BNN Training Algorithm with Ternary Gradients and BNN based on MRAM Array

Yuya Fujiwara
Department of Electric Engineering
Tokyo University of Science
Tokyo, Japan
4322544@ed.tus.ac.jp

Takayuki Kawahara
Department of Electric Engineering
Tokyo University of Science
Tokyo, Japan
kawahara@ee.kagu.tus.ac.jp

***Abstract*—Internet of Things (IoT) devices have only limited computing resources, which means we need to reduce the scale of operation circuits and energy consumption to build a neural network (NN). The binarized neural network (BNN) and computing-in-memory (CiM) have been proposed to fulfill these requirements, and recently, magnetic random access memory (MRAM), next-generation memory for CiM-based architectures has attracted interest. In this study, we utilize a CiM architecture based on an MRAM array to build a BNN on the edge side. We also implement an XNOR gate on our MRAM array using voltage-controlled magnetic anisotropy (VCMA)-based magnetization switching to reduce the scale of the multiply-and-accumulate (MAC) operation circuits by half. Further, we propose a BNN training algorithm utilizing ternary gradients to enable both training and inference on the edge side using only binary weights and ternary gradients. Experiments on the MNIST dataset showed that our MRAM array can achieve an accuracy of around 80%.**

***Keywords*—MRAM, BNN, SOT, VCMA**

I. Introduction

One of the key challenges in the modern Internet of Things (IoT) society is how to build a neural network (NN) on the edge side. Since edge devices have only limited computing resources, it is first necessary to reduce the scale of the operation circuits and energy consumption. On the software side, the binarized neural network (BNN) has been proposed to address this challenge [1]. A BNN is an NN that utilizes binary value inputs, weights, and activations to enable multiply-and-accumulate (MAC) operation, which is a core operation on an NN, using only XNOR gates and bit counts. On the hardware side, computing-in-memory (CiM) is the main proposal [2]. CiM is based on the concept that memory devices can be utilized not only for memory but also as a processor. CiM-based BNN hardware has been proposed to implement BNNs on edge devices [3]. Recently, magnetic random access memory (MRAM), a next-generation memory based on the CiM architecture, has been attracting interest thanks to its low power consumption and fast write operation [4].

In this study, we propose a BNN training algorithm that utilizes ternary gradients to enable both training and inference on an MRAM array using only binary weights and ternary gradients. For training the MRAM array, we utilize a voltage-controlled magnetic anisotropy (VCMA)-based XNOR gate, which is an improved version of the earlier VCMA-based XOR gate [5], to reduce the scale of the MAC operation circuits. We implemented the proposed BNN training algorithm on our MRAM array and evaluated its performance on the MNIST dataset.

II. Training and Inference Phases of BNN

A. Training of BNN

In the training phase of a BNN, we can use stochastic gradient decent (SGD), Adam, or AdaGrad, the same as with a regular NN. However, real-valued weights and real-valued gradients of the weights are necessary to update the weights [1]. In addition, batch normalization must be used for normalizing the activation values to properly train the network [1], [6]. Thus, processing of real numbers is necessary in the BNN training phase (Fig. 1). On the other hand, in the inference phase of the BNN, real-valued weights and real-valued gradients are not necessary because we do not have to update weights or calculate gradients in this phase. However, batch normalization is necessary, so we still need real number processing in the inference phase.

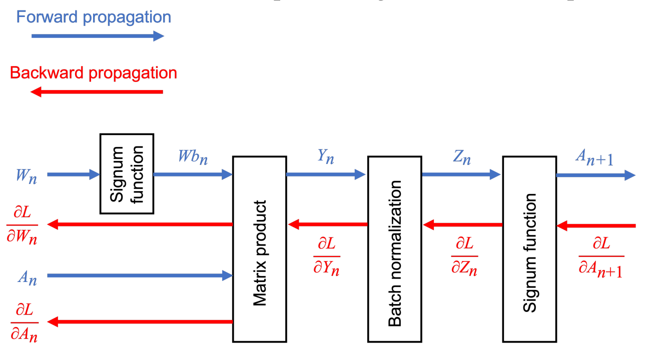
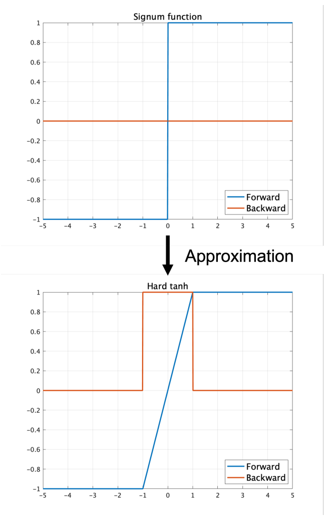

Forward propagation
Backward propagation
W_n
Signum function
Wb_n
$\frac{\partial L}{\partial W_n}$
A_n
$\frac{\partial L}{\partial A_n}$
Matrix product
Y_n
$\frac{\partial L}{\partial Y_n}$
Batch normalization
Z_n
$\frac{\partial L}{\partial Z_n}$
Signum function
A_{n+1}
$\frac{\partial L}{\partial A_{n+1}}$

Fig. 1. Conventional BNN architecture. In the training phase, real-valued weights and real-valued gradients are necessary if SGD, Adam, AdaGrad, etc. are used. In the inference phase, only binarized weights are used.

B. Signum function and straight-through estimator

In a BNN, activation values are binarized by a signum function, but since the gradients of the signum function are mostly 0 (Fig. 2), they are mostly not propagated in backward propagation. We therefore use hard tanh instead of the signum function in the backward propagation [1]. This approximation is known as a straight-through estimator (STE) [1]. The gradient values of hard tanh are 1 if the input values are in a closed interval [–1, 1] (Fig. 2), which enables the gradients to be moderately propagated in the backward propagation.



$$y = \mathrm{sign}(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

$$\frac{dy}{dx} = \begin{cases} \infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

$$y = \text{hard}\tanh(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \leq 1 \\ 1, & x > 1 \end{cases}$$

$$\frac{dy}{dx} = \begin{cases} 1, & -1 \leq x \leq 1 \\ 0, & |x| > 1 \end{cases}$$

Fig. 2. Straight-through estimator. Signum function mostly does not have gradients whereas hard tanh does have gradients in [−1, 1].

III. SOT and VCMA

A. *Spin orbit torque*

Spin orbit torque (SOT) is generated on a layered structure that consists of a magnetic tunnel junction (MTJ) and heavy metal thin film. If an electric current flows through the heavy metal layer, a spin current with a direction vertical to the electric current is created by the spin hall effect (SHE) (Fig. 3-(a)). As a result, a certain spin is injected into the free layer of the MTJ, which gives magnetization of the free layer torque, so that the magnetization is parallel with the injected spin. This torque is called SOT (Fig. 3-(b)). The direction of the injected spin depends on which direction the electric current flowing in the heavy metal layer. Thus, when an appropriate electric current is flowing in the heavy metal layer, we can control the magnetization direction of the free layer of the MTJ. However, on an MTJ that has perpendicular magnetic anisotropy (PMA), the direction of the injected spin generated by SOT is vertical to the magnetization of the free layer, and SOT cannot switch the magnetization of the free layer. Thus, the magnetization randomly switches as determined by the thermal fluctuation [7]. If we want to switch the magnetization deterministically, we need to apply an external magnetic field in the direction of the electric current to break the symmetry [8]. In addition, when we use an electric current that is smaller than a threshold current, the magnetization switches stochastically. The switching probability depends on the current magnitude [4].

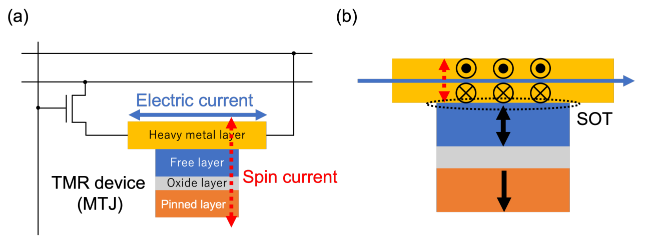


Fig. 3. Spin orbit torque. (a) The SOT-MRAM cell structure has a heavy metal layer, a free layer with a magnetization that can switch, an oxide layer, and a pinned layer with a magnetization that is fixed. (b) SOT is created by spin current originating from SHE, which is generated by electric current flows through the heavy metal layer.

B. *Voltage-controlled magnetic anisotropy*

Voltage-controlled magnetic anisotropy (VCMA) is an effect that can control the interfacial magnetic anisotropy of ferromagnetic thin film with PMA when a bias voltage is applied to it [9] (Fig. 4-(a)). On ferromagnetic thin film, in a stable state, the direction of the magnetization is either up or down, and there is an energy barrier between up and down states. We therefore apply energy to the ferromagnetic thin film that is larger than the energy barrier. The magnitude of this energy barrier is in proportion to the interfacial magnetic anisotropy. As a result, on ferromagnetic thin film with PMA, we can control the magnitude of the energy barrier by means of the VCMA effect (Fig. 4-(b)). If we apply a threshold voltage to ferromagnetic thin film with PMA that modulates the energy barrier to around 0, its magnetization can go back and forth between the up and down states and if an extra horizontal magnetic field is applied to it, the magnetization begins precession around this extra field (Fig. 4-(c)). If we stop the bias voltage halfway through the precession period, the magnetization is switching [10] (Fig. 4-(d)). We use this switching method on the free layer of the MTJ to enable the MRAM write operation. However, this switching method can only reverse the state of the magnetization, so we need to keep the previous state to memorize the optional value on MRAM.

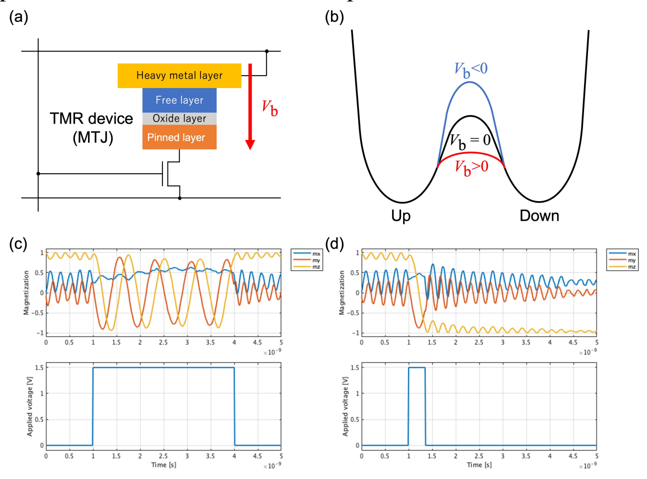


Fig. 4. VCMA effect. (a) VCMA-MRAM cell structure. (b) Energy barrier controlled by applied voltage. (c) Magnetization and applied voltage pulse of the free layer when threshold voltage is applied to the MRAM cell (pulse width = 3 ns). (d) Magnetization and applied voltage pulse of the free layer when threshold voltage is applied the MRAM cell (pulse width = half of precession period).

IV. Proposed Algorithm and MRAM Cell

A. *BNN training algorithm using ternary gradients*

When training a BNN, real-valued weights and real-valued gradients are generally used in update operation, and batch normalization is used in the forward propagation. In this study, we propose a new BNN training algorithm utilizing ternary gradients to reduce these real-value calculations.

In our proposal, we use only binary inputs, binary weights, and ternary gradients, so real-valued weights and real-valued gradients never appear in either the training or the inference phase. We introduce three new techniques into the BNN training algorithm, as follows.

The first is to ternarize gradients. In our proposal, after every calculation of gradients, the gradients are ternarized to {–1, 0, +1} (Fig. 5). Thus, in backward propagation, the gradients propagating to the back are always "–1" or "0" or "+1". However, we implement backward propagation of the loss function layer and the softmax layer at the same time, using cross entropy error as the loss function, which means the results of backward propagation of the loss function layer and the softmax layer is difference in value between the teacher data and the output data. This difference propagates back after ternarizing. In other words, if the teacher data equal the output data, "0" propagates back, and if the teacher data do not equal the output data, "–1" or "+1" propagates back, as determined by the sign of the difference.

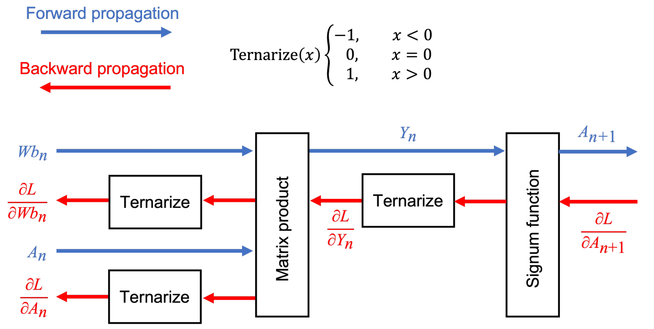

Forward propagation
Backward propagation
$\mathrm{Ternarize}(x)\begin{cases}-1, & x<0\\ 0, & x=0\\ 1, & x>0\end{cases}$
Wb_n
$\frac{\partial L}{\partial Wb_n}$
Ternarize
A_n
$\frac{\partial L}{\partial A_n}$
Ternarize
Matrix product
Y_n
$\frac{\partial L}{\partial Y_n}$
Ternarize
Signum function
A_{n+1}
$\frac{\partial L}{\partial A_{n+1}}$

Fig. 5. Proposed BNN architecture. The gradients are ternarized every layer.

The second technique is to improve the STE. In general STE, hard tanh is used instead of the signum function so that gradients can exit in closed interval [–1, 1]. In our proposal, we change the range in which the gradients are exist, to close interval [–*n*, *n*]. In this case, the gradients whose value is 1/*n* exist (Fig. 6). In our algorithm, we do not use batch normalization, which means the distribution of the activation values might be biased and, our training thus involves the risk of losing the gradients. Thus, we use a new STE and *n* as a hyperparameter to control the ease of gradient propagation.

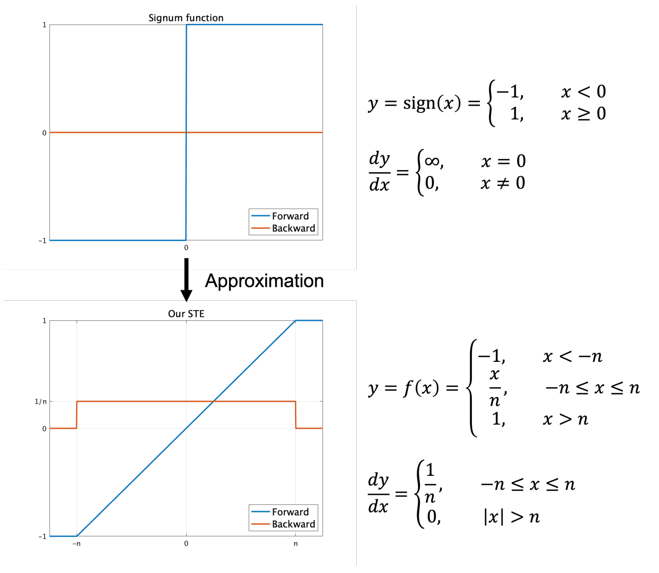


Fig. 6. Proposed straight-through estimator, where gradients exist in close interval [–*n*, *n*].

The third technique is to update the parameters stochastically. If we use only binary weights and ternary gradients without real-valued weights or real-valued gradients, the binary weight is either reversed or kept. Thus, when the value of a ternary gradient equals the value of a binary weight, the value of the binary weight is reversed, and in other cases, the value of the binary weight is kept the same (TABLE I). However, if all the binary weights that equal ternary gradients are reversed, the structure of the network would change too much. We therefore decide that if the binary weights equal the ternary gradients, the binary weights are reversed stochastically to absorb the change of the network. We utilize this reverse probability as a hyperparameter to train the network.

TABLE I. TRUTH TABLE OF UPDATE PHASE OF ALGORITHM

Previous weight	Ternary gradient	New weight
–1	–1	+1
–1	0	–1
–1	+1	–1
+1	–1	+1
+1	0	+1
+1	+1	–1

B. *MRAM cell on BNN*

The BNN enables the XNOR gate and bit count to implement MAC operation because the inputs, weights, and activations are binarized values. Conventionally, an MRAM-based XNOR gate requires two complementary MRAM cells [3]. An MRAM cell-based XOR gate using VCMA magnetization switching has thus been proposed [5].

In our work, we utilize an MRAM cell-based XNOR gate based on this MRAM cell-based XOR gate, as shown in Fig. 7-(a).

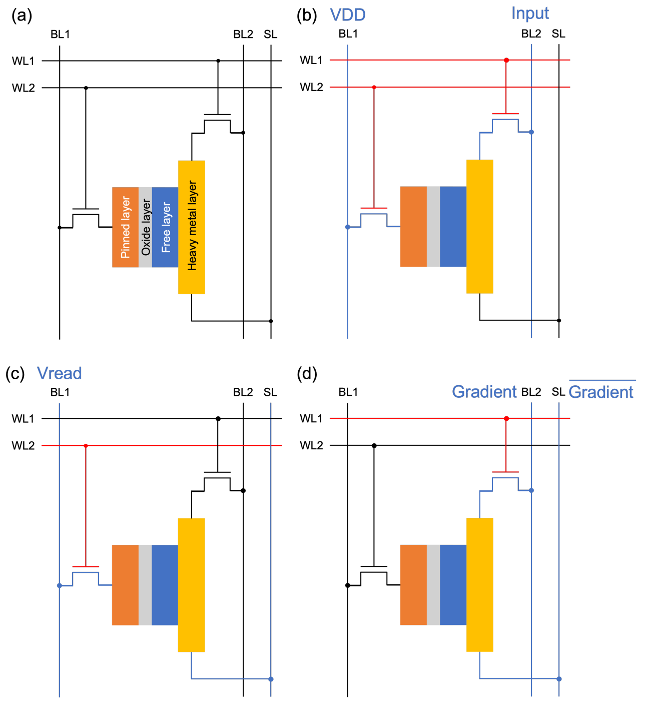


Fig. 7. Structure and each phase of our MRAM cell. (a) Cell structure. (b) XNOR (multiply) operation. (c) Read (accumulate) operation. (d) Weight update operation.

Fig. 7-(b) shows the MRAM cell in the XNOR (multiply) operation phase. In this phase, WL1 and WL2 are active (red line), BL2 represents input value, BL1 is VDD, and the current value of the MRAM cell represents the weights value. When the input value is "–1" (i.e., BL2 is low level (= GND)), VDD is applied. If we set VDD to the VCMA threshold voltage, the magnetization of the free layer of the MRAM is reversed (i.e., the current value of MRAM is reversed). In contract, when the input value is "+1" (i.e., BL2 is high level (= VDD)), no voltage is applied, the magnetization of the free layer is kept the same (i.e., the current value of MRAM is maintained). This operation is essentially the same as the XNOR operation (TABLE II), and by using this method, we can build an XNOR gate using only one MRAM cell. However, after this XNOR operation, the current value of the MRAM vanishes and, we need to add the same input to the MRAM cell to restore the current value. After this operation, the MRAM cell has multiple inputs and weights.

TABLE II. TRUTH TABLE OF XNOR PHASE

Current MRAM value (= weight)	Input (= BL2)	New MRAM value
–1	–1	+1
–1	+1	–1
+1	–1	–1
+1	+1	+1

Fig. 7-(c) shows the operation of our MRAM cell in the read (accumulate) operation phase. In this phase, WL1 is active (red line) and BL1 is set to the read voltage. The magnitude of the read current depends on the value that the MRAM cell memorizes: specifically, due to tunnel magnetoresistance (TMR) effect, a large read current flows if the value is "+1" and a small read current flows if the value is "–1" because of the read current corresponds to the value of the MRAM cell.

Fig. 7-(d) shows the operation of our MRAM cell in the weight updating phase. In this phase, we first need to restore the weight value (i.e., the previous value of the MRAM). This is done by means of XNOR operation in which the same input value in the previous XNOR operation phase is input again. After restoring operation, the value of the MRAM cell (i.e., the value of the weights) is updated in accordance with the propagated ternary gradients. In this step, WL1 is active (red line). The electric current flowing through the heavy metal layer of the MRAM cell corresponds to the ternary gradient; in other words, the electric current that generates SOT can write "+1" to the MRAM cell correspond to "+1" and write "–1" to the one corresponds to "–1", and no electric current corresponds to "0". For example, assume the direction of BL2 to SL is the "–1" electric current direction and the reverse direction is the "+1" electric current direction (in reality, the relation between the direction of the electric current and the direction of SOT depends on the direction of the external magnetic field). Considering this assumption, when BL2 is high level and SL is low level, the electric current flows from BL2 to SL; in other words, the "–1" electric current flows to the heavy metal layer and an SOT that can switch the magnetization to "–1" is generated. In contract, when BL2 is low level and SL is high level, the electric current flows from SL to BL2; in other words, the "+1" electric current flows to the heavy metal layer and an SOT that can switch the magnetization to "+1" is generated. In addition, when BL2 and SL are the same level, no electric current flows through the heavy metal layer and no SOT is generated. Thus, BL2 corresponds to the ternary gradient while SL corresponds to complementary value of the gradients. As shown in TABLE III, when the weight equals the gradient, an SOT that can reverse the value of the weight is generated, and opposite case, an effective SOT to reverse the weight is not generated. Thus, if we set the appropriate magnitude of the electric current, we can control the SOT writing probability; in other words, we can control the probability of updating the weight value to train our BNN.

TABLE III. TRUTH TABLE OF UPDATE PHASE

Weights	Gradient (= BL2)	Complementary value of the gradient (= SL)	Direction of SOT
–1	–1 (= L)	+1 (= H)	+1
–1	0 (= SL)	0 (= BL2)	–
–1	+1 (= H)	–1 (= L)	–1
+1	–1 (= L)	+1 (= H)	+1
+1	0 (= SL)	0 (= BL2)	–
+1	+1 (= H)	–1 (= L)	–1

On the BNN, in the training phase, we use all the phases of our MRAM cell, while in the inference phase, we use only the XNOR operation phase and read operation phase. Thus, we can use the MRAM cell in both the training phase and inference phase on the BNN.

V. BNN BASED ON MRAM ARRAY

In an NN, MAC operation of inputs (activations) and weights are expressed by the matrix product of an input vector and weight matrix. The CiM-based NN concept is that this weight matrix is memorized in a memory array and an input signal is represented as an input vector. In this study, we use the MRAM array (Fig. 8) and perform multiply and update operations on each MRAM cell (Section IV *B*) while the accumulate operation is done on each MRAM array column. The matrix product of the input vector and weight matrix is output as a vector containing an element that is the sum of the element-wise product of the input vector and a column of the weight matrix. The XNOR (multiply) operation on our MRAM cell corresponds to this element-wise product and the read (accumulate) operation on our MRAM array correspond to summation of this element wise product. In the read (accumulate) operation on our MRAM array, the read current flows through all of the MRAM cells in a column of the MRAM array. Thus, the summation of the read current of the MRAM cells in a column of the MRAM array flows through the SL of the MRAM array, and this sum current represents the sum of the element-wise product of the input vector and a column of the weight matrix. This sum current is then compared with a threshold current by a sense amp (SA) to output a binarized value "–1" or "+1". The SA operates as a signum function to output binarized activation values and propagate them to the next layer. Thus, we can use the MRAM array in both training phase and inference phase on the BNN.

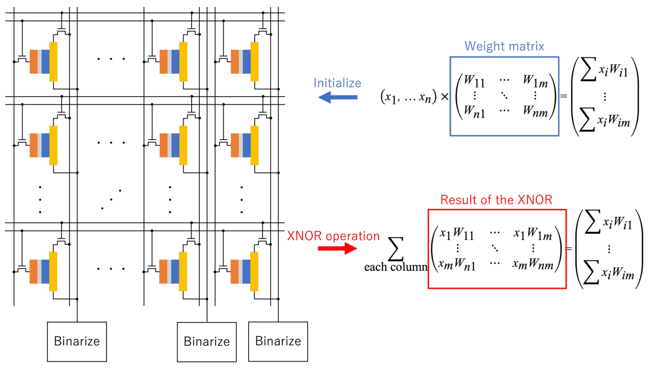

Weight matrix
Initialize
$(x_1,\dots x_n)\times\begin{pmatrix}W_{11}&\cdots&W_{1m}\\\vdots&\ddots&\vdots\\W_{n1}&\cdots&W_{nm}\end{pmatrix}=\begin{pmatrix}\sum x_iW_{i1}\\\vdots\\\sum x_iW_{im}\end{pmatrix}$
Result of the XNOR
XNOR operation
$\sum_{\text{each column}}\begin{pmatrix}x_1W_{11}&\cdots&x_1W_{1m}\\\vdots&\ddots&\vdots\\x_mW_{n1}&\cdots&x_mW_{nm}\end{pmatrix}=\begin{pmatrix}\sum x_iW_{i1}\\\vdots\\\sum x_iW_{im}\end{pmatrix}$
Binarize
Binarize
Binarize

Fig. 8. Our MRAM array. It is initialized by the weight matrix, and memorizes multiply result after XNOR operation.

VI. Simulation Results

A. *MAC operation on a small-scale MRAM array*

We simulate a MAC operation on our MRAM array using a 3 × 3 MRAM array. Fig. 9 shows the input vector and the weight matrix. After the XNOR operation, our MRAM array memorizes a new 3 × 3 matrix (lower part of Fig. 9).

Initial MRAM value (Weight matrix)

Input vector

$$\boxed{(-1,-1,1)}\times\boxed{\begin{pmatrix}-1&1&-1\\1&1&-1\\-1&1&1\end{pmatrix}}=\begin{pmatrix}-1\\-1\\3\end{pmatrix}$$

New MRAM value (XNOR weight with input)

$$\sum_{\text{each column}}\boxed{\begin{pmatrix}1&-1&1\\-1&-1&1\\-1&1&1\end{pmatrix}}=\begin{pmatrix}-1\\-1\\3\end{pmatrix}$$

Fig. 9. Example of 3 × 3 weight matrix used in XNOR opearation.

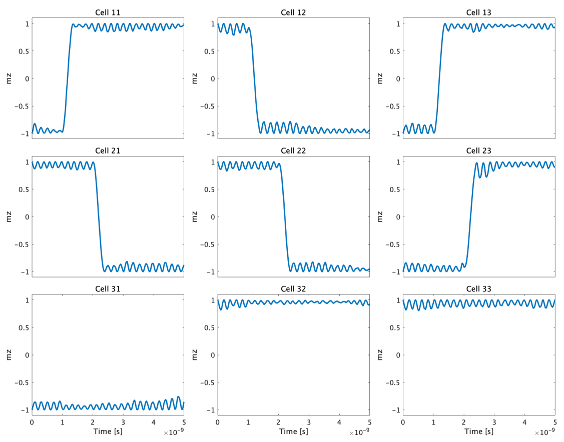


Fig. 10. Simulation results of XNOR operation using 3 × 3 MRAM array.

Fig. 10 shows the simulation results depicted as the time evolution of the z component of the magnetization of the free layer of our MRAM cell on the 3 × 3 MRAM array. When the value of the y axis of the plot is –1, the MRAM cell memorizes "–1", and when it is 1, the MRAM cell memorizes "+1". In the first 1 ns, no operation is performed on the MRAM array, and between 1 ns and 4 ns, the XNOR operation is active. As we can see in Fig. 10, in the first 1 ns, the MRAM array memorizes the weight matrix (as shown in lower part of Fig. 9), and after XNOR operation finishes (after 4 ns), the MRAM array memorizes the XNOR result (as shown in upper part of Fig. 9). These results demonstrate that the MRAM array operates as XNOR gates correctly.

B. *Weight update on a small-scale MRAM array*

Next, we simulate a weights update operation on our MRAM array using a 3 × 3 MRAM array. Fig. 11 shows the current weight matrix and the ternary gradient matrix. After the update operation, weight matrix is updated by the ternary gradient matrix. In the case shown here, red elements of the weight matrix are reversed stochastically.

Weight matrix Gradient matrix

$$\boxed{\begin{pmatrix}-1&-1&-1\\-1&-1&1\\-1&1&1\end{pmatrix}}\xleftarrow{\text{Update}}\boxed{\begin{pmatrix}0&-1&1\\0&-1&1\\-1&0&1\end{pmatrix}}$$

Fig. 11. Example of 3 × 3 weight and gradient matrix used in update opearation.

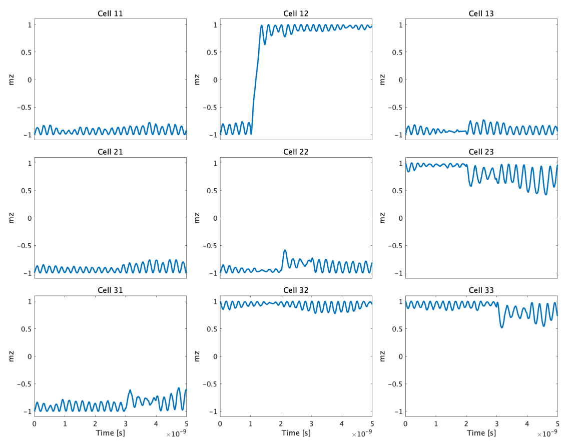


Fig. 12. Simulation results of update operation using 3 × 3 MRAM array.

Fig. 12 shows the simulation results depicted as the time evolution of the z component of the magnetization of the free layer of our MRAM cell on the 3 × 3 MRAM array. In the first 1 ns, no operation is performed on the MRAM array, and between 1 ns and 4 ns, the update operation is active. As we can see in Fig. 12, in the first 1 ns, the MRAM array memorizes the weight matrix (as shown Fig. 11), and after the update operation finishes (after 4 ns), the red elements of the weight matrix (as shown in Fig. 11) are reversed stochastically (note that in this case, only W_{12} is reversed, and other red elements have magnetization fluctuation but do not achieve magnetization switching). These results demonstrate that the MRAM array operates the stochastic update correctly.

C. *BNN training using MNIST dataset*

Finally, we simulate MNIST dataset training on our MRAM array using our BNN training algorithm. Fig. 13 compares the accuracy of MNIST dataset training using our BNN training algorithm on our MRAM array with that using a general BNN training algorithm [1] (which does not on an MRAM array). Each BNN has an input layer, a hidden layer, and an output layer containing 784, 3,136, and ten neurons, respectively.

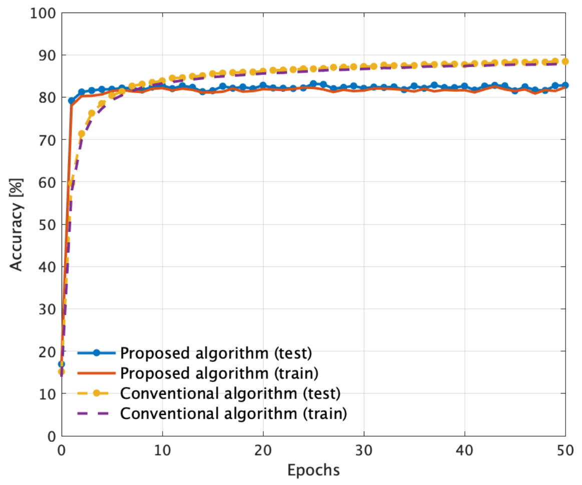


Fig. 13. Result of simulated training with MNIST dataset.

After training for 50 epochs, our algorithm showed an accuracy of around 80% and a faster training speed than the conventional algorithm, as shown in Fig. 13. TABLE IV lists the accuracy of each BNN after training for 50 epochs.

TABLE IV. SIMULATED TAINING ACCURACY OF CONVENTIONAL AND PROPOSED ALGORITHMS

	Conventional	Proposed
Test	88.43 %	82.79 %
Train	87.88 %	82.34 %

VII. CONCLUSION

In this study, we proposed a BNN training algorithm using ternary gradients and applied it to an MRAM array-based BNN training/inference machine. Our algorithm consists of three key elements: ternary gradients, an improved STE, and stochastic weight updating. In the training phase, we use only binary weights and ternary gradients and do not use real-valued weights or real-valued gradients necessary in conventional algorithms. In our MRAM array, we utilize VCMA magnetization switching to reduce the scale of the MAC operation circuits by half compared to SOT-based MAC operation. We also implement SOT stochastic switching to update weights and thereby enable training on the MRAM array.

Our experimental results using the MNIST dataset demonstrate that our MRAM array can achieve an accuracy of around 80%. A BNN training/inference machine can therefore be built on the edge side based on the MRAM array thanks to our BNN training algorithm and the smaller MAC operation circuit based on the MRAM array using VCMA magnetization switching.

REFERENCES

[1] M. Courbariaux I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized Neural Network: Training Neural Network with Weights and Activations Constrained to +1 or –1”, arXiv preprint, arXiv: 1602.02830, Feb. 2016

[2] H. Jiang, X. Peng, S. Huang, and S. Yu, “CIMAT: A Computing-in-Memory Architecture for On-chip Training Based on Transpose SRAM Arrays”, IEEE Trans. Computers, vol. 69, no. 7, July 2020

[3] H. Wang, W. Kang, B. Pan, H. Zhang, E. Deng, and W. Zhao, “Spintronic Computing-in-Memory Architecture Based on Voltage-Controlled Spin-Orbit Torque Devices for Binary Neural Network”, IEEE Trans. Electron Devices, vol. 68, no. 10, Oct. 2021

[4] Y. Kishi, A. Yamada, M. Ke, and T. Kawahara, “Examination of Magnetization Switching Behavior by Bi-Directional Read of Spin-Orbit-Torque MRAM”, IEEE Trans. Magnetics, vol. 58, no.5, May 2022

[5] Akhikesh Jaiswal, Amogh Agrawal, and Kaushik Roy, “In-situ, In-Memory Stateful Vector Logic Operations based on Voltage Controlled Magnetic Anisotropy”, nature Sci Rep 8, 5738 (2018)

[6] Sergy Iofee, and Christian Szegedy, “Batch Normalization: Accelerating Deep Neural Training by Reducing Internal Covariate Shift”, arXiv: 1502.03167, Mar. 2016

[7] M. Kazemi, Graham E. Rowlands, S. Shi, Robert A. Buhrman, and Edy G. Friedman, “All-Spin-Orbit Switching of Perpendicular Magnetization”, IEEE Trans. Electron Devices, vol. 63, no. 11, Nov. 2016

[8] X. Han, X. Wang, C. Wan, G. Yu, and X. Lv, “Spin-orbit torques: Materials, physics, and devices”, Appl. Phys. Lett. 118, 120502 (2021)

[9] W. Kang, Y. Ran, Y. Zhang, W. Lv, and W. Zhao, “Modeling and Exploration of the Voltage-Controlled Magnetic Anisotropy Effect for the Next-Generation Low-Power and High-Speed MRAM Application”, IEEE Trans. NanoTechnology, vol. 16 no. 3, May 2017

[10] Venkata Pavan Kumar Miriyala, Xuanyao Fong, and Gengchiau Liang, “Influence of Size and Shape on Performance of VCMA-Based MTJs”, IEEE Trans. Electron Devices, vol.66, no. 2, Feb. 2019