NOLTA, IEICE

Paper

Heuristic model for configurable polymer wire synaptic devices

Yoshiki Amemiya[1a)], *Ali Emiliano Jose*[1], *Naruki Hagiwara*[1], *Megumi Akai-Kasaya*[2,3], and *Tetsuya Asai*[2]

[1] *Graduate School of Information Science and Technology, Hokkaido University Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan*

[2] *Faculty of Information Science and Technology, Hokkaido University Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan*

[3] *Department of Chemistry, Graduate School of Science, Osaka University 1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan*

[a)] *amemiya.yoshiki.du@ist.hokudai.ac.jp*

Received October 18, 2021; Revised December 18, 2021; Published April 1, 2022

Abstract: Recently, there has been considerable research on nonvolatile analog devices for artificial intelligence (AI); however, it focuses on all-coupled neural networks. In contrast, polymer wire-type synaptic devices, which can be expected to be arbitrarily wired similar to a biological neural network, have already been proposed and demonstrated. In this study, we model a polymer wire synaptic device based on the results of previous research, and demonstrate an example of applying simple perceptron (AI) to the model. The results of our study show that it is possible to predict effective methods of using polymer wire synaptic elements in AI.

Key Words: analog ai, nonvolatile analog devices, organic polymers, three-dimensional ai devices

1. Introduction

We are at the dawn of the Internet of things (IoT) era, with various household items becoming Internet-connected. In the widespread use of IoT, it is necessary to process data on edge machines to reduce the communication load with cloud artificial intelligence (AI), process data, and make decisions only on edge machines when rapid responses are required. To meet these objectives, there is a growing demand for edge AI computing. Research has focused on dedicated AI hardware with lower power consumption, smaller size, and faster data processing because the power and space available for edge machines are limited.

Recently, two-dimensional (2D) crossbar models have been extensively studied for AI-specific hardware. These models have a planar structure with many synaptic elements that can hold the resistance value, changed by the application of voltage, for a long time[1]. Such a model can be considered an all-coupled neural network with synaptic elements as weights in the neural network. However, when implementing neighborhood joins, such as convolution, in a 2D crossbar model, preparing all the joins in advance results in waste [2]. The field of neuromorphology, which closely mimics neural structures, has the potential to solve this problem.[3]

2. Concept of Brain-type Devices

Conductive bridging RAM (CBRAM) is a synaptic device that has recently received significant attention [4] [5]. When a voltage is applied between the electrodes, the CBRAM generates conductive filaments in the insulating layer, as shown in Fig. 1(a). Inspired by this phenomenon, we sought to control the location of conductive filament generation by passing the current from point to point, as shown in Fig. 1(b), rather than from surface to surface. If we regard conductive filaments as wires in CBRAM, we can realize arbitrary wiring. We also devised a brain-type device that mimics the structure of an actual brain by integrating CBRAMs in three dimensions, as shown in Fig. 1(c). In an actual neuron, the signals that are received by dendrites travel down axons, and are transmitted to the next neuron, with the efficiency of transmission controlled by synapses. In contrast, in brain-type devices, the wiring of conductive filaments is a weighted conduction path, corresponding to the dendrites, axons, and synapses of a neuron. In other words, this synaptic device mimics the structure of a nerve more closely. For example, we have already reported a synaptic device made of conductive polymer wires that was grown via electropolymerization in a solution (polymer wire synaptic device) [6] [7].

This study aims to predict the problems that occur when learning AI with polymer wire synaptic devices. To achieve this, polymer wire synaptic devices need to be modeled and simulated, and then, applied to AI. In this study, a 2D simulator for a polymer wire synaptic device and implemented AI was created. In the future, we will create a three-dimensional (3D) simulator based on this 2D simulator.

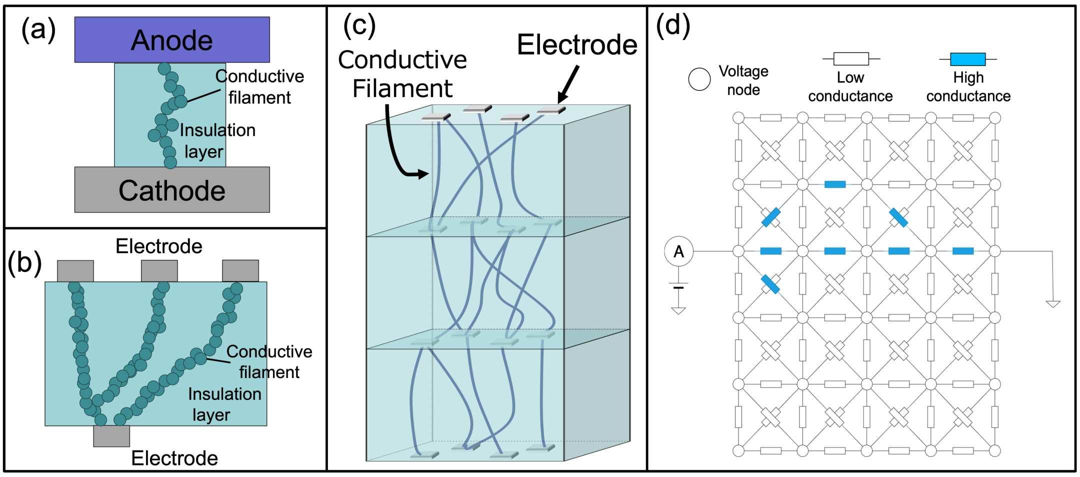

Fig. 1. (a) Schematic of a typical CBRAM. (b) Schematic of the proposed CBRAM. (c) Brain-type device. (d) Diagram of our simulator.

3. Development of Simulation Model of PEDOT:PSS Wire Growth

3.1 Basic principle

A CBRAM simulator based on the study "Evolving conductive polymer neural networks on wetware," [6] was created. In that study, a conductive polymer polystyrene sulfonate (PSS) wire was grown via electropolymerization by using a 3,4-ethylenedioxythiophene (EDOT) monomer solution, with PSS as a dopant, as a precursor solution, and by applying a square-wave AC potential to an electrode immersed in the solution. This CBRAM can be quickly reset and reconstituted by washing it away.

When a square-wave AC voltage is applied to this CBRAM, a conductive polymer wire grows from the anode along the potential gradient, generated between the electrodes, when the AC voltage is switched. To incorporate these features into the simulation, we simplify them into a form of "transforming the place where the most current flowed in the quantized solution space into a conductive polymer wire." Fig. 1(d) shows the overall view of the simulator. In this simulator, the real space is quantized, potential nodes are set up, and resistors are placed between the potential points. High resistance resistors correspond to the solutions, while low resistance resistors correspond to the conductive polymers. This network circuit, comprising potential nodes and resistors, corresponds to one CBRAM. The conductive polymer wires were grown using the following procedure: ① applying voltage and grounding to the potential nodes corresponding to the electrodes; ② calculating the potential of each potential node using the nodal method of the network circuit; ③ calculating the current value flowing through each resistor; and ④ changing the resistance of one resistor with the maximum current value (from the calculated current value) from high to low resistance. The growth of the conductive polymer wire was represented by repeating the calculation of each current value and updating the resistance at the maximum current location.

3.2 Setting various values in the simulation

To make the simulator resemble the actual PEDOT:PSS wire growth, we chose two different values. The first value is the dispersion value given to the resistor corresponding to the solution. The principle of "transforming the place where the most current flowed in the quantized solution space into a conductive polymer wire" was insufficient for selecting the diagonal resistor as the maximum current location. This is because this simulator does not perform the shortest path search when updating from high to low resistance. Therefore, the resistance value has a random variance such that the diagonal direction is chosen as the maximum current point. To select the optimal variance, simulations were performed, wherein the ratio of high resistance to low resistance was set to 100:1 in a region with potential nodes of $vertical : horizontal = 90 : 65$, and the resistance was updated 500 times using different variances. Fig. 2 shows a gray-scale representation of the resistance of the simulated area. The normalization of shading is based on the upper limit of the variance for high resistance values. The whiter the color, the higher the resistance value (solution); the blacker the color, the lower the resistance value (conductive polymer or electrode element). The black blocks at both ends of the figure are pre-set electrode elements. Fig. 2(a) shows the results when variance ±10 % is given. Figs. 2(b), (c), and (d) show variances of ±20 %, ±30 %, and ±40 %, respectively. These figures show that the shade of white in the solution region changes by changing the dispersion of the resistance value. From these results, we chose ±30 %, which provides a dispersion close to the spatial distribution of the real PEDOT:PSS wires.

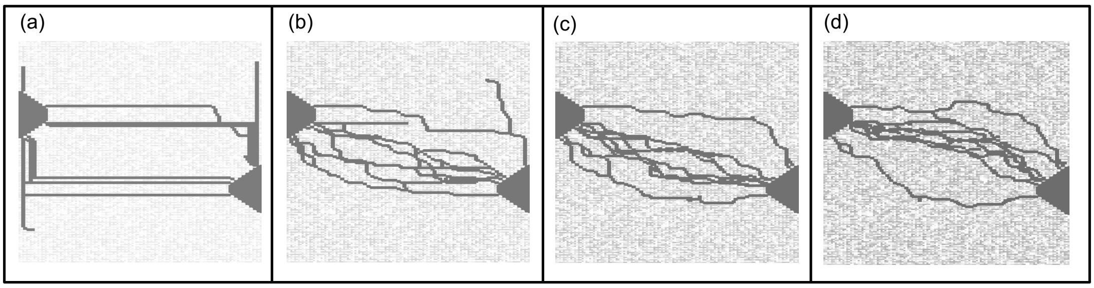

Fig. 2. Simulation results with different variances: (a) ±10 %; (b) ±20 %; (c) ±30 %; (d) ±40 %.

The second value is the ratio of high resistance to low resistance. This resistance ratio changes the conductive polymer wire growth. To select the optimal resistance ratio, we set the variance to ±30 %, and updated the resistance value 500 times with different resistance ratios. Fig. 3(a) shows a high:low resistance ratio of 10:1. Figs. 3(b), (c), and (d) depict ratios of 100:1, 1000:1, and 10000:1, respectively. Based on these results, we selected a resistance ratio of 100:1, which provides a distribution close to the spatial distribution of the actual PEDOT:PSS wires.

The other setting values listed below are tentative because they do not affect the overall shape of the simulation. First, we explain the value of the high resistors when measuring AI weights. The conductance value from the applied node to the ground node was used as the AI weight value. This

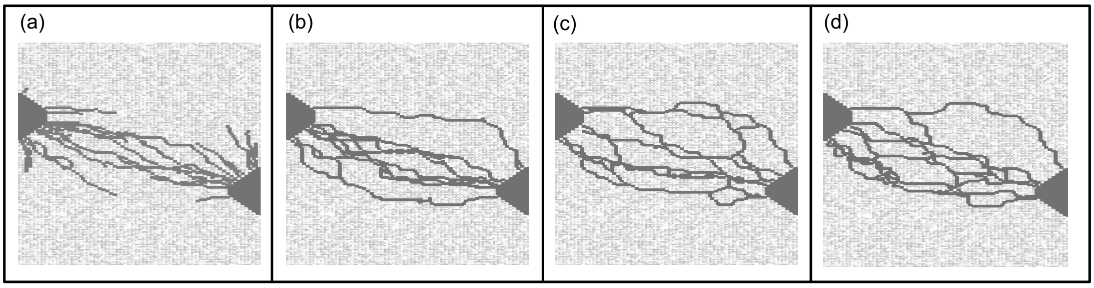

Fig. 3. Simulation results with different resistance ratios: (a) 10:1; (b) 100:1; (c) 1000:1; (d) Resistance ratio 10000:1.

allows us to express that the weights increase as the conductive polymer wire grows. However, when the resistance ratio was set to 100:1 for the weight measurement, the variation in weight values owing to wire growth was excessively small, and the initial placement of the electrodes had a significant effect on the size of the weight values. To address this problem, we set the value of the high resistors to 10000 Ω when measuring the weights for learning; consequently, the variation of the weight values owing to wire growth became large, allowing learning to take place.

Next, we explain the rounding of digits for the weight values in AI learning. The weight values change, albeit minutely, with each polymer update, even if the conductive polymer wire is not cross-linked. We round minute values to stabilize the size relationship between the weights. In the simple perceptron using the simulator, rounding the AI learning weight values to the fourth decimal place further stabilized the learning.

Because it is evident that the resistance value of the electrode part resistor is negligible compared to that of the solution and conductive polymer, it was set to 0.0001 Ω. The variance of the resistance between the polymer and electrode resistors was set to ±10 %; further, a supply voltage of 10 V along with an input and output resistance of 10 Ω were set.

4. Implementing AI (Neural Networks) in the Simulator

We now specify how the electrodes are set up, the degree to which conductive polymer wires are grown in one epoch, how the weight values are handled, and how the weights are calculated for the simulation of learning. First, we explain how to set up the electrodes. To simplify the structure, the electrodes were placed on the left and right sides of the simulator, with the left side as the working electrode and the right side as the counter electrode. Next, we discuss the degree to which the conductive polymer wire grows in one epoch. When updating weights in AI learning, although we can increase a certain weight w in this simulator, we cannot increase it by a specific value such as Δw. This is because the simulator is a complex network circuit, and we cannot estimate the change when we change the value of a single resistor. Therefore, we set the growth of the conductive polymer wire to be performed in one epoch, once for each weight. Next, we explain how to handle the weight values for AI learning. As mentioned in the previous section, because the conductance value from the applied node to the ground node is used for the weight value, this value can only increase as the wire grows. However, the AI learning weights repeatedly increase or decrease until training is completed. To reflect this, we prepared positive (w^+) and negative (w^-) values, and expressed one weight w as $w = w^+ - w^-$. Finally, we explain the method for calculating the weights for AI learning. We used OpenBLAS to calculate the nodal method for the networked circuit. In the execution of the program, we found that it takes the considerable amount of time to calculate the values of all potential nodes of the network circuit using the nodal method. Therefore, the weight calculation method was modified to reduce the number of calculations for the nodal method. That is, instead of determining the conductance values between the electrodes of a working electrode and counter electrode, all the working electrodes and one counter electrode were used to calculate the potential. The current flowing into each working electrode, which was calculated using each potential and input resistance, was used to calculate the conductance value between each electrode. This made it possible to obtain multiple conductance values for a single simulator using a single-nodal method calculation.

4.1 Simple perceptron

We used the simple perceptron [8] as an AI implementation. The teacher data were an AND function with two inputs. The bias term was set to -1. A sign function was used as the activation function. When the two inputs are X_1 and X_2, the weights of each are w_1 and w_2, respectively, and the weight of the bias term is w_0. The output function to obtain output Y is expressed as follows:

$$y' = (w_1^+ - w_1^-)X_1 + (w_2^+ - w_2^-)X_2 - (w_0^+ - w_0^-) \tag{1}$$

$$Y = \begin{cases} 1(y' > 0) \\ 0(y' \leq 0) \end{cases} \tag{2}$$

For the learning method, we used batch learning. Because it is a two-input AND function, the number of teacher data points is four. Because the activation function is a sign function, the amount of weight update Δw is expressed as follows:

$$\Delta w = \sum_{k=0}^{4} (TrainingData_k - Output_k) * (Y) \tag{3}$$

When Δw is positive, w^+ increases; when Δw is negative, w^- increases to the electrode corresponding to its unsigned weight value, and conductive polymer wire growth is performed once. Learning was completed when all inference results matched the teacher data.

The results of learning the AND function with a simple perceptron using the simulator are shown in Fig. 4. Here, white represents the solution, and black represents the conductive polymer or electrode. The scale of the simulator is *height* 190 × *width* 110. The blocks on the left side are the working poles, corresponding to W_1^+, W_1^-, W_2^+, W_2^-, W_0^+, and W_0^- from the top. The block on the right-hand side is the counter electrode. The learning process was completed in 323 epochs. The weighted values are $w_0^+ = 0.007$, $w_0^- = 0.005$, $w_1^+ = 0.005$, $w_1^- = 0.003$, $w_2^+ = 0.010$, and $w_2^- = 0.008$. It was demonstrated that the learning of the AND function with two inputs was complete; hence, we can conclude that AI learning implementing a simple perceptron was successful.

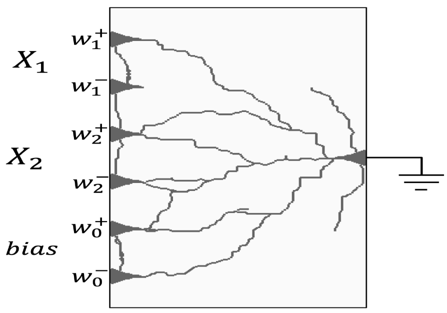

Fig. 4. Results of AND function learning with a simple perceptron using the simulator.

5. Considerations

5.1 Differences between PEDOT:PSS wire growth and simulator

Although we have imitated the PEDOT:PSS wire growth, the simulation of the growth process is still problematic. PEDOT:PSS wires are solids in a liquid, and behave as if they are attracted by electric charges. Additionally, the PEDOT:PSS wires shrank, and before cross-linking occurs, the force of attraction between the charges stretches the shrunken part, resulting in instant cross-linking. Because the simulation does not reproduce the movement of the wire, owing to this attraction and stretching of the frizz during cross-linking, the simulation cannot foresee the problems of the PEDOT:PSS wire. Furthermore, the simulation cannot foresee the problem of the wires becoming tangled owing to the attraction of the PEDOT:PSS wires when they are made 3D.

5.2 Simulation using for learning

In the region where multiple electrodes were concentrated on the left side of Fig. 4 in the simple perceptron, multiple cross-links between the electrodes occurred. This is because the resistance of the electrodes is lower, and coupling to an already cross-linked electrode can be a shortcut. However, the cross-linking between electrodes is not a problem for AI learning. Because the weight values were measured between the working and counter electrodes, the weight values, including the effect of cross-linking, were only calculated and did not affect the AI learning.

In some trials, it took thrice as many epochs as the learning epoch, as shown in Fig. 4. In such cases, the overall percentage of low resistance increased. In the case of complex AI learning, the percentage of low resistance is expected to increase even more, which may lead to the limit of the differential. To handle this, we need to place the targeted conductive polymer back into the solution, desorb the targeted conductive polymer to reduce the conductance value, or reset everything when the percentage of conductive polymers in the solution reaches a certain level.

6. Conclusions

In this study, we show the creation of a simulator based on PEDOT:PSS wires and the possibility of AI learning using a simulator. We modeled and simulated the growth of a PEDOT:PSS wire as a simple process for transforming the most current flowing place in the quantized solution space into a conductive polymer wire. We also implemented simple AI learning using this simulator. However, we determined that some phenomena cannot be reproduced by this simulator; further, issues related to the limitation of the differential in weights became apparent. In future research, it will be necessary to discuss how to handle such problems in PEDOT:PSS wires. Additionally, we intend to create a 3D simulator based on the 2D simulator, and predict the problems that may occur when the PEDOT:PSS wire is structured in 3D.

Acknowledgments

This work was supported in part by a Grant-in-Aid for Scientific Research (21H03506, PI: Tetsuya Asai).

References

[1] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015.

[2] S. Mittal, "A survey of ReRAM-based architectures for processing-in-memory and neural networks," *AI learning and knowledge extraction*, vol. 1, no. 1, pp. 75–114, August 2019.

[3] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629-1636, October. 1990.

[4] Q. Xia, and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature materials*, vol. 18, no. 4, pp. 309-323, March 2019.

[5] I. Valov, R. Waser, J. R. Jameson, and M. N. Kozicki, "Electrochemical metallization memories—fundamentals, applications, prospects," *Nanotechnology*, vol. 22, no. 25, pages. 254003, May 2011.

[6] M. Akai-Kasaya, N. Hagiwara, W. Hikita, M. Okada, Y. Sugito, Y. Kuwahara, and T. Asai T, "Evolving conductive polymer neural networks on wetware," *Japanese journal of applied physics*, vol. 59, no. 5, pp. 060601(1)–(9), May 2020.

[7] N. Hagiwara, S. Sekizaki, Y. Kuwahara, T. Asai, and M. Akai-Kasaya, "Long- and short-term conductance control of artificial polymer wire synapses," *Polymers*, vol. 13, no. 2, pp. 312(1)–(10), January 2021.

[8] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, pp. 386–408, November 1958.