micromachines

Article

Memristor-CMOS Hybrid Neuron Circuit with Nonideal-Effect Correction Related to Parasitic Resistance for Binary-Memristor-Crossbar Neural Networks

Tien Van Nguyen, Jiyong An and Kyeong-Sik Min *

School of Electrical Engineering, Kookmin University, Seoul 02707, Korea; tiennv@kookmin.ac.kr (T.V.N.); sunday1903@kookmin.ac.kr (J.A.)
* Correspondence: mks@kookmin.ac.kr

Abstract: Voltages and currents in a memristor crossbar can be significantly affected due to nonideal effects such as parasitic source, line, and neuron resistance. These nonideal effects related to the parasitic resistance can cause the degradation of the neural network's performance realized with the nonideal memristor crossbar. To avoid performance degradation due to the parasitic-resistance-related nonideal effects, adaptive training methods were proposed previously. However, the complicated training algorithm could add a heavy computational burden to the neural network hardware. Especially, the hardware and algorithmic burden can be more serious for edge intelligence applications such as Internet of Things (IoT) sensors. In this paper, a memristor-CMOS hybrid neuron circuit is proposed for compensating the parasitic-resistance-related nonideal effects during not the training phase but the inference one, where the complicated adaptive training is not needed. Moreover, unlike the previous linear correction method performed by the external hardware, the proposed correction circuit can be included in the memristor crossbar to minimize the power and hardware overheads for compensating the nonideal effects. The proposed correction circuit has been verified to be able to restore the degradation of source and output voltages in the nonideal crossbar. For the source voltage, the average percentage error of the uncompensated crossbar is as large as 36.7%. If the correction circuit is used, the percentage error in the source voltage can be reduced from 36.7% to 7.5%. For the output voltage, the average percentage error of the uncompensated crossbar is as large as 65.2%. The correction circuit can improve the percentage error in the output voltage from 65.2% to 8.6%. Almost the percentage error can be reduced to ~1/7 if the correction circuit is used. The nonideal memristor crossbar with the correction circuit has been tested for MNIST and CIFAR-10 datasets in this paper. For MNIST, the uncompensated and compensated crossbars indicate the recognition rate of 90.4% and 95.1%, respectively, compared to 95.5% of the ideal crossbar. For CIFAR-10, the nonideal crossbars without and with the nonideal-effect correction show the rate of 85.3% and 88.1%, respectively, compared to the ideal crossbar achieving the rate as large as 88.9%.

Keywords: neuron circuit; nonideal-effect correction; binary memristor crossbar; neural networks; edge intelligence

Citation: Nguyen, T.V.; An, J.; Min, K.-S. Memristor-CMOS Hybrid Neuron Circuit with Nonideal-Effect Correction Related to Parasitic Resistance for Binary-Memristor-Crossbar Neural Networks. *Micromachines* **2021**, *12*, 791. https://doi.org/10.3390/mi12070791

Academic Editors: Jung Ho Yoon and Nam-Trung Nguyen

Received: 4 May 2021
Accepted: 28 June 2021
Published: 1 July 2021

1. Introduction

Neural networks draw many interests nowadays, as they have been verified useful in various cognitive tasks such as natural language processing, image recognition, object classification, etc. [1–3]. As the applications of neural networks become more complex, the demand for high-performance computing becomes increasingly more. For meeting the need for heavy computation capability, general-purpose digital systems based on CMOS technology have been used widely so far. However, CMOS device scaling has slowed down recently and VDD scaling becomes no longer effective in reducing power consumption [4,5]. Moreover, in terms of computing architecture, the traditional Von Neumann machine has been suffering the memory bottleneck problem [6–9]. The bottleneck problem becomes

more severe especially for computation of neural networks, where memory access for large amounts of data occurs frequently between memory and computing units [10].

One of the important applications of neural networks can be found in edge intelligence such as Internet of Things (IoT) sensors and edge devices, where a massive number of various sensors collect huge amounts of unstructured data everywhere and every time to make human life more comfortable and safe [11–13]. If all the data sensed from IoT sensors and edge devices are sent to data centers, amounts of energy for communication and computation for the data centers may be exploding as much as an unbearable level. To avoid the explosion of communication and computing energy at the cloud servers, energy-efficient computing is indispensable in implementing neural networks at IoT sensors and edge devices for edge intelligence [13].

To achieve both high-performance and energy-efficient computing for edge intelligence, memristor crossbars can be considered as good candidates of computing hardware for possible applications such as neural networks, neuromorphic computing, processing-in-memory, etc. [14–16]. Memristors demonstrated experimentally in 2008 [17] are nonvolatile memories, where both binary and multi-level values can be stored [18–20]. For the architecture, memristor crossbars can be a built-in 3-dimensional multi-layer structure that seems very similar to the biological neuronal structure observed in the human brain [21–24]. For the fabrication process, Back-end-of-line (BEOL) has been reported in many kinds of literature, to make it possible for memristors fabricated with CMOS devices on the same wafer [10,25,26]). The crossbars made of memristors can perform low-power, parallel, and binary/multi-valued computation similarly with the biological nervous systems. From these properties mentioned above, the crossbars can be considered very suitable particularly for the computational acceleration of neural networks at the edge [20,27].

Vector Matrix Multiplication (VMM) is a core computational function used in both the training and inference phases of neural networks. The VMM operation can be realized in memristor crossbars, where a matrix multiplication operation can be performed using the memristor's voltage-current relationship according to Ohm's law [10,28]. The VMM carried out by the memristor crossbar is one example of "computing by physics". One big advantage of the memristor-crossbar VMM is that the computing capability can be expanded in parallel by adding more columns to the memristor crossbar. The parallel computing of the memristor crossbar is desirable for handling the heavy computational load of VMM operation in data-centric processing systems such as deep-learning neural networks, etc. Moreover, the memory access bottleneck of the Von Neumann architecture can be alleviated significantly in the VMM, because both the memory and computing functions can be merged in the memristor crossbar [10]. In the traditional computing systems, the data fetching and updating operations should take place frequently between the computing and memory units, which are separated in the physical distance as long as ~mm. One more thing to comment here is that memristor's non-volatility can help IoT sensors extend battery lifetime very long because the synaptic weights of neural networks can be maintained very long time in the crossbar even during the power-off time, without refreshing the stored data.

Figure 1a shows a conceptual schematic of Artificial Neural Network (ANN) which is composed of neurons and synaptic connections. Here X_1, X_2, etc. represent input neurons. Y_1, Y_2, etc. are hidden neurons between input and output neurons. Z_1, Z_2, etc. are output neurons of the network in Figure 1a. Here 'm', 'n', and 'k' are the numbers of input, hidden, and output neurons, respectively. The hidden-neuron layers can be more than one. W_{111} means a synaptic weight between two neurons, X_1 and Y_1 for the first synapse layer. W_{121} is a weight between X_1 and Y_2. Similarly, W_{112} is a synapse between Y_1 and Z_1 for the second synapse layer. W_{1k2} is between Y_1 and Z_k. The hidden neurons, Y_1, Y_2, etc. can be calculated with Vector-Matrix Multiplication (VMM) of the input-neuron vector and the 1st-layer weight matrix. The output neurons, Z_1, Z_2, etc. are obtained from the VMM operation of the hidden-neuron vector and the 2nd-layer weight matrix. The VMM

operation can consume a large amount of computing power if the VMM is performed using digital CMOS logic.

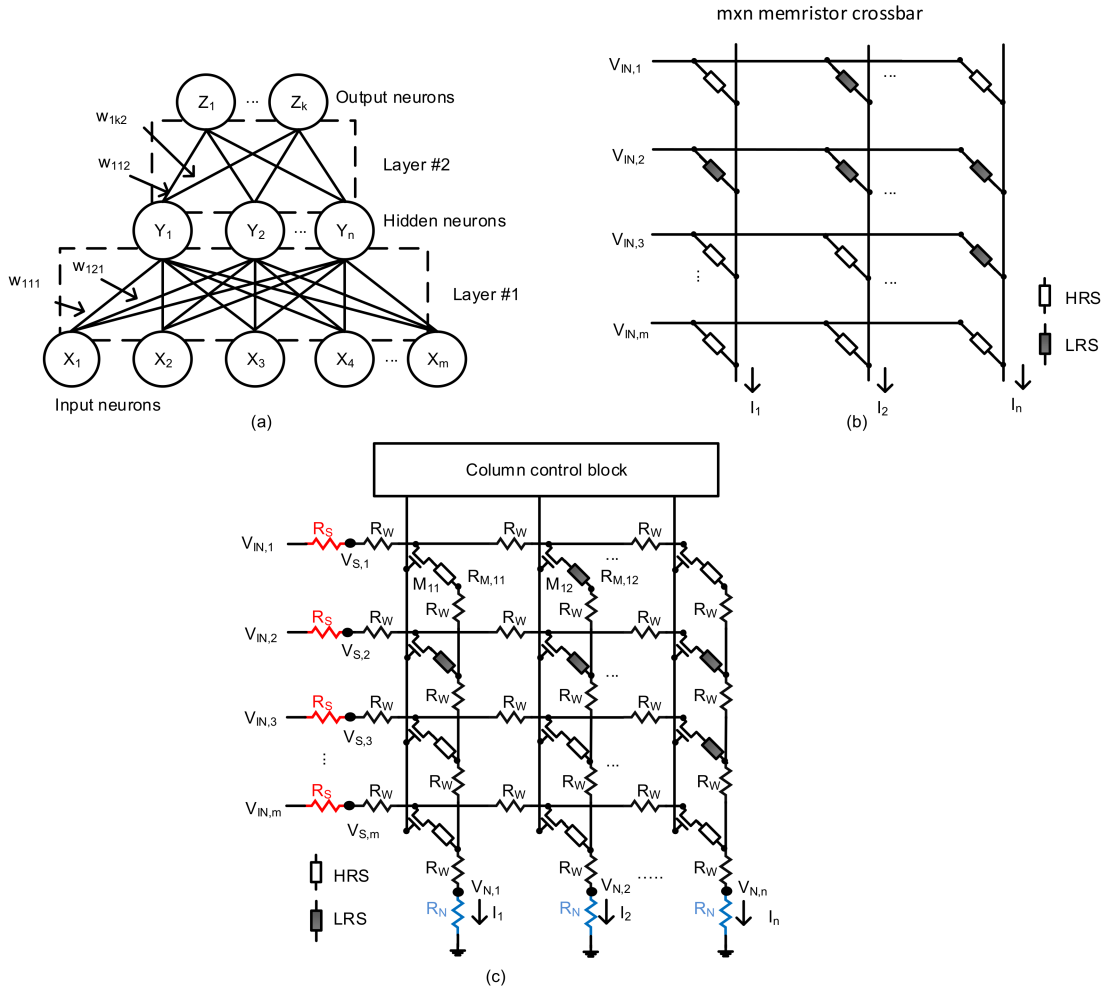

Figure 1. (**a**) Conceptual schematic of two-layer neural network with input, hidden, and output neurons. (**b**) Schematic of ideal memristor crossbar circuit with HRS and LRS memristor synapses. The numbers of rows and columns are 'm' and 'n', respectively. The parasitic resistance such as source, line, and neuron resistance are assumed zero, in the ideal crossbar. (**c**) Schematic of nonideal memristor crossbar circuit with parasitic crossbar resistance such as source, line, and neuron resistance.

Figure 1b shows a schematic of the ideal memristor crossbar with HRS and LRS memristor cells. HRS and LRS mean High Resistance State and Low Resistance State of memristors, respectively. HRS and LRS are represented with white and gray boxes, respectively. Here the memristor crossbar in Figure 1b is used to store the synaptic weights of ANN in Figure 1a. The synaptic weights stored in the crossbar are assumed binary (-1 and $+1$) or ternary (-1, 0, and $+1$) in this paper. $V_{IN,1}$, $V_{IN,2}$, and $V_{IN,m}$ are input voltages applied to Row #1, Row #2, and Row #m, respectively. I_1, I_2, and I_N are column currents for Column #1, Column #2, and Column #n, respectively. In this figure, 'm' and 'n' are the numbers of rows and columns in the crossbar, respectively. Parasitic crossbar resistance such as line resistance, neuron resistance, and source resistance are not considered in Figure 1b, where the ideal crossbar is assumed without any parasitic resistance.

Figure 1c shows a schematic of nonideal memristor crossbar, where parasitic crossbar resistance such as R_S, R_W, and R_N are considered. R_S, R_W, and R_N represent source resistance, line resistance, and neuron resistance, respectively. Here HRS and LRS are represented with white and gray boxes, respectively. Like Figure 1b, the nonideal memristor

crossbar is used to store the synaptic weights of ANN in Figure 1a. The synaptic weights stored in the crossbar of Figure 1c are assumed binary (−1 and +1) or ternary (−1, 0, and +1), as mentioned in Figure 1b. In Figure 1c, $V_{IN,1}$, $V_{IN,2}$, and $V_{IN,m}$ are input row voltages for Row #1, Row #2, and Row #m, respectively. Here 'm' means the number of rows in the crossbar. $V_{S,1}$, $V_{S,2}$, and $V_{S,m}$ are source voltages on the rows, which are degraded due to R_S and R_W, for Row #1, Row #2, and Row #m, respectively. I_1, I_2, and I_N are column currents of Column #1, Column #2, and Column #n, respectively. 'n' is the number of columns in the crossbar. Similarly, $V_{N,1}$, $V_{N,2}$, and $V_{N,n}$ are neuron voltages on the columns affected due to R_N and R_W, for Column #1, Column #2, and Column #n, respectively. In the crossbar, M_{11} and M_{12} are transistors for controlling memristors of $R_{M,11}$ and $R_{M,12}$, respectively. $R_{M,11}$ is a memristor cell connected with Row #1 and Column #1. $R_{M,12}$ is a memristor cell connected with Row #1 and Column #2. The access-controlling transistors of M_{11} and M_{12} are turned on or off by the signals from the column control block, shown in Figure 1c.

The source, line, and neuron resistance in the nonideal crossbar in Figure 1c can affect the source voltages and the column currents. First, let us consider the nonideal effect due to source resistance. From Kirchhoff law, the source voltage, v.s. can be calculated by dividing the input voltage, V_{IN}, between R_S and the rest part of the row line including the line resistance and LRS cells along the row line. Usually, HRS cells affect the source voltage very little, because the conductance is negligibly small. If the number of LRS cells for a row becomes larger, the source voltage on the row line is affected more by the source resistance, R_S, not by the LRS cells. This leads to the degradation of the source voltage. Similarly, we can consider the nonideal effect due to neuron resistance, R_N. If the number of LRS cells for a column is increased larger and the parallel combination of LRS for the column becomes much smaller than R_N, the column current begins to be dominated by R_N, not by the parallel combination of LRS cells. If so, the column current is observed to be degraded compared to the column current of the ideal crossbar with $R_N = 0$. A more detailed analysis of the nonideal effects due to R_S and R_N will be explained in the next section.

The degradation of source voltage and column current due to the nonideal effects such as source, line, and neuron resistance can affect neural network's performance significantly. This is because the synaptic weights calculated from the backpropagation algorithm assume the memristor crossbar for the network is ideal not suffering the nonideal effects. To mitigate the performance gap between the ideal and nonideal crossbars, adaptive training methods have been proposed to consider the nonideal effects in the crossbar during the training phase [29,30]. For doing this, however, the training algorithm of the nonideal crossbar should be more complicated and the computational load becomes heavier to consider the voltage and current degradation due to the nonideal effects. The complicated training algorithm with heavy computational load is a big disadvantage, in terms of the training energy and time, particularly, for the on-device training applications such as edge devices, IoT sensors, etc.

Unlike the adaptive training methods, the compensation of the nonideal effects can be performed during the inference phase, not relying on the complicated training algorithm of the nonideal crossbar. The linear correction method was proposed to compensate for the nonideal effects during the inference time [10]. Though the computational burden becomes smaller compared to the adaptive training method, the linear correction proposed previously was performed by an external Trans-Impedance Amplifier (TIA) not being included in the memristor crossbar circuit [10]. The external TIA should be equipped with programmable gain and offset to calibrate the column current for compensating the nonideal effects [10]. The correction by the external TIA causes the calibration overhead because each TIA should be programmed with different gain and offset values for compensating the corresponding column's nonideal behavior [10]. One more thing to note here is that the linear correction was proposed only for compensating the column current not the source voltage [10]. Unlike the previous linear correction by the external TIA, a new correction circuit proposed in this paper can be included in the memristor crossbar for

compensating not only the column current but also the row voltage degradation due to the nonideal effects.

Particularly, implementing the nonideal-effect correction in the memristor crossbar circuit is very important for realizing the edge intelligence at IoT sensors, where the hardware and power overheads are critically important. For avoiding the hardware and power overheads due to the external hardware, the compensation of nonideal effects should be realized inside the memristor crossbar circuit not relying on the external hardware. For doing this, a new memristor-CMOS hybrid circuit for realizing the nonideal-effect correction is proposed in this paper. This proposed circuit can compensate for the nonideal effects in the inference phase, not in the training phase. Thus, the training algorithm in this paper can be as simple as the normal backpropagation of the ideal crossbar, not using the complicated adaptive training algorithm. In addition, the correction circuit is implemented in the memristor crossbar not using the external hardware. By doing so, the power and hardware overheads can be avoided in this paper. In the next section, a new neuron circuit with the nonideal-effect correction is explained in detail. In Section 3, the proposed correction neuron circuit is tested and discussed for MNIST (Modified National Institute of Standards and Technology database) and CIFAR-10 data sets [12,31]. In Section 4, we conclude this work finally.

Figure 2a shows the source voltage degradation due to the nonideal effects such as R_S, R_N, and R_W, as mentioned just earlier. In the ideal crossbar, v.s. seems constant despite increasing the percentage of LRS cells among all the memristor cells per row. On the contrary, the source voltage in the nonideal crossbar is degraded with increasing the percentage of LRS per row, as shown in Figure 2a. Assuming the source voltage is affected little by R_W and R_N, the source voltage of Row #i, $V_{S,i}$ can be simply approximated with the following equation.

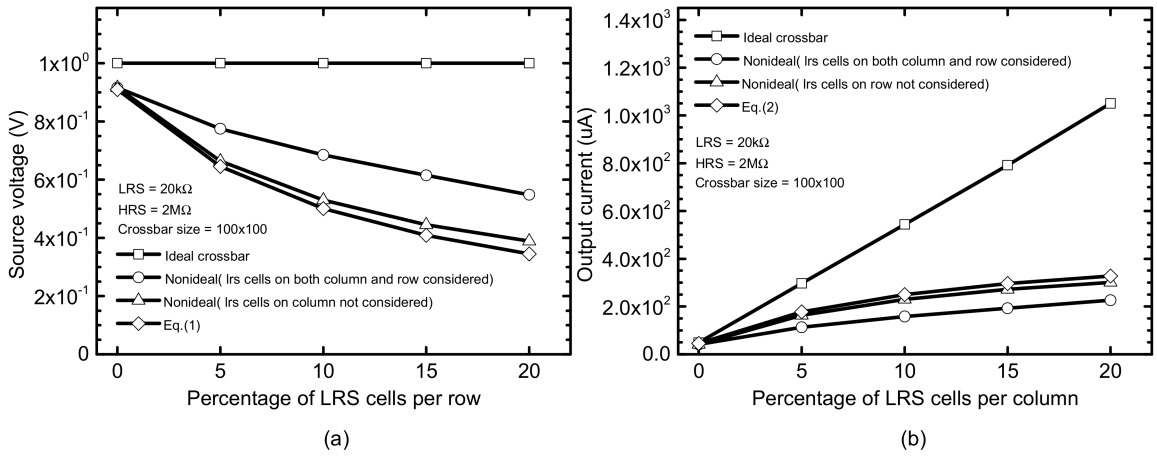

Figure 2. (**a**) Comparison of the source voltage of the ideal crossbar, the nonideal one (LRS cells on both column and row considered), the nonideal one (LRS cells on a column not considered), and the calculation with Equation (1). (**b**) Comparison of the output current of the ideal crossbar, the nonideal one (LRS cells on both column and row considered), the nonideal one (LRS cells on row not considered), and the calculation with Equation (2). Here, the ideal crossbar is assumed zero parasitic resistance. The nonideal crossbar is assumed with R_S = 2 kΩ, R_N = 2 kΩ, and R_W = 1 Ω. The ideal and nonideal crossbars are simulated with a CADENCE SPECTRE version 6.1.6 circuit simulator.

2. Method

Here $V_{IN,i}$ is the input voltage of Row #i and assumed as large as 1 V, as shown in Figure 2a. In Equation (1), G_S means the inverse of source resistance, R_S. G_{LRS} and G_{HRS} are the conductance of LRS and HRS cells, respectively. 'l_i' means the number of LRS cells of Row #i. 'n' means the number of columns in the crossbar. Thus, '$n-l_i$' represents the number of HRS cells of Row #i in Equation (1). In Figure 2a, the X-axis is the percentage of LRS cells among all the memristor cells of Row #i. Here the percentage of LRS of Row #i is

calculated with 'l_i/n(%)'. As expected from Equation (1), $V_{S,i}$ is degraded with increasing the number of LRS cells for Row #i. This is because the voltage drop on R_S becomes larger, as the number of LRS cells of the row is increased. In Figure 2a, we compared the source voltage of the ideal crossbar, the nonideal one (LRS cells on both column and row considered), the nonideal one (LRS cells on a column not considered), and the calculation with Equation (1). Here, the nonideal crossbar is assumed with R_S = 2 kΩ, R_N = 2 kΩ, and R_W = 1 Ω. The ideal and nonideal crossbars are simulated with a CADENCE SPECTRE version 6.1.6 circuit simulator. In Figure 2a, Equation (1) is calculated with MATLAB. The gap between the Equation (1) and the nonideal crossbar (LRS cells on a column not considered) is very little. This is because R_W = 1Ω can affect the source voltage very little. Here the end-to-end line resistance is as small as 100 Ω, for 100 × 100 crossbar. Compared to the end-to-end line resistance of 100 Ω, R_S is as large as 2 kΩ in Figure 2a. The nonideal crossbar (LRS cells on both column and row considered) shows a larger source voltage than Equation (1). This is because more LRS cells on the column can boost up the source voltage higher than Equation (1).

$$V_{S,i} \approx \frac{G_S \cdot V_{IN,i}}{G_S + \frac{l_i}{R_{LRS}+R_N} + \frac{n-l_i}{R_{HRS}+R_N}} \quad (1)$$

The column current can be changed due to R_S, R_N, and R_W like the source voltage, too. Figure 2b shows the column current degradation due to the nonideal effects. For the ideal crossbar with zero parasitic resistance, the column current seems proportional to the number of LRS cells per column. However, as indicated in Figure 2b, the column current seems saturated due to R_N, in the nonideal crossbar. Similarly, with Equation (1), assuming the column current is affected little by R_S and R_W, the column current can be expressed with

$$I_j \approx \frac{G_N \cdot \sum_{i=1}^{m} \left(G_{M,ij} \cdot V_{S,i} \right)}{G_N + \frac{k_j}{R_{LRS}+R_S} + \frac{m-k_j}{R_{HRS}+R_S}} \quad (2)$$

I_j is the column current for Column #j. G_N is the inverse of R_N. $G_{M,ij}$ is memristor's conductance for Row #i and Column #j. $V_{S,i}$ is the source voltage of Row #i, as mentioned in Equation (1). The term of $\sum_{i=1}^{m} \left(G_{M,ij} \cdot V_{S,i} \right)$ calculates the summation of the conductance-voltage multiplications from Row #1 to Row #n, for Column #j. 'm' is the number of rows in the crossbar. In the denominator term, 'k_j' is the number of LRS cells for Column #j. Thus, 'm-k_j' means the number of HRS cells for Column #j.

The percentage of LRS cells for Column #j is calculated with 'k_j/m(%)'. In the ideal crossbar, it is expected that the column current, Ij is proportional to the number of LRS cells for its own column. However, in the nonideal crossbar with parasitic neuron resistance R_N, the column current begins to be saturated, when the number of LRS cells is increased. From Equation (2), it is obvious that the saturation comes from the term $k_j \cdot G_{LRS}$ in the denominator term. In Figure 2b, we compared the source voltage of the ideal crossbar, the nonideal one (LRS cells on both column and row considered), the nonideal one (LRS cells on a column not considered), and the calculation with Equation (2). Here, the nonideal crossbar is assumed with R_S = 2 kΩ, R_N = 2 kΩ, and R_W = 1 Ω. The ideal and nonideal crossbars are simulated with a CADENCE SPECTRE version 6.1.6 circuit simulator. In Figure 2b, Equation (2) is calculated with MATLAB. Like Figure 2a, the gap between the Equation (2) and the nonideal crossbar (LRS cells on row not considered) seems very small. This is because R_W = 1 Ω can affect the column current very little. Here the end-to-end line resistance is as small as 100 Ω, for 100 × 100 crossbar. Compared to the end-to-end line resistance of 100 Ω, R_N is as large as 2 kΩ in Figure 2b. The nonideal crossbar (LRS cells on both column and row considered) shows a lower column current than Equation (2). This is because more LRS cells on a row can suppress the column current more severely than Equation (2).

Here the memristor array size simulated in Figure 2a,b is assumed 100 rows and 100 columns. Here LRS and HRS are assumed 20 kΩ and 2 MΩ, respectively.

To compensate for the voltage and current degradation due to the nonideal effects in the nonideal crossbar, a new memristor-CMOS hybrid circuit for realizing the nonideal-effect correction neuron is proposed, as indicated in Figure 3. The proposed circuit in Figure 3 can correct both the source voltage and column current degradation from the nonideal effects, in the inference phase, not in the training phase. Thus, the training procedure in this paper can be as simple as the normal backpropagation of the ideal neural networks, not using the complicated adaptive training algorithm [30,31]. In addition, the correction can be implemented in the memristor crossbar circuit in Figure 3 without using the complicated digital control block such as the external correction method [10]. By doing so, the power and hardware overhead due to the external digital controller can be avoided in the proposed neuron circuit with nonideal-effect correction.

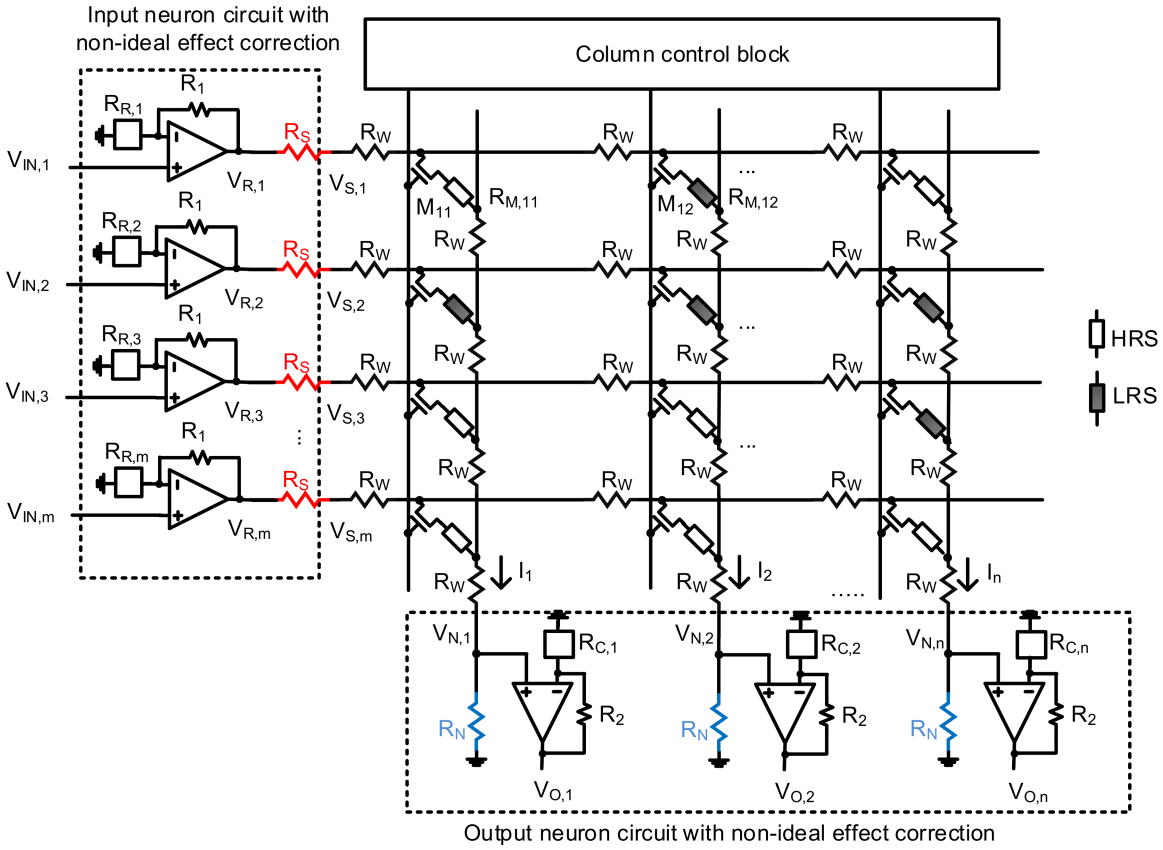

Figure 3. Schematic of the proposed memristor crossbar circuit with correction of nonideal effects. Here the nonideal-effect correction circuit in the left is for compensating the loss of source voltage. The correction circuit added to the bottom is for reducing the loss of output voltage.

Figure 3 shows a schematic of the memristor crossbar circuit, where the proposed correction circuit is added for compensating the nonideal effects due to source and neuron resistance. Here the memristor crossbar is used to store the synaptic weights of neural networks. In Figure 3, memristor cells with HRS and LRS are represented with open and solid styles, respectively. R_W, R_S, and R_N mean parasitic line, source, and neuron resistance of the nonideal crossbar, respectively. In Figure 3, M_{11} and M_{12} are transistors for accessing the memristors of $R_{M,11}$ and $R_{M,12}$, respectively. $R_{M,11}$ is an HRS cell connected with Row #1 and Column #1. $R_{M,12}$ is the LRS cell connected with Row #1 and Column #2. The access transistors of M_{11} and M_{12} are turned on or off by the signals from the column control block, shown in Figure 3.

Looking at the row lines in Figure 3, $V_{IN,1}$, $V_{IN,2}$, and $V_{IN,m}$ are external input voltages applied to Row #1, Row #2, and Row #m, respectively. The input voltages enter the correction circuit, where the input voltages are converted to the correction voltages. The

correction circuit for Row #1 is made of the non-inverting amplifier with R_1 and $R_{R,1}$. Similarly, the correction circuit for Row #2 has the noninverting amplifier with R_1 and $R_{R,2}$. Here it should be noted that R_1 has the same resistance value for all the rows in the crossbar, while $R_{R,i}$ made of a memristor can be programmed with a different value for each row. $V_{R,1}$, $V_{R,2}$, and $V_{R,m}$ represent the correction voltages for compensating the nonideal effect due to source resistance. $V_{R,i}$ for Row #i, can be calculated with $\left(1+\frac{G_{R,i}}{G_1}\right)\cdot V_{IN,i}$. $G_{R,i}$, and G_1 are the inverse of $R_{R,i}$ and R_1, respectively. After passing through source resistance of R_S, $V_{S,1}$, $V_{S,2}$, and $V_{S,m}$ represent the source voltages for Row #1, Row #2, and Row #m, respectively. Using Equation (1), the source voltage of Row #i in Figure 3 can be approximated with

$$V_{S,i} \approx \frac{G_S\cdot\left(1+\frac{G_{R,i}}{G_1}\right)\cdot V_{IN,i}}{G_S+\frac{l_i}{R_{LRS}+R_N}+\frac{n-l_i}{R_{HRS}+R_N}} = \frac{\left(1+\frac{G_{R,i}}{G_1}\right)\cdot V_{IN,i}}{\left(1+l_i\cdot\frac{1}{G_S}\cdot\left(\frac{1}{R_{LRS}+R_N}-\frac{1}{R_{HRS}+R_N}\right)+n\cdot\frac{1}{G_S(R_{HRS}+R_N)}\right)} \tag{3}$$

As mentioned in Equation (1), 'l_i' and '$n-l_i$' are the numbers of LRS and HRS cells for Row #i, respectively. 'n' means the number of columns in the crossbar. G_S is the inverse of R_S. The nonideal effect due to R_S is caused by $l_i\cdot\frac{1}{G_S}\cdot\left(\frac{1}{R_{LRS}+R_N}-\frac{1}{R_{HRS}+R_N}\right)$ in the denominator term of Equation (3). From this equation, the source voltage, $V_{S,i}$ is expected to be lowered, as the number of LRS cells for Row #i becomes large. By adjusting the resistance $R_{R,i}$ for Row #i, in the noninverting op amp, we can make the condition of $\frac{G_{R,i}}{G_1}\approx l_i\cdot\frac{1}{G_S}\cdot\left(\frac{1}{R_{LRS}+R_N}-\frac{1}{R_{HRS}+R_N}\right)$. By doing so, the source voltage loss due to source resistance can be compensated in Equation (3). Here it should be noted that $R_{R,i}$ of Row #i in Figure 3 can be made of a memristor.

Looking at the column lines in Figure 3, I_1, I_2, and I_N are the column currents for Column #1, Column #2, and Column #n, respectively. $V_{N,1}$, $V_{N,2}$, and $V_{N,n}$ are the neuron voltages for Column #1, Column #2, and Column #n, respectively, which are generated from the column currents. $V_{O,1}$, $V_{O,2}$, and $V_{O,n}$ are the corrected column voltages for taking into account the parasitic neuron resistance. The correction circuit for Column #1 is composed of the non-inverting amplifier with $R_{C,1}$ and R_2. The correction circuit for the next Column #2 has $R_{C,2}$ and R_2. Here it should be noted that R_2 has the same resistance for all the columns in the crossbar, while $R_{C,j}$ made of a memristor can be programmed with a different value for each column. The corrected output voltage of $V_{O,j}$ for Column #j can be calculated with $\left(1+\frac{G_{C,j}}{G_2}\right)\cdot V_{N,j}$. $G_{C,j}$ and G_2 are the inverse of $R_{C,j}$ and R_2, respectively. Using $V_{N,j}$ from the Equation (2), the output voltage of the correction circuit for Column #j is expressed roughly with

$$\begin{aligned} V_{O,j} \approx V_{N,j}\cdot\left(1+\frac{G_{C,j}}{G_2}\right) &\approx \frac{\sum_{i=1}^{m}\left(G_{M,ij}\cdot V_{S,i}\right)\cdot\left(1+\frac{G_{C,j}}{G_2}\right)}{G_N+\frac{k_j}{R_{LRS}+R_S}+\frac{m-k_j}{R_{HRS}+R_S}} \\ &= \frac{\sum_{i=1}^{m}\left(G_{M,ij}\cdot V_{S,i}\right)\cdot\left(1+\frac{G_{C,j}}{G_2}\right)}{G_N\left(1+k_j\cdot\frac{1}{G_N}\cdot\left(\frac{1}{R_{LRS}+R_S}-\frac{1}{R_{HRS}+R_S}\right)+m\cdot\frac{1}{G_N(R_{HRS}+R_S)}\right)} \end{aligned} \tag{4}$$

As mentioned in Equation (2), 'k_j' and 'm-k_j' are the numbers of LRS and HRS cells for Column #j, respectively. 'm' means the number of rows in the crossbar. G_N is the inverse of R_N. Similarly, with Equation (3), the output voltage degradation due to R_N is caused from $k_j\cdot\frac{1}{G_N}\cdot\left(\frac{1}{R_{LRS}+R_S}-\frac{1}{R_{HRS}+R_S}\right)$ in the denominator term of Equation (4). As a column has more LRS cells, the column current becomes degraded more significantly. If $G_{C,j}$ is adjusted according to the number of LRS cells for Column #j, the degradation due to $k_j\cdot\frac{1}{G_N}\cdot\left(\frac{1}{R_{LRS}+R_S}-\frac{1}{R_{HRS}+R_S}\right)$ can be reduced. Here $G_{C,j}$ implemented with a memristor can be programmed with different values to adjust its conductance for compensating the column current loss due to neuron resistance.

Figure 4a–d compares the ideal crossbar, the nonideal without compensation, and the nonideal with compensation. Among these four figures, Figure 4a,b indicate the source and output voltages, respectively, for R_W = 1Ω. Similarly, Figure 4c,d show the simulated source and output voltages for R_W = 2Ω. The source voltage compensation is achieved by the correction circuit shown on the left in Figure 3. The correction of the column's output voltage is performed by the circuit shown at the bottom in Figure 3.

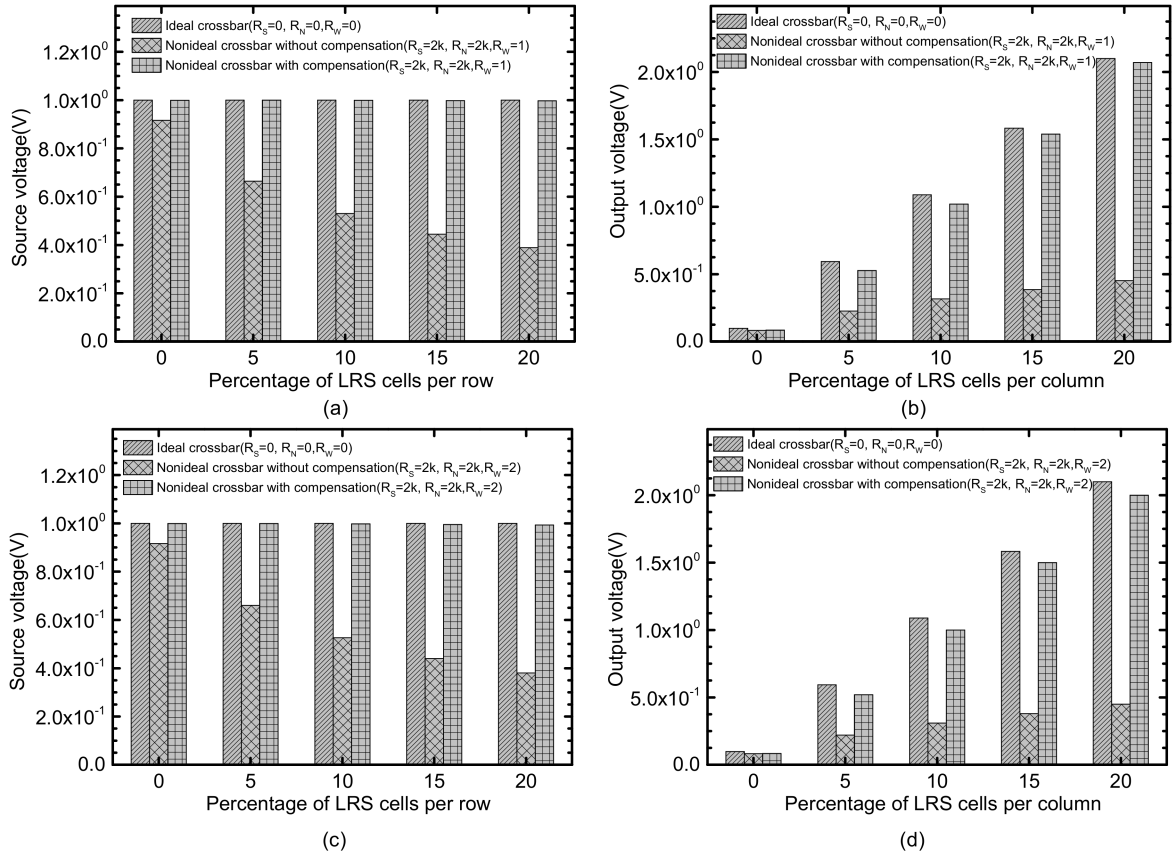

Figure 4. (**a**) Comparison of the source voltages of the nonideal crossbars between without and with the correction circuit for R_S = 2 kΩ and R_W = 1 Ω per cell. (**b**) Comparison of the output voltages of the nonideal crossbars between without and with the correction circuit for R_N = 2 kΩ and R_W = 1 Ω per cell. (**c**) Comparison of the source voltages of the nonideal crossbars between without and with the correction circuit for R_S = 2 kΩ and R_W = 2 Ω per cell. (**d**) Comparison of the output voltages of the nonideal crossbars between without and with the correction circuit for R_N = 2 kΩ and R_W = 2 Ω per cell.

Figure 4a indicates the source voltage can be compensated by using the correction circuit in Figure 3. Here, the nonideal crossbar without the correction shows the source voltage becomes degraded as small as 38.9% compared to the ideal crossbar when the percentage of LRS cells per row is 20%. However, the correction circuit added to the nonideal crossbar can improve the source voltage from 38.9% to 99.7% of the ideal crossbar, as shown in Figure 4a.

Figure 4b compares the output voltages of the three crossbar circuits, which are the ideal crossbar, the nonideal one without the correction circuit, and the nonideal one with the correction circuit, respectively. The correction circuit used in Figure 4b is shown at the bottom in Figure 3. The output voltage of the ideal crossbar is proportional to the number of LRS cells per column. However, the nonideal crossbar without compensation indicates the output voltage begins to be saturated when the number of LRS cells per column is increased. The correction circuit can restore the output voltage from 21.5% to 98.5% of the ideal crossbar when the percentage of LRS cells per row is 20%.

In Figure 4c,d, the ideal crossbar, the nonideal without compensation, and the nonideal with compensation, are simulated for R_W = 2 Ω. As expected, R_W = 2 Ω degrades the source and output voltages more severely than R_W = 1 Ω. In Figure 4c, the nonideal without compensation has the source voltage as small as only 38%, compared to the ideal crossbar. The correction circuit can recover the output voltage from 38% to 99.3% of the ideal crossbar, as indicated in Figure 4c. Similarly, Figure 4d shows the output voltage of the uncompensated crossbar is as small as 21.4% of the ideal crossbar. If the correction circuit is used in Figure 4d, the output voltage can reach as large as 95.2% of the ideal crossbar, when the percentage of LRS cells per column is assumed 20%.

In Figure 5, the percentage errors between the ideal and nonideal crossbars are simulated using the CADENCE SPECTRE version 6.1.6 circuit simulator. The simulated memristor crossbar is assumed to be 64 rows and 64 columns. Here the white and black pixels represent LRS and HRS, respectively, in Figure 5a.

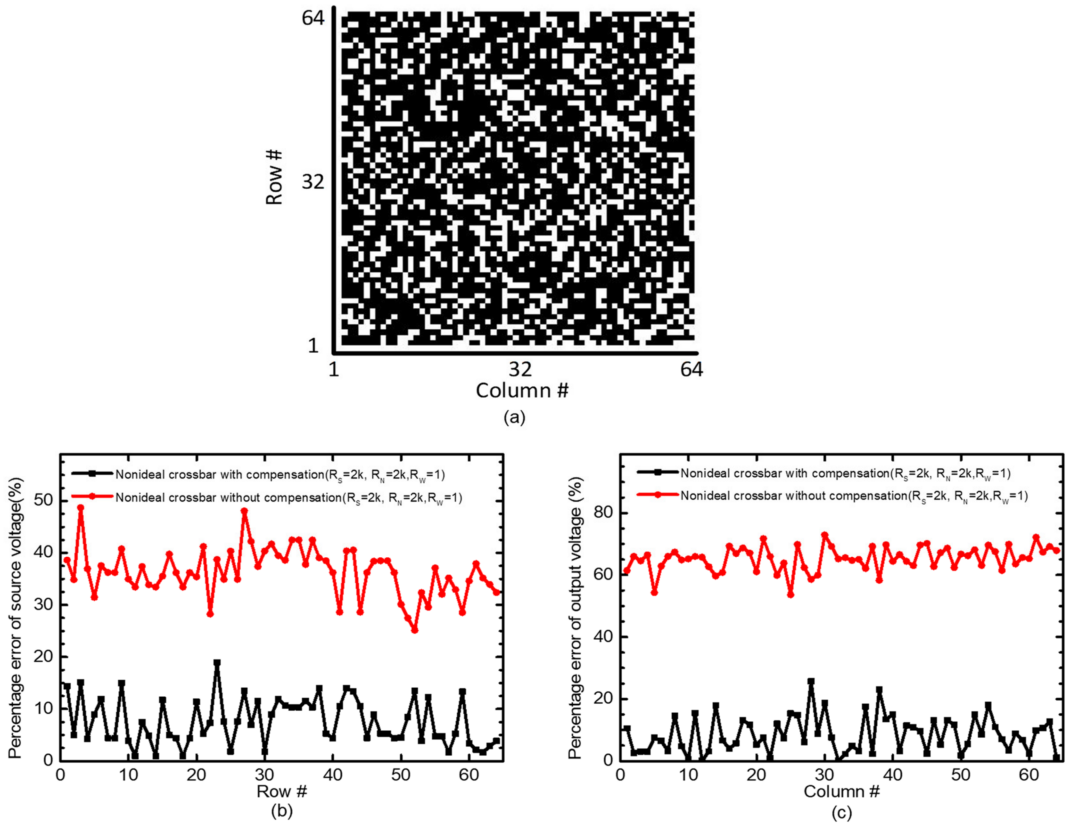

Figure 5. (**a**) Memristor crossbar with 64 rows and 64 columns. Here the white and black pixels represent LRS and HRS, respectively. (**b**) The percentage error of source voltage from Row #1 to Row #64. Here R_S = 2 kΩ, R_N = 2 kΩ, and R_W = 1 Ω. The percentage error between the ideal and nonideal without the correction circuit is shown in the red line. The percentage error with the correction is shown in the black line. (**c**) The percentage error of output voltage from Column #1 to Column #64.

Figure 5b indicates the percentage error of source voltage from Row #1 to Row #64. Here R_S = 2 kΩ, R_N = 2 kΩ, and R_W = 1 Ω. The percentage error between the ideal and nonideal without the correction circuit is shown in the red line. The percentage error with the correction is shown in the black line, in Figure 5b. The average percentage error of the uncompensated crossbar is as large as 36.7%. If the correction circuit is used, the percentage error in the source voltage can be reduced from 36.7% to 7.5%. Figure 5c shows the percentage error of output voltage from Column #1 to Column #64. The average percentage error of the uncompensated crossbar is as large as 65.5%. The correction circuit

can improve the percentage error from 65.5% to 8.6%. Almost the percentage error can be reduced to ~1/7, if the correction circuit is used.

3. Results

Figure 6a shows a simple schematic of the proposed correction circuit for compensating the voltage loss due to the nonideal effects, which was already shown in the left and bottom in Figure 3. R_1 represents feedback resistance of noninverting op amp. It should be noted R_1 has the same resistance for all the rows in the nonideal crossbar. In Figure 6a, $V_{IN,i}$ means the input voltage for Row #i. $V_{R,j}$ is the compensated voltage by the correction circuit. As explained earlier, the $V_{R,i}$ can be obtained with $V_{R,i} = (1 + G_{R,i}/G_1) \cdot V_{IN,i}$, where $R_{R,i}$ can be made of a memristor cell for adjusting its conductance according to the number of LRS cells for Row #i.

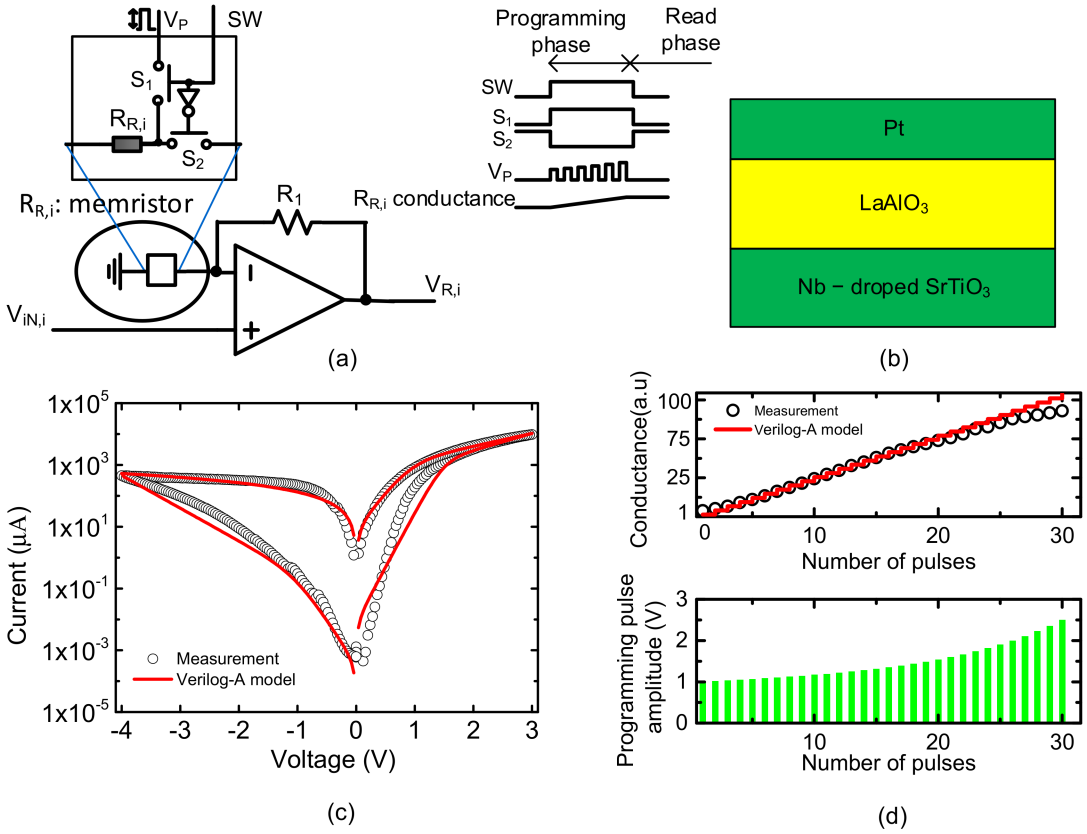

Figure 6. (**a**) Schematic of the correction circuit for source voltage of Row #i. Here $R_{R,i}$ is implemented with a memristor device. (**b**) Cross-sectional view of the memristor measured in this paper. (**c**) Comparison of the measurement and simulated Verilog-A model of the memristor device in Figure 6b. (**d**) Memristor programming for adjusting its conductance with increasing the number of programming pulses.

The white box in Figure 6a shows the memristor, $R_{R,i}$ with the programming circuit. $R_{R,i}$ is put between the minus terminal and ground node. $R_{R,i}$ should be programmed with different conductance values for each row according to the number of LRS cells for the corresponding row. SW is a switch control signal for programming $R_{R,i}$. S_1 and S_2 are the switches for programming and reading operations of the memristor, respectively. V_P is a memristor programming pulse train voltage, which can be modulated for linear and precise programming of memristor's conductance, as explained in Figure 6d. The simple waveforms of the memristor programming circuit are also shown in Figure 6a, where the memristor's conductance is changed according to the programming pulse train, V_P, during the programming phase. In the read phase, the memristor's conductance is used to compensate for the nonideal effect.

Figure 6b shows the cross-sectional view of the memristor device used in this paper. The memristor in Figure 6b has a film structure composed of Pt/LaAlO3/Nb-doped SrTiO stacked layers [18,20]. Figure 6c shows the measurement and the Verilog-A model for the memristor device in Figure 6b. The measured data are represented with open circles. The calculated current-voltage relationship from the Verilog-A model is shown in the red line. For the entire operation region, the Verilog-A model of the memristor is in good agreement with the measured data. In Figure 6c, Keithley-4200 (Semiconductor Characterization System) was used for the measurement of the current-voltage relationship of the memristor device. The model equations for the memristor device in Figure 6b are programmed with Verilog-A language and simulated by CADENCE SPECTRE version 6.1.6 circuit simulator in Figure 6c. The model equations used in Figure 6c were explained in detail in the previous publication [18]. For simulating the hybrid circuit of memristors and CMOS devices, we need to have CMOS parameters, in this paper. The CMOS model parameters used here were obtained from the 0.18-um CMOS process of CMOS logic Foundry Company. Figure 6d shows the programmed conductance of the memristor device according to the number of programming pulses. Here, the program-verify method with the fine Pulse Amplitude Modulation (PAM) is used for adjusting the conductance of the memristor precisely, as shown in Figure 6d. As the number of fine-modulated programming pulses is increased, the conductance of the memristor seems to change in proportion to the number of programming pulses [18]. This linear relationship between the programmed conductance and the number of programming pulses is very helpful in adjusting $R_{R,i}$ accurately in Figure 6a.

One more concern of programming $R_{R,i}$ may be endurance. Fortunately, the memristor in the correction circuit is programmed once according to the number of LRS cells of the corresponding row during the training phase. In the inference operation, the memristor $R_{R,i}$ in the correction circuit is only read not be programmed. The endurance cycles have been experimentally measured within 10^5–10^7, which can be enough not only for the inference phase but also for the training phase [10,32].

Figure 7 shows a tiled architecture of memristor crossbar composed of sub-crossbars for implementing a large network [30,32,33] For designing a neural network with a large number of neurons and synapses, it is better to put together sub-crossbars, not building a single big crossbar.

The tile architecture in Figure 7 is very useful in limiting the nonideal effects under a certain level. Usually, the size of sub-crossbar is within 128 × 128 array [30,32,33]. In this paper, we used the sub-crossbar with 100 × 100 memristor cells. In Figure 7, a sub-crossbar has 'a × b' memristor cells. 'a' and 'b' mean the numbers of rows and columns for sub-crossbar, respectively. If the total number of rows and columns needed for implementing a neural network is 'm × n', 'm' and 'n' can be calculated simply with 'm = M × a' and 'n = N × b', respectively [30]. Here, 'M' and 'N' are the numbers of sub-crossbars for rows and columns, respectively. For example, if we try to implement the MNIST neural network with 784 input neurons and 200 hidden neurons, 'M' and 'N' should be 8 and 2, respectively.

Considering practical parameters of memristor crossbars, we used LRS = 20 kΩ and HRS = 2 MΩ, for the circuit simulation, in this paper. These LRS and HRS values are obtained from the experimental parameters of real fabricated crossbars [30]. The line resistance used in the circuit simulation is 1 Ω per cell obtained from TSMC RRAM data [33]. The sub-crossbar size used in the MNIST and CIFAR-10 simulation is 100 × 100 [30,32,33].

To estimate the neural network's performance realized with the nonideal-effect correction, the memristor crossbar in Figure 3 is tested here for the MNIST dataset [12]. The MNIST data set is composed of 60,000 training images and 10,000 testing ones. Each image has 28 × 28 = 784 pixels. The classified items are hand-written digits from '0′ to '9′. Thus, each digit has 6000 training and 1000 testing vectors. The neural network simulated here for testing the MNIST dataset has 784 input and 200 hidden neurons. The number of output neurons is 10. The fully connected neural network (784-200-10 neurons) for testing MNIST

vectors is implemented with the tiled architecture of multiple sub-crossbars, as shown in Figure 7.

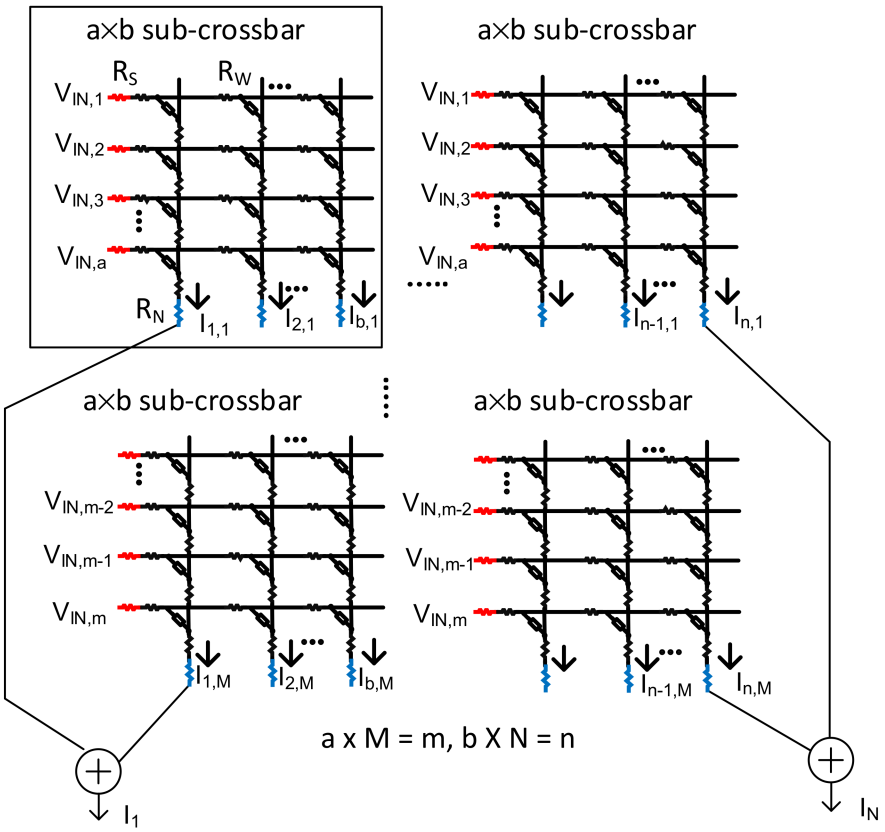

Figure 7. Tile architecture of multiple sub-crossbars for implementing a memristor crossbar with 'm × n' cells. Here each sub-crossbar size is 'a × b'. The numbers of sub-crossbars for rows and columns are 'M' and 'N', respectively. 'M' and 'N' can be calculated from 'm = M × a' and 'n = N × b', respectively. Here RS, RW, and RN are the source, line, and neuron resistance, respectively, contributing to the nonideal effects in the memristor crossbar.

Figure 8a compares the recognition rate of the ideal crossbar, the nonideal without the correction, and the nonideal with the correction circuit. Here the source and line resistance are assumed 2 kΩ and 1 Ω, respectively. The neuron resistance is increased from 1 kΩ to 3 kΩ. In Figure 8a, the first, second, and third bars are for the ideal, the nonideal without correction, and the nonideal with correction, respectively. When the neuron resistance is 1 kΩ, the uncompensated crossbar and compensated one indicates the recognition rate, 92.5% and 95.4%, respectively, compared to the rate of the ideal crossbar, 95.5%. Here, it should be noted that the correction circuit could improve the rate of the compensated crossbar by 2.9% compared to the uncompensated. If the neuron resistance is as large as 3 kΩ, the rate of the crossbars without and with the correction circuit are 90.4% and 95.1%, respectively. The gap in recognition rate between the uncompensated and compensated crossbars is increased from 2.9% to 4.7%, as shown in Figure 8a. One thing to note here is that the recognition rate of 95.5% in Figure 8a seems lower than the state-of-the-art rate as high as 99%. This is mainly due to the ternary synaptic weights used in the memristor crossbar instead of the floating-point numbers. When the memristor crossbar is tested with the integer synaptic weights, the rate could be as high as 98% for MNIST vectors.

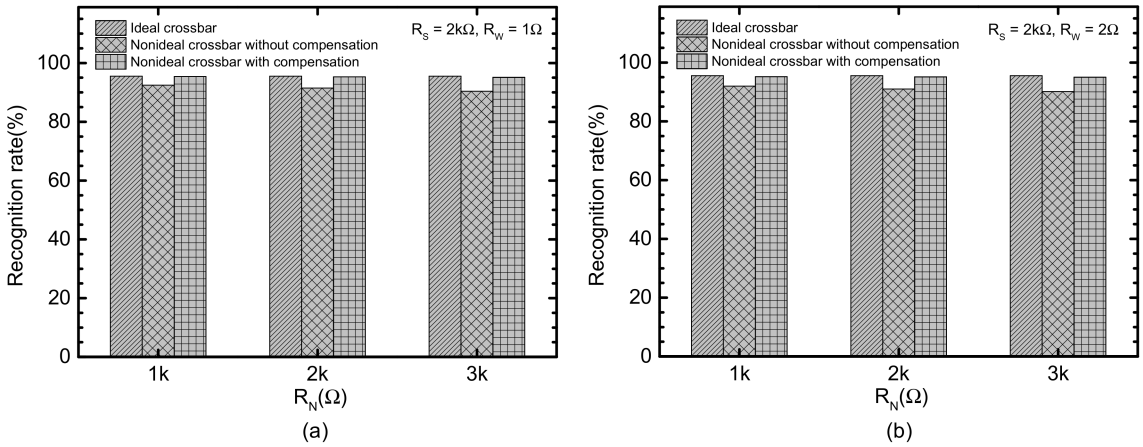

Figure 8. (**a**) Comparison of the recognition rate of the nonideal crossbars between without and with the correction circuit for MNIST data set with R_W = 1 Ω, (**b**) Comparison of the recognition rate of the nonideal crossbars between without and with the correction circuit for MNIST data set with R_W = 2 Ω.

Similarly, Figure 8b compares the nonideal crossbars without and with the correction circuit, for R_W = 2 Ω. Here RW in Figure 8b is 2× larger than R_W in Figure 8a. The recognition rate without the correction is as small as 90.1%. However, the correction circuit can increase the rate as large as 95%. The gap between without and with the correction circuit is 4.9% in Figure 8b.

Next, the proposed correction circuit is tested for the CIFAR-10 (Canadian Institute For Advanced Research) data set. Like the MNIST dataset, CIFAR-10 has 50,000 training images and 10,000 testing ones for classifying 10 image objects. Each image is composed of 32 × 32 RGB pixels. For each image object, there are 5000 training and 1000 testing vectors. The ten image objects are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Here the ResNet CNN architecture is used for testing the CIFAR-10 dataset [34]. Here only the fully-connected layers in the ResNet are implemented with the tile architecture of sub-crossbars in Figure 7. Here, in the fully connected layer in ResNet, the number of hidden neurons is 200. The output neurons for the neural network should be 10 for classifying the 10 image objects such as airplanes, automobiles, and so on in the CIFAR-10 dataset.

Figure 9a compares the recognition rate of the ideal crossbar, the nonideal without the correction, and the nonideal with correction circuit, for the CIFAR-10 dataset. In Figure 9a, the first, second, and third bars are for the ideal, the nonideal without correction, and the nonideal with correction, respectively. When the neuron resistance is as small as 1 kΩ, the uncompensated crossbar and compensated one indicates the recognition rate, 87.4% and 88.7%, respectively, compared to the rate of the ideal crossbar, 88.9%. The gap between the uncompensated and the compensated is as small as 1.3% for the neuron resistance = 1 kΩ. For R_N = 3 kΩ, the correction circuit seems to improve the recognition rate from 85.3% to 88.1%, compared to the nonideal without compensation. The gap between the uncompensated and compensated becomes as large as 2.8%, for R_N = 3 kΩ.

Figure 9b indicates the recognition rate of the nonideal crossbars without and with the correction circuit, for R_W = 2Ω. For R_N = 3 kΩ, the recognition rate without the correction is as small as 85.1%. However, the correction circuit can increase the rate as large as 88%. The gap between without and with the correction circuit is as large as 2.9% in Figure 9b, for R_N = 3 kΩ.

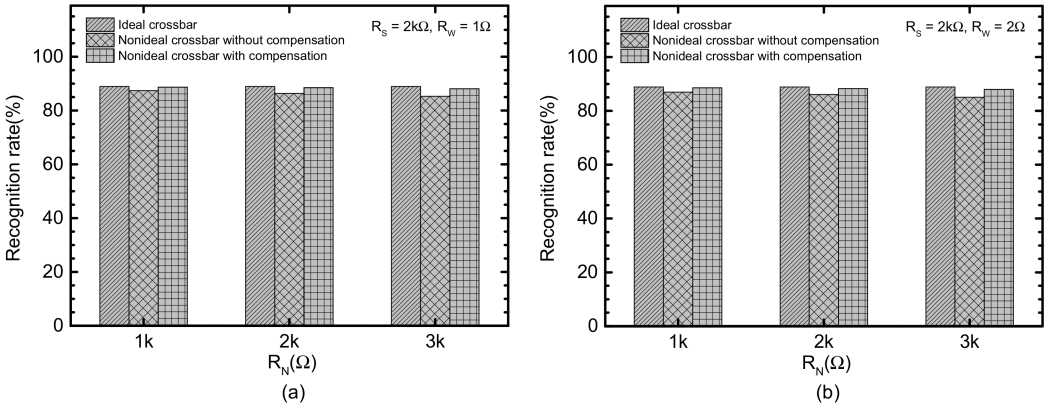

Figure 9. (**a**) Comparison of the recognition rate of the nonideal crossbars between without and with the correction circuit for CIFAR-10 data set with R_W = 1 Ω (**b**) Comparison of the recognition rate of the nonideal crossbars between without and with the correction circuit for CIFAR-10 data set with R_W = 2 Ω.

One more thing to discuss here is the power overhead due to the correction circuit in Figure 3. Using the circuit simulator of CADENCE SPECTRE version 6.1.6, the power consumption of the correction circuit is estimated at roughly ~0.76 uA for each noninverting amplifier. The power overhead ratio due to the correction circuit can be approximated with the following equation.

$$R_{overhead} \approx \frac{(m+n)\cdot V_{DD}\cdot I_{correction,avg}}{m\cdot n\cdot V_{DD}\cdot\left(I_{LRS,avg}\cdot\alpha + I_{HRS,avg}\cdot(1-\alpha)\right)} \quad (5)$$

Here $R_{overhead}$ means the overhead ratio of the power due to the correction circuit with respect to the power consumption of the memristor crossbar. 'm' and 'n' are the numbers of rows and columns, respectively, in the crossbar. $I_{correction,avg}$ means the average current consumption for each noninverting op amp used in the correction circuit. VDD is the power supply voltage. $I_{LRS,avg}$ and $I_{HRS,avg}$ are the average currents for LRS cells and HRS cells in the crossbar, respectively. 'α' means the activity ratio of LRS cells during the inference phase. '$1-\alpha$' is the activity ratio of HRS cells during the inference time. As expected in Equation (5), the overhead ratio depends on the crossbar size with 'm' and 'n'. Assuming LRS = 20 kΩ and HRS = 2 MΩ, the overhead ratio is estimated at less than 1%. Here the sub-crossbar size is assumed 100 ×100. The activity ratio of LRS cells is assumed at 30% on average, which is obtained from the synaptic weights calculated from MATLAB simulation. If LRS becomes higher, the overhead ratio in Equation (5) becomes higher, too.

In addition, it should be discussed the power overhead due to the mixed-mode interface circuit such as an analog-to-digital (ADC) converter [32,35]. As mentioned, the memristor crossbar can perform VMM operation based on the analog current-voltage relationship of memristors. For performing the VMM operation in memristor crossbars, the mixed-mode interface circuit should be added to the memristor crossbar. The ADC is used for converting the VMM result to the digital data for delivering them to the other networks. For doing this, the ADC should consume an amount of computing power. Comparing the power consumption between the mixed-mode interface circuit and the memristor crossbar, it has been reported that the mixed-mode interface circuit including ADC consumes 9× larger power than the memristor crossbar [32]. Though the ADC consumes most of the computing power, the overall computational energy efficiency of the memristor crossbar-based neural networks can easily outperform the conventional digital-based neural networks relying on the digital VMM operation [35]. Thus, despite the power overhead of the mixed-mode interface circuit, the memristor crossbar-based neural networks can be very suitable particularly for the edge intelligence hardware such as IoT sensors, where energy-efficient computing is very critical [13,35].

Finally, it should be mentioned that we focused on solving the nonideal-effect problems related to the parasitic resistance such as source resistance, neuron resistance, and line resistance, in this paper. Of course, other nonideal effects are not considered in this paper, such as yield-limited memristor defects, nonlinear current-voltage behaviors, etc. [36,37]. Actually, for considering the yield-limited memristor defects, a defect-aware in-situ crossbar training can be used with the parasitic-resistance correction circuit proposed in this paper [37]. By putting the defect-aware training and parasitic-resistance correction circuit together, the nonideal-effect correction circuit can compensate the voltage and current loss due to the parasitic resistance and the defect-aware training can compensate the performance loss due to memristor defects such as stuck-at-0 and stuck-at-1 simultaneously.

For the nonideal-effect correction circuit shown in Figure 3, the added OP amp may have the offset voltage problem and the added resistors such as RR and RC may suffer process variation. However, the nonideal effects caused by the OP amp's offset voltage and added RR and RC variations can be thought to cause smaller degradation of neural network's performance than the gain of the recognition rate with the correction circuit. From the MNIST simulation, the variations of OP amp's offset voltage, RR, and RC can degrade the MNIST recognition rate by about ~0.6%. This rate loss of ~0.6% is much smaller than ~4.7% that is the gain of the recognition rate due to the correction circuit.

4. Conclusions

The nonideal effects such as parasitic source, line, and neuron resistance can affect the input voltage and the output current of the memristor crossbar significantly. The nonideal effects cause the degradation of the neural network's performance realized with the nonideal memristor crossbar. To avoid performance degradation due to the nonideal effects, the adaptive training methods were proposed previously [29,30]. However, the complicated training algorithm can add a heavy computational burden to the neural network hardware. Especially, this hardware and algorithmic burden can be more serious for edge intelligence applications such as IoT sensors.

In this paper, the memristor-CMOS hybrid neuron circuit was proposed for compensating the voltage loss due to the nonideal effects during the inference phase, not the training phase. Unlike the previous linear correction method performed by the external hardware [10], the proposed memristor-CMOS hybrid neuron circuit can be included in the memristor crossbar to minimize the power and hardware overheads caused by the nonideal-effect correction.

The proposed correction circuit was verified to be able to restore both the source voltage and the output voltage degradation due to the nonideal effects. For the source voltage, the average percentage error of the uncompensated crossbar is as large as 36.7%. If the correction circuit is used, the percentage error in the source voltage can be reduced from 36.7% to 7.5%. For the output voltage, the average percentage error of the uncompensated crossbar is as large as 65.2%. The correction circuit can improve the percentage error in the output voltage from 65.2% to 8.6%. Almost the percentage error can be reduced to ~1/7 if the correction circuit is used.

The nonideal memristor crossbar with the correction circuit was tested for MNIST and CIFAR-10 data sets in this paper. For MNIST, the uncompensated and compensated crossbars indicated the recognition rate of 90.5% and 95.1%, respectively, compared to 95.5% of the ideal crossbar. For CIFAR-10, the nonideal crossbars without and with the nonideal effect correction showed the rate of 85.4% and 88.3%, respectively, compared to the ideal crossbar not suffering the nonideal effects achieving the rate as large as 88.9%.

Author Contributions: K.-S.M. defined the research topic. T.V.N. and J.A. performed the simulation and measurement. T.V.N. and K.-S.M. wrote the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The work was financially supported by NRF-2015R1A5A7037615, NRF-2019K1A3A1A25000279, NRF-2021R1A2C1011631, Research Grant (Incremental Funds)-N62909-20-1-2021-P00001-Memristor-based Neural Network Circuits, and SRFC-TA1903-01.

Acknowledgments: The CAD tools were supported by IC Design Education Center (IDEC), Daejeon, Korea.

Conflicts of Interest: The authors declare that they have no competing interests.

References

1. Pouyanfar, S.; Sadiq, S.; Yan, Y.; Tian, H.; Tao, Y.; Reyes, M.P.; Shyu, M.-L.; Chen, S.-C.; Iyengar, S.S. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.* **2018**, *51*, 1–36. [CrossRef]
2. Shrestha, A.; Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **2019**, *7*, 53040–53065. [CrossRef]
3. Jo, J.; Kung, J.; Lee, Y. Approximate LSTM Computing for Energy-Efficient Speech Recognition. *Electronics* **2020**, *9*, 2004. [CrossRef]
4. Bohr, M.T.; Young, I. CMOS Scaling Trends and Beyond. *IEEE Micro* **2017**, *37*, 20–29. [CrossRef]
5. Dean, J.; Patterson, D.; Young, C. A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution. *IEEE Micro* **2018**, *38*, 21–29. [CrossRef]
6. Linn, E.; Rosezin, R.; Tappertzhofen, S.; Böttger, U.; Waser, R. Beyond von Neumann—Logic operations in passive crossbar arrays alongside memory operations. *Nanotechnology* **2012**, *23*, 305205. [CrossRef]
7. Wright, C.D.; Hosseini, P.; Diosdado, J.A.V. Beyond von-Neumann computing with nanoscale phase-change memory devices. *Adv. Funct. Mater.* **2013**, *23*, 2248–2254. [CrossRef]
8. Indiveri, G.; Liu, S.-C. Memory and Information Processing in Neuromorphic Systems. *Proc. IEEE* **2015**, *103*, 1379–1397. [CrossRef]
9. Sebastian, A.; Le Gallo, M.; Eleftheriou, E. Computational phase-change memory: Beyond von Neumann computing. *J. Phys. D Appl. Phys.* **2019**, *52*, 443002. [CrossRef]
10. Hu, M.; Graves, C.E.; Li, C.; Li, Y.; Ge, N.; Montgomery, E.; Davila, N.; Jiang, H.R.; Williams, S.; Yang, J.J.; et al. Memristor-based analog computation and neural network classification with a dot product engine. *Adv. Mater.* **2018**, *30*, 1705914. [CrossRef]
11. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
12. Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal. Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
13. Krestinskaya, O.; James, A.P.; Chua, L.O. Neuromemristive Circuits for Edge Computing: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4–23. [CrossRef]
14. Jo, S.H.; Chang, T.; Ebong, I.; Bhadviya, B.B.; Mazumder, P.; Lu, W. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* **2010**, *10*, 1297–1301. [CrossRef]
15. Hu, M.; Li, H.; Chen, Y.; Wu, Q.; Rose, G.; Linderman, R.W. Memristor Crossbar-Based Neuromorphic Computing System: A Case Study. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 1864–1878. [CrossRef] [PubMed]
16. Li, Y.; Wang, Z.; Midya, R.; Xia, Q.; Yang, J.J. Review of memristor devices in neuromorphic computing: Materials sciences and device challenges. *J. Phys. D Appl. Phys.* **2018**, *51*, 503002. [CrossRef]
17. Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, S. The missing memristor found. *Nature* **2008**, *453*, 80–83. [CrossRef]
18. Truong, S.N.; Van Pham, K.; Yang, W.; Shin, S.; Pedrotti, K.; Min, K.-S. New pulse amplitude modulation for fine tuning of memristor synapses. *Microelectron. J.* **2016**, *55*, 162–168. [CrossRef]
19. Song, C.; Liu, B.; Wen, W.; Li, H.; Chen, Y. A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system. In Proceedings of the IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA), Hsinchu, Taiwan, 16–18 August 2017; pp. 1–6.
20. Pham, K.V.; Tran, S.B.; Nguyen, T.V.; Min, K.-S. Asymmetrical training scheme of binary-memristor-crossbar-based neural networks for energy-efficient edge-computing nanoscale systems. *Micromachines* **2019**, *10*, 141. [CrossRef]
21. Adam, G.C.; Hoskins, B.D.; Prezioso, M.; Merrikh-Bayat, F.; Chakrabarti, B.; Strukov, D.B. 3-D Memristor Crossbars for Analog and Neuromorphic Computing Applications. *IEEE Trans. Electron. Devices* **2017**, *64*, 312–318. [CrossRef]
22. Chakrabarti, B.; Lastras-Montaño, M.A.; Adam, G.; Prezioso, M.; Hoskins, B.; Payvand, M. A multiply-add engine with monolithically integrated 3D memristor crossbar/CMOS hybrid circuit. *Sci. Rep.* **2017**, *7*, 1–10.
23. Wang, T.-Y.; Meng, J.-L.; Rao, M.-Y.; He, Z.-Y.; Chen, L.; Zhu, H.; Sun, Q.-Q.; Ding, S.-J.; Bao, W.-Z.; Zhou, P.; et al. Three-Dimensional Nanoscale Flexible Memristor Networks with Ultralow Power for Information Transmission and Processing Application. *Nano Lett.* **2020**, *20*, 4111–4120. [CrossRef] [PubMed]
24. Lin, P.; Li, C.; Wang, Z.; Li, Y.; Jiang, H.; Song, W.; Rao, M.; Zhuo, Y.; Upadhyay, N.K.; Barnell, M.; et al. Three-dimensional memristor circuits as complex neural networks. *Nat. Electron.* **2020**, *3*, 225–232. [CrossRef]
25. Sheng, X.; Graves, C.E.; Kumar, S.; Li, X.; Buchanan, B.; Zheng, L. Low-Conductance and Multilevel CMOS-Integrated Nanoscale Oxide Memristors. *Adv. Electron. Mater.* **2019**, *5*, 1800876. [CrossRef]
26. Graves, C.E.; Li, C.; Sheng, X.; Miller, D.; Ignowski, J.; Kiyama, L.; Strachan, J.P. In-Memory Computing with Memristor Content Addressable Memories for Pattern Matching. *Adv. Mater.* **2020**, *32*, 2003437. [CrossRef]

27. Qin, Y.-F.; Bao, H.; Wang, F.; Chen, J.; Li, Y.; Miao, X.-S. Recent Progress on Memristive Convolutional Neural Networks for Edge Intelligence. *Adv. Intell. Syst.* **2020**, *2*, 2000114. [CrossRef]
28. Li, C.; Hu, M.; Li, Y.; Jiang, H.; Ge, N.; Montgomery, E.; Zhang, J.; Song, W.; Dávila, N.; Graves, C.E.; et al. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **2018**, *1*, 52–59. [CrossRef]
29. Liu, B.; Li, H.; Chen, Y.; Li, X.; Huang, T.; Wu, Q.; Barnell, M. Reduction and IR-drop compensations techniques for re-liable neuromorphic computing systems. In Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2–6 November 2014; pp. 63–70.
30. Chakraborty, I.; Roy, D.; Roy, K. Technology aware training in memristive neuromorphic systems for noni-deal synaptic crossbars. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 335–344. [CrossRef]
31. Krizhevsky, A.; Hinton, G. Convolutional deep belief networks on cifar-10. *Unpubl. Manuscr.* **2010**, *40*, 1–9.
32. Cai, F.; Correll, J.M.; Lee, S.H.; Lim, Y.; Bothra, V.; Zhang, Z.; Flynn, M.P.; Lu, W.D. A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. *Nat. Electron.* **2019**, *2*, 290–299. [CrossRef]
33. Murali, G.; Sun, X.; Yu, S.; Lim, S.K. Heterogeneous Mixed-Signal Monolithic 3-D In-Memory Computing Using Resistive RAM. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 386–396. [CrossRef]
34. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
35. Amirsoleimani, A.; Alibart, F.; Yon, V.; Xu, J.; Pazhouhandeh, M.R.; Ecoffey, S.; Beilliard, Y.; Genov, R.; Drouin, D. In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives. *Adv. Intell. Syst.* **2020**, *2*, 2000115. [CrossRef]
36. Wang, W.; Song, W.; Yao, P.; Li, Y.; Van Nostrand, J.; Qiu, Q.; Yang, J.J. Integration and Co-design of Memristive Devices and Algorithms for Artificial Intelligence. *iScience* **2020**, *23*, 101809. [CrossRef]
37. Li, C.; Belkin, D.; Li, Y.; Yan, P.; Hu, M.; Ge, N.; Jiang, H.; Montgomery, E.; Lin, P.; Wang, Z.; et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nat. Commun.* **2018**, *9*, 1–8. [CrossRef]