frontiers in Computational Neuroscience

BRIEF RESEARCH REPORT
published: 25 February 2020
doi: 10.3389/fncom.2020.00014

Probabilistic Circuits for Autonomous Learning: A Simulation Study

Jan Kaiser, Rafatul Faria, Kerem Y. Camsari and Supriyo Datta*

Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, United States

Modern machine learning is based on powerful algorithms running on digital computing platforms and there is great interest in accelerating the learning process and making it more energy efficient. In this paper we present a fully autonomous probabilistic circuit for fast and efficient learning that makes no use of digital computing. Specifically we use SPICE simulations to demonstrate a clockless autonomous circuit where the required synaptic weights are read out in the form of analog voltages. This allows us to demonstrate a circuit that can be built with existing technology to emulate the Boltzmann machine learning algorithm based on gradient optimization of the maximum likelihood function. Such autonomous circuits could be particularly of interest as standalone learning devices in the context of mobile and edge computing.

Keywords: on-device learning, Boltzmann machine algorithm, probabilistic computing, magnetic tunnel junction (MTJ), machine learning, analog circuit

OPEN ACCESS

Edited by:
Lei Deng,
University of California, Santa Barbara, United States

Reviewed by:
Youhui Zhang,
Tsinghua University, China
Jeongjun Lee,
Texas A&M University, United States

Correspondence:
Jan Kaiser
kaiser32@purdue.edu

Received: *25 November 2019*
Accepted: *03 February 2020*
Published: *25 February 2020*

Citation:
Kaiser J, Faria R, Camsari KY and Datta S (2020) Probabilistic Circuits for Autonomous Learning: A Simulation Study. Front. Comput. Neurosci. 14:14. doi: 10.3389/fncom.2020.00014

1. INTRODUCTION

Machine learning, inference, and many other emerging applications (Schuman et al., 2017) make use of stochastic neural networks comprising (1) a binary stochastic neuron (BSN) (Ackley et al., 1985; Neal, 1992) and (2) a synapse that constructs the inputs I_i to the ith BSN from the outputs m_j of all other BSNs.

The output m_i of the ith BSN fluctuates between +1 and −1 with a probability controlled by its input

$$m_i(t + \tau_N) = \text{sgn}\left[\tanh\left(I_i(t)\right) - r\right] \tag{1}$$

where r represents a random number in the range $[-1, +1]$, and τ_N is the time it takes for a neuron to provide a stochastic output m_i in accordance with a new input I_i[1].

Usually the synaptic function, $I_i(\{m\})$ is linear and is defined by a set of weights W_{ij} such that

$$I_i(t + \tau_S) = \sum_j W_{ij} m_j(t) \tag{2}$$

where τ_S is the time it takes to recompute the inputs $\{I\}$ everytime the outputs $\{m\}$ change. Typically Equations (1), (2) are implemented in software, often with special accelerators for the synaptic function using GPU/TPUs (Schmidhuber, 2015; Jouppi, 2016).

The time constants τ_N and τ_S are not important when Equations (1) and (2) are implemented on a digital computer using a clock to ensure that neurons are updated sequentially and the synapse is updated between any two updates. But they play an important role in clockless operation of autonomous hardware that makes use of the natural physics of specific systems to implement Equations (1) and (2) approximately. A key advantage of using BSNs is that Equation (1) can be

[1]Equation (1) can be written in binary notation with a unit step function and a sigmoid function.

implemented compactly using stochastic magnetic tunnel junctions (MTJs) as shown in Camsari et al. (2017a,b), while resistive or capacitive crossbars can implement Equation (2) (Hassan et al., 2019a). It has been shown that such hardware implementations can operate autonomously without clocks, if *the BSN operates slower than the synapse*, that is, if $\tau_N >> \tau_S$ shown by Sutton et al. (2019).

Stochastic neural networks defined by Equations (1) and (2) can be used for inference whereby the weights W_{ij} are designed such that the system has a very high probability of visiting configurations defined by $\{m\} = \{v\}_n$, where $\{v\}_n$ represents a specified set of patterns. However, the most challenging and time-consuming part of implementing a neural network is not the inference function, but the learning required to determine the correct weights W_{ij} for a given application. This is commonly done using powerful cloud-based processors and there is great interest in accelerating the learning process and making it more energy efficient so that it can become a routine part of mobile and edge computing.

In this paper we present a new approach to the problem of fast and efficient learning that makes no use of digital computing at all. Instead it makes use of the natural physics of a fully autonomous probabilistic circuit composed of standard electronic components like resistors, capacitors, and transistors along with stochastic MTJs.

We focus on a fully visible Boltzmann machine (FVBM), a form of stochastic recurrent neural network, for which the most common learning algorithm is based on the gradient ascent approach to optimize the maximum likelihood function (Carreira-Perpinan and Hinton, 2005; Koller and Friedman, 2009). We use a slightly simplified version of this approach, whereby the weights are changed incrementally according to

$$W_{ij}(t+\Delta t) = W_{ij}(t) + \epsilon[v_i v_j - m_i m_j - \lambda W_{ij}(t)]$$

where ϵ is the learning parameter and λ is the regularization parameter (Ng, 2004). The term $v_i v_j$ is the correlation between the *i*th and the *j*th entry of the training vector $\{v\}_n$. The term $m_i m_j$ corresponds to the sampled correlation taken from the model's distribution. The advantage of this network topology is that the learning rule is local since it only requires information of the two neurons *i* and *j* connected by weight W_{ij}. In addition, the learning rule can tolerate stochasticity for example in the form of sampling noise which makes it an attractive algorithm to use for hardware machine learning (Carreira-Perpinan and Hinton, 2005; Fischer and Igel, 2014; Ernoult et al., 2019).

For our autonomous operation we replace the equation above with its continuous time version (τ_L: learning time constant)

$$\frac{dW_{ij}}{dt} = \frac{v_i v_j - m_i m_j - \lambda W_{ij}}{\tau_L} \tag{3}$$

which we translate into an RC circuit by associating W_{ij} with the voltage on a capacitor C driven by a voltage source $(V_{v,ij} - V_{m,ij})$ with a series resistance R (**Figure 1**):

$$C\frac{dV_{ij}}{dt} = \frac{V_{v,ij} - V_{m,ij} - V_{ij}}{R} \tag{4}$$

with $v_i v_j = V_{v,ij}/(V_{DD}/2)$ and $m_i m_j = V_{m,ij}/(V_{DD}/2)$. From **Figure 1** and comparing Equations (3), (4) it is easy to see how the weights and the learning and regularization parameters are mapped into circuit elements: $W_{ij} = A_v V_{ij}/V_0$, $\lambda = V_0/(A_v V_{DD}/2)$, and $\tau_L = \lambda RC$ where A_v is the voltage gain of OP3 in **Figure 1** and V_0 is the reference voltage of the BSN. For proper operation the learning time scale τ_L has to be much larger than the neuron time τ_N to be able to collect enough statistics throughout the learning process.

A key element of this approach is the representation of the weights W with *voltages* rather than with programmable *resistances* for which memristors and other technologies are still in development (Li et al., 2018b). By contrast the charging of capacitors is a textbook phenomenon, allowing us to design a learning circuit that can be built today with established technology. The idea of using capacitor voltages to represent weights in neural networks has been presented by several authors for different network topologies in analog learning circuits (Schneider and Card, 1993; Card et al., 1994; Kim et al., 2017; Sung et al., 2018). The use of capacitors has the advantage of having a high level of linearity and symmetry for the weight updates during the training process (Li et al., 2018a).

In section 2, we will describe such a learning circuit that emulates Equations (1)–(3). The training images or patterns $\{v\}_n$ are fed in as electrical signals into the input terminals, and the synaptic weights W_{ij} can then be read out in the form of voltages from the output terminals. Alternatively the values can be stored in a non-volatile memory from which they can subsequently be read and used for inference. In section 3, we will present SPICE simulations demonstrating the operation of this autonomous learning circuit.

2. METHODS

The autonomous learning circuit has three parts where each part represents one of the three Equations (1)–(3). On the left hand side of **Figure 1**, the training data is fed into the circuit by supplying a voltage $V_{v,ij}$ which is given by the *i*th entry of the bipolar training vector v_i multiplied by the *j*th entry of the training vector v_j and scaled by the supply voltage $V_{DD}/2$. The training vectors can be fed in sequentially or as an average of all training vectors. The weight voltage V_{ij} across capacitor C follows Equation (4) where $V_{v,ij}$ is compared to voltage $V_{m,ij}$ which represents correlation of the outputs of BSNs m_i and m_j. Voltage $V_{m,ij}$ is computed in the circuit by using an XNOR gate that is connected to the output of BSN *i* and BSN *j*. The synapse in the center of the circuit connects weight voltages to neurons according to Equation (2). Voltage V_{ij} has to be multiplied by 1 or -1 depending on the current value of m_j. This is accomplished by using a switch which connects either the positive or the negative node of V_{ij} to the operational amplifiers OP1 and OP2. Here, OP1 accumulates all negative contributions and OP2 accumulates all positive contributions of the synaptic function. The differential amplifier OP3 takes the difference between the output voltages of OP2 and OP1 and amplifies the voltage by amplification factor A_v. This voltage conversion is used to control the voltage level of V_{ij} in relation to the input voltage of each BSN. The voltage level at the input of the BSN is fixed by the reference voltage

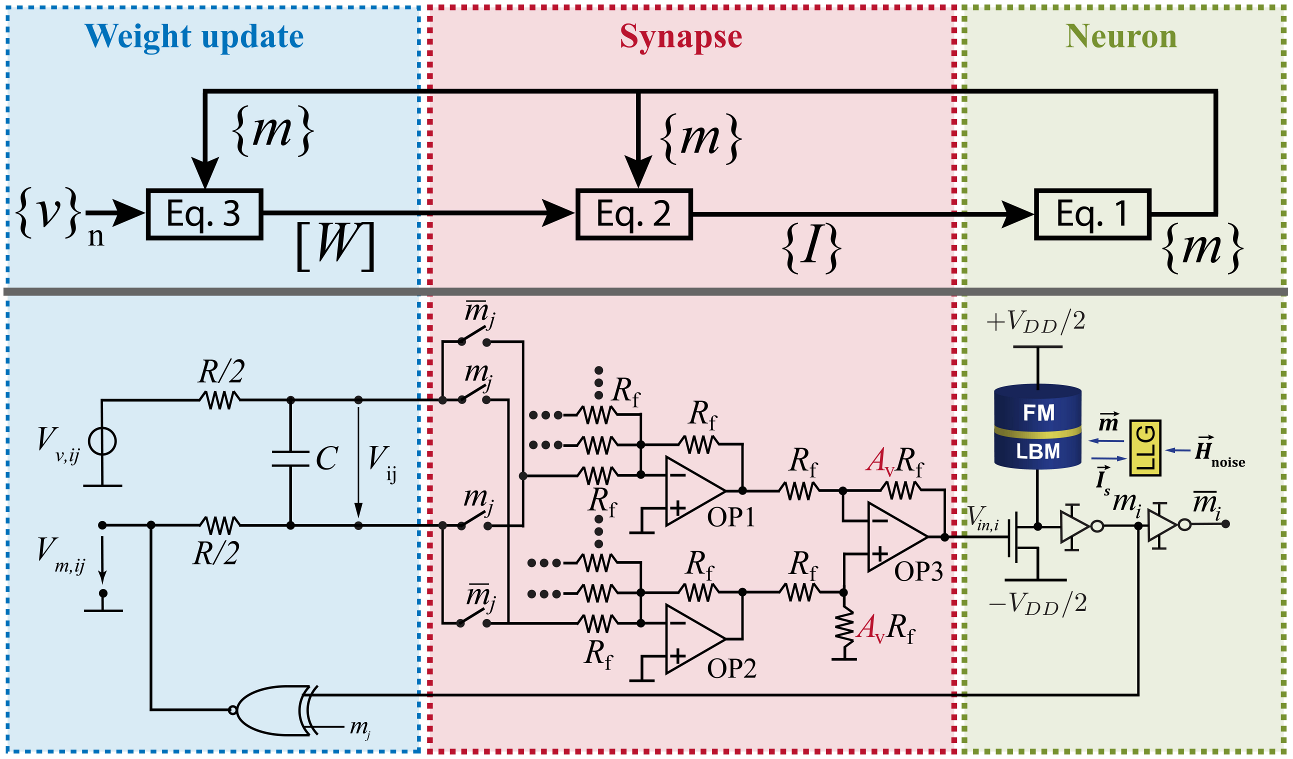

FIGURE 1 | Clockless learning circuit designed to emulate Equations (1)–(3) autonomously.

of the BSN which is V_0. However, the voltage level of V_{ij} can be adjusted and utilized to adjust the regularization parameter λ in the learning rule (Equation 3). The functionality of the BSN is described by Equation (1) where the dimensionless input is given by $I_i(t) = V_{i,in}(t)/V_0$. This relates the voltage V_{ij} to the dimensionless weight by $W_{ij} = A_v V_{ij}/V_0$. The hardware implementation of the BSN uses a stochastic MTJ in series with a transistor as presented by Camsari et al. (2017b). Due to thermal fluctuations of the low-barrier magnet (LBM) of the MTJ the output voltage of the MTJ fluctuates randomly but with the right statistics given by Equation (1). The time dynamics of the LBM can be obtained by solving the stochastic Landau-Lifshitz-Gilbert (LLG) equation. Due to the fast thermal fluctuations of the LBM in the MTJ, Equation (1) can be evaluated on a subnanosecond timescale leading to fast generation of samples (Hassan et al., 2019b; Kaiser et al., 2019b).

Figure 1 just shows the hardware implementation of one weight and one BSN. The size of the whole circuit depends on the size of the training vector N. For every entry of the training vector one BSN is needed. The number of weights which is the number of RC-circuits is given by $N(N-1)/2$ where every connection between BSNs is assumed to be reciprocal. To learn biases another N RC-circuits are needed.

The learning process is captured by Equations (3) and (4). The whole learning process has similarity with the software implementation of persistent contrastive divergence (PCD) (Tieleman, 2008) since the circuit takes samples from the model's distribution ($V_{m,ij}$) and compares it to the target distribution ($V_{v,ij}$) without reinitializing the Markov Chain after a weight update. During the learning process voltage V_{ij} reaches a constant average value where $\frac{dV_{ij}}{dt} \approx 0$. This voltage $V_{ij} = V_{ij,\text{learned}}$ corresponds to the learned weight.

For inference the capacitor C is replaced by a voltage source of voltage $V_{ij,\text{learned}}$. Consequently, the autonomous circuit will compute the desired functionality given by the training vectors. In general, training and inference have to be performed on identical hardware in order to learn around variations (see **Supplementary Material** for more details). It is important to note that in inference mode this circuit can be used for optimization by performing electrical annealing. This is done by increasing all weight voltages V_{ij} by the same factor over time. In this way the ground state of a Hamiltonian like the Ising Hamiltonian can be found (Sutton et al., 2017; Camsari et al., 2019).

3. RESULTS

In this section the autonomous learning circuit in **Figure 1** is simulated in SPICE. We show how the proposed circuit can be used for both inference and learning. As examples, we demonstrate the learning on a full adder (FA) and on 5×3 digit images. The BSN models are simulated in the framework developed by Camsari et al. (2015). For all SPICE simulations the

TABLE 1 | Truth table of a full adder.

A v_1	B v_2	C_{in} v_3	S v_4	C_{out} v_5	Dec	P_{Ideal}
−1	−1	−1	−1	−1	0	0.125
−1	−1	1	1	−1	6	0.125
−1	1	−1	1	−1	10	0.125
−1	1	1	−1	1	13	0.125
1	−1	−1	1	−1	18	0.125
1	−1	1	−1	1	21	0.125
1	1	−1	−1	1	25	0.125
1	1	1	1	1	31	0.125

Every 0 in the binary representation of the full adder is replaced by −1 in the bipolar representation. "Dec" represents the decimal conversion of each line. P_{Ideal} is the ideal probability distribution were every line's probability is p=1/8=0.125.

following parameters are used for the stochastic MTJ in the BSN implementation: Saturation magnetization M_S = 1,100 emu/cc, LBM diameter D = 22 nm, LBM thickness l = 2 nm, TMR = 110%, damping coefficient α = 0.01, temperature T = 300 K and demagnetization field $H_D = 4\pi M_S$ with $V = (D/2)^2\pi l$. For the transistors, 14 nm HP-FinFET Predictive Technology Models (PTM)[2] are used with fin number fin = 1 for the inverters and fin = 2 for XNOR-gates. Ideal operational amplifiers and switches are used in the synapse. The characteristic time of the BSNs τ_N is in the order of 100 ps (Hassan et al., 2019b) and much larger than the time it takes for the synaptic connections, namely the resistors and operational amplifiers, to propagate BSN outputs to neighboring inputs. It has to be noted that in principle other hardware implementations of the synapse for computing Equation (2) could be utilized as long as the condition $\tau_N \gg \tau_S$ is satisfied.

3.1. Learning Addition

As first training example, we use the probability distribution of a full adder. The FA has 5 nodes and 10 weights that have to be learned. In the case of the FA training, no biases are needed. The probability distribution of a full adder with bipolar variables is shown in **Table 1**. To learn this distribution the correlation terms $v_i v_j$ in the learning rule have to be fed into the voltage node $V_{v,ij}$. The correlation is dependent on what training vector/truth table line is fed in. For the second line of the truth table for example $v_1 v_2 = -1 \cdot -1 = 1$ and $v_1 v_3 = -1 \cdot 1 = -1$ with A being the first node, B the second node and so on. In **Figure 2B** the correlation $v_1 v_5$ is shown. For the sequential case the value of $v_1 v_5$ is obtained by circling through all lines of the truth table where each training vector is shown for 1 ns. A and C_{out} in **Table 1** only differ in the fourth and fifth line for which $v_1 v_5 = -1$. For all other cases $v_1 v_5 = 1$. The average of all lines is shown as red solid line. **Figure 2A** shows the weight voltage V_{ij} with $i = 1$ and $j = 5$ for FA learning and the first 1,000 ns of training. The following learning parameters have been used for the FA: τ_L = 62.5 ns where C = 1 nF and R = 5 kΩ, A_v = 10, and R_f = 1 MΩ. This choice of learning parameters

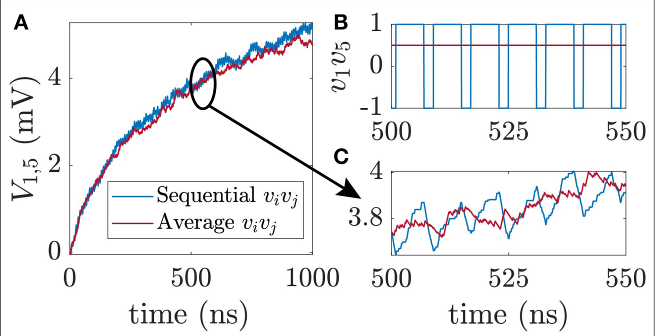

FIGURE 2 | Feeding of training data into the circuit. **(A)** Weight voltage $V_{1,5}$ over time for sequential and average feeding in of the correlation between visible unit i and visible unit j for training a full adder. **(B)** Correlation $v_1 v_5$ vs. time t. All eight lines of the truth table of a full adder are cycled through where every vector is shown for time T = 1 ns at a time. **(C)** Enlarged version of subfigure **(A)**. For sequential feeding in of data, the voltage change in $v_1 v_5$ directly affects $V_{1,5}$.

ensures that $\tau_L \gg \tau_N$. Due to the averaging effect of the RC-circuit both sequential and average feeding of the training vector result in similar learning behavior as long as the RC-constant is much larger than the timescale of sequential feeding. **Figure 2C** shows the enlarged version of **Figure 2A**. For the sequential feeding, voltage $V_{1,5}$ changes substantially every time $v_1 v_5$ switches to −1.

At the start of training all weight voltages are initialized to 0 V and the probability distribution is uniform. The training is performed for 5,500 ns. In **Figure 3A** the ideal probability distribution of the FA P_{Ideal} is shown together with the normalized histogram P_{SPICE} of the sampled BSN configurations taken from the last 500 ns of learning and compared to the ideal distribution P_{Ideal}. The training vector is fed in as an average. For P_{SPICE} the eight trained configurations of **Table 1** are the dominant peaks. To monitor the training process, the Kullback-Leibner divergence between the trained and the ideal probability distribution $KL(P_{\text{Ideal}}||P_{\text{SPICE}}(t))$ is plotted as a function of training time t in **Figure 3B** where $P_{\text{SPICE}}(t)$ is the normalized histogram taken over 500 ns. P_{SPICE} at t = 0 corresponds to the histogram taken from t = 0 to t = 500 ns. During training the KL divergence decreases over time until it reaches a constant value at about 0.1. It has to be noted that after the weight matrix is learned correctly for a fully visible Boltzmann machine, the KL divergence can be reduced further by increasing all weights uniformly by a factor I_0 which corresponds to inverse temperature of the Boltzmann machine (Aarts and Korst, 1989). **Figure 3** shows that the probability distribution of a FA can be learned very fast with the proposed autonomous learning circuit. In addition, the learning performance is robust when components of the circuit are subject to variation. In the **Supplementary Material**, additional figures of the learning performance are shown when the diameter of the magnet and the resistances of the RC-circuits are subject to variation. The robustness against variations can be explained by the fact that the circuit can learn around variations. BSNs using LBMs under variations

[2]http://ptm.asu.edu/

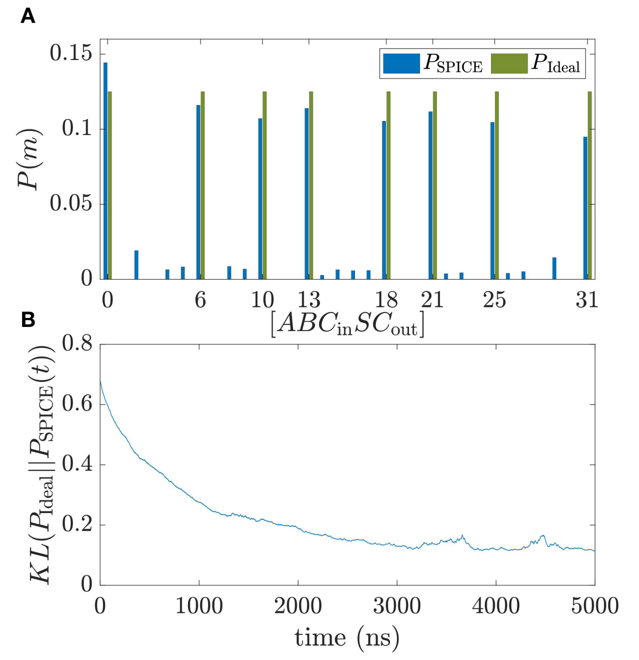

FIGURE 3 | Training of a full adder in SPICE. **(A)** Probability distribution of a trained full adder is compared to the ideal distribution with binary inputs A, B, C_{in} and outputs S and C_{out}. The training is performed for 5,500 ns. Blue bars are the probability distribution extracted from SPICE simulations by creating a histogram of the configurations of m over the last 500 ns of training. **(B)** Kullback–Leibler divergence between P_{SPICE} obtained by doing a moving average of 500 ns and the target distribution defined as $KL(P_{\mathrm{Ideal}}||P_{\mathrm{SPICE}}(t)) = \sum_m P_{\mathrm{Ideal}}(m)\log(P_{\mathrm{Ideal}}(m)/P_{\mathrm{Train}}(m,t))$. Following parameters have been used in the simulations: $C = 1$ nF, $R = 5$ kΩ, $R_F = 1$ MΩ, $A_v = 10$, $V_0 = 50$ mV.

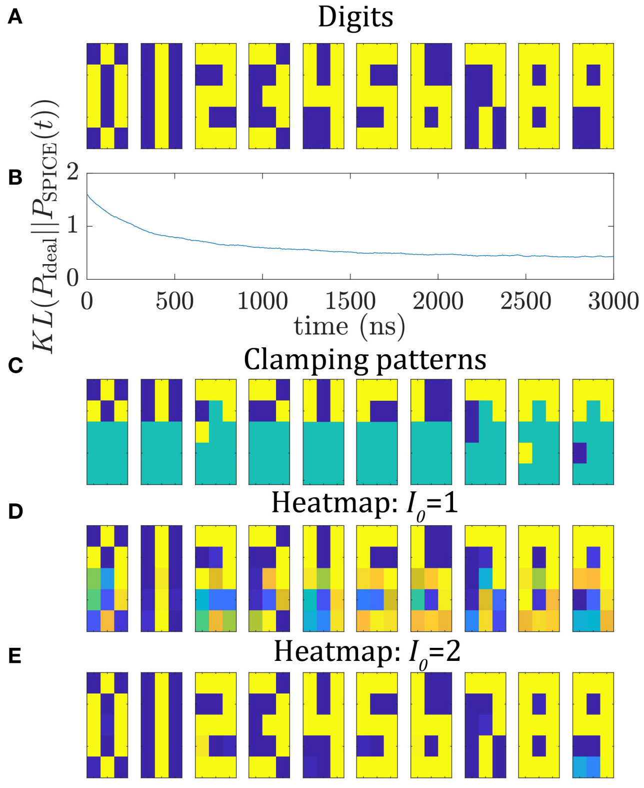

FIGURE 4 | Training and testing of 5 × 3 digit images. **(A)** 5 × 3 digit images from 0 to 9. **(B)** Kullback Leibner divergence during training for 3,000 ns using the autonomous circuit. **(C–E)** Image completion: For inference, six unique pixels are clamped for every digit (as shown in **C**). **(D,E)** show the heatmap of BSN outputs during inference for running the circuit for 100 ns for **(D)** $I_0 = 1$ and **(E)** $I_0 = 2$.

have also been analyzed by Abeed and Bandyopadhyay (2019) and Drobitch and Bandyopadhyay (2019).

3.2. Learning Image Completion

As second example, the circuit is utilized to train 10 5 × 3 pixel digit images shown in **Figure 4A**. Here, 105 reciprocal weights and 15 biases have to be learned. The network is trained for 3,000 ns and the bipolar training data is fed in as average of the 10 $v_i v_j$ terms for every digit. The same learning parameters as in the previous section are used here. In **Figure 4B**, the KL divergence is shown as a function of time between the SPICE histogram and the ideal probability distribution where the ideal distribution has 10 peaks with each peak being 10% for each digit. Most of the learning happens in the first 1,500 ns of training, however, the KL divergence still reduces slightly during the later parts of learning. After 3,000 ns the KL divergence reaches a value of around 0.5.

For inference we replace the capacitor by a voltage source where every voltage is given by the previously learned voltage V_{ij}. The circuit is run for 10 instances where every instance has a unique clamping pattern of 6 pixels representing one of the 10 digits. The clamped inputs are shown in **Figure 4C**. The input of a clamped BSN is set to $\pm V_{DD}/2$. Each instance is run for 100 ns and the outputs of the BSNs are monitored. The BSNs fluctuate between the configurations given by the learned probability distribution. In **Figure 4D**, the heat map of the output of the BSNs is shown. For every digit the most likely configuration is given by the trained digit image. To illustrate this point, the amount of BSN fluctuations is reduced by increasing the learned weight voltages by a factor of $I_0 = 2$. The circuit is again run in inference mode for 100 ns with the same clamping patterns. In **Figure 4E** the heatmap is shown. The circuit locks into the learned digit configuration. This shows that in inference mode the circuit can be utilized for image completion.

4. DISCUSSION

In this paper we have presented a framework for mapping a continuous version of Boltzmann machine learning rule (Equation 3) to a clockless autonomous circuit. We have shown full SPICE simulations to demonstrate the feasibility of this circuit running without any digital component with the learning parameters set by circuit parameters. Due to the fast BSN operation, samples are drawn at subnanosecond speeds leading to fast learning, as such the learning speed should be at least multiple orders of magnitudes faster compared to other computing platforms (Adachi and Henderson, 2015; Korenkevych et al., 2016; Terenin et al., 2019). The advantage of this autonomous architecture is that it produces random numbers naturally and does not rely on pseudo random number generators like linear-feedback shift register (LFSRs) (which are for example used in Bojnordi and Ipek, 2016). These LFSRs have overhead and are not as compact and efficient as the hardware BSN used in this paper. As shown by Borders et al. (2019), typical LFSRs need about 10x more energy per flip and more than 100x more area than an MTJ-based BSN. Another advantage of this approach is that the interfacing with digital hardware only needs to be performed after the learning has been completed. Hence, no expensive analog-to-digital conversion has to be performed during learning. We believe this approach could be extended to other energy based machine learning algorithms like equilibrium propagation introduced by Scellier and Bengio (2017) to design autonomous circuits. Such standalone learning devices could be particularly of interest in the context of mobile and edge computing.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

JK and SD wrote the paper. JK performed the simulations. RF helped setting up the simulations. KC developed the simulation modules for the BSN. All authors discussed the results and helped refine the manuscript.

FUNDING

This work was supported in part by ASCENT, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. KC gratefully acknowledges support from Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant CCF-0939370.

ACKNOWLEDGMENTS

This manuscript has been released as a prepint at arXiv (Kaiser et al., 2019a).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fncom.2020.00014/full#supplementary-material

REFERENCES

Aarts, E., and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. New York, NY: John Wiley & Sons, Inc.

Abeed, M. A., and Bandyopadhyay, S. (2019). Low energy barrier nanomagnet design for binary stochastic neurons: design challenges for real nanomagnets with fabrication defects. *IEEE Magn. Lett.* 10, 1–5. doi: 10.1109/LMAG.2019.2929484

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cogn. Sci.* 9, 147–169. Available online at: https://www.sciencedirect.com/science/article/abs/pii/S0364021385800124

Adachi, S. H., and Henderson, M. P. (2015). Application of quantum annealing to training of deep neural networks. *arXiv*: 1510.06356. Available online at: https://arxiv.org/abs/1510.06356

Bojnordi, M. N., and Ipek, E. (2016). "Memristive Boltzmann machine: a hardware accelerator for combinatorial optimization and deep learning," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (IEEE), 1–13. Available online at: https://ieeexplore.ieee.org/abstract/document/7446049

Borders, W. A., Pervaiz, A. Z., Fukami, S., Camsari, K. Y., Ohno, H., and Datta, S. (2019). Integer factorization using stochastic magnetic tunnel junctions. *Nature* 573, 390–393. doi: 10.1038/s41586-019-1557-9

Camsari, K. Y., Chowdhury, S., and Datta, S. (2019). Scalable emulation of sign-problem–free Hamiltonians with room-temperature p-bits. *Phys. Rev. Appl.* 12:034061. doi: 10.1103/PhysRevApplied.12.034061

Camsari, K. Y., Faria, R., Sutton, B. M., and Datta, S. (2017a). Stochastic p -bits for invertible logic. *Phys. Rev. X* 7:031014. doi: 10.1103/PhysRevX.7.031014

Camsari, K. Y., Ganguly, S., and Datta, S. (2015). Modular approach to spintronics. *Sci. Rep.* 5:10571. doi: 10.1038/srep10571

Camsari, K. Y., Salahuddin, S., and Datta, S. (2017b). Implementing p-bits with embedded MTJ. *IEEE Elect. Device Lett.* 38, 1767–1770. doi: 10.1109/LED.2017.2768321

Card, H. C., Schneider, C. R., and Schneider, R. S. (1994). Learning capacitive weights in analog CMOS neural networks. *J. VLSI Signal Process.* 8, 209–225. doi: 10.1007/BF02106447

Carreira-Perpinan, M. A., and Hinton, G. E. (2005). "On contrastive divergence learning," in *Aistats*, Vol. 10, 33–40. Available online at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.221.8829&rep=rep1&type=pdf#page=42

Drobitch, J. L., and Bandyopadhyay, S. (2019). Reliability and scalability of p-bits implemented with low energy barrier nanomagnets. *IEEE Magn. Lett.* 10, 1–4. doi: 10.1109/LMAG.2019.2956913

Ernoult, M., Grollier, J., and Querlioz, D. (2019). Using memristors for robust local learning of hardware restricted Boltzmann machines. *Sci. Rep.* 9:1851. doi: 10.1038/s41598-018-38181-3

Fischer, A., and Igel, C. (2014). Training restricted Boltzmann machines: an introduction. *Pattern Recogn.* 47, 25–39. doi: 10.1016/j.patcog.2013.05.025

Hassan, O., Camsari, K. Y., and Datta, S. (2019a). Voltage-driven building block for hardware belief networks. *IEEE Design Test* 36, 15–21. doi: 10.1109/MDAT.2019.2897964

Hassan, O., Faria, R., Camsari, K. Y., Sun, J. Z., and Datta, S. (2019b). Low-barrier magnet design for efficient hardware binary stochastic neurons. *IEEE Magn. Lett.* 10, 1–5. doi: 10.1109/LMAG.2019.2910787

Jouppi, N. (2016). *Google Supercharges Machine Learning Tasks With TPU Custom Chip*. Google Cloud Platform Blog. Google. Available online at: https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip

Kaiser, J., Faria, R., Camsari, K. Y., and Datta, S. (2019a). Probabilistic circuits for autonomous learning: a simulation study. *arXiv [Preprint]. arXiv: 1910.06288.* Available online at: https://arxiv.org/abs/1910.06288

Kaiser, J., Rustagi, A., Camsari, K. Y., Sun, J. Z., Datta, S., and Upadhyaya, P. (2019b). Subnanosecond fluctuations in low-barrier nanomagnets. *Phys. Rev. Appl.* 12:054056. doi: 10.1103/PhysRevApplied.12.054056

Kim, S., Gokmen, T., Lee, H. M., and Haensch, W. E. (2017). "Analog CMOS-based resistive processing unit for deep neural network training," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (IEEE), 422–425. Available online at: https://ieeexplore.ieee.org/abstract/document/8052950

Koller, D., and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press. Available online at: https://mitpress.mit.edu/books/probabilistic-graphical-models

Korenkevych, D., Xue, Y., Bian, Z., Chudak, F., Macready, W. G., Rolfe, J., et al. (2016). Benchmarking quantum hardware for training of fully visible Boltzmann machines. *arXiv*:1611.04528. Available online at: https://arxiv.org/abs/1611.04528

Li, Y., Kim, S., Sun, X., Solomon, P., Gokmen, T., Tsai, H., et al. (2018a). "Capacitor-based cross-point array for analog neural network with record symmetry and linearity," in *2018 IEEE Symposium on VLSI Technology*, 25–26. Available online at: https://ieeexplore.ieee.org/document/8510648

Li, Y., Wang, Z., Midya, R., Xia, Q., and Yang, J. J. (2018b). Review of memristor devices in neuromorphic computing: materials sciences and device challenges. *J. Phys. D Appl. Phys.* 51:503002. doi: 10.1088/1361-6463/aade3f

Neal, R. M. (1992). Connectionist learning of belief networks. *Artif. Intell.* 56, 71–113. doi: 10.1016/0004-3702(92)90065-6

Ng, A. Y. (2004). "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *Twenty-First International Conference on Machine Learning - ICML '04* (Banff, AB: ACM Press), 78.

Scellier, B., and Bengio, Y. (2017). Equilibrium propagation: bridging the gap between energy-based models and backpropagation. *Front. Comput. Neurosci.* 11:24. doi: 10.3389/fncom.2017.00024

Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Netw.* 61, 85–117. doi: 10.1016/j.neunet.2014.09.003

Schneider, C. R., and Card, H. C. (1993). Analog CMOS deterministic Boltzmann circuits. *IEEE J. Solid State Circ.* 28, 907–914. doi: 10.1109/4.231327

Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., et al. (2017). A survey of neuromorphic computing and neural networks in hardware. *arXiv [Preprint]. arXiv:1705.06963 [cs]*. Available online at: https://arxiv.org/abs/1705.06963

Sung, C., Hwang, H., and Yoo, I. K. (2018). Perspective: a review on memristive hardware for neuromorphic computation. *J. Appl. Phys.* 124:151903. doi: 10.1063/1.5037835

Sutton, B., Camsari, K. Y., Behin-Aein, B., and Datta, S. (2017). Intrinsic optimization using stochastic nanomagnets. *Sci. Rep.* 7:44370. doi: 10.1038/srep44370

Sutton, B., Faria, R., Ghantasala, L. A., Camsari, K. Y., and Datta, S. (2019). Autonomous probabilistic coprocessing with petaflips per second. *arXiv [Preprint]. arXiv:1907.09664*. Available online at: https://arxiv.org/abs/1907.09664

Terenin, A., Dong, S., and Draper, D. (2019). GPU-accelerated Gibbs sampling: a case study of the Horseshoe Probit model. *Stat. Comput.* 29, 301–310. doi: 10.1007/s11222-018-9809-3

Tieleman, T. (2008). "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *Proceedings of the 25th International Conference on Machine learning - ICML '08* (Helsinki: ACM Press), 1064–1071. doi: 10.1145/1390156.1390290

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.