SCIENCE CHINA
Information Sciences

• Supplementary File •

A Bio-inspired Tactile-Olfactory Fusion Perception System Based on Memristive Spiking Neural Network

Chao Yang[1,2], Xiaoping Wang[1,3,4*], Zhanfei Chen[1,3,4], Jiang Li[1,3,4], Nan Qin[5], Tingwen Huang[6] & Zhigang ZENG[1,3,4]

[1]*School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan* 430074*, China;*
[2]*School of Computer Science and Technology, Guangxi University of Science and Technology, Liuzhou* 545006*, China;*
[3]*Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan* 430074*, China;*
[4]*Hubei Key Laboratory of Brain-inspired Intelligent Systems, Wuhan* 430074*, China;*
[5]*State Key Laboratory of Transducer Technology, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai,* 200050*, China;*
[6]*School of Computer Science and Control Engineering, Shenzhen University of Advanced Technology, Shenzhen* 518055*, China*

Appendix A Functionality of Humanoid Hand

To mimic the perception structure of the star-nosed mole's nose, a humanoid hand that integrates 14 force sensors on each fingertip and six gas sensors into the palm has been developed in the previous work [1]. Each force sensor has a footprint of 0.5×0.5 mm^2, which allows for high resolution. Its design considers the sensitive hexagonal silicon membrane and piezoresistive Wheatstone-bridge to achieve high sensitivity and accuracy. The membrane deforms when the sensors contact the test object, with the extent of deformation influenced by the stiffness of the object and the applied force. The deformation can lead to a resistance change in the piezoresistors, further causing a change in the output voltage of the Wheatstone-bridge. When the same amount of force is applied, the different output voltage of a Wheatstone-bridge reflects the stiffness and local topology of the test object. The sensitivity of the force sensor is 0.375 mV kPa^{-1} over a range of 0-400 kPa during a typical object interaction procedure, which enables the tactile sensing array to detect the minimum height difference of about 0.3 mm.

As for the olfactory perception, six gas sensors (3×3 mm^2 for each unit) are functionalized to be highly sensitive to ethanol, acetone, ammonia, carbon monoxide, hydrogen sulfide, and methane, respectively. The gas molecules can be absorbed by the gas-sensitive material inside the sensors and change its resistance, thus distinguishing the concentration and type of the detected gas by measuring the resistance change and analyzing the gas sensing array, similar to the biological olfactory receptors [2]. Notably, a gas sensor responds not only to one sensitive gas but also to other gases to a certain extent, which allows the olfactory sensing array to sense various gases depending on the degree of response. In addition, the sensors can rapidly respond, which can reach the steady phase in about 10 seconds during contact with the measured gas and return to the initial state within 10 seconds after the gas is cut off. A more detailed introduction about this humanoid hand can refer to the previously reported work [1].

Appendix B Memristor-based computing platform and the Computing diagram of SNN

The photograph of the customized memristor-based computing platform is shown in Figure B1(c). The platform mainly consists of a dual-core ARM Cortex-A9 processor equipped on XILINX's Zynq 7015 and two memristor-based processing elements (MPEs). The ARM processor schedules all calculations of SNN, such as data preparation, driver circuit configuration, analog-to-digital converters (ADCs) control, digital-to-analog converters (DACs) control, and neuron state updates (Figure B1(a)). The data preparation includes data quantization, data bit-wise unfolding, data rearrangement, and so on.

The computations of convolutional and fully connected layers are performed in the MPEs, which can realize parallel matrix-vector multiplication based on a memristor array. As demonstrated in Figure B1(b), the MPEs are integrated in a 2-transistor-2-memristor (2T2R) structure [3], which has the advantage of reducing IR drop. Each MPE consists of 1152×128 memristors. That is, 576×128 weights can be mapped on the array since a positive and a negative memristor represent one weight. Each memristor has eight stability levels. Therefore, the precision of the memristive weight is 4 bits. The driver circuit configuration involves configuring the word line (WL) driver, bit line (BL) driver, and ADC operating parameters of source line (SL) according to read/write operation, input bit, and selected memristor area. VDD or GND in the WL driver can be selected by the corresponding programmable multiplexer to open or close a memristor branch. The 16-bit DACs in the BL and WL drivers are utilized together to program memristors by the write-verify method [4].

* Corresponding author (email: wangxiaoping@hust.edu.cn)

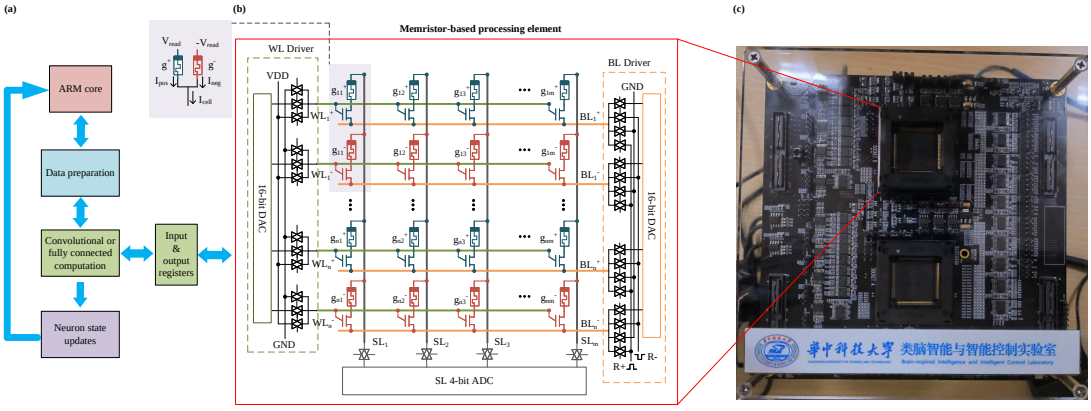


Figure B1 (a) Computing diagram of the SNN deployed on the memristor-based digital-analog hybrid system. (b) Schematic of the memristor-based processing element. (c) Photograph of the customized integrated printed circuit board system.

Positive (R^+, 0.1 V) or negative (R^-, -0.1 V) pulses are selected as the read voltage for calculation according to the input bit (only -1, 0, or 1) during the computing phase. As shown in the inset of (b), the conductance of g^+ and g^- can represent the positive weight and negative weight, respectively. If the input bit is 1, R^+ and R^- will be applied to g^+ and g^- via BL^+ and BL^-, respectively, then $I_{cell} = I_{pos} + I_{neg}$, where $I_{pos} = V_{read}g^+$ and $I_{neg} = -V_{read}g^-$, i.e., $I_{cell} = V_{read}(g^+ - g^-) = V_{read}W_{cell}$. If the input bit is -1, R^- and R^+ will be applied to BL^+ and BL^-, then $I_{cell} = -V_{read}W_{cell}$. A weight can be mapped as $(g^+ - g^-)$, which can be positive, negative, or zero and is proportional to the cell current. Eventually, the current of SL_j can be expressed as

$$\begin{aligned} I_{SL_j} &= V_{read}\sum_i x_i W_{cell}(i,j) \\ &= \mathbf{X}\mathbf{W_{cell}}(:,j), \end{aligned} \tag{B1}$$

where x_i is the i-th element of $\mathbf{X}$, serving as an input bit; $\mathbf{W_{cell}}(:,j)$ represents the mapped weights on SL_j. Likewise, the equation exists for other specified SLs. The 4-bit LPAR-ADCs [3] are used to convert the computing result carried on the currents of SLs into digital. Therefore, vector (only -1, 0, or 1) matrix multiplication operations can be computed in parallel on the MPEs with one step. The neuron state updates refer to the membrane potential update and spike firing control.

Appendix C Data preprocessing

To implement object recognition with the humanoid hand, a customized data set aiming at rescue task is built [1], which contains 100,000 samples covering five categories, i.e., human arm (main target), mouse (tactile interference), towel (olfactory interference), orange (soft object), and stone (rigid object). To facilitate the calculation and recognition on the MPE, the collected recordings need to be preprocessing. Figure C1 illustrates the data preprocessing at each stage when the humanoid hand grabs a human arm.

Firstly, a max-min normalization is applied to the force sensor recordings to mitigate data variability.

$$U_n = \frac{U_t - U_{min}}{U_{max} - U_{min}}, \tag{C1}$$

where U_{max} and U_{min} are the maximum and minimum output voltages of the force sensor, respectively; U_n is the normalized voltage of U_t. Since the measured resistances of the gas sensor array are positive values, a max-based normalization method is employed to process olfactory recordings for each gas sensor.

$$R_n = \frac{R_t}{R_{max}}, \tag{C2}$$

where R_{max} represents the max resistance of the gas sensor, and R_n is the normalized resistance of R_t.

Next, a random sampling recombination strategy is employed, which is conducive to containing more grasp gestures. Namely, each sample mapping is composed of N (here, N = 10) sampling points that are randomly resampled from the recording sequence when the sensors contact a detected object. Therefore, the size of the tactile sample mapping is 5 × 14 × 10 since there are 5 force arrays in total, and each force array consists of 14 sensors. The size of the olfactory sample mapping is 6 × 10 because six gas sensors are mounted.

Finally, the sample mappings are processed with a 4-bit quantization method to balance accuracy and efficiency. The formula is as follows.

$$x_i^q = \text{Round}(\frac{x_i \cdot C}{S}). \tag{C3}$$

Here, x_i^q is the quantization value of x_i, and x_i is the i-th value in the sample mapping $\mathbf{X}$. S represents the maximum value of the absolute value of all elements in $\mathbf{X}$. $C = 2^{N-1} - 1$, where $N = 4$ means that 4-bit quantization is performed. Round$(\cdot)$ means round the value to the nearest integer.

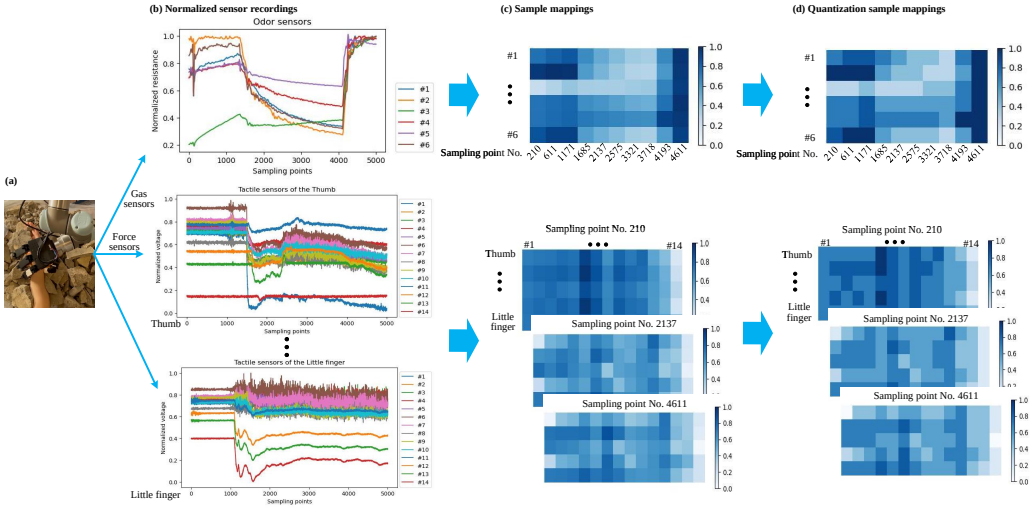


Figure C1 Data processing framework and samples at each stage. (a) Photograph of the humanoid hand grabbing a human arm. The gas and force sensors gather the odor and tactile information of the arm simultaneously. (b) Normalized odor and tactile sensor recordings. (c) Mappings of the olfactory and tactile samples. (d) 4-bit quantization of the sample mappings in (c).

Appendix D Integrate-and-fire neuron

The type of neurons used in this work is the Integrate-and-Fire (IF) model. Compared with the other models, it significantly simplifies the description of ion channel and membrane potential dynamics but preserves the key dynamic description of neurons. This math-ematically simpler model is conducive to reducing on-chip computing costs and being applied in resource-constrained scenarios. The IF neuron can be characterized with an iterative formula.

$$\begin{cases} \mathbf{H}_t^n = \mathbf{V}_{t-1}^n + \mathbf{X}(t), \\ \mathbf{S}_t^n = \mathrm{Hea}(\mathbf{H}_t^n - V_{th}), \\ \mathbf{V}_t^n = V_{reset}\mathbf{S}_t^n + \mathbf{H}_t^n \odot (1 - \mathbf{S}_t^n), \end{cases} \tag{D1}$$

where n and t mean the layer and time step, respectively; $\mathbf{H}_t^n$ represents the membrane potential produced by the previous temporal potential $\mathbf{V}_{t-1}^n$ and the current spatial input $\mathbf{X}(t)$; $\mathbf{S}_t^n$ is the output spike tensor governed by $\mathrm{Hea}(\cdot)$ that satisfies $\mathrm{Hea}(x) = 1$ when $x \geqslant 0$, otherwise $\mathrm{Hea}(x) = 0$; $\odot$ denotes the element-wise multiplication. In this work, V_{reset} and V_{th} take 0 and 1, respectively.

Appendix E Training of the SNN

In this work, an arc tangent function is utilized to approximate the spiking activation function and generate surrogate gradients [5]. To match the memristive weight precision, 4-bit quantization-aware training is performed. First, perform fake quantization on the SNN using the following formula.

$$W_{i,n}^q = \mathrm{Round}(\frac{W_{i,n}C}{S}) \cdot \frac{S}{C}, \tag{E1}$$

where $W_{i,n}$ denotes the i-th weights in the n-th layer; $C = 2^{N-1} - 1$ and $S = \max(\mathrm{abs}(\mathbf{W}_n))$, which are similar to Eq. (C3); N is 4 for specifying the signed quantization bits. Note that the derivative of the above formula is 0 everywhere. This work uses straight through estimator [6] to solve the problem of untrainability.

Then, add a certain intensity of random noise to the updated weights during training to improve the adaptation of the trained SNN to the variability of the on-chip memristor [7]. In this work, the noise is generated by the following equation.

$$noise = s(\max(\mathbf{W}) - \min(\mathbf{W}))N(0, 1), \tag{E2}$$

where $N(0, 1)$ represents a normal distribution with a mean of 0 and a variance of 1, which has the same shape as $\mathbf{W}$; s is a coefficient related to the weight mapping offset and takes 0.08 in this work, which means that the weight offset caused by variability is basically within 1.28 when the weight has 16 levels (4-bit). The 4-bit quantization-aware training of the SNN is described as in Algorithm E1.

Algorithm E1 4-bit quantization-aware training

Inputs: Training set ($\mathrm{X}_1,\ \mathrm{Y}_1$), test set ($\mathrm{X}_2,\ \mathrm{Y}_2$), SNN net
Outputs: Optimized SNN net^* with 4-bit quantization
$N = 4,\ C = 2^{N-1} - 1,\ s = 0.8,\ best\_acc = 0$
for $epoch = 1, 2, ..., N_{epoch}$ **do**
  Phase 1: Quantize the samples and SNN and add noise to the SNN (Line 6-10)
  $\mathrm{W},\ \mathrm{b} \leftarrow$ Obtain weights and biases of the net
  $S_x,\ S_w,\ S_b \leftarrow max(abs(\mathrm{X}_1)),\ max(abs(\mathrm{W})),\ max(abs(\mathrm{b}))$
  $\mathrm{X}_1,\ \mathrm{W},\ \mathrm{b} \leftarrow \mathrm{Round}(\frac{\mathrm{X}_1 C}{S_x}) \cdot \frac{S_x}{C},\ \mathrm{Round}(\frac{\mathrm{W}C}{S_w}) \cdot \frac{S_w}{C},\ \mathrm{Round}(\frac{\mathrm{b}C}{S_b}) \cdot \frac{S_b}{C}$
  Generate $noise_w$ and $noise_b$ for W and b with E2
  $\mathrm{W},\ \mathrm{b} \leftarrow \mathrm{W} + noise_w,\ \mathrm{b} + noise_b$
  Phase 2: Calculate SNN error and backpropagate it (Line 12-18)
  $out\_spikes = 0$
  for $t = 1, 2, ..., T_{step}$ **do**
    $out\_spikes \leftarrow out\_spikes + net(X_1)$
  end for
  $out\_spikes \leftarrow out\_spikes / T_{step}$
  $loss = loss\_function(out\_spikes, Y_1)$
  Backpropagate $loss$ in the net with surrogate gradient and straight through estimator method
  Phase 3: Test the net and save it when it reaches its optimal performance (Line 20-30)
  Perform **Phase 1** on the test set and the net.
  $out\_spikes = 0$
  for $t = 1, 2, ..., T_{step}$ **do**
    $out\_spikes \leftarrow out\_spikes + net(X_2)$
  end for
  $out\_spikes \leftarrow out\_spikes / T_{step}$
  $test\_acc = accuracy\_function(out\_spikes, Y_2)$
  if $test\_acc > best\_acc$ **then**
    $best\_acc = test\_acc$
    $net^* \leftarrow net$
  end if
end for
return net^*

The total 100,000 samples are divided into training and test sets according to 3:1. Considering the trade-off of accuracy and latency, the rate coding with 4 time steps ($T_{step} = 4$) is employed in this work. Adam is used to optimize the model with the mean squared error loss over 30 epochs. The SNN is built and trained based on SpikingJelly [8]. The training curves are shown in Figure E1. The model achieves the best performance at epoch 12, where the test accuracy is about 99.84%. As a comparison, the best test accuracy when using only tactile or olfactory information is 98.7% and 91.74%, respectively. Both are lower than that when tactile and olfactory information are used simultaneously. This means that integrating multi-modal information is conducive to recognition.

Appendix F Deployment of the SNN

The weights of each layer of the trained SNN are first converted to signed 4-bit integers using Eq. (C3). Next, the quantized weights are programmed as the conductance of the memristors through a write-verify method [4]. Figure F1(a) illustrates the process of the convolutional kernels mapping to the memristor array. Each channel of kernel i and each weight in each channel are mapped to the SL_i column in turn. All kernels of the convolutional layer are mapped sequentially in rows. To average the weight mapping offset [4], the kernels is repeated K times along the columns. The weights in the fully connected layer with O_N output neurons and I_N input neurons are mapped to the area with O_N rows and I_N cols. As shown in Figure F1(a) right, the quantized feature map needs to be rearranged in accordance with the mapping weights for parallel computation. Specifically, the input of the TConv1 and OConv1 should be unfolded into its absolute value 1 or -1 according to its sign, then the unfolded 1 or -1 is randomly filled in a sequence with length $2^{N-1} - 1$ to avoid possible current overload caused by applying all 1 or -1 to the BLs. Finally, the multiplication of the input vector $\mathbf{X}$ (signed N bits) and the weight $\mathbf{W}$ (represented by the memristors) can be obtained by accumulating the results read out by the ADCs of SLs over $2^{N-1} - 1$ times.

The mapped area and mapping offset are shown in Figure F1(b). The mapped areas of the convolutional layers are represented by $[K_N, C \times W_N \times K]$, where K_N, C, W_N, and K denote the kernel number, kernel channels, weight number

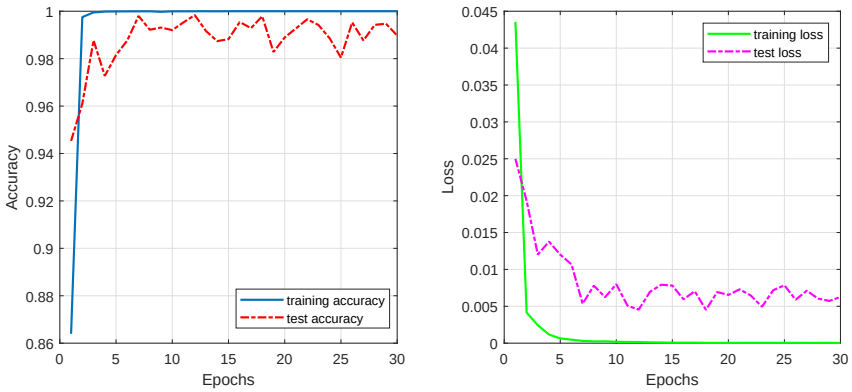


Figure E1 Training and test curves over 30 epochs. The best performance is achieved at epoch 12, where the training and test accuracy are 100% and 99.84%, respectively.

per channel, and repeated number, respectively. The mapped areas of the fully connected layers are defined by $[O_N, I_N \times K]$, where O_N and I_N mean the number of output neurons and the number of input neurons, respectively. K is the number of repeats. The mapping offset histogram of all weights is shown in Figure F1(c). The offset is within [-1, 0, 1] with a probability of about 99%.

Appendix G Layer-by-layer fitting and IF neuron remodeling

Since the calculation results of the MPE are converted from the currents of SLs via the LPAR-ADCs and the resolution of the ADC is related to the integration time [3], there is an unavoidable gap between the standard results (computed by CPU) and the converted results. This gap will directly affect the activation of the next IF neurons. Here, we assume that there is a linear relationship between the two results and adopt a layer-by-layer fitting strategy to minimize the gap.

First, N (50) samples are randomly selected from each category to serve as the network input. Next, the calculations for TConv1 and OConv1 are performed on a computer, and the same operations are conducted on the MPE using different ADC integration times. Then, we fit the linear relationship and calculate the correlation between the above two results under different integration times. The integration time can be configured as t_n times the sampling clocks, where $t_n = 1, 2, \cdots, 10$, and one sampling clock is 100 ns. Afterwards, the t_n and fitting coefficients that maximize the correlation and uniformize the result distribution are selected to map the converted result as an approximation of the standard result (fitting result) via the following formula.

$$\hat{V}_{cpu} = aV_{mem} + b, \tag{G1}$$

where a and b are the fitting coefficients, V_{mem} and $\hat{V}_{cpu}$ are the result computed by the MPE and fitting result, respectively. Finally, the fitting result is input to the next layer to continue fitting.

Figure G1 illustrates an example of how to select the number of sampling clocks and the fitting coefficients. As $t_n = 1$, the correlation 0.9208 is relatively minimal among the three, and only nine discrete levels are used due to the small integration time. While $t_n = 10$, there are too many results falling in the smallest level, which means that the converted results are generally smaller. Based on the correlation and uniformity of the converted results, $t_n = 5$ and the fitting coefficients $a = 1.5980$, $b = -0.2593$ are selected.

Table G1 lists the number of sampling clocks (t_n), the results correlation (cor), and the fitting coefficients of each layer in the SNN.

Table G1 Correlation and fitting parameters of each layer

Layers	cor	t_n	a	b
TConv1	0.8591	2	0.5567	0.6374
OConv1	0.9854	6	0.1526	-0.0537
TConv2	0.9577	5	1.5980	-0.2593
OConv2	0.9785	3	1.4092	-0.2157
FC1	0.9122	6	1.9167	-1.1182
FC2	0.8633	1	6.4949	-0.9937

Analyzing the IF neuron model (Eq. D1), it is feasible to integrate the fitting coefficients into the model to avoid fitting calculations during inference. Before the neurons fire spikes, $\mathbf{H}_t^n$ can be rewritten as

$$\mathbf{H}_t^n = \mathbf{V}_0^n + \sum_{\tau=1}^{t} \mathbf{X}(\tau), \tag{G2}$$

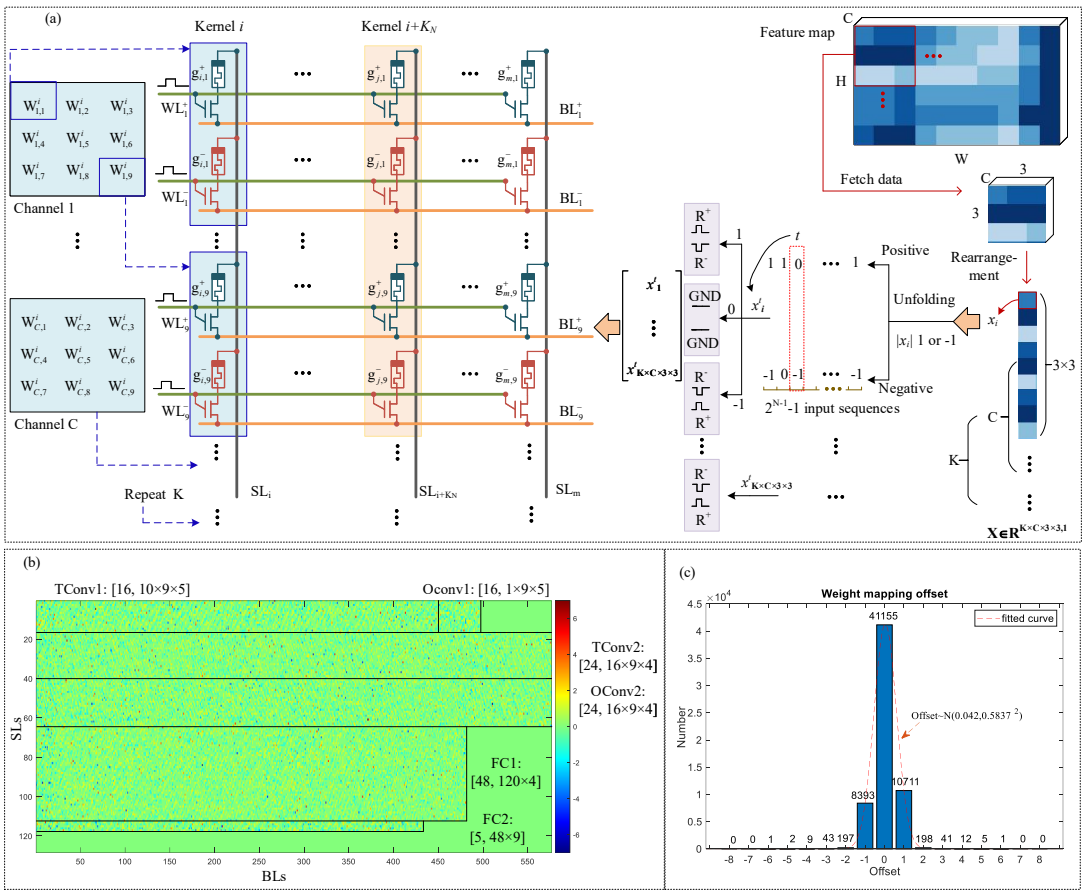


Figure F1 (a) Left: Schematic of convolutional layer weights mapping to the 2T2R cells. Right: Scheme showing a feature map how to be rearranged and a signed N-bit integer x_i how to be unfolded and input the memristor array in order. (b) Weight mapping area and offset. (c) Weight mapping offset histogram of all weights.

where $\mathbf{V}_0^n$ is the initial membrane potential. Then, the $\mathbf{S}_t^n$ can be expressed as

$$\mathbf{S}_t^n = \mathrm{Hea}(\sum_{\tau=1}^{t} \mathbf{X}(\tau) - (V_{th} - \mathbf{V}_0^n)). \tag{G3}$$

Considering $\mathbf{X}(\tau) = a\mathbf{X}_{mem}(\tau) + b$, $\mathbf{S}_t^n$ can be further expressed as

$$\begin{aligned} \mathbf{S}_t^n =& \mathrm{Hea}(\sum_{\tau=1}^{t} (a\mathbf{X}_{mem}(\tau) + b) - (V_{th} - \mathbf{V}_0^n)) \\ \equiv& \mathrm{Hea}(\sum_{\tau=1}^{t} \mathbf{X}_{mem}(\tau) - \frac{1}{a}[(V_{th} - \mathbf{V}_0^n) - b]). \end{aligned} \tag{G4}$$

Therefore, fitting calculations can be avoided by reconfiguring the firing thresholds of neurons in each layer. Considering that $\mathbf{V}_0^n = \mathbf{0}$ and $V_{th} = 1$ in this work, the firing thresholds of neurons in each layer can be recalculated via $(1-b)/a$ by refer to the fitting coefficients listed in Table G1.

Algorithm G1 illustrates the method to remodel the IF neuron with fitting coefficients.

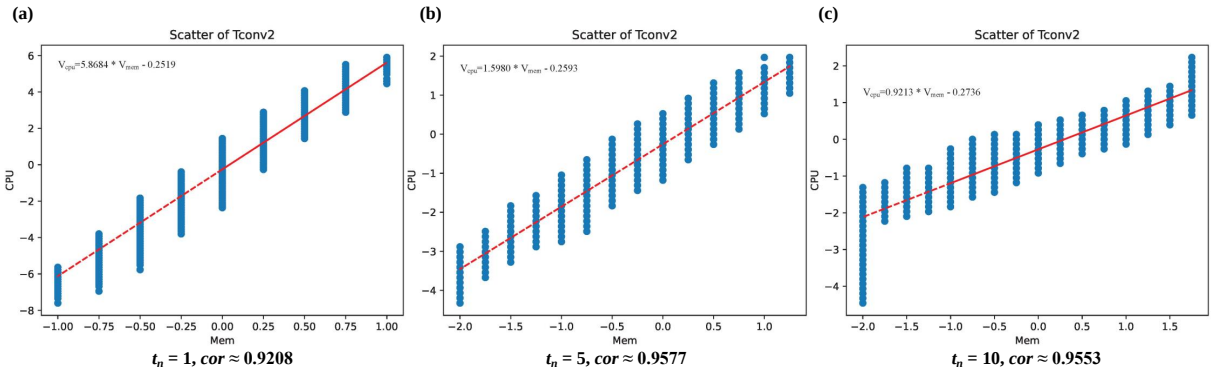


Figure G1 Scatter plot and fitted linear relationship between standard results and the results calculated by the MPE at different integration times. *cor* represents the correlation between the calculated results and the CPU-based results.

Algorithm G1 Remodeling IF neurons with fitting coefficients

Inputs: Fitting sample D, SNN net
Outputs: Remodeled SNN net_m deployed on the memristor-based computing platform
for $l = 1, 2, ..., L$ **do**
  $sub\_net^l \leftarrow$ Obtain the subnetwork of the net preceding the l-th layer neuron
  $neuron_m^l,\ sub\_net_m^l \leftarrow$ Obtain the IF neurons in the l-th layer from the net_m and the subnetwork preceding it
  X, Xm, $\leftarrow sub\_net^l(D),\ sub\_net_m^l(D)$
  Fit the linear relationship between X and Xm
  $a,\ b \leftarrow$ Select the fitting coefficients that maximize the correlation and uniformize the result distribution
  $V_0^l,\ V_{th}^l \leftarrow$ Obtain the initial membrane potential and the firing threshold from the $neuron_m^i$
  $V_{th}^l \leftarrow \frac{1}{a}[(V_{th}^l - V_0^l) - b]$
end for
return net_m

Appendix H Performance analysis

Generally, a larger time step will result in higher recognition accuracy due to higher information encoding accuracy, but the latency and computational cost will also be greater.

The computing latency is the sum of all layer computing latencies, while the computing latency of one layer is related to the unfolded bit of the input feature map. More unfolded bits require more array operations. The time to perform an array operation in the MPE mainly comprises the current sampling time and the ADC quantization output time, which can be estimated as $(0.1t_n + 0.01)$ μs, where t_n is the number of the ADC sampling clocks, 0.1 is the time of one sampling clock, and 0.01 is the ADC quantization output time. Assuming T_s is the time step, the time to complete a sample inference is

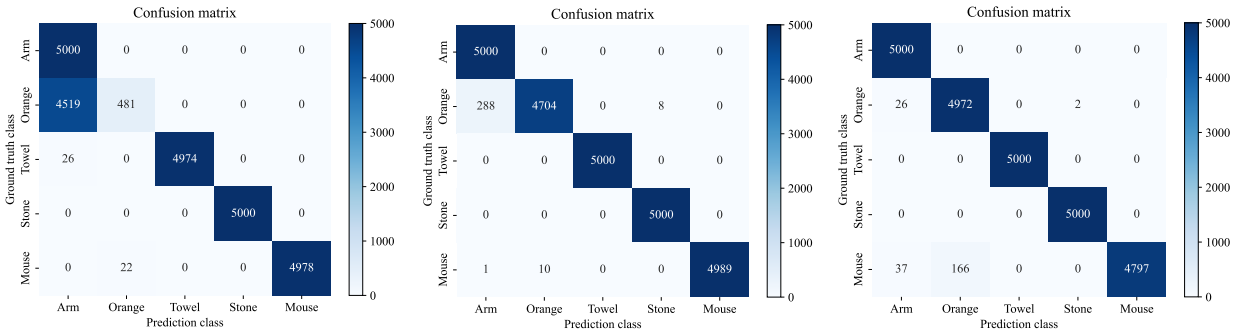
$$T = T_s \sum_{i=1}^{N} (0.1t_n(i) + 0.01)Nc(i), \tag{H1}$$

where $t_n(i)$ denotes the number of sampling clocks of the ADC used for i-th layer, which is listed in Table G1. $Nc(i)$ is the number of array operations of the i-th layer in a time step inference. There are 7 array operations required for TConv1 and OConv1 since their input data is unfolded into 7 input sequences, while other layers only need 1 array operation based on spike tensors.

Therefore, the estimated computing latency for 4 time steps is 29.12 μs. As a comparison, Figure H1(a)-H1(c) shows the recognition confusion matrix, recognition accuracy, and computing latency for time steps 2, 6, and 8.

On-chip power consumption is estimated by measuring the voltage and average current of the external power supply connected to the MPE during calculations. Table H1 presents the power consumption for each module within the MPE. The energy consumed for inferring one sample with 4 time steps is calculated to be 0.72 μJ. Note that the weights of the SNN are repeated along the SLs and the time step is 4. The actual amount of operations performed in the MPE is about 3.84 M for completing inference. The energy efficiency is about 5.34 TOPS/W when the SNN running on MPE.

As a comparison, we estimate the energy consumption and computational latency of a typical digital accelerator-based (HNPU-based) system to complete the inference of one sample. The typical computing power, energy efficiency, energy consumption of memory operation, and memory bandwidth of the HNPU-based system are 3.07 TOPS, 2.64 TOPS/W, 55 pJ/bit, and 19.2 GB/s respectively [9]. During the inference process, energy consumption and computing latency are mainly contributed by calculation operations and memory-fetching operations. The energy consumption of the calculation



(a) 2 time steps, ACC = 81.73%, T = 14.52 μs.

(b) 6 time steps, ACC = 98.77%, T = 43.68 μs.

(c) 8 time steps, ACC = 99.08%, T = 58.24 μs.

Figure H1 Recognition confusion matrix when the time step takes 2, 6, and 8. ACC means recognition accuracy.

operations is the total number of operations divided by the energy efficiency, and the latency is estimated by the total number of operations divided by the computing power. The energy consumption of the memory fetching is obtained by multiplying the data size by the energy consumption per bit, and the latency is evaluated by dividing the data size by the memory bandwidth. The number of operations and the transferred data size for one inference in the HNPU-based system are 851.54 K and 231.94 Kbit (4 bits per weight), respectively. Hence, the estimated energy consumption and the computing latency of the HNPU-based system for one inference are 13.08 μJ, and 1.79 μs, respectively. The energy consumption in the MPE is reduced by 94.50%, and the MPE is also more energy efficient.

Table H1 Power consumption of the modules within MPE

Module	Driver	Buffer	Array	Sample& Hold& ADC	Total
Power (mW)	8.31	2.32	0.51	13.55	24.69

Appendix I Comparison with other works

Table I1 gives a comparison between this work and related works. In recent years, many works [4] have focused on the energy-efficient computing and hardware deployment of traditional ANNs based on memristors. Traditional ANNs can integrate spatial information well, but are not good at processing multi-modal and spatiotemporal data. To effectively process such data, sophisticated structures are usually introduced in the networks [11]. However, this also increases the difficulty of deployment on memristor-based computing platforms. SNNs are considered more energy-efficient and can integrate multi-modal information from both temporal and spatial dimensions. However, there is little work on their on-chip deployment and verification, and most of the studies on memristive SNNs are implemented via software simulation [12, 13]. Since the membrane potential of spiking neurons changes nonlinearly when they are activated and emit spikes, SNNs are sensitive to computational errors. Therefore, it remains to be revealed to deploy the SNNs on memristor-based computing platforms. This work proposes effective SNN on-chip deployment methods drawing inspiration from traditional ANN on-chip deployment, especially correcting the results of each layer through layer-by-layer fitting and incorporating the fitting coefficients into the neuron model to avoid additional computational overhead. In addition, we find that SNNs are more suitable for the customized memristor-based computing platform than traditional ANNs because the outputs (only 0 or 1) of SNNs can be fed into the next memristor array directly without additional processing. The proposed tactile-olfactory fusion perception system based on memristive SNN achieves higher recognition accuracy with lower energy consumption than the previous works [11].

Table I1 Comparison with related works

Refs.	Network model	Muti-modal	Spatiotemporal fusion	IMC with spikes	Memristive on-chip verification
[4]	Traditional ANNs	✗	✗	✓	✓
[11]	Bioinspired ANN	✓	✗	✗	✗
[12]	SNN	✗	✓	✓	✗
[13]	SNN	✓	✓	✗	✗
This work	SNN	✓	✓	✓	✓

References

1 LIU M, ZHANG Y, WANG J, et al. A star-nose-like tactile-olfactory bionic sensing array for robust object recognition in non-visual environments. Nature Communications, 2022, 13(1): 79

2 SU C Y, MENUZ K, CARLSON J R. Olfactory perception: receptors, cells, and circuits. Cell, 2009, 139(1): 45–59.
3 LIU Q, GAO B, YAO P, et al. 33.2 A fully integrated analog ReRAM based 78.4 TOPS/W compute-in-memory chip with fully parallel MAC computing. In: 2020 IEEE International Solid-State Circuits Conference-(ISSCC). San Francisco, CA, USA, 2020. 500–502.
4 LIU Y, GAO B, TANG J, et al. Architecture-circuit-technology co-optimization for resistive random access memory-based computation-in-memory chips. SCIENCE CHINA Information Sciences, 2023, 66(10): 200408.
5 YAO M, ZHAO G, ZHANG H, et al. Attention spiking neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(8): 9393–9410.
6 BENGIO Y, LÉONARD N, COURVILLE A. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432, 2013.
7 Wan W, Kubendran R, Schaefer C, et al. A compute-in-memory chip based on resistive random-access memory. Nature, 2022, 608(7923): 504-512.
8 FANG W, CHEN Y, DING J, et al. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. Science Advances, 2023, 9(40): eadi1480.
9 ZHANG W, YAO P, GAO B, et al. Edge learning using a fully integrated neuro-inspired memristor chip. Science, 2023, 381(6663): 1205–1211.
10 Dong Z, Ji X, Lai C S, et al. Memristor-based hierarchical attention network for multimodal affective computing in mental health monitoring. IEEE Consumer Electronics Magazine, 2022, 12(4): 94-106.
11 WANG J, LI X, LIU M, et al. Bionic mechanical hand integrated with artificial olfactory sensor array for enhanced object recognition. In: 2023 IEEE 36th International Conference on Micro Electro Mechanical Systems (MEMS), Munich, Germany, 2023. 209-212
12 Kim T, Hu S, Kim J, et al. Spiking neural network (SNN) with memristor synapses having non-linear weight update. Frontiers in computational neuroscience, 2021, 15: 646125.
13 Zhu J, Zhang X, Wang R, et al. A heterogeneously integrated spiking neuron array for multimode?fused perception and object classification. Advanced Materials, 2022, 34(24): 2200481.