synaptic Z-traces:

$$\frac{dZ_i}{dt} = \frac{S_i - Z_i}{\tau_{zi}} \quad \frac{dZ_j}{dt} = \frac{S_j - Z_j}{\tau_{zj}} \tag{1}$$

Here, i denotes pre- and j denotes post-synaptic variables and S represents incoming and generated spiking activity. These Z-traces in turn drive the E-traces and P-traces following the same kind of dynamics with different time constants:

$$\frac{dE_i}{dt} = \frac{Z_i - E_i}{\tau_e} \quad \frac{dE_j}{dt} = \frac{Z_j - E_j}{\tau_e} \quad \frac{dE_{ij}}{dt} = \frac{Z_i Z_j - E_{ij}}{\tau_e} \tag{2}$$

$$\frac{dP_i}{dt} = \frac{(E_i - P_i)}{\tau_p}\kappa \quad \frac{dP_j}{dt} = \frac{(E_j - P_j)}{\tau_p}\kappa \quad \frac{dP_{ij}}{dt} = \frac{(E_{ij} - P_{ij})}{\tau_p}\kappa \tag{3}$$

The learning rate κ in the dynamics of P traces modulates the learning. The E-traces form a synaptic tag important for delayed reinforcement learning. In many cases, the E-traces can be omitted and the P-traces can be updated according to equation (4) with an added parameter κ. The simplified BCPNN learning rule without E trace is shown in **Figure 2B**.

$$\frac{dP_i}{dt} = \frac{(Z_i - P_i)}{\tau_p}\kappa \quad \frac{dP_j}{dt} = \frac{(Z_j - P_j)}{\tau_p}\kappa \quad \frac{dP_{ij}}{dt} = \frac{(Z_i Z_j - P_{ij})}{\tau_p}\kappa \tag{4}$$

Finally, as shown in equation (5), the P-traces are used to update network unit biases, and weights with an additional parameter ε, which originates from a minimum spiking activity assumed for the pre- and postsynaptic units:

$$\beta_j = \log(P_j + \varepsilon) \quad W_{ij} = \log\left(\frac{P_{ij} + \varepsilon^2}{(P_i + \varepsilon)\cdot(P_j + \varepsilon)}\right) \tag{5}$$

2.1.3. BCPNN Application and Implementation

The BCPNN model has been used for neural computation and machine learning applications as well as to model the synaptic plasticity like long-term potentiation (LTP) and long-term depression (LTD) in SNN models of cortical associative memory. In the case of neural computation, BCPNN has been used to model scalable self-organizing associative memory (Johansson and Lansner, 2007). As for the classification of the MNIST machine learning benchmarking, an accuracy of 98.6% can be achieved while maintaining a high neurobiological plausibility (Ravichandran et al., 2020, 2021a). In the latter case, the hidden layer had 200 HCUs, each with 100 MCUs. Recent cortical associative memory models have focused on synaptic working memory using BCPNN as a model for rapid cortical synaptic plasticity (Fiebig and Lansner, 2017; Fiebig et al., 2020). The positive BCPNN weights are used as excitatory connections between pyramidal cells, while the negative ones are disynaptically inhibiting pyramidal cells in distant HCUs via e.g., double bouquet cells. These SNN models are tiny compared to their biological counterparts, typically comprising up to a thousand MCUs partitioned into some 30 HCUs.

The BCPNN model has been implemented in software packages, GPU, and supercomputer clusters. It has also been

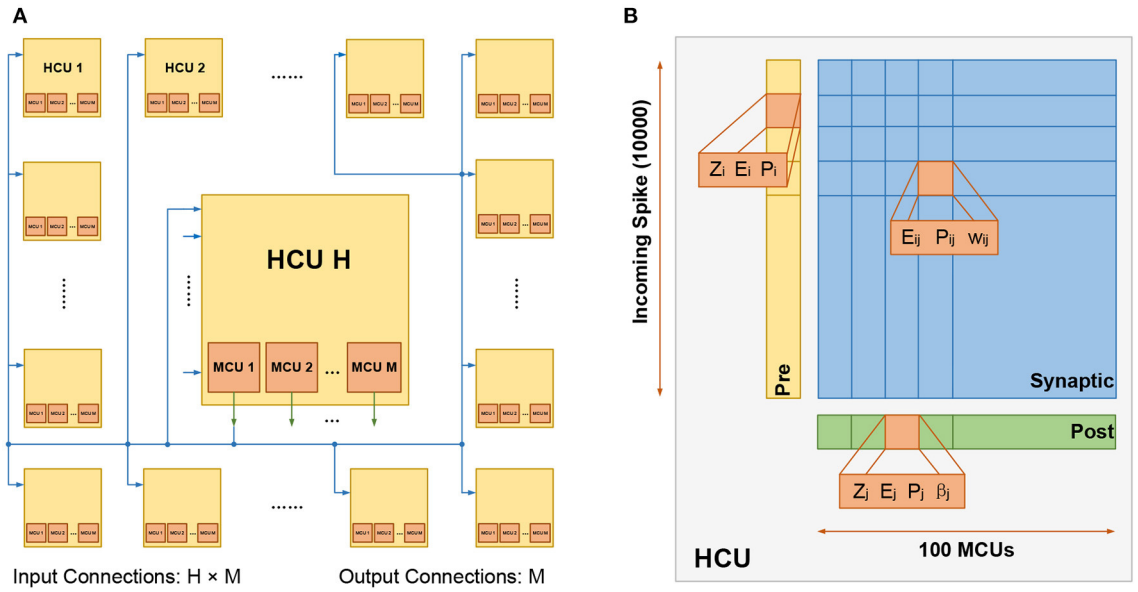

FIGURE 1 | (A) The fully connected HCUs. The number of HCUs is *H*, each HCU contains *M* MCUs. For each HCU, the number of input connections is $H \times M$, and the number of output connections is *M*. **(B)** The structure of HCU. The upper limit of input connections and MCUs is 10,000 and 100, respectively.

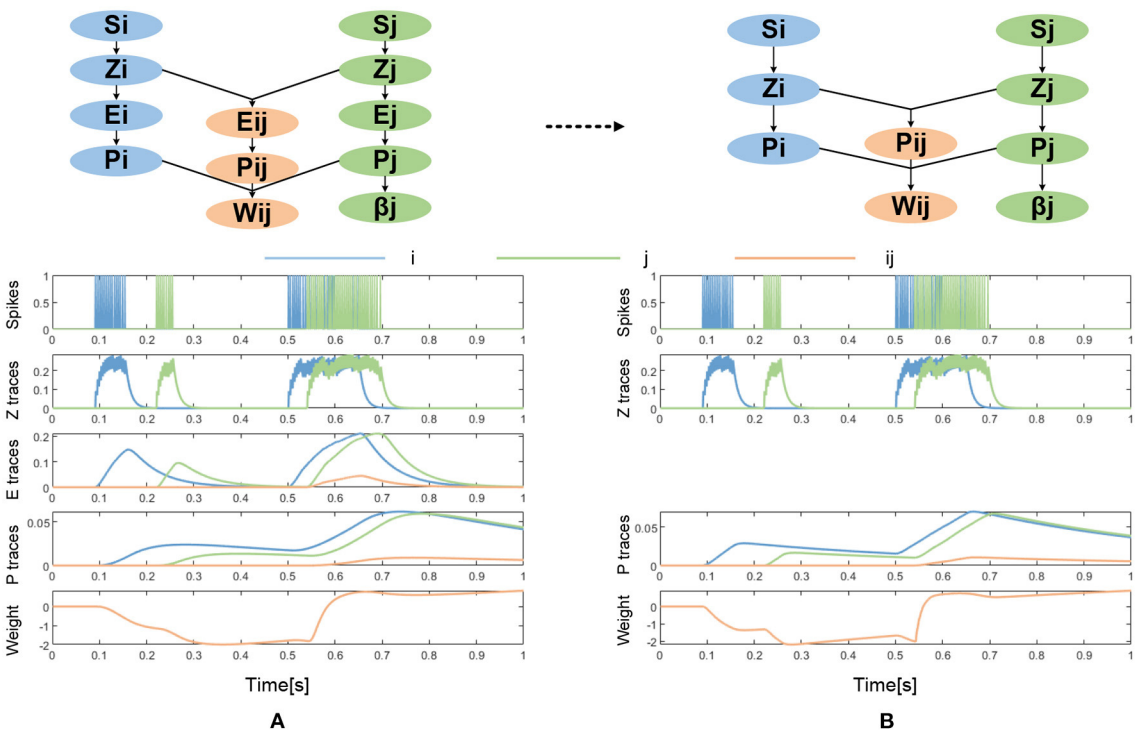

FIGURE 2 | (A) The original BCPNN learning rule (adapted after Tully et al., 2014). **(B)** The simplified BCPNN learning rule without E trace.

implemented as custom hardware with 3D integration of DRAM for the synaptic weights (Farahini et al., 2014; Lansner et al., 2014; Stathis et al., 2020; Yang et al., 2020). The BCPNN learning rule is amenable to low-precision implementation (Vogginger et al., 2015), and the cortical memory models have proven quite robust and tolerant to external as well as to intrinsic noise

and imprecision in weights and unit biases. Therefore, it is a highly scalable, modular, and hardware-friendly neuromorphic architecture with the potential for compact and low-power digital or mixed-signal design.

2.2. The Memristor

The memristor was predicted as a fourth fundamental circuit element following the resistor, capacitor, and inductor by Chua (1971). In 2008, HP Labs demonstrated and fabricated a memristor for the first time (Strukov et al., 2008). The HP Memristor was based on a nanoscale TiO_2 thin film, with a doped region and an undoped region, as shown in **Figure 3A**. The total resistance of the device is determined by the variable resistances of these two regions. When an external bias voltage is applied across the device, the charged dopants will drift, moving the boundary between the two regions. The HP memristor produces rich hysteretic current-voltage behavior, which can be observed in many nanoscale electronic devices. However, in nanoscale devices, a small voltage can yield enormous electric fields, secondarily producing significant non-linearities in ionic transport, which is called the non-linear dopant drift phenomenon. This phenomenon can be represented with a window function model, as shown in **Figure 3B**.

The memristor has many characteristics that can be utilized in a variety of applications. To begin with, as suggested by its name, a memristive device remembers the charge that passes through it rather than storing the charge so that the memristor is nonvolatile. What is more, the memristor device can store multi-valued rather than binary values. The ability to represent multi-bit values stems from the memristor's ability to have multiple intermediate points in its transfer curve. The transfer curve, with its hysteretic behavior and ability to represent multiple values, resembles biological synapses. This is the reason for memristors attracting attention as ideal building blocks for neuromorphic structures. The ability to remember multi-valued quantities in response to voltages applied to its terminals mimics analog computation. This *in-situ* computation has also been exploited to build general-purpose computers (Zidan et al., 2018), content addressable memory (Li et al., 2020a), and to implement neural networks, both spiking and non-spiking, as discussed next.

For both non-spiking artificial neural networks and spiking neural networks, the core operation is to reinforce or weaken the synaptic weights. The algorithms used for deciding the time, magnitude, and sign of reinforcement vary from one algorithm to another. The commonality is in applying appropriate voltages for an appropriate duration to the two terminals of the memristors.

In the ANN space, several memristor-based ANNs have been studied and implemented. A single-layer perceptron (Prezioso et al., 2015) was constructed based on transistor-free metal-oxide memristor crossbars, performing the perfect classification of images. The feasibility of a three-layer fully connected neural network on MNIST and a 13-layer Convolutional Neural Network (CNN) on CIFAR-10 using the flexible memristor are studied and evaluated (Xu et al., 2018). A five-layer memristor-based CNN (Yao et al., 2020) was demonstrated to perform image recognition on MNIST, achieving an accuracy of over 96%. It is worth noting that it is challenging to take *in-situ* training on memristor-based ANNs due to non-ideal device characteristics. Prezioso's work takes *in-situ* learning with a simple learning rule called the Manhattan update rule. In Yao's work, a hybrid-training method is taken to compensate for existing device imperfections.

In-situ computation in memristors has also been widely studied for spiking neural networks. A supervised learning model (Nishitani et al., 2015) that enables error backpropagation for spiking neural network hardware was proposed, and the memristor was employed as an electric synapse to store the analog synaptic weight in the circuit. An all-memristor stochastic SNN architecture (Wijesinghe et al., 2018) was proposed in which the inherent stochasticity of nanoscale devices is utilized to emulate the functionality of a spiking neuron. An area-efficient memristor SNN for hardware implementation (Zhou et al., 2019) was presented based on the modified SpikeProp-like supervised learning algorithm. An STDP-based SNN (Zhao et al., 2020) was proposed to achieve the mechanism of lateral inhibition and homeostasis by memristor-based inhibitory synapses. A novel memristive synapse model based on the HP memristor was proposed, and a spiking neural network hardware fragment was constructed (Huang et al., 2021).

However, the majority of the state-of-the-art memristor-based SNNs are limited in scale and employ simple learning rules such as STDP. Compared with small-scale SNNs using STDP, the BCPNN learning rule is more complex, and its computational structure is modular and cascaded. This paper exploits the non-linearity of the memristor and elaborates how *in-situ* analog computation in the memristor has been utilized to implement the BCPNN learning rule.

3. A MEMRISTOR-BASED BCPNN LEARNING RULE

3.1. BCPNN Model

The BCPNN learning rule has been depicted with ordinary differential equations, representing the update process of Z, E, P traces. To facilitate the hardware implementation, the ordinary differential equations (1,2,3) are further transformed to equations (6,7,8), respectively, with Euler's method, as shown below:

$$\begin{aligned} Z_i(t) &= Z_i(t-1) \times (1 - kz_i) + S_i(t-1) \times kz_i \\ Z_j(t) &= Z_j(t-1) \times (1 - kz_j) + S_j(t-1) \times kz_j \end{aligned} \tag{6}$$

$$\begin{aligned} E_i(t) &= E_i(t-1) \times (1 - ke) + Z_i(t-1) \times ke \\ E_j(t) &= E_j(t-1) \times (1 - ke) + Z_j(t-1) \times ke \\ E_{ij}(t) &= E_{ij}(t-1) \times (1 - ke) + Z_i(t-1) \times Z_j(t-1) \times ke \end{aligned} \tag{7}$$

$$\begin{aligned} P_i(t) &= P_i(t-1) \times (1 - kp) + E_i(t-1) \times kp \\ P_j(t) &= P_j(t-1) \times (1 - kp) + E_j(t-1) \times kp \\ P_{ij}(t) &= P_{ij}(t-1) \times (1 - kp) + E_{ij}(t-1) \times kp \end{aligned} \tag{8}$$

where,

$$kz_i = \frac{dt}{\tau_{z_i}} \; kz_j = \frac{dt}{\tau_{z_j}} \; ke = \frac{dt}{\tau_e} \; kp = \frac{dt}{\tau_p} \cdot \kappa \tag{9}$$

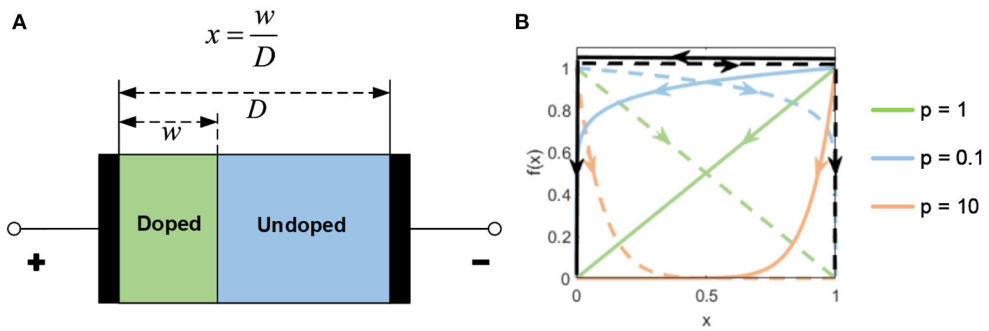

FIGURE 3 | (A) The HP model. **(B)** The window function.

The current values of Z, E, and P traces are all calculated from their previous values. The kz_i, kz_j, ke and kp are all constants. The simplified equation (4) can be transformed to equation (10) in the same manner, as follows:

$$\begin{aligned} P_i(t) &= P_i(t-1) \times (1-kp) + Z_i(t-1) \times kp \\ P_j(t) &= P_j(t-1) \times (1-kp) + Z_j(t-1) \times kp \\ P_{ij}(t) &= P_{ij}(t-1) \times (1-kp) + Z_i(t-1) \times Z_j(t-1) \times kp \end{aligned} \tag{10}$$

It should be noted that we do not consider the E trace in this work to facilitate hardware implementation. Therefore, we adopt simplified equation (10), whose update process and curve of traces can be seen in **Figure 2B**.

3.2. The Memristor Model

In 2008, HP Labs proposed the linear model of a memristor (Strukov et al., 2008). Following the HP model, a variety of memristor models have been devised, such as the non-linear ion drift model (Yang et al., 2008), Simmons Tunnel Barrier Model (Pickett et al., 2009), the TEAM model (Kvatinsky et al., 2013) and the VTEAM model (Kvatinsky et al., 2015). To emulate the non-linear dopant drift phenomenon, the window function is introduced as an essential part of a memristor model, and dozens of window functions have been proposed so far. However, most window functions are facing one or more of the following problems: the boundary effect, the boundary lock, and inflexibility (Xu et al., 2021). Joglekar's window function (Joglekar and Wolf, 2009) considers the boundary effect but suffers from the boundary lock problem. Biolek's window function (Biolek et al., 2009) takes the current direction into account to solve the boundary lock issue, but its parameter setting is not flexible enough. Recently, Li's window function (Li et al., 2020b) is proposed to consider all three issues. However, Li's window function is complex, and its expression is associated with six controlling parameters, which may increase the effort required for simulation. The window function that we proposed in Xu et al. (2021) is introduced to address this problem, which is both flexible and concise.

The VTEAM model is adopted for this work because of the following advantages: 1) the VTEAM model has a good fitting effect for the nonlinear bipolar physical memristor devices that we are concerned with (Johnson et al., 2010; Chanthbouala et al., 2012; Li et al., 2018); 2) this memristor model is voltage-controlled, and the threshold voltage phenomenon has been observed in many physical devices; 3) the VTEAM model is compatible with many window functions, which demonstrates great flexibility to simulate the non-linear dopant drift phenomenon. Besides, the window function that we proposed in Xu et al. (2021) is used in this work due to its flexibility and simplicity.

The VTEAM model is shown as follows:

$$\frac{dw(t)}{dt} = \begin{cases} k_{\text{off}} \cdot \left(\frac{v(t)}{v_{\text{off}}} - 1\right)^{\alpha_{\text{off}}} \cdot f(x(t)), & 0 < v_{\text{off}} < v \\ 0, & v_{\text{on}} < v < v_{\text{off}} \\ k_{\text{on}} \cdot \left(\frac{v(t)}{v_{\text{on}}} - 1\right)^{\alpha_{\text{on}}} \cdot f(x(t)), & v < v_{\text{on}} < 0 \end{cases} \tag{11}$$

$$x(t) = \frac{w(t)}{W} \tag{12}$$

$$R(t) = R_{\text{on}} + (R_{\text{off}} - R_{\text{on}}) \cdot x(t) \tag{13}$$

$$v(t) = R(t) \cdot i(t) \tag{14}$$

where $w(t)$ is an internal state variable in $[0, W]$, W is the maximum value of $w(t)$, $x(t)$ is an internal state variable in $[0, 1]$, $f(x)$ is the proposed window function, $v(t)$ is the voltage across the memristor, $i(t)$ is the current passing through the memristor, $R(t)$ is the resistance of the memristor, and t is the time. The parameters v_{on} and v_{off} are threshold voltages, R_{on} and R_{off} are the corresponding resistances of the memristor when $w(t)$ is 0 and W, respectively. The parameters k_{on}, k_{off}, α_{on} and α_{off} are constants.

The proposed window function is provided as below:

$$\begin{aligned} f(x) &= j[\text{sgn}(-i) \cdot (x-1) + \text{stp}(-i)]^p \\ \text{sgn}(i) &= \begin{cases} 1, & i \geq 0 \\ -1, & i < 0 \end{cases} \quad \text{stp}(i) = \begin{cases} 1, & i \geq 0 \\ 0, & i < 0 \end{cases} \end{aligned} \tag{15}$$

where i is the memristor current, and j and p are two tuning parameters. The memristor current i is positive when the internal

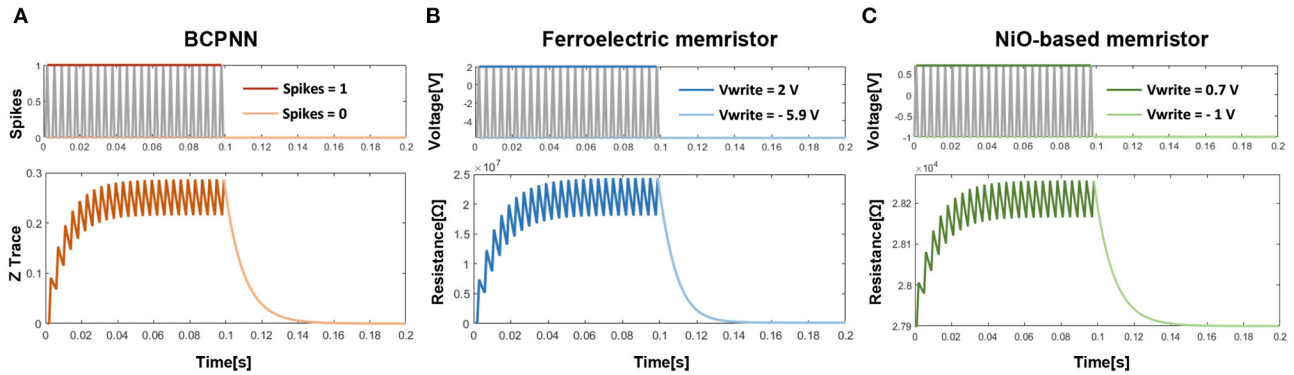

FIGURE 4 | Similarity between the BCPNN trace and the memristor nonlinearity: **(A)** The curve of Z trace. **(B)** The curve of resistance of the Ferroelectrc memristor. **(C)** The curve of resistance of the NiO-based memristor.

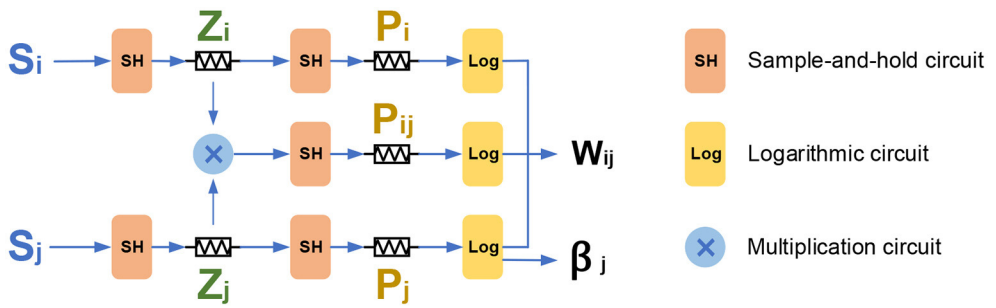

FIGURE 5 | The basic memristor-based architecture for the BCPNN learning rule.

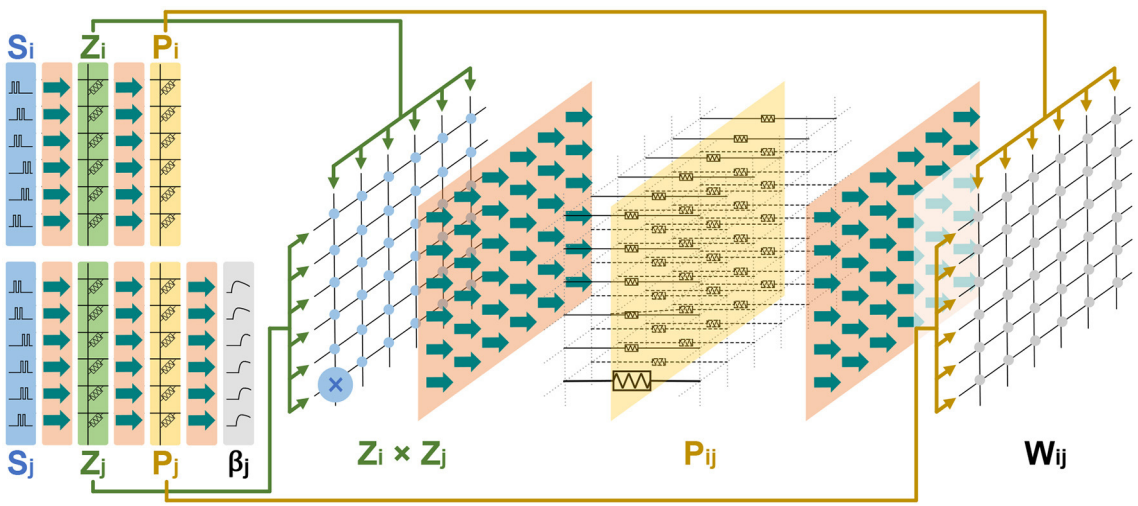

FIGURE 6 | The memristor-based architecture for an HCU with a configuration of 6 × 6.

state x is moving toward 1. The parameter j determines the magnitude, and the parameter p controls the decrease rate of the window function when approaching the boundaries. When p approaches 0, the non-linearity is weakened.

3.3. Similarity Between BCPNN Synaptic Traces and the Memristor Non-linearity

To explore the similarity of the BCPNN traces and the memristor non-linearity, the curve of the BCPNN trace (take

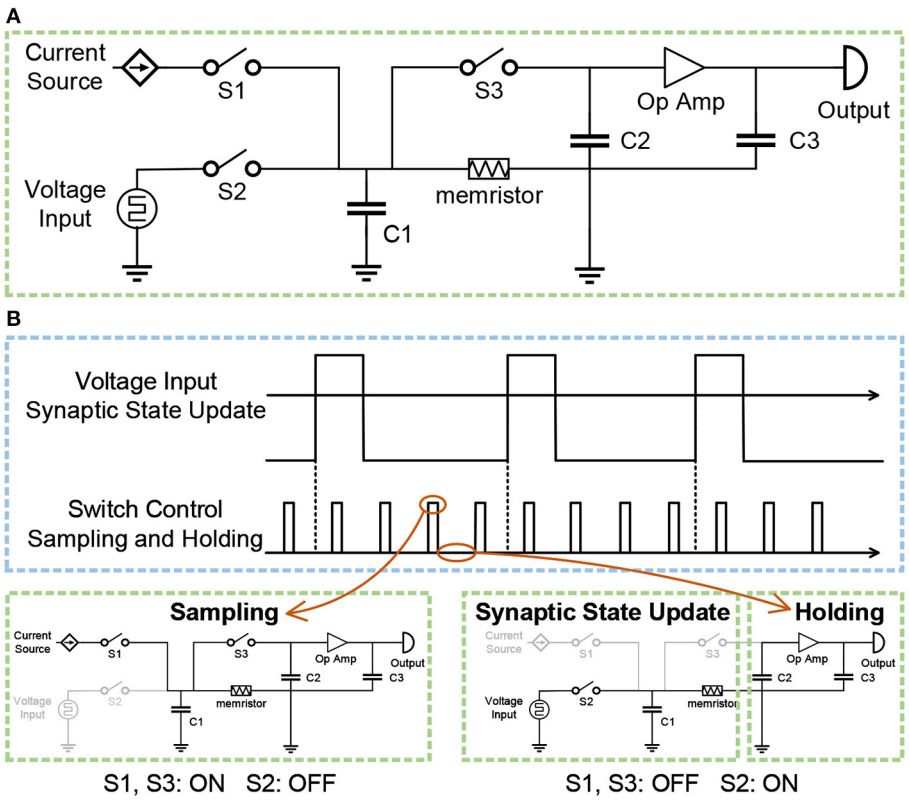

FIGURE 7 | (A) The diagram of the sample-and-hold circuit. **(B)** The switch between the sampling state and the holding state, controlled by switches S1, S2, and S3.

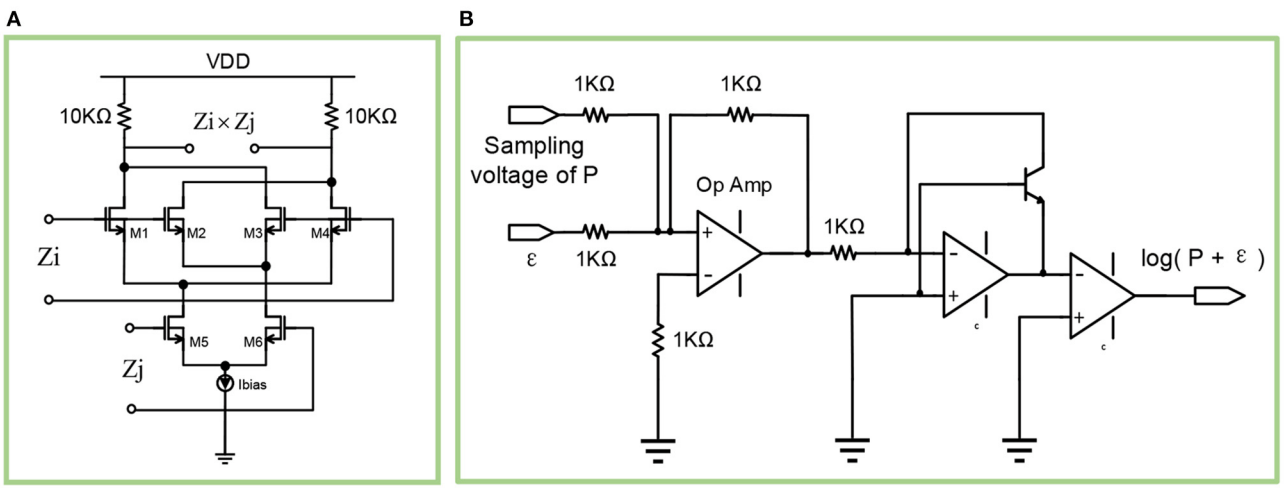

FIGURE 8 | (A) The diagram of the multiplication circuit. **(B)** The diagram of the logarithmic circuit.

Z trace as an example) and the curves of the resistances of two physical memristor devices are depicted in **Figure 4**. As shown in **Figure 4A**, the Z trace of BCPNN increases when there is a spike and decreases when there is no spike. While **Figures 4B,C** show that the resistances of the ferroelectric memristor (Chanthbouala et al., 2012) and the NiO-based memristor (Li et al., 2018) both increase when a positive voltage is applied and decrease when a negative voltage is applied. Therefore, a similarity can be found from **Figure 4** that both the BCPNN trace and the resistance of memristor change in a similar non-linear manner.

To further analyze the similarity between the BCPNN traces and the memristor non-linearity, their respective formulas are listed and compared. Take the Z trace of BCPNN as an example, when there is a spike or not, the formula of the Z trace is as follows:

$$\begin{aligned} S = 1 &: \; Z(t) = A \cdot Z(t-1) + B \\ S = 0 &: \; Z(t) = A \cdot Z(t-1) \end{aligned} \tag{16}$$

Correspondingly, when the voltage is positive or negative, the formula for the internal state variable of the memristor is as follows:

$$\begin{aligned} \nu\text{positive} &: \; w(t) = C \cdot w(t-1) + D \\ \nu\text{negative} &: \; w(t) = E \cdot w(t-1) \end{aligned} \tag{17}$$

where A, B, C, D and E are all constants expressed as:

$$\begin{cases} A = 1 - kz \\ B = kz \end{cases} \quad \begin{cases} C = 1 - dt \cdot \frac{k_{\text{off}}}{W} \cdot \left(\frac{v(t)}{v_{\text{off}}} - 1\right)^{\alpha_{\text{off}}} \\ \text{D} = dt \cdot k_{\text{off}} \cdot \left(\frac{v(t)}{v_{\text{off}}} - 1\right)^{\alpha_{\text{off}}} \\ E = 1 + dt \cdot \frac{k_{\text{on}}}{W} \cdot \left(\frac{v(t)}{v_{\text{on}}} - 1\right)^{\alpha_{\text{on}}} \end{cases} \tag{18}$$

Comparing formulas (16) and (17), a significant similarity can be observed, which also demonstrates the similarity between BCPNN traces and memristor non-linearity. Consequently, it is inspired that the non-linearity dopant drift phenomenon found in the memristor can be utilized to simulate the traces in the BCPNN learning rule.

4. MEMRISTOR-BASED ARCHITECTURE AND IMPLEMENTATION

4.1. Memristor-Based Architecture

The BCPNN learning rule involves the update of synaptic traces, the bias, and the weight. **Figure 5** presents the basic memristor-based architecture for the BCPNN learning rule. In the basic structure, five memristors are used to mimic the traces Z_i, Z_j, P_i, P_j, and P_{ij} respectively, and a multiplication circuit is used to calculate the product of Z_i and Z_j. Besides, five sample-and-hold circuits are used to provide the converted voltage input for the memristors, and three logarithmic circuits are used to calculate the weight w_{ij} and the bias β_j. The circuit design of the sample-and-hold circuit, logarithmic circuit, and the multiplication circuit will be explained in section 4.2.

As illustrated in **Figure 5**, the incoming presynaptic spike S_i is filtered into the Z_i trace through a sample-and-hold circuit. Then the Z_i trace is further filtered into the P_i trace with the same sample-and-hold circuit. Similarly, the postsynaptic spike S_j is first filtered into the Z_j trace, and then the Z_j trace is filtered into the P_j trace, both via a sample-and-hold circuit. Besides, the Z_i, Z_j traces are multiplied with each other, and then the obtained $Z_i \times Z_j$ is filtered into the P_{ij} through a sample-and-hold circuit. Last but not least, the P_{ij}, together with the P_i and P_j is used to calculate the weight W_{ij} through a logarithmic circuit. The P_j trace is calculated through a logarithmic circuit to obtain the value of bias β_j. It should be noted that although the E trace is removed in this work, it could be added without any issue by adding another level in the cascade if the E trace is needed.

TABLE 1 | Parameters for the Simulations.

Parameters	Value	Parameters	Value
BCPNN Model			
kz_i	1/11	kz_j	1/11
kp	1/500	ε	0.01
Memristor Model			
p	1	j	1
α_{off}	1	α_{on}	1
v_{off}	0.02 V	v_{on}	–0.02 V
R_{off}	200 kΩ	R_{on}	2 kΩ
k_{off}	21 nm/s	k_{on}	–28 nm/s
W	1 nm	w_{init}	0 nm
dt	1 ms		

What's more, the basic memristor-based architecture described above can be reused and scaled to build a memristor-based HCU that includes more synaptic traces. As a typical case for demonstration, **Figure 6** presents the memristor-based architecture for an HCU with a 6 × 6 configuration. The HCU contains a presynaptic vector of length 6, a postsynaptic vector of length 6, and a synaptic matrix of size 6 × 6.

It should be noted that the intention of **Figure 6** is to illustrate the scalability of the basic architecture in **Figure 5**. In this work, we focus on simulating and implementing the basic memristor-based architecture for the BCPNN learning rule in **Figure 5**.

4.2. Analog Circuit Implementation

4.2.1. Pre- and Post-synaptic Trace

The spike-based BCPNN is implemented with local synaptic state variables Z_i, Z_j, P_i, P_j and P_{ij}, which keep track of presynaptic, postsynaptic and synaptic activities. The implementation of pre- and post-synaptic trace Z_i, Z_j, P_i, P_j can be divided into two cascaded processing stages. In the first stage, the incoming pre- and post-synaptic spike trains S_i, S_j are low pass filtered into the Z_i, Z_j traces, with time constants τ_{zi} and τ_{zj}. In the second stage, the Z_i, Z_j traces are low pass filtered into the P_i, P_j traces with time constant τ_p. To implement the above two cascaded processing stages, two sample-and-hold circuits are cascaded in the analog circuit implementation. **Figure 7A** presents the diagram of the sample-and-hold circuit. The voltage input is used to represent the incoming spike trains S_i, S_j. The input spike is either 0 or 1, while the voltage input is either excitatory 193.2 mV or inhibitory –149.9 mV. The input of the current source is constant, which is used to transform the resistance of the memristor into a voltage value. Switches S1, S2, S3 are used to control the switch between the sampling state and the holding state. Three capacitors C1, C2, C3, are used to store voltage. Besides, an operational amplifier is utilized to amplify the voltage value.

Figure 7B illustrates the switch between the sampling state and the holding state. When switches S1, S3 are on and switch S2 is off, the circuit is in the sampling state. The constant current

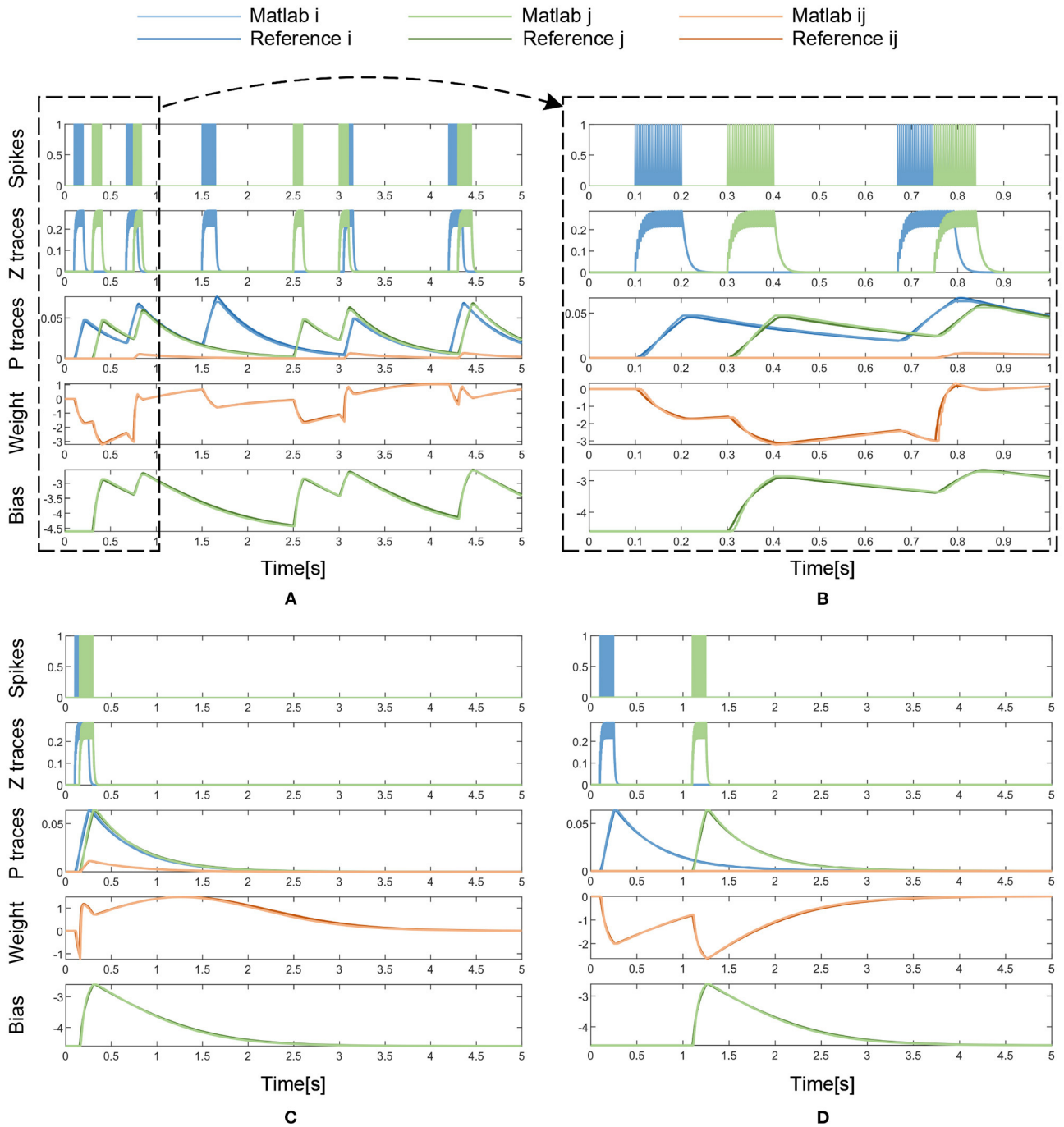

FIGURE 9 | Matlab simulation results with dense or sparse incoming spikes. **(A)** Dense Spikes: 5-s simulation. **(B)** Dense Spikes: 1-s simulation. **(C)** Sparse Spikes: presynaptic spike train S_i overlaps with postsynaptic spike train S_j. **(D)** Sparse Spikes: presynaptic spike train S_i is seperated with postsynaptic spike train S_j.

of the current source passes through the memristor to obtain the voltage, which is stored in capacitor C1. Since switch S3 is on, the voltage stored in capacitor C2 is equal to the voltage stored in capacitor C1. Therefore, the resistance of the memristor is converted into the corresponding voltage value, and the voltage value is stored in the capacitor C2, thus completing a sampling process. When switch S2 is on and switches S1, S3 are off, the left part of the circuit is responsible for the update of synaptic traces, and the right part of the circuit is in the holding state. The voltage source is the input excitation of the memristor, thereby changing the resistance of the memristor. At the same time, the sampled voltage stored in the capacitor C2 is amplified by the operational

amplifier, and the obtained voltage is stored in the capacitor C3 as the input voltage of the next-stage circuit. Similarly, the second stage adopts the same circuit, and the only difference is that the voltage input of the second circuit is the output of the first circuit rather than a voltage source.

4.2.2. Synaptic Trace Pij

The pre- and postsynaptic traces Z_i, Z_j, P_i, P_j can be obtained with the mentioned sample-and-hold circuit. However, to get the value of synaptic trace P_{ij}, an extra multiplier is required to calculate the product of Z_i and Z_j, as illustrated in formula (10).

As shown in **Figure 8A**, the multiplication circuit is based on the classic Gilbert cell. The resistance of the two resistors are both 10kΩ. The aspect ratios W/L for $M1$, $M2$, $M3$, $M4$ are 1μm/0.18μm, while the aspect ratios W/L for $M5$, $M6$ are 2μm/0.18μm. Besides, the bias voltage Vdc for Z_i and Z_j are 1.5 v and 1.3 v respectively.

4.2.3. Weight and Bias Computation

The three P traces P_i, P_j, P_{ij} represent the exponentially weighted moving averages of firing probability for presynaptic spikes, postsynaptic spikes, and spike co-activation respectively, which are used to compute the weight w_{ij} and the bias β_j. The calculation formula (5) for w_{ij} and β_j can be further rewritten as:

$$\begin{aligned} W_{ij} &= \log(P_{ij} + \varepsilon^2) - \log(P_i + \varepsilon) - \log(P_j + \varepsilon) \\ \beta_j &= \log(P_j + \varepsilon) \end{aligned} \tag{19}$$

The key to the calculation of w_{ij} and β_j lies in the logarithmic calculation of the sum of P trace and the constant ε, as shown in formula (19). Therefore, a logarithmic calculation circuit is required. As shown in **Figure 8B**, the sampling voltage of P trace (P_i, P_j, P_{ij}) is added with the constant parameter ε, then the sum is logarithmically calculated through a triode and an operational amplifier. Using such a circuit, the bias β_j can be obtained with an input pair of P_i and ε. Similarly, using three such circuits, whose input pairs are P_{ij} and ε^2, P_i and ε, P_j and ε respectively, the results of the three circuits can be used to get the value of weight w_{ij}.

5. EXPERIMENTAL RESULTS

In this section, we conduct simulations to verify the feasibility of the memristor-based implementation for the BCPNN learning rule at both the algorithm level and the circuit level. From the algorithmic perspective, we conducted simulations in Matlab. From the circuit-level perspective, we conducted SPICE-level simulations. The typical values of the parameters used in the simulations are shown in **Table 1**, including the parameters of the BCPNN model and the memristor model.

5.1. Matlab Simulation Results

To verify the effectiveness of the memristor-based solution for the BCPNN learning rule from an algorithmic perspective, a simulation of the Z traces, P traces, the weight w_{ij}, and the bias β_j is conducted using a model of the memristor device in Matlab. In the simulation, the results of the memristor-based solution are compared with those of the BCPNN reference model. The simulation lasts for 5 s with a simulation step of 1 ms, which is the simulation step in BCPNN.

TABLE 2 | Five-second simulation results with dense spikes in matlab.

Trace	Mean error	Max error	RMSE	Correlation coefficient
Z_i	0.0000	0.0000	0.0000	1.0000
Z_j	0.0000	0.0000	0.0000	1.0000
P_i	0.0015	0.0064	0.0019	0.9961
P_j	0.0013	0.0045	0.0015	0.9973
P_{ij}	0.0001	0.0008	0.0002	0.9984
w_{ij}	0.0418	1.4643	0.0862	0.9972
β_j	0.0408	0.2795	0.0489	0.9979

In **Figure 9** and **Table 2**, the simulation results are visualized and analyzed. **Figure 9A** presents the memristor-based 5-s Matlab simulation results with dense incoming spikes. To take a closer look at the difference between the memristor-based results and the reference model of BCPNN, the period from 0 to 1 s in **Figure 9A** has been enlarged, as shown in **Figure 9B**. With the same incoming pre- and post-synaptic spikes, the Z traces (Z_i, Z_j) of the memristor-based solution are the same as those of the BCPNN model. Therefore, in the Z traces part, the Z_i, Z_j curves of the two models completely coincide. As for the P traces (P_i, P_j, P_{ij}), the weight w_{ij} and the bias β_j, the curves of the two models are not the same but very close. In particular, simulations with sparse spikes are carried out to observe the change of the weight in the long-lasting silent state. When the presynaptic spike train S_i overlaps with the postsynaptic spike train S_j, the weight rises and finally decays to 0 in the long-lasting silent state, as shown in **Figure 9C**. Similarly, when the presynaptic spike train S_i is separated from the postsynaptic spike train S_j, the weight drops and gradually returns to 0 in a long-lasting silent state, as shown in **Figure 9D**. In the analysis of the simulation results, the average error, maximum error, Root Mean Square Error (RMSE), and correlation coefficient are used as the main evaluation metrics, as shown in **Table 2**. Due to the nonlinearity of the memristor, the memristor-based emulation of the BCPNN learning rule is accurate with a correlation coefficient of over 0.99.

5.2. SPICE Simulation Results

To further validate the feasibility of the memristor-based design from a circuit-level perspective, a SPICE-level simulation is conducted for the analog circuit implementation. In the SPICE simulation, the cascade circuit in **Figure 5** is implemented, where 5 sample-and-hold circuit modules, 3 logarithmic circuit modules, and 1 multiplication circuit module described in section 4.2 are used. The parameters for the BCPNN model and the memristor model used in the SPICE simulation are the same as those used in the Matlab simulation, as shown in **Table 1**. It is worth noting that the timestep in the Matlab simulation is 1 ms, while the timestep in the SPICE simulation is 100 ns because of the limitation of the timestep for transistors in the simulation environment.

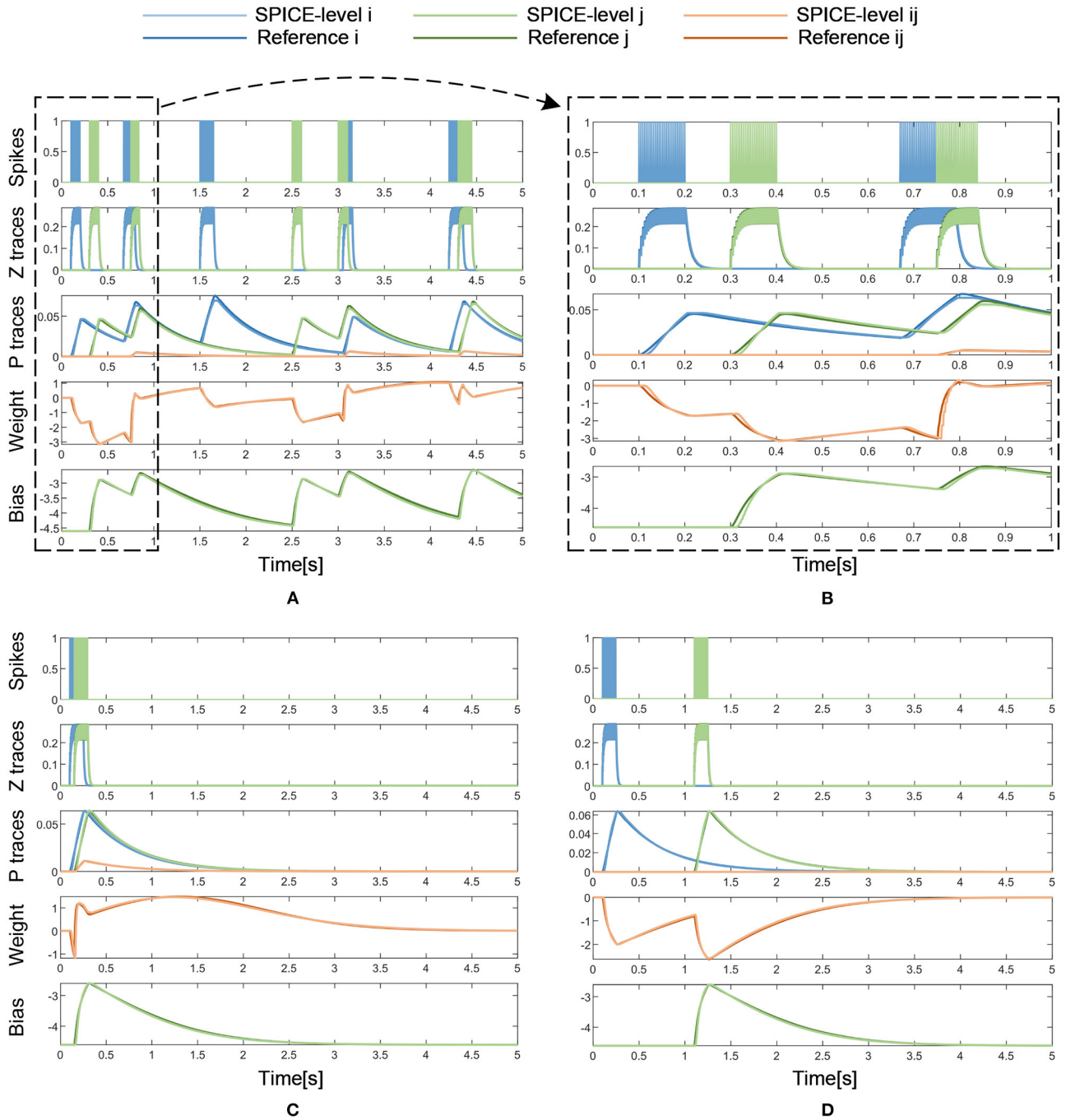

FIGURE 10 | Analog circuit implementation results with dense or sparse incoming spikes. **(A)** Dense Spikes: 5-s simulation. **(B)** Dense Spikes: 1-s simulation. **(C)** Sparse Spikes: presynaptic spike train S_i overlaps with postsynaptic spike train S_j. **(D)** Sparse Spikes: presynaptic spike train S_i is seperated with postsynaptic spike train S_j.

With the same input of pre- and post-synaptic spikes, the results of the SPICE-level simulation are compared with those of the reference model and the error is analyzed. **Figure 10A** presents the SPICE simulation results with the same dense incoming spikes as in the Matlab simulation. Similarly, the period from 0 to 1 s of the simulation results is magnified to show more details of the curves, as shown in **Figure 10B**. Besides, **Figures 10C,D** also demonstrates that the weight increases with a pair of correlated S_i and S_j and decreases with a pair of uncorrelated S_i and S_j. In a long-lasting silent state, the

weight returns to 0 eventually. As shown in **Table 3**, the SPICE simulation results of memristor-based solution demonstrate a fairly high degree of fit with the reference model of BCPNN, and a correlation coefficient of over 0.98 is achieved. All in all, it is validated that the memristor-based solution for BCPNN can achieve a high degree of fit with the reference BCPNN model in the analog circuit implementation.

TABLE 3 | Analog circuit implementation results of 5-s simulation with dense spikes.

Trace	Mean error	Max error	RMSE	Correlation coefficient
Z_i	0.0046	0.0909	0.0147	0.9830
Z_j	0.0041	0.0909	0.0138	0.9830
P_i	0.0014	0.0067	0.0018	0.9965
P_j	0.0012	0.0056	0.0015	0.9974
P_{ij}	0.0001	0.0011	0.0002	0.9981
w_{ij}	0.0432	1.5765	0.1003	0.9957
β_j	0.0483	0.3733	0.0631	0.9971

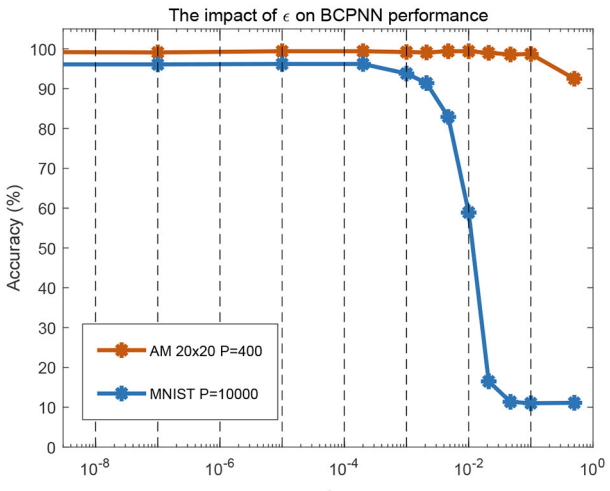

FIGURE 11 | Software simulation of the impact of ε on BCPNN performance in associative memory (AM) storage capacity and handwritten digit recognition (MNIST). For AM, a network configuration of $H \times M = 20 \times 20$ was tested for storage of 400 patterns. For MNIST, all 10,000 test patterns were used.

6. DISCUSSION

In this paper, the BCPNN learning rule is mapped to a memristor model and implemented with a memristor-based architecture. The similarity between the nonlinearity of the memristor and the trace update rule of BCPNN is explored and analyzed. The strong correlation between the simulated memristor-based BCPNN traces and the reference BCPNN traces has been validated in the Matlab simulation with a correlation coefficient over 0.99. Moreover, the analog circuit design of the memristor-based architecture is implemented, and the SPICE-level implementation for the BCPNN learning rule can achieve a decent emulation effect with a correlation coefficient of over 0.98.

6.1. Cumulative Error Analysis

The cumulative error of the memristor-based implementation can be analyzed from three aspects: the BCPNN algorithm, the memristor-based solution for BCPNN, and the analog circuit implementation. Firstly, as described before, BCPNN employs a correlation-based learning rule, which is robust and tolerant to the intrinsic noise and imprecision. BCPNN has proven to be able to function using lower precision (Vogginger et al., 2015). Secondly, as shown in **Table 2**, the memristor-based solution for the BCPNN learning rule presents a good simulation effect with the reference model. Moreover, it can be seen from **Figure 9** that the simulation effect does not deteriorate with the increasing simulation time, which means that there is no significant increase of cumulative error. Thirdly, the same is true for the analog circuit implementation, as can be seen in **Figure 10** and **Table 3**. Therefore, the cumulative error will not affect the stability of the memristor-based implementation for the BCPNN learning rule.

6.2. Setting of the Parameter ε

In the BCPNN model, the setting of the parameter ε has an impact on the performance of BCPNN-based tasks, as shown in **Figure 11**. With ε less than 0.001, good performance was achieved in an associative memory task and a standard machine learning classification benchmark (MNIST, LeCun et al., 1998). With ε equal to 0.01, the associative memory task still maintained good performance, but the performance in the MNIST task dropped a lot. For the experiments in section 5, the parameter ε was set to be 0.01, due to the limitation of the resolution of the logarithmic circuit. Later work will seek a higher-precision analog logarithmic circuit design or adopt digital methods to implement the logarithmic calculation of weight. In this way, the value of ε can be set to be less than 0.001, which can likely meet the requirement of most BCPNN-based tasks.

It should be noted that the intention of **Figure 11** is to analyze the impact of the value of ε (a parameter in the BCPNN learning rule) on the BCPNN performance from the perspective of software simulation. For the details about the working mechanism of the whole BCPNN network and how it implements associative memory tasks and practical recognition functions like MNIST classification, these works can be referred to (Johansson and Lansner, 2007; Meli and Lansner, 2013; Ravichandran et al., 2020, 2021a). The realization of the whole memristor-based BCPNN network and the algorithmic benchmarking is outside the scope of this paper and is what we plan to do in follow-up work.

6.3. Consideration of Device Variation

In this paper, we focus on mapping the BCPNN learning rule to a memristor model and validating the feasibility of the memristor-based implementation at the algorithm and circuit level. However, in reality, memristor-based structures suffer from device variations due to process variation and age degradation. These two factors lead to two different types of variations in the memristors devices (Park et al., 2013; Le et al., 2018). The first is spatial variations, where different devices in the crossbar react differently to the applied voltage, i.e., identical voltage pulse

can drive different devices to different resistances. The second is temporal variations, where the behavior of the same device will change over time. Neural networks can adapt to such variations by taking them into account during the training of the network. This method has been used in deep neural networks (Long et al., 2019) and spiking neural networks (Querlioz et al., 2013). The authors in Querlioz et al. (2013) identify non-supervised learning as one of the fundamental benefits of the STDP learning rule that helps when dealing with device variations. Previous work indicates that the BCPNN learning rule is amenable to low-precision implementation (Vogginger et al., 2015), and the cortical memory models have proven quite robust and tolerant to external as well as to intrinsic noise and imprecision in weights and unit biases. In the follow-up work, we will focus on the non-idealities of memristors and study to what extent BCPNN's robustness can absorb the non-idealities and what other measures could be needed to cope with the non-idealities.

7. FUTURE WORK

As a follow-up to this paper, we plan to rigorously address the issue of nonidealities in memristors. Specifically, its variance in both space and time. We plan to quantify the extent to which BCPNN's robustness can cope with the variances and if that is not sufficient, we will study how the behavior diverges and use these experiments to devise techniques to counter the nonidealities.

Next to addressing nonidealities, the aspect on our priority list is to make the implementation more complete. This would involve implementing the control logic in CMOS, data converters, drive circuits, etc. It is also obvious, a large stack of memristor devices cannot be driven by single drivers. For this reason, we plan to experiment with and find out fragments of memristor fabrics that can be stacked with scalable drive circuits. Besides the above, we might also need to implement compensation logic to deal with nonidealities in the spirit of pre-distortion.

Having a good grip on nonidealities and more complete implementation, we will then be in a position to have a fair comparison of performance and energy efficiency between a memristor-based implementation of BCPNN and pure digital implementations that we have been experimenting with (Stathis et al., 2020; Yang et al., 2020). Besides providing realistic comparison, such an experiment will also provide us with inputs to create a more optimized implementation.

Designing memristor-based systems, at present, is a circuit-level effort. This is cumbersome and not accessible to everyone. We plan to develop, a Lego-inspired design flow called SiLago (Hemani et al., 2017), to enable automation of memristor-based designs from higher abstractions. Some work toward building such infrastructure has happened for CMOS-based conventional digital designs (Gonzalez et al., 2021; Hemani et al., 2021). We plan to enhance this for the memristors.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

The initial idea proposed in the manuscript came from DS, AL, and AH. DW performed experiments and was responsible for the manuscript writing. JX proposed the methodology and guided the overall experimental design. LZ and FL contributed in the SPICE and Matlab simulations. ZZ provided supervision on DW, JX, and FL's work. AL, AH, DS, YY, PH, and ZZ helped with the refinement of this work and the revision of the manuscript. All authors contributed to the article and approved the submitted version.

FUNDING

This work was supported in part by the National Natural Science Foundation of China under Grant 61876039 and 62011530132 (NSFC-STINT project), and Shanghai Municipal Science and Technology Major Project No. 2021SHZDZX0103 and No. 2018SHZDZX01, and in part by the Shanghai Platform for Neuromorphic and AI Chip under Grant 17DZ2260900. In part, this work was financed by the mobility grant from STINT Sweden Dnr: CH2019-8357.

REFERENCES

Biolek, Z., Biolek, D., and Biolkova, V. (2009). SPICE model of memristor with nonlinear dopant drift. *Radioengineering* 18, 201–214. doi: 10.1049/el.2010.0358

Chanthbouala, A., Garcia, V., Cherifi, R. O., Bouzehouane, K., Fusil, S., Moya, X., et al. (2012). A ferroelectric memristor. *Nat. Mater.* 11, 860–864. doi: 10.1038/nmat3415

Chrysanthidis, N., Fiebig, F., Lansner, A., and Herman, P. (2021). Traces of semantization-from episodic to semantic memory in a spiking cortical network model. *bioRxiv*. doi: 10.1101/2021.07.18.452769

Chua, L. (1971). Memristor-The missing circuit element. *IEEE Trans. Circ. Theory* 18, 507–519. doi: 10.1109/TCT.1971.1083337

Ciregan, D., Meier, U., and Schmidhuber, J. (2012). "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (Providence, RI: IEEE), 3642–3649.

DeFelipe, J., Ballesteros-Yá nez, I., Inda, M. C., and Mu noz, A. (2006). Double-bouquet cells in the monkey and human cerebral cortex with special reference to areas 17 and 18. *Prog. Brain Res.* 154, 15–32. doi: 10.1016/S0079-6123(06)54002-6

Farahini, N., Hemani, A., Lansner, A., Clermidy, F., and Svensson, C. (2014). "A scalable custom simulation machine for the Bayesian confidence propagation neural network model of the brain," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)* (Singapore: IEEE), 578–585.

Fiebig, F., Herman, P., and Lansner, A. (2020). An indexing theory for working memory based on fast hebbian plasticity. *eNeuro* 7, 1–22. doi: 10.1523/ENEURO.0374-19.2020

Fiebig, F., and Lansner, A. (2017). A spiking working memory model based on Hebbian short-term potentiation. *J. Neurosci.* 37, 83–96. doi: 10.1523/JNEUROSCI.1989-16.2016

Gonzalez, J. A., Hemani, A., and Stathis, D. (2021). "Synthesis of predictable global NoC by abutment in synchoros VLSI design," in *Proceedings 15th*

IEEE/ACM International Symposium on Networks-on-Chip – NOCS 2021 (Virtual Conference).

Hemani, A., Jafri, S. M. A. H., and Masoumian, S. (2017). "Synchoricity and NOCs could make billion gate custom hardware centric SOCs affordable," in *Proceedings2017 Eleventh IEEE/ACM International Symposium on Networks-on-Chip (NOCS)* (Seoul), 1–10.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-,r., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process Mag.* 29, 82–97. doi: 10.1109/MSP.2012.2205597

Huang, Y., Liu, J., Harkin, J., McDaid, L., and Luo, Y. (2021). An memristor-based synapse implementation using BCM learning rule. *Neurocomputing* 423, 336–342. doi: 10.1016/j.neucom.2020.10.106

Hubel, D. H., and Wiesel, T. N. (1977). The functional architecture of the macaque visual cortex. *Ferrier Lect*. 198, 1–59. doi: 10.1098/rspb.1977.0085

Joglekar, Y. N., and Wolf, S. J. (2009). The elusive memristor: properties of basic electrical circuits. *Eur. J. Phys*. 30, 661–675. doi: 10.1088/0143-0807/30/4/001

Johansson, C., and Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Netw*. 20, 48–61. doi: 10.1016/j.neunet.2006.05.029

Johnson, S., Sundararajan, A., Hunley, D., and Strachan, D. (2010). Memristive switching of single-component metallic nanowires. *Nanotechnology* 21, 125204. doi: 10.1088/0957-4484/21/12/125204

Kvatinsky, S., Friedman, E. G., Kolodny, A., and Weiser, U. C. (2013). TEAM: ThrEshold adaptive memristor model. *IEEE Trans. Circ. Syst. I Regul. Pap*. 60, 211–221. doi: 10.1109/TCSI.2012.2215714

Kvatinsky, S., Ramadan, M., Friedman, E. G., and Kolodny, A. (2015). VTEAM: a general model for voltage-controlled memristors. *IEEE Trans. Circ. Syst. II Express Briefs* 62, 786–790. doi: 10.1109/TCSII.2015.2433536

Lansner, A., and Ekeberg, Ö. (1989). A one-layer feedback artificial neural network with a Bayesian learning rule. *Int. J. Neural Syst*. 1, 77–87. doi: 10.1142/S0129065789000499

Lansner, A., Hemani, A., and Farahini, N. (2014). "Spiking brain models: computation, memory and communication constraints for custom hardware implementation," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)* (Singapore), 556–562.

Lansner, A., and Holst, A. (1996). A higher order Bayesian neural network with spiking units. *Int. J. Neural Syst*. 7, 115–128. doi: 10.1142/S0129065796000816

Le, B. Q., Grossi, A., Vianello, E., Wu, T., Lama, G., Beigne, E., et al. (2018). Resistive RAM with multiple bits per cell: array-level demonstration of 3 bits per cell. *IEEE Trans. Electron. Devices* 66, 641–646. doi: 10.1109/TED.2018.2879788

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Li, C., Graves, C. E., Sheng, X., Miller, D., Foltin, M., Pedretti, G., et al. (2020a). Analog content-addressable memories with memristors. *Nat. Commun*. 11, 1–8. doi: 10.1038/s41467-020-15254-4

Li, J., Dong, Z., Luo, L., Duan, S., and Wang, L. (2020b). A novel versatile window function for memristor model with application in spiking neural network. *Neurocomputing* 405, 239–246. doi: 10.1016/j.neucom.2020.04.111

Li, Y., Chu, J., Duan, W., Cai, G., Fan, X., Wang, X., et al. (2018). Analog and digital bipolar resistive switching in solution-combustion-processed nio memristor. *ACS Appl. Mater. Interfaces* 10, 24598–24606. doi: 10.1021/acsami.8b05749

Long, Y., She, X., and Mukhopadhyay, S. (2019). "Design of reliable DNN accelerator with un-reliable ReRAM," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)* (Florence: IEEE), 1769–1774.

Lundqvist, M., Herman, P., and Lansner, A. (2011). Theta and gamma power increases and alpha/beta power decreases with memory load in an attractor network model. *J. Cogn. Neurosci*. 23, 3008–3020. doi: 10.1162/jocn_a_00029

Meli, C., and Lansner, A. (2013). A modular attractor associative memory with patchy connectivity and weight pruning. *Network* 24, 129–150. doi: 10.3109/0954898X.2013.859323

Nishitani, Y., Kaneko, Y., and Ueda, M. (2015). Supervised learning using spike-timing-dependent plasticity of memristive synapses. *IEEE Trans. Neural Netw. Learn. Syst*. 26, 2999–3008. doi: 10.1109/TNNLS.2015.2399491

Park, J.-K., Kim, S.-Y., Baek, J.-M., Seo, D.-J., Chun, J.-H., and Kwon, K.-W. (2013). "Analysis of resistance variations and variance-aware read circuit for cross-point ReRAM," in *2013 5th IEEE International Memory Workshop* (Monterey, CA: IEEE), 112–115.

Pickett, M. D., Strukov, D. B., Borghetti, J. L., Yang, J. J., Snider, G. S., Stewart, D. R., et al. (2009). Switching dynamics in titanium dioxide memristive devices. *J. Appl. Phys*. 106, 074508. doi: 10.1063/1.3236506

Podobas, A., Svedin, M., Chien, S. W., Peng, I. B., Ravichandran, N. B., Herman, P., et al. (2021). "Streambrain: an hpc framework for brain-like neural networks on cpus, gpus and fpgas," in *Proceedings of the 11th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, 1–6.

Prezioso, M., Merrikh-Bayat, F., Hoskins, B., Adam, G. C., Likharev, K. K., and Strukov, D. B. (2015). Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* 521, 61–64. doi: 10.1038/nature14441

Querlioz, D., Bichler, O., Dollfus, P., and Gamrat, C. (2013). Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Trans. Nanotechnol*. 12, 288–295. doi: 10.1109/TNANO.2013.2250995

Ravichandran, N. B., Lansner, A., and Herman, P. (2020). "Learning representations in bayesian confidence propagation neural networks," in *2020 International Joint Conference on Neural Networks (IJCNN)* (Glasgow: IEEE), 1–7.

Ravichandran, N. B., Lansner, A., and Herman, P. (2021a). "Brain-like approaches to unsupervised learning of hidden representations-a comparative study," in *International Conference on Artificial Neural Networks* (Bratislava: Springer), 162–173.

Ravichandran, N. B., Lansner, A., and Herman, P. (2021b). Semi-supervised learning with bayesian confidence propagation neural network. *arXiv [Preprint] arXiv:2106.15546*. doi: 10.14428/esann/2021.ES2021-156

Sandberg, A., Lansner, A., Petersson, K., and Ekeberg. (2002). A Bayesian attractor network with incremental learning. *Network* 13, 179–194. doi: 10.1080/net.13.2.179.194

Stathis, D., Chaourani, P., Jafri, S. M. A. H, and Hemani, A. (2021). "Clock tree generation by abutment in synchoros VLSI design," in *Proceedings 2021 Nordic Circuits and Systems Conference (NorCAS)* (Oslo).

Stathis, D., Sudarshan, C., Yang, Y., Jung, M., Weis, C., Hemani, A., et al. (2020). eBrainII: a 3 kW realtime custom 3D DRAM integrated ASIC implementation of a biologically plausible model of a human scale cortex. *J. Signal Process Syst*. 92, 1323–1343. doi: 10.1007/s11265-020-01562-x

Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *Nature* 453, 80–83. doi: 10.1038/nature06932

Tully, P. J., Hennig, M. H., and Lansner, A. (2014). Synaptic and nonsynaptic plasticity approximating probabilistic inference. *Front. Synaptic. Neurosci*. 6:8. doi: 10.3389/fnsyn.2014.00008

Vogginger, B., Schüffny, R., Lansner, A., Cederström, L., Partzsch, J., and Höppner, S. (2015). Reducing the computational footprint for real-time BCPNN learning. *Front. Neurosci*. 9:2. doi: 10.3389/fnins.2015.00002

Wijesinghe, P., Ankit, A., Sengupta, A., and Roy, K. (2018). An all-memristor deep spiking neural computing system: a step toward realizing the low-power stochastic brain. *IEEE Trans. Emerg. Top. Comput. Intell*. 2, 345–358. doi: 10.1109/TETCI.2018.2829924

Xu, J., Huan, Y., Yang, K., Zhan, Y., Zou, Z., and Zheng, L.-R. (2018). Optimized near-zero quantization method for flexible memristor based neural network. *IEEE Access* 6:29320–29331. doi: 10.1109/ACCESS.2018.2839106

Xu, J., Wang, D., Li, F., Zhang, L., Stathis, D., Yang, Y., et al. (2021). "A memristor model with concise window function for spiking brain-inspired computation," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (Washington DC: IEEE), 1–4.

Yang, J. J., Pickett, M. D., Li, X., Ohlberg, D. A., Stewart, D. R., and Williams, R. S. (2008). Memristive switching mechanism for metal/oxide/metal nanodevices. *Nat. Nanotechnol*. 3, 429–433. doi: 10.1038/nnano.2008.160

Yang, Y., Stathis, D., Jord ao, R., Hemani, A., and Lansner, A. (2020). Optimizing BCPNN learning rule for memory access. *Front. Neurosci*. 14:878. doi: 10.3389/fnins.2020.00878

Yao, P., Wu, H., Gao, B., Tang, J., Zhang, Q., Zhang, W., et al. (2020). Fully hardware-implemented memristor convolutional neural network. *Nature* 577, 641–646. doi: 10.1038/s41586-020-1942-4

Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. *arXiv [Preprint] arXiv:1702.01923*.

Zhao, Z., Qu, L., Wang, L., Deng, Q., Li, N., Kang, Z., et al. (2020). A memristor-based spiking neural network with high scalability and learning efficiency. *IEEE Trans. Circ. Syst. II Express Briefs* 67, 931–935. doi: 10.1109/TCSII.2020.2980054

Zhou, E., Fang, L., Liu, R., and Tang, Z. (2019). Area-efficient memristor spiking neural networks and supervised learning method. *Sci. China Inf. Sci*. 62, 1–3. doi: 10.1007/s11432-018-9607-8

Zidan, M. A., Jeong, Y., Lee, J., Chen, B., Huang, S., Kushner, M. J., et al. (2018). A general memristor-based partial differential equation solver. *Nat. Electron*. 1, 411–420. doi: 10.1038/s41928-018-0100-6

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.