scientific reports

Check for updates

OPEN

Alternative negative weight for simpler hardware implementation of synapse device based neuromorphic system

Geonhui Han, Chuljun Lee, Jae-Eun Lee, Jongseon Seo, Myungjun Kim, Yubin Song, Young-Ho Seo & Daeseok Lee✉

Lately, there has been a rapid increase in the use of software-based deep learning neural networks (S-DNN) for the analysis of unstructured data consumption. For implementation of the S-DNN, synapse-device-based hardware DNN (H-DNN) has been proposed as an alternative to typical Von-Neumann structural computing systems. In the H-DNN, various numerical values such as the synaptic weight, activation function, and etc., have to be realized through electrical device or circuit. Among them, the synaptic weight that should have both positive and negative numerical values needs to be implemented in a simpler way. Because the synaptic weight has been expressed by conductance value of the synapse device, it always has a positive value. Therefore, typically, a pair of synapse devices is required to realize the negative weight values, which leads to additional hardware resources such as more devices, higher power consumption, larger area, and increased circuit complexity. Herein, we propose an alternative simpler method to realize the negative weight (named weight shifter) and its hardware implementation. To demonstrate the weight shifter, we investigated its theoretical, numerical, and circuit-related aspects, following which the H-DNN circuit was successfully implemented on a printed circuit board.

Recently, there has been a substantial increase in the consumption of unstructured data, such as images, movies, songs, sensory signals, and others[1,2]. To effectively analyze this data, software-based deep learning neural networks (S-DNN) are widely utilized[3]. However, in hardware respect, the conventional Von-Neumann structural computing system has insufficient analog-support structure for the S-DNN, due to several inherent limitations such as high energy consumption, low data processing speed, and etc[4–6]. Thus, there is an urgent need for human–brain-inspired, potent, and efficient computing systems[7]. As one of the promising approaches, synapse-device-based hardware DNN (H-DNN) has been proposed to overcome the limitations[8,9].

In contrast to the S-DNN, negative weight which is one of core values in the deep learning neural network (DNN) cannot be directly implemented in the H-DNN. Considering that weight is normally represented as conductance of the synapse device which always has a positive value, a pair of synapse device is employed to express the negative weight: named as pair-synapse method[10,11]. Conductance values of the two synapse devices are subtracted from each other through additional circuitry to indirectly express the negative weight, which requires additional devices, larger area, higher power consumption, and increased circuit complexity[12–14]. Hence, in this research, we propose a simpler way to implement the negative weight.

During a vector matrix multiplication (VMM) of the DNN, each weight (conductance) is multiplied to input bias, then all output currents are summed through column line (bit line)[15–20]. Because the weight can have positive or negative values in the S-DNN, the summed output currents can increase or decrease. To hardware implement the VMM, in a typical way, the negative weight value is expressed by the pair-synapse method[10,11]. In other words, conductance of one synapse device plays a role of reference conductance while conductance of the other synapse device is subtracted from the reference conductance. In the pair-synapse method, each weight needs to be expressed as two synapse devices with additional circuits; the negative weight is implemented at device level.

However, we can simply implement the negative weight by utilizing the summed output current. Based on the fact that all weights are multiplied to input bias and summed through column lines, we can make references for each output current of column line (named as weight shifter). Thus, for each column, the reference output

Department of Electronic Materials Engineering, Kwangwoon University, Seoul 01897, Republic of Korea. ✉email: leeds@kw.ac.kr

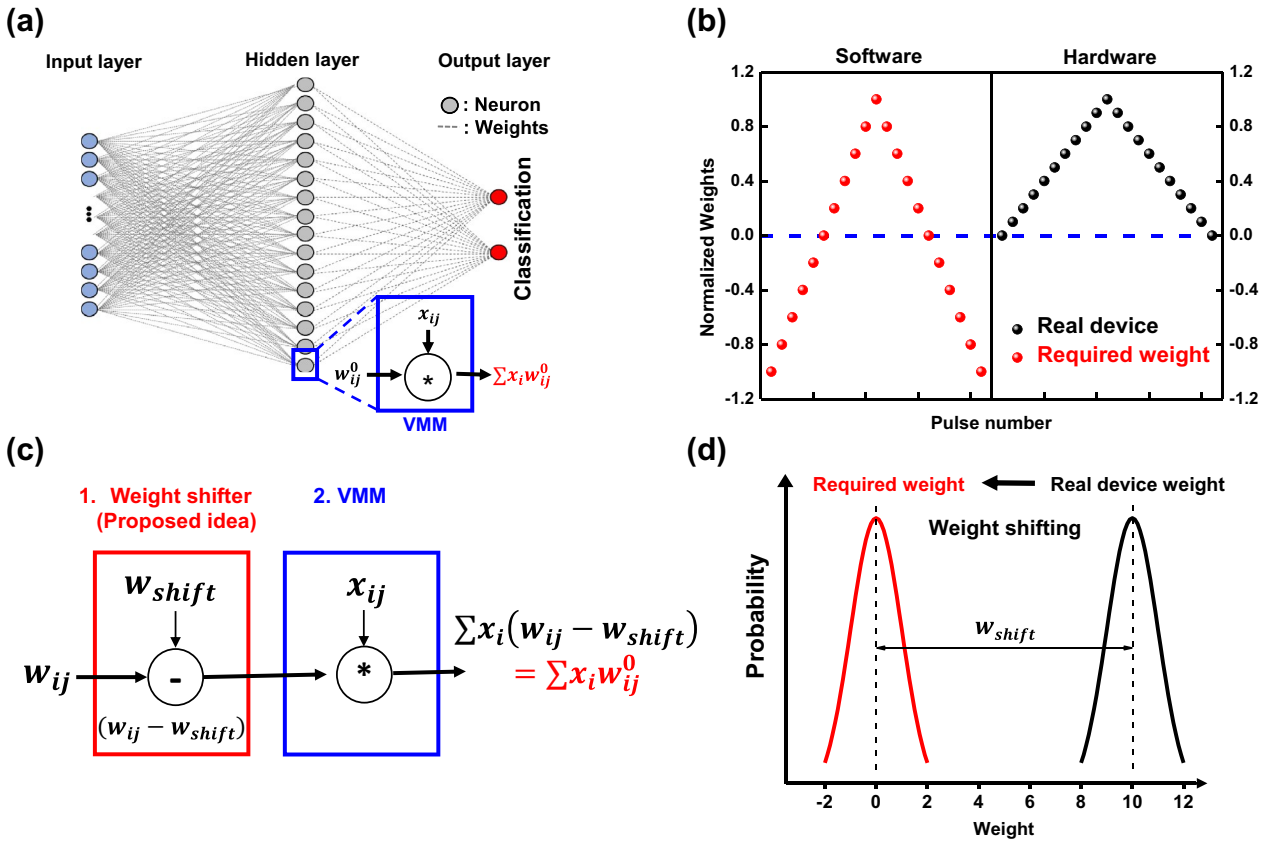


Figure 1. (**a**) Simple illustration of employed deep learning neural network (DNN) with the vector matrix multiplication (VMM). (**b**) Normalized required weight values in software based DNN (S-DNN), and weight values of real synapse device. Contrast to the S-DNN, the synapse device has only positive conductance values which is expressed as weight values. (**c**) Concept of proposed weight shifter to realize negative weight in the hardware based DNN (H-DNN). The positive conductance values of synapse device can be considered as positively shifted weight values. The difference between required weight values of the S-DNN and positively shifted weight values of the H-DNN (w_{shift}) can be subtracted during the VMM. (**d**) Distributions of weight values in the S-DNN and H-DNN. The w_{shift} can be compensated by the weight shifter during VMM.

current can be subtracted from the summed output current. In the array aspect, the negative weight can be implemented as the output current of bit line through the weight shifter. We can implement the H-DNN with less synapse devices, simple circuits, and low power consumption.

Results and discussion

To realize the DNN, the VMM is necessary, as shown in Fig. 1a. Therefore, in the H-DNN, arrays of synapse device are utilized as the VMM. However, the negative weight can not be implemented efficiently because the conductance of synapse device is always a positive value (Fig. 1b). In other words, the S-DNN has both positive and negative weight values ($\pm w_{ij}^0$), and they are multiplied to input data (x_{ij}). Then, the multiplied results ($\pm w_{ij}^0 x_{ij}$) are summed in parallel through the VMM[21–23]. Based on the $\pm w_{ij}^0$ of S-DNN, final values could be increased or decreased. In contrast to the S-DNN, the synapse device can express only positive weight values. Thus, in typical way, conductance values of two synapse devices are subtracted for expression of one negative weight[12,24]. This method consumes more power and requires various additional resources, such as additional synapse devices and subtraction circuits[10].

Therefore, as shown in Fig. 1c, d, we proposed the weight shifter; the median of weight values in the S-DNN is moved from zero to positive region to make all weight values become positive. Based on the positive weight values of the S-DNN, it can be directly expressed by the conductance of synapse devices (w_{ij}). In this case, because all weight values are positive, the output current will be increased. Therefore, the changed value of median should be considered; we returned the median back to its original value in output current respect.

As an example, in Fig.1d, when the weight values of the S-DNN are in the range from − 2 to 2, we can shift the median of the weight values from 0 to 10. Then the weight values are in the range from 8 to 12; the positive weight values (8–12) of the S-DNN can be directly expressed by the conductance values of synapse devices. Based on these values, the VMM can be conducted through the synapse-device array. After the VMM, the shifted median of weight value needs to be returned back to 0. Thus, the difference between the original and shifted median of weight values ($w_{shift} = 10$) needs to be subtracted from the VMM results (output current at each column).

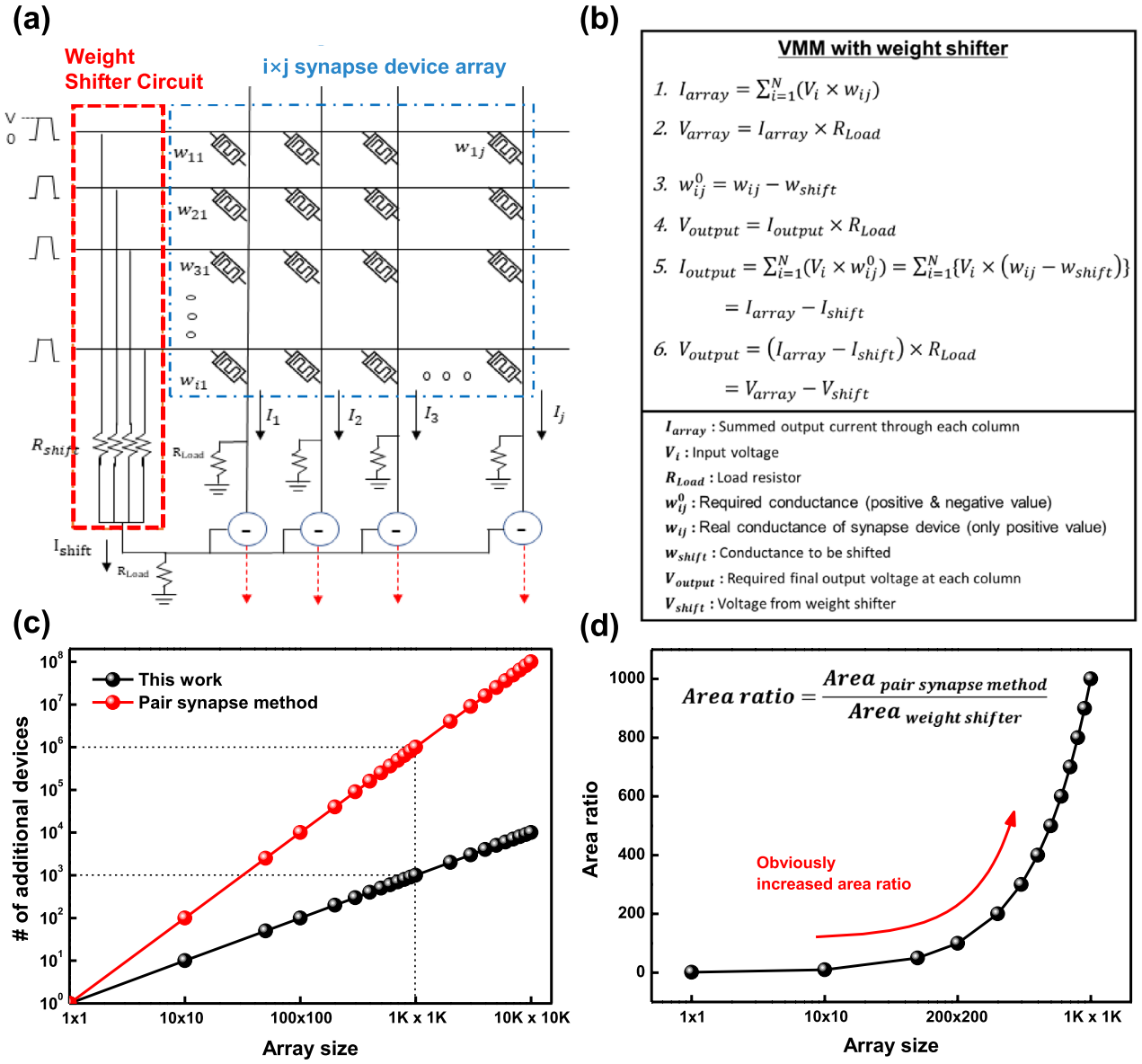


Figure 2. (**a**) Schematic of the weight shifter with i×j synapse-device array. (**b**) Theoretical equation of the weight shifter. The conductance value of synapse device (w_{ij}) can be considered as positively shifted weight value ($=w_{ij}^0+w_{shift}$). When we consider that output results of the VMM will be total columnar current (I_{array}), it is derived from multiplication between input data (V_i) and conductance of synapse device (w_{ij}). Then, output voltage of one column (V_{array}) can result from multiplying the I_{array} by load resistor (R_{Load}). In the same manner, output voltage of the weight shifter can be derived by multiplying output current of the weight shifter (I_{shift}) by R_{Load}. As a result, the final output voltage of the VMM (V_{output}) can be obtained by subtracting the V_{shift} from V_{array}. (**c**) Comparison of additionally required devices between conventional pair synapse method and proposed weight shifter for realization of the negative weight. (**d**) Area ratio (= area of conventional method/area of weight shifter) depending on array size.

The simplified H-DNN circuit with the proposed weight shifter and theoretical equation of the weight shifter are presented with details in Fig. 2a, b, respectively. Based on the input pulses (indicating input data: v_i) and synapse devices having various positive conductance values (w_{ij}), we can obtain results of the VMM as output current (I_{array}). And it leads to the output voltage (V_{array}) across the load resistor (R_{Load}); note that the w_{ij} implies conductance of the synapse device, which has only positive values.

During this process, the w_{ij} can be considered as already positively shifted weight because the employed conductance values are all positive. Thus, we can drive the required conductance value (w_{ij}^0) which is representing the weight value of S-DNN as below.

$$w_{ij}^0 = w_{ij} - w_{shift}$$

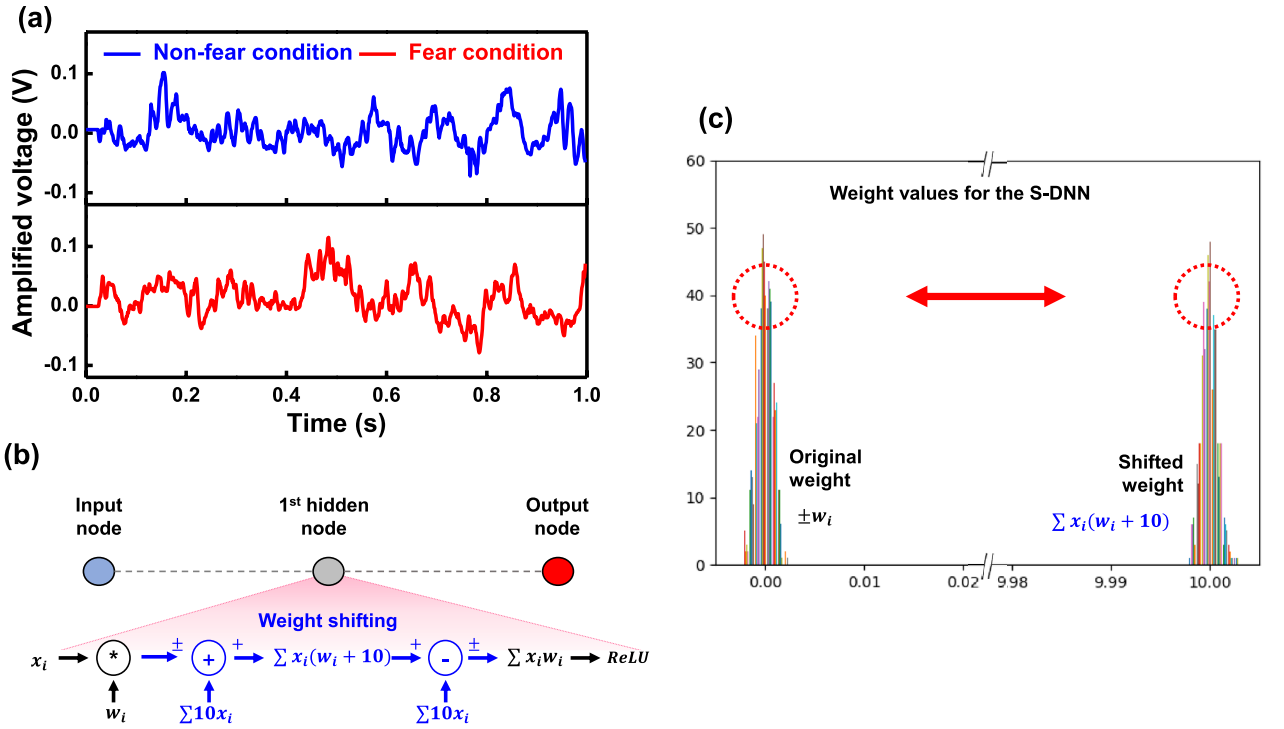


Figure 3. (**a**) Part of rat's neural signals in non-fear and fear conditions. (**b**) Sequence of the weight shifter in S-DNN. Firstly, the weight values are positively shifted; then it returns back to origin before the ReLU function. (**c**) Weight values utilized in the S-DNN (original weights and shifted weights).

Then, required output voltage of each column (V_{output}) can be derived from output current (I_{output}) and R_{Load}. Because the w_{ij}^0 is defined by w_{ij} and w_{shift}, the I_{output} is derived as $I_{array} - I_{shift}$. Consequently, for one column of synapse-device array, the V_{output} can be simply expressed as below.

$$V_{output} = (I_{array} - I_{shift}) \times R_{shift} = V_{array} - V_{shift}$$
$$= \sum_{i=1}^{n} \{(V_i \times (w_{ij} - w_{shift})) \times R_{Load}\}$$

Comparing to the typical expression of the negative weight (:pair-synapse method), we can minimize hardware resources by the weight shifter for realization of the negative weight, as shown in Fig. 2c, d[25,26]. Figure 2c shows a comparison of the number of additional devices, between the conventional pair-synapse method and the proposed weight shifter. When the synapse-device array size becomes larger, the number of additional devices increases exponentially for the pair-synapse method. In contrast, the weight shifter needs only a small number of additional devices because the size of weight shifter is only dependent on the number of rows in the synapse-device array (Fig. 2a). From this result, we can estimate the additional area for both methods, and shown in Fig. 2d. The ratio of area between both methods increased drastically, and it is obvious that a larger number of devices is required for the pair-synapse method.

$$Area\ ratio = \frac{Area_{\ pair\ synapse\ method}}{Area_{\ weight\ shifter}}$$

We constructed the S-DNN to demonstrate proposed weight shifter, as shown in supplementary Fig. S1. As an application, rat's neural signals were recognized by the constructed S-DNN which has the weight shifter. Figure 3a exhibits parts of detected neural signals that were obtained in both fear and non-fear conditions. Detailed description of the neural signals is presented in the supplementary Figs. S2 and S3[27], and the process of constructed S-DNN is simply described in Fig. 3b. From typical S-DNN, the weight shifting process was added; the VMM was conducted with a shifted median of weight. After the VMM, shifted values were returned back ($-\sum 10x_i$) when x_i means input bias.

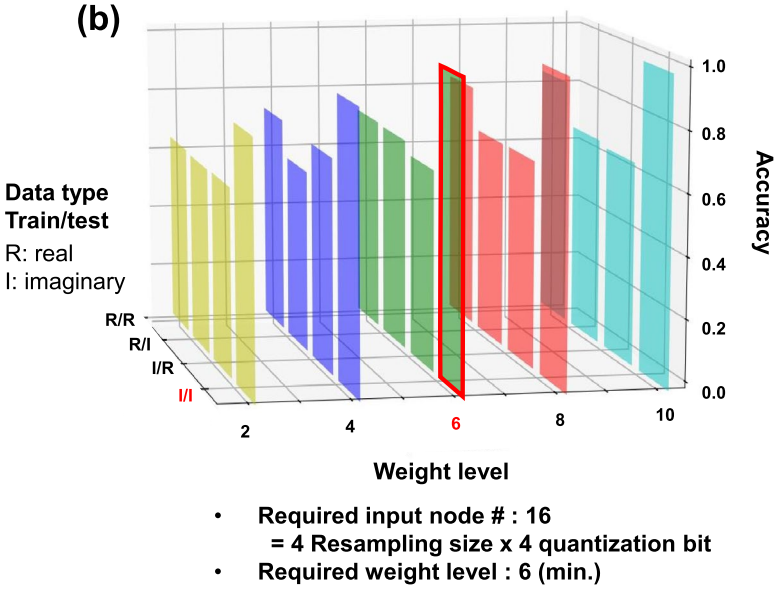
In other words, during both training and inference processes, the weight values remained positive because they were shifted as much as 10 towards the positive side. On completion of the VMM, the shifted values were returned back. All the original and shifted weight values utilized through the S-DNN are exhibited in Fig. 3c.

Moreover, the S-DNN was optimized to minimize the hardware resources for implementation of the weight shifter on a printed circuit board (PCB), as shown in Fig. 4. We compared various parameters such as the number of input nodes, number of weight levels, and types of data sets, besides the methods of input data preparation, viz., resampling size and quantization bit, details of which are explained in Figs. S3 and S4. The S-DNN was optimized for a high accuracy level (93.81%) with reduced hardware resources for the PCB level implementation, as

(a)

Input node	Accuracy (%)
8	83.829
16	93. 81
32	99.923
64	99.965
128	99.99

(b)



- Required input node # : 16
  = 4 Resampling size x 4 quantization bit
- Required weight level : 6 (min.)

Figure 4. Optimized conditions of the S-DNN. For implementation of the H-DNN, the S-DNN was optimized to minimize hardware sources. (**a**) Input node dependence of recognition accuracy. For more than 90% of accuracy, at least 16 input nodes are required. (**b**) Optimization of various parameters such as data type, weight level, and input data preparation method (resampling size and quantization bit). More details about optimization of the S-DNN is described in the supplementary figures (Figs. S2–S4). For high recognition accuracy with minimized hardware source, 16 input nodes (4 resampling size and 4 quantization bit), imaginary input data, and 6 weight levels were utilized.

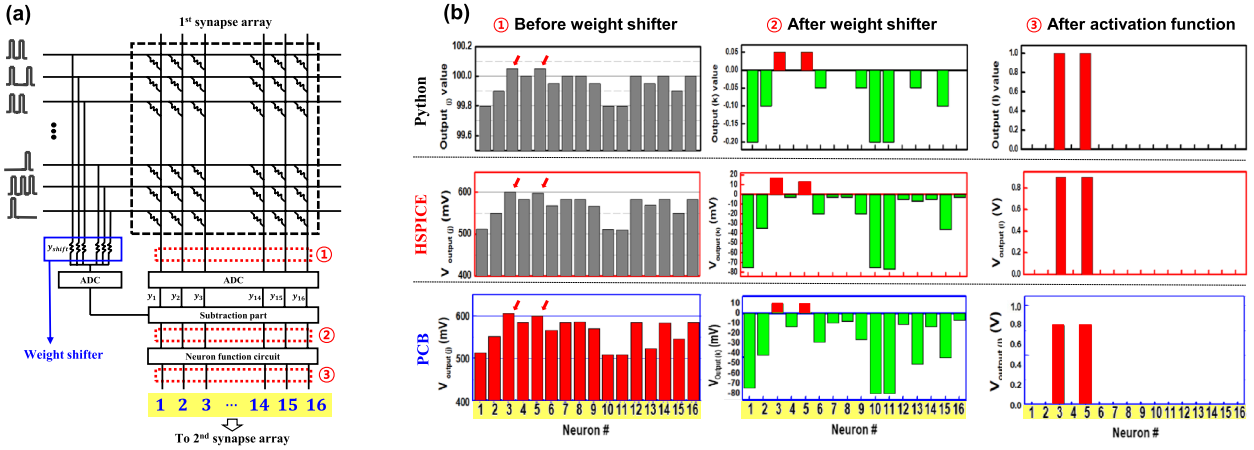


Figure 5. (**a**) Simplified schematic of realized H-DNN. To demonstrate the weight shifter, various output results (at 1 before weight shifter, 2 after weight shifter, and 3 after activation function) are compared by the S-DNN (Python), circuit simulator (HSPICE), and H-DNN (PCB). (**b**) The obtained output results were the same for all three cases: Python, HSPICE, and PCB.

shown in Fig. 4b (Figs. S2–S4). The higher number of input node showed higher recognition accuracy. However, for hardware implementation, minimum number of input node which can exhibit more than 90% accuracy was selected (Fig. 4a). There were also various dependence of input data type, resampling size, and quantization bit on the accuracy, as shown in Figs. S2–S4. To effectively minimize the hardware resources, we considered all above parameters (number of input node, input data type, resampling size, and quantization bit).

Based on the optimized condition of Fig. 4, the weight shifter was evaluated and compared in three respects: S-DNN, circuit simulator, and PCB circuit (Figs. 5 and S5). The hardware implemented circuit having weight shifter is composed of input layer, 1st synapse arrays (16 × 16), weight shifter, neuron parts (analog to digital circuit (ADC), subtraction part, and activation function), 2nd synapse array (16 × 2), and output layer, as shown in Figs. 5a and S5.

As utilized bias, 16 input voltages which are prepared from rat's neural signals are applied to the input layer (Fig. S2), and they are multiplied with conductance of the 1st synapse array. The results of multiplication: output currents (I_{array}) are converted to voltages (V_{array}), and the V_{shift} is subtracted from the V_{array} through the ADC and subtraction part. It implies the returning shifted weight back.

Consequently, the V_{output} (output voltage of each column) is recognized at the activation function part which is constructed as a Rectified Linear Unit function (ReLU) circuit by comparator and integrator (Fig. S5c). When the V_{output} exceeds a threshold voltage, the V_{output} is applied to the next synapse array. In the other case, output voltage of the activation function part is suppressed to zero voltage. After these 1st synapse array and neuron parts, output voltages of the activation function part is applied to the 2nd synapse array (16×2).

In the same manner, applied voltages are multiplied with conductance of 2nd synapse array, then two output currents of the 2nd synapse array are converted to output voltages by the R_{Load}. These output voltages are compared each other at the end of neural network; one represents the fear condition and the other represents the non-fear condition.

Even though whole H-DNN was successfully implemented (Fig. S5d), we detected the output voltages at near the weight shifter to confirm whether the weight shifter can work well through the VMM. To evaluate the weight shifter, for all cases (S-DNN, HSPICE, and PCB), we compared the output values at three points: ① before weight shifter, ② after weight shifter, and ③ after activation function. As described in Fig. S1, 16×16 synapse-device array was utilized to connect the input layer and hidden layer, which leads to 16 output values at the three points (highlighted 16 results of Fig. 5b).

Before the weight shifter, output values showed all positive values for all cases because the median of weight values was shifted to positive values. Then, through the weight shifter, the median of weight values was returned back to their original value, which led to the negative output values. Finally, after the activation function (designated as the ReLU), only the positive output values (number 3 and 5) remained. At the three points (①–③), all cases such as the S-DNN, HSPICE, and PCB exhibited the same results. This conclusively demonstrates effective transfer of the weight shifter function from software to hardware levels. Complete details about the hardware implementation are described in Fig. S5.

Conclusion

Hardware implementation of the negative weight was realized in a simpler way by the proposed weight shifter. In comparison to the conventional pair-synapse method, the weight shifter minimized various hardware resources such as additional requirement of devices and power consumption, which resulted in less area-requirement and simplicity of circuits. To demonstrate the weight shifter, rat's neural signals were recognized by the S-DNN, HSPICE, and PCB level H-DNN. During the recognition, employed weight values were positively shifted to be directly expressed by the conductance of the synapse device. For all cases (S-DNN, HSPICE, and PCB), the same output results were observed; this conclusively demonstrates that the proposed weight shifter can more effectively implement the negative weight with less hardware resources.

Methods

To evaluate the proposed weight shifter, the S-DNN was constructed with 16 input nodes, one hidden layer, and 2–10 weight levels by using Python (Fig. S1). The employed weight values were positively shifted to confirm the weight shifter in the S-DNN. Through the constructed S-DNN, we recognized a rat's neural signals in fear or non-fear conditions, as shown in Fig. S2. For the hardware implementation of the weight shifter, we optimized the S-DNN to reduce required hardware resources (Figs. S3, S4). Following this, the optimized S-DNN including weight shifter was realized as a circuit through a circuit simulator: HSPICE. For the weight shifter, 6.25 kΩ fixed resistors are employed as R1–R16. To express the conductance derived from the weight map of S-DNN (Fig. S5b), resistors (3.2–100 kΩ) are utilized. In addition, op-amp (LM 338) was used to construct the ADC, activation function, and comparator. Finally, the developed circuit composed of the weight shifter, ADC, subtraction part, and transimpedance amplifier was implemented on the PCB (Fig. S5).

Received: 8 June 2021; Accepted: 28 October 2021
Published online: 01 December 2021

References

1. Luo, Q. *et al.* Self-rectifying and forming-free resistive-switching device for embedded memory application. *IEEE Electron. Dev. Lett.* **39**, 664–667 (2018).
2. Kim, M. *et al.* Energy-storing hybrid 3d vertical memory structure. *IEEE Electron. Dev. Lett.* **40**, 1622–1625 (2019).
3. Lee, J.-E., Lee, C., Kim, D.-W., Lee, D. & Seo, Y.-H. An on-chip learning method for neuromorphic systems based on non-ideal synapse devices. *Electronics* **9**, 1946 (2020).
4. Tang, J. et al. Ecram as scalable synaptic cell for high-speed, low-power neuromorphic computing. In *2018 IEEE International Electron Devices Meeting* (IEDM), 13–1 (IEEE, 2018).
5. Lee, J., Lim, S., Kwak, M., Song, J. & Hwang, H. Understanding of proton induced synaptic behaviors in three-terminal synapse device for neuromorphic systems. *Nanotechnology* **30**, 255202 (2019).
6. Merolla, P. A. *et al.* A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).
7. Lee, D., Moon, K., Park, J., Park, S., Hwang, H. Trade-off between number of conductance states and variability of conductance change in pr0. 7ca0. 3mno3-based synapse device. *Appl. Phys. Lett.* **106**, 113701 (2015).
8. Burr, G. W. *et al.* Neuromorphic computing using non-volatile memory. *Adv. Phys. X* **2**, 89–124 (2017).
9. Yu, S. Neuro-inspired computing with emerging nonvolatile memory. *Proc. IEEE* **106**, 260–285 (2018).

10. Truong, S. N., Min, K.-S. New memristor-based crossbar array architecture with 50% area reduction and 48% power saving for matrix–vector multiplication of analog neuromorphic computing. *JSTS* **14**, 356–363 (2014).
11. Hu, M., Li, H., Wu, Q., Rose, G. S., Chen, Y. Memristor crossbar based hardware realization of bsb recall function. in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 1–7 (IEEE, 2012).
12. Burr, G. W. *et al.* Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Trans. Electron. Dev.* **62**, 3498–3507 (2015).
13. Kwak, M., Park, J., Woo, J. & Hwang, H. Implementation of convolutional kernel function using 3-d tio x resistive switching devices for image processing. *IEEE Trans. Electron Dev.* **65**, 4716–4718 (2018).
14. Park, Y. J. *et al.* 3-d stacked synapse array based on charge-trap flash memory for implementation of deep neural networks. *IEEE Trans. Electron Dev.* **66**, 420–427 (2018).
15. Hu, M., Li, H., Wu, Q., Rose, G. S. Hardware realization of bsb recall function using memristor crossbar arrays. in: *DAC Design Automation Conference 2012*, 498–503 (IEEE, 2012).
16. Kim, M. et al. Multinary data processing based on nonlinear synaptic devices. *J. Electron. Mater.* 1–7 (2021).
17. Yeo, I., Chu, M., Gi, S.-G., Hwang, H. & Lee, B.-G. Stuck-at-fault tolerant schemes for memristor crossbar array-based neural networks. *IEEE Trans. Electron Dev.* **66**, 2937–2945 (2019).
18. Lee, C. *et al.* Li memristor-based mosfet synapse for linear i–v characteristic and processing analog input neuromorphic system. *Jpn. J. Appl. Phys.* **60**, 024003 (2021).
19. Choi, Y. *et al.* Structural engineering of li-based electronic synapse for high reliability. *IEEE Electron. Dev. Lett.* **40**, 1992–1995 (2019).
20. Lee, C. *et al.* Two-terminal structured synaptic device using ionic electrochemical reaction mechanism for neuromorphic system. *IEEE Electron. Dev. Lett.* **40**, 546–549 (2019).
21. Choi, W. *et al.* Wo x-based synapse device with excellent conductance uniformity for hardware neural networks. *IEEE Trans. Nanotechnol.* **19**, 594–600 (2020).
22. Choi, W. *et al.* Impact of operating temperature on pattern recognition accuracy of resistive array-based hardware neural networks. *IEEE Electron. Dev. Lett.* **42**, 763–766 (2021).
23. Yeon, H. *et al.* Alloying conducting channels for reliable neuromorphic computing. *Nat. Nanotechnol.* **15**, 574–579 (2020).
24. Choi, H.-S., Park, Y. J., Lee, J.-H. & Kim, Y. 3-d synapse array architecture based on charge-trap flash memory for neuromorphic application. *Electronics* **9**, 57 (2020).
25. Ielmini, D. & Ambrogio, S. Emerging neuromorphic devices. *Nanotechnology* **31**, 092001 (2019).
26. Kim, S., Lee, H.-m., Gokmen, T., Han, S.-J. In-cell differential read-out circuitry for reading signed weight values in resistive processing unit architecture (2019). US Patent 10,340,002.
27. Lee, J. H., Lee, S. & Kim, J.-H. Amygdala circuits for fear memory: A key role for dopamine regulation. *Neuroscientist* **23**, 542–553 (2017).

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MSIP) (NRF-2020R1C1C1005925), by Nano-Material Technology Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (2009-0082580), by the Research Grant of Kwangwoon University in 2020, and by the excellent researcher support project of Kwangwoon University in 2021.

Author contributions

D.S.L. conceived and the directed the research. Y.H. conceived experiments. C.J. and J.L. conducted the experiment(s). G.H., J.S., M.K. and Y.B. analysed the results. G.H. wrote the manuscript. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41598-021-02176-4.

Correspondence and requests for materials should be addressed to D.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.