nature communications

Article

https://doi.org/10.1038/s41467-025-56079-3

Random resistive memory-based deep extreme point learning machine for unified visual processing

Received: 21 December 2023

Accepted: 6 January 2025

Published online: 23 January 2025

Check for updates

Shaocong Wang[1,2,3,10], Yizhao Gao[1,10], Yi Li[1,2,4,5,10], Woyu Zhang[2,4,5], Yifei Yu[1,3], Bo Wang[1,3], Ning Lin[1,3], Hegan Chen[1,3], Yue Zhang[1,3], Yang Jiang[1,3], Dingchen Wang[1,3], Jia Chen[1,3], Peng Dai[1], Hao Jiang[6], Peng Lin[7], Xumeng Zhang[6], Xiaojuan Qi[1], Xiaoxin Xu[2,4,5], Hayden So[1], Zhongrui Wang[8] ✉, Dashan Shang[2,4,5] ✉, Qi Liu[2,6], Kwang-Ting Cheng[3,9] & Ming Liu[2,6]

Visual sensors, including 3D light detection and ranging, neuromorphic dynamic vision sensor, and conventional frame cameras, are increasingly integrated into edge-side intelligent machines. However, their data are heterogeneous, causing complexity in system development. Moreover, conventional digital hardware is constrained by von Neumann bottleneck and the physical limit of transistor scaling. The computational demands of training ever-growing models further exacerbate these challenges. We propose a hardware-software co-designed random resistive memory-based deep extreme point learning machine. Data-wise, the multi-sensory data are unified as point set and processed universally. Software-wise, most weights are exempted from training. Hardware-wise, nanoscale resistive memory enables collocation of memory and processing, and leverages the inherent programming stochasticity for generating random weights. The co-design system is validated on 3D segmentation (ShapeNet), event recognition (DVS128 Gesture), and image classification (Fashion-MNIST) tasks, achieving accuracy comparable to conventional systems while delivering 6.78 × /21.04 × /15.79 × energy efficiency improvements and 70.12%/89.46%/85.61% training cost reductions.

The burgeoning field of intelligent machines has seen a rapid integration of diverse visual sensors such as conventional frame cameras, light detection and rangings (LiDARs), and dynamic vision sensors (DVS). These sensors enable machines to better perceive and comprehend the surrounding environment. However, the data generated by these sensors can exhibit considerable heterogeneity. Specifically, frame camera-derived images embody grid structures, whereas LiDAR-derived point clouds are

[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, China. [2]Key Lab of Fabrication Technologies for Integrated Circuits Institute of Microelectronics, Chinese Academy of Sciences, 100029 Beijing, China. [3]ACCESS—AI Chip Center for Emerging Smart Systems, InnoHK Centers, Hong Kong Science Park, Hong Kong, China. [4]Key Laboratory of Microelectronic Devices & Integrated Technology, Institute of Microelectronics, Chinese Academy of Sciences, 100029 Beijing, China. [5]University of Chinese Academy of Sciences, 100049 Beijing, China. [6]Frontier Institute of Chip and System, Fudan University, Shanghai 200433, China. [7]College of Computer Science and Technology, Zhejiang University, Zhejiang 310027, China. [8]School of Microelectronics, Southern University of Science and Technology, Shenzhen 518055, China. [9]Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. [10]These authors contributed equally: Shaocong Wang, Yizhao Gao, Yi Li.
✉e-mail: wangzr@sustech.edu.cn; shangdashan@ime.ac.cn

irregular, unordered[1]. DVS output is characterized by asynchronous and sparse event streams[2].

Consequently, this significant dispersion in data structure leads to huge complexity and segregation in algorithm design, training strategy, and hardware optimization, making the system development extremely costly for edge-side intelligent devices[3,4]. This complexity is further exacerbated by the exhaustive training required for each data modality. Although efforts have been made to unify the processing of point cloud, event data, and images using CNNs[5] or transformers[6], they lead to information degradation (e.g., for converting event data) or memory cost (e.g., for voxelizing the 3D point cloud)[7,8], not to mention their prohibitive training complexity.

The performance of the multi-sensory intelligent machine is further limited by its hardware. Conventional digital hardware—currently the de facto platform for most machine learning software—suffers from drastic energy inefficiency, which can be fatal for edge-side intelligence machines. The separation of the processing and memory units in conventional digital hardware, known as the von Neumann bottleneck, results in heavy data traffic between the two units, contributing a considerable amount of energy consumption of AI algorithms[9]. In addition, the size of complementary metal-oxide semiconductor (CMOS) is approaching its physical limit, causing the slowing down of Moore's law and limiting the further fundamental improvement in the energy efficiency of conventional digital hardware.

To address the aforementioned challenges, we unify the processing of heterogeneous multi-sensory data from the perspectives of data interpretation, software design, and hardware development.

Data from these diverse sensors can be interpreted into a unified data structure, e.g., point set, with minimal pre-processing overhead and no information loss. Point cloud, as a collection of three-dimensional spatial points, can be represented as a point set by nature. Event streams, by interpreting the time dimension as a spatial one, can also be construed as a point set in (x, y, t) three-dimensional space, with each 3D point representing an event[10]. For images, each pixel is actually a point in (x, y, *grayscale*) space, enabling images to be seamlessly interpreted as a set containing these pixels as points[11]. Consequently, the processing of point clouds from LiDARs, event streams from DVS sensors, and images from frame cameras can be unified. This unification minimizes computational complexity and data memory footprint, simplifies sensing pipeline, and reduces in-memory hardware footprint using weight sharing, as discussed in Supplementary Note 3.

In terms of software, drawing inspiration from extreme learning machines (ELM)[12,13] that mimic the human neural networks, wherein general low-level cortices are fixed, and task-specific higher-level cortices exhibit greater flexibility, we propose a deep extreme point learning machine (DEPLM). This DEPLM facilitates the processing of varied modal data using identical software and hardware architectures while obviating the need for tedious training, and thus the frequent write operations on resistive memory hardware.

From a hardware perspective, we resort to an in-memory computing system based on emerging memory. Such systems integrate nanoscale resistive memory cells in a crossbar configuration, enabling vector-matrix multiplication (VMM) via voltage-amplitude-vector multiplying with conductance-matrix using Ohm's law and Kirhoff's law. As computation is performed right where the data is stored, it leads to minimal data traffic and energy consumption[14–33].

In addition, we leverage the programming stochasticity of resistive memory to produce sparse Gaussian distributed random weights of DEPLM, transforming the disadvantage into a benefit. The sparsity also endows the hardware with enhanced robustness against the cycle-to-cycle read noise in analog computing[34,35].

In this Article, we present such a deep extreme point learning machine with 40 nm resistive memory array on edge learning different data modalities. The system is evaluated on the 3D point cloud segmentation task ShapeNet, the event-based gesture recognition task on the DVS128 Gesture dataset, and the image classification task Fashion-MNIST. Compared to the conventional digital hardware-based systems, our co-design system achieves 6.78 × , 21.04 × , and 15.79 × energy consumption improvements on these three tasks, respectively. Additionally, the system achieves 70.12, 89.46, and 85.61% reduction in training cost compared to conventional digital systems. Our work may pave the way for future energy-efficient and training affordable edge AI with multi-sensory data.

Results

Hardware-software co-designed deep extreme point learning machine

Figure 1 illustrates the hardware-software co-design of the deep extreme point learning machine. Figure 1a shows the algorithmic schematic of DEPLM. In general, the DEPLM is composed of several layers of mapping-aggregating operation, following the PointNet++ styled point-based methods[36,37]. Without loss of generality, we assume that the input data, resembling the digit "8" in this figure, is a set of points in three-dimensional Euclidean space $S = \{\mathbf{p_i} | \mathbf{p} \in \mathbb{R}^3, i \in \{1, \ldots, N\}\}$, where N is the number of points in the set, and each point $\mathbf{p_i}$ is represented by its 3D coordinate vector $\mathbf{p_i} = (x_i, y_i, z_i)$. The coordinate vector of each point is first mapped to a higher dimensional feature space with a resistive-memory-array-implemented fully connected layer $\mathbf{W}^{(in)} \in \mathbb{R}^{d \times 3}$ shown in Fig. 1b, mathematically $\mathbf{f_p} = \mathbf{W}^{(in)}\mathbf{p} \in \mathbb{R}^d$, where d is dimension of the mapped space. The point feature vectors are then grouped into overlapping subsets according to their 3D coordinate distance (see details of grouping in "Method"). Point feature vectors in the same subset are then aggregated, using sum pooling, into a single feature vector representing the information of the entire subset. The subset feature vectors form a new point set which is the input to the next layer. After L layers of mapping-aggregating operation, the entire point set is abstracted into a single representation vector, which can be used for downstream tasks, like 3D segmentation and classification in this article. The fully connected layers $\{\mathbf{W}^l | l \in \{1, \ldots, L\}\}$ in DEPLM are physically implemented on the random resistive memory arrays, where L is the number of layers. Figure 1c shows a 50×50 sub-array of the weight matrix of a fully connected layer. The weights follow a zero-inflated Gaussian mixture distribution with three Gaussian sub-distributions centered at −34.12, 34.43, and 0.00, respectively. Physically, the weights are conductance difference of two half-sparse 50×50 resistive memory sub-arrays, which are stochastically electroformed to be 50% sparse. The resistive memory conductance follows a zero-inflated Gaussian distribution with the Gaussian model centered at ~35.85 μS show in Fig. 1d (their implementation in resistive crossbar shown in Fig. 1e). As such, DEPLM leverages the programming stochasticity of resistive memory in generating high density, large-scale and true random weights, and can be robust to read noise (to be discussed in the later section). Figure 1f shows the photo of such a resistive memory chip. Its micro structure is shown in Fig. 1g, h, corresponding to the cross-sectional transmission electron microscopy (TEM) of the resistive memory crossbar structure and the composing cell, respectively (see details of the system in Supplementary Fig. 1 and device characteristics in Supplementary Fig. 2).

3D point cloud segmentation

The system is first evaluated on a prominent three-dimensional point cloud part segmentation benchmark, ShapeNet[38] (see Supplementary Fig. 4 for 3D point cloud classification task on ModelNet[39]). Part segmentation is a challenging fine-grained 3D task to recognize a specific part of an object to which a certain point belongs. For example, given a 3D chair point cloud sample, the system is expected to discern which points correspond to the chair's back, seat, and legs. The ShapeNet dataset contains 16 types of 3D objects that can be segmented into a total of 50 parts. Figure 2a shows the data flow of

https://doi.org/10.1038/s41467-025-56079-3

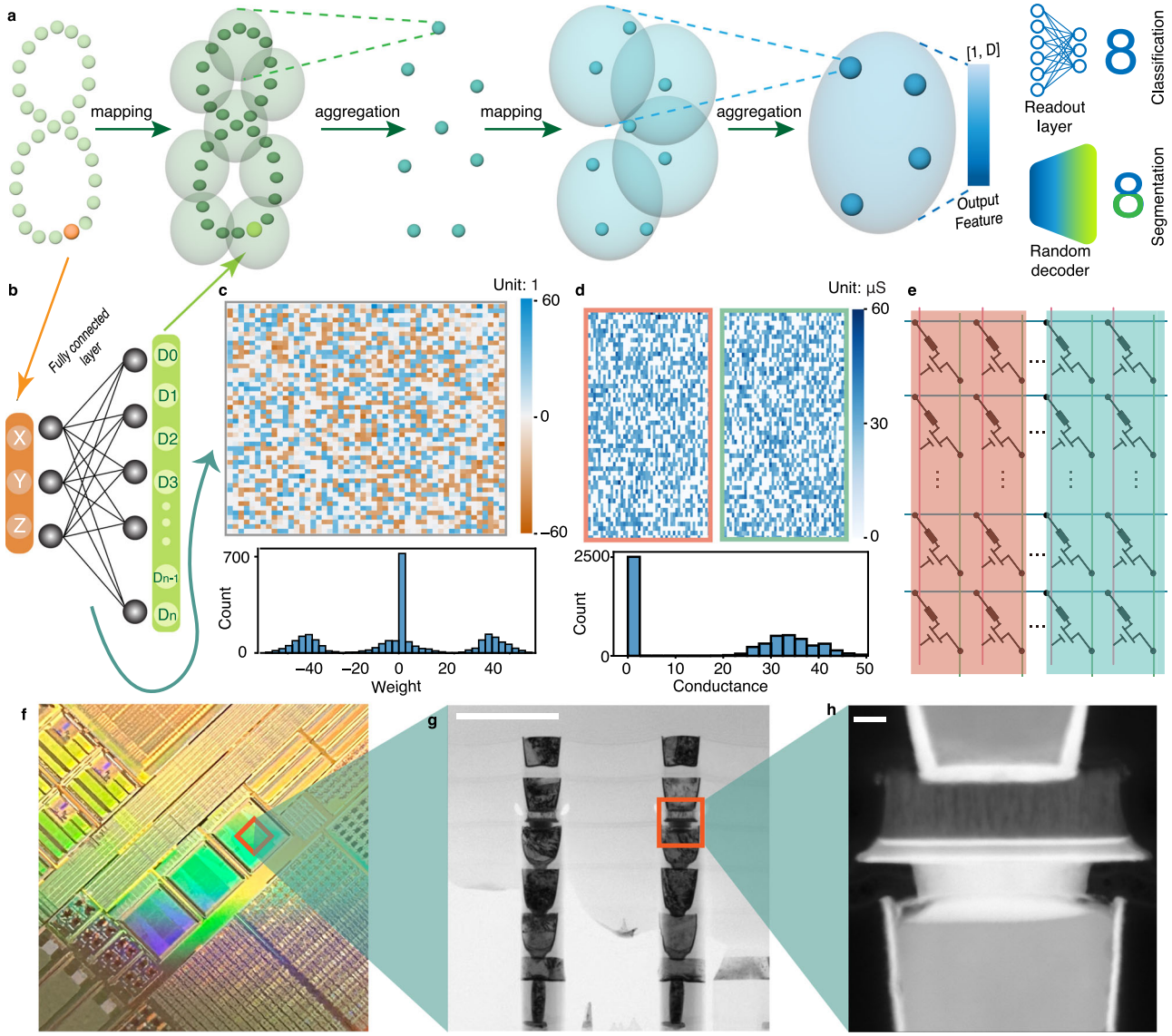

Fig. 1 | Hardware-software co-design the random resistive memory-based deep extreme point learning machine. a The point set encoding process of DEPLM. **b** The fully connected layer. Each three-dimensional (in the input layer in this example) point in the point set is fed into the layer and mapped into an n-dimensional vector. **c** A 50 × 50 random weight matrix and its distribution histogram. The weight matrix is implemented using two 50 × 50 random resistive memory arrays in (**d**). The matrix with red (green) bounding box represents the positive (negative) part of the differential conductance matrix. The resistive memory conductance distribution follows a zero-inflated Gaussian distribution, with half conductance being zero (remaining insulating). **e** The circuit schematic of resistive memory crossbar array separated into the positive (red) and negative (green) sub-arrays for the differential conductance matrix. **f** Optical photo of the resistive memory chip. **g** The cross-sectional TEM of resistive memory array. (Scale bar: 500 nm). **h** The cross-sectional TEM of a single resistive memory cell. (Scale bar: 20 nm).

DEPLM. Given a 3D point cloud sample, the system first encodes its feature hierarchically with the random encoder, producing a single feature vector as the representation of the entire sample, shown in the middle of Fig. 2a. This representation is then hierarchically decoded to get the feature of each point (see "Method" for details of the random decoder and Supplementary Fig. 14 for the feature of each layer). The features are subsequently fed into a single-layered readout map (classification head) to determine which object parts the points correspond to. With the color of points representing their feature distances, the output feature of the decoder shows a color discrepancy among the points on the chair's back, seat, and legs, indicating the discriminative nature of the point feature in segmenting these three object parts. Figure 2b shows the selected feature vector of each point (each column), grouped according to its part class. It is clear that feature vectors of points from the same object part are similar across channels, while those from different parts of the object are distinct (see Supplementary Fig. 5 for the original feature vector). The readout layer segments the 3D point cloud samples into different parts. The mean intersection over union (mIoU) of each object type is shown in Fig. 2c. The mIoU of the hardware experiment is shown in yellow bars, while that of the software simulated DEPLM and the trainable baseline is shown in green and blue bars, respectively. The performance of hardware DEPLM on the majority of classes is comparable to the fully-trained baseline and the software-simulated DEPLM, but exhibits a noticeable decline for long tail classes like bag, car, and motorbike, due to the cycle-to-cycle conductance fluctuation and scarcity of samples in these classes (see Supplementary Fig. 3 for mIoU distributions of the hardware outperformed categories). The overall instance average mIoU (accuracy) of the hardware experiment is 66.28% (83.79%),

https://doi.org/10.1038/s41467-025-56079-3

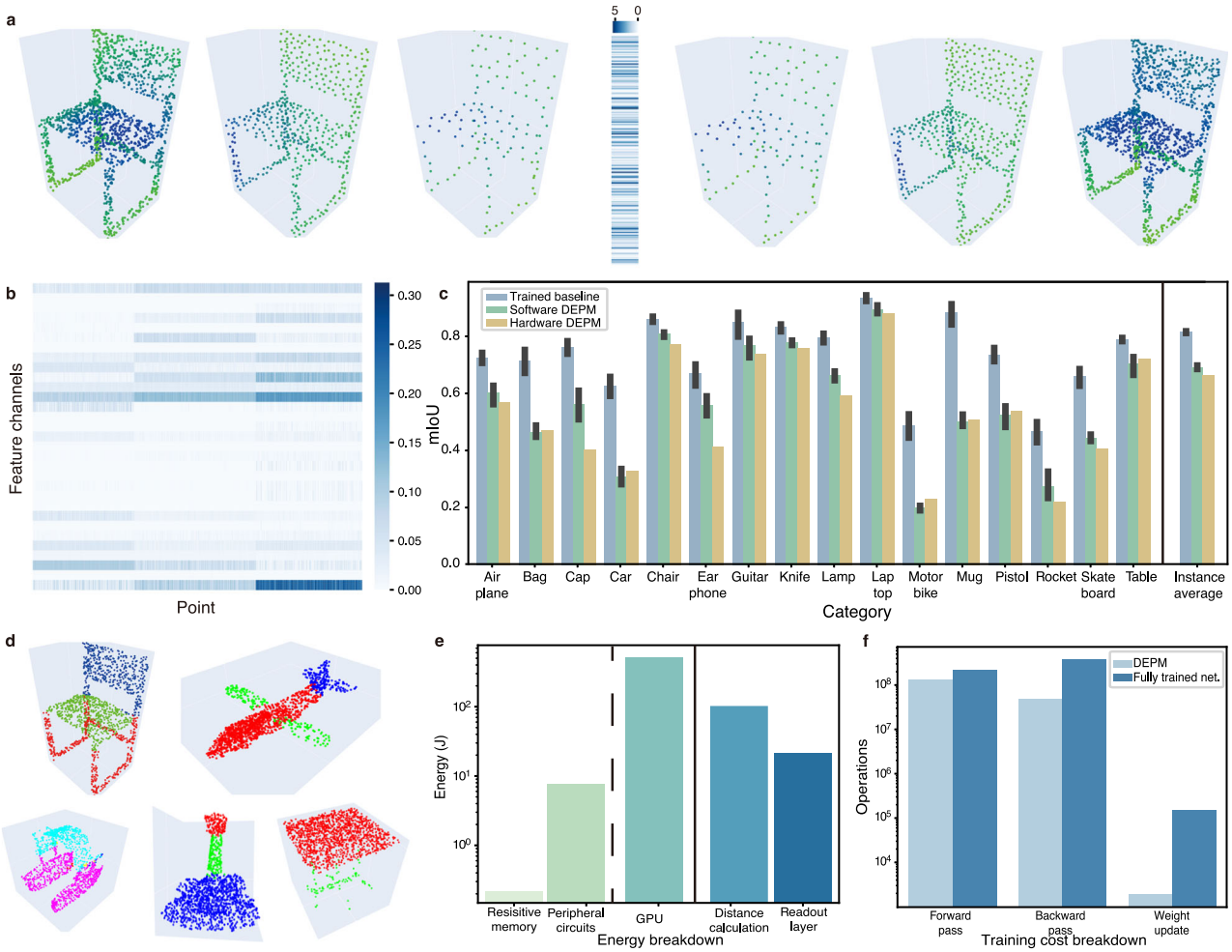

Fig. 2 | Experimental point cloud part segmentation using ShapeNet dataset. **a** Segmentation dataflow of the DEPLM. The input 3D point cloud is hierarchically encoded into a single representation vector, shown in the middle, using the DEPLM. The representation vector is then decoded with the random decoder with skip connection. The color of the points shows the distance of their features. **b** Selected feature vectors. **c** The class-wise mean intersection over union (mIoU) and the instance average mIoU of the hardware experimented DEPLM (yellow bars), software simulated DEPLM (green bars), and trained baseline (blue bars). The hardware DEPLM performance is sub-optimal on the classes with limited samples, while similar to the trained counterpart and the software DEPLM on the majority of classes. The overall instance average mIoU of the hardware DEPLM is similar to that of the trained baseline and the software simulated DEPLM. Error bar: standard deviation over 10 runs. **d** Visualization of segmentation results. The color of points indicates the part to which they belong. **e** Energy breakdown of the DEPLM on inferring a sample, compared to a state-of-the-art digital system. The energy reduction is attributed to in-memory computing with resistive memory. **f** Training cost breakdown, showing complexity reduction due to random weights in DEPLM.

compared to 81.49% (92.79%) of the trained baselines and 69.14% (86.97%) of the software simulated DEPLM (see Supplementary Table 13 for baseline comparison). Figure 2d visualizes the segmentation results. The part of a point is marked with its color (see upscaled simulation result on large scaled point cloud datasets in Supplementary Table 8).

In addition to the segmentation performance, we conducted a comprehensive analysis of our system's efficiency. Figure 2e shows the energy breakdown of the DEPLM when segmenting a sample from the ShapeNet dataset in comparison to a state-of-the-art digital system. In our system, the VMM operations of both the random encoder and decoder are executed on the random resistive memory. The energy for segmenting a single 3D sample on the resistive memory and its peripheral circuits is 218.09 nJ and 7.68 μJ, respectively, significantly lower than that of VMM on digital system (522.24 μJ). The distance calculation occurred in point set grouping and the final readout layer is performed on the digital component of our hybrid system, contributing 102.45 μJ and 21.20 μJ to the energy consumption, respectively. This is consistent with the conventional digital system (see Supplementary Note 1 for layer-wise energy breakdown in Supplementary Note 1). The overall energy consumption of the DEPLM is 135.11 μJ, realizing ~6.78 × improvement in energy efficiency, compared to 916.08 μJ of the digital system. Figure 2f presents the detailed breakdown of estimated DEPLM training complexity for a single sample, compared to a fully trained baseline. The complexity of the forward pass is 131.40 MOPs, which is lower than the 214.43 MOPs observed on the fully trained baseline, due to the swapped "grouping" and "mapping" operations (see Supplementary Fig. 6). The complexity of backward pass (weight update) for DEPLM is 47.92 MOPs (11.70 KOPs), significantly lower than 385.53 MOPs (161.04 KOPs) of the trained baseline. The overall training complexity for the DEPLM is reduced by 70.12% compared to that of the fully trained baseline.

Event-based gesture recognition

To prove the effectiveness of the DEPLM system in event stream learning task, the system is evaluated on the DVS128 Gesture[40] classification task. The event stream is treated as a series of 3D points within

https://doi.org/10.1038/s41467-025-56079-3

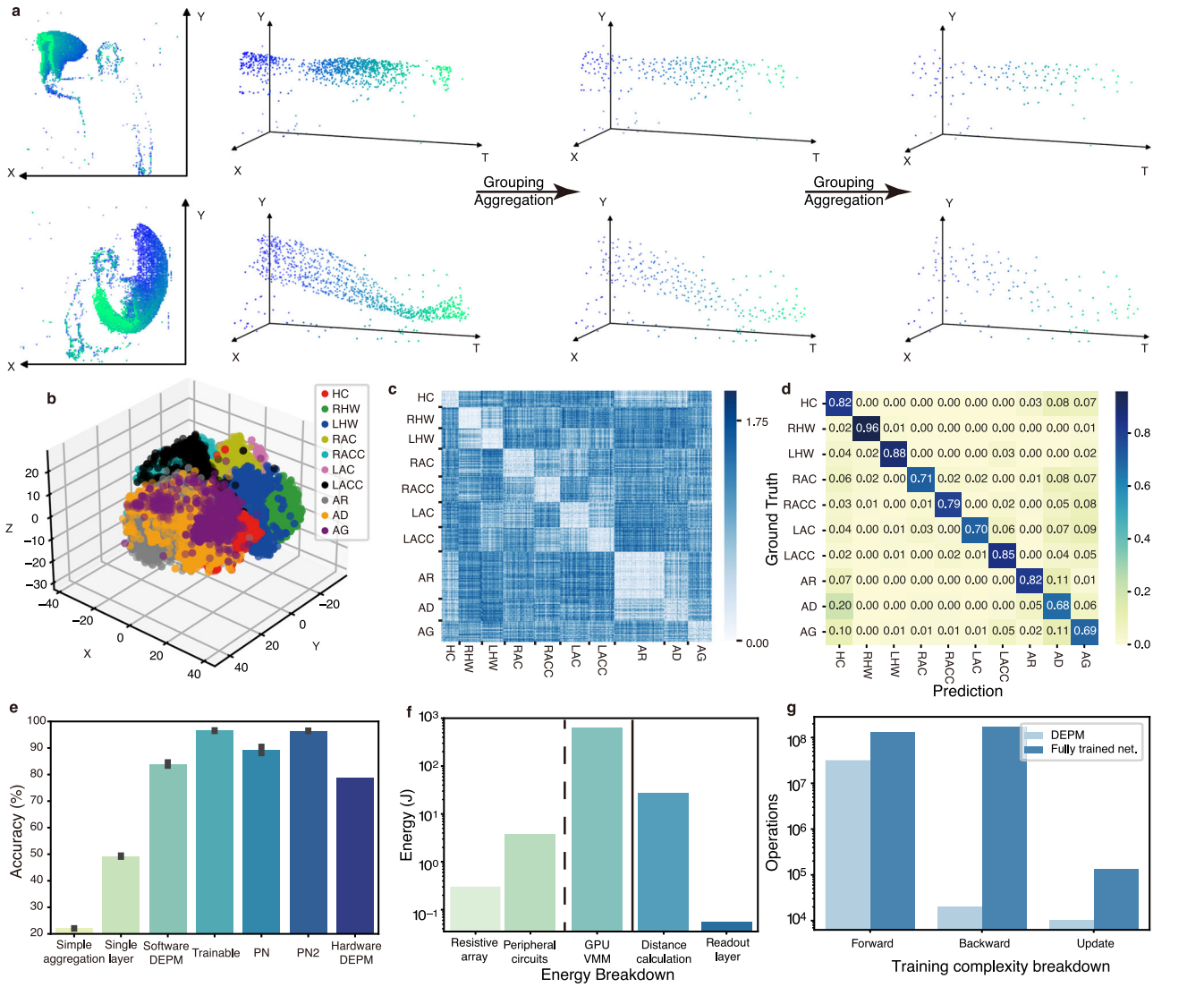

Fig. 3 | Experimental event-based gesture classification with DVS128 Gesture dataset. a Samples from the DVS128 Gesture dataset. The top and bottom samples belong to the Right-hand-wave and Left-arm-counterclockwise gesture, respectively. The spatial projections and the three-dimensional spatial-temporal visualization of the input sample with 1024 events are shown in the left. The event stream samples are aggregated into 256 and 128 points during the forward pass in our co-design. They are finally abstracted into a single representation vector and classified by the trainable readout layer. **b** Linear discriminative analysis with reduced three-dimensional features, showing clear clustering of same category features. **c** Distance matrix of the feature vectors in the dataset. The sub-matrices along the diagonal show smaller distances within each class. **d** Confusion matrix of the datasets, with dominating diagonal elements. **e** Accuracy comparison with other software solutions. The whistle represents the standard deviation. Error bar: standard deviation over 10 runs. **f** Forward pass energy breakdown. The significant reduction is due to in-memory computing with resistive memory. **g** Training complexity breakdown, showing complexity reduction due to random weights in DEPLM.

the spatial-temporal domain (see "Method" for data processing). In Fig. 3a, we illustrate two instances from the DVS128 Gesture dataset, showcasing the Right-hand-wave (RHW) and Left-arm-counterclockwise (LACC) gestures. The event stream sample is fed into three layers of mapping-aggregating operation to hierarchically abstract its semantic feature, with the weights of mapping operations (fully connected layers) physically implemented on random resistive memory. A single representation vector is generated for each event stream sample and subsequently classified by the readout layer (see Supplementary Fig. 7 for representation vectors of all samples in the dataset). Figure 3b shows the linear discriminative analysis (LDA) for the representation vectors of entire datasets. Notably, the representation vectors generated by the random resistive memory from different gesture classes (indicated by different colors) roughly form into separated clusters, enabling the readout map (linear classifier) to differentiate. Furthermore, Fig. 3c presents an L2 distance matrix of the representation vectors of samples, gathered according to their classes. Most diagonal sub-matrices exhibit higher intensity, indicating smaller L2 distances for intra-class features. This corroborates the capability of random resistive memory to effectively extract features for different gestures in the event stream. The confusion matrix in Fig. 3d outlines classification outcomes, with the diagonals dominating. (see Supplementary Fig. 9 for unnormalized confusion matrix). Figure 3e compares the accuracy of our hardware implemented DEPLM with others. Our hardware DEPLM achieves 78.73% accuracy on the DVS128 Gesture Recognition dataset, which is slightly lower compared to 83.97% (96.50%) of the software-simulated (fully trained) version due to read noise impact (fixed weights). In comparison, PointNet[36] and PointNet++[37] with a comparable number of parameters yield accuracies of 89.27% and 96.46%, respectively (see Supplementary Table 14). To further demonstrate the effectiveness of the hierarchical random projection, we conduct two ablation studies. If the

input point set is directly abstracted with global sum pooling without random-projection-based feature extraction, the accuracy is significantly dropped to 22.04% (the leftmost bar). On the other hand, when a single (instead of deep) random mapping layer is implemented, the accuracy will also drop to 49.28% (second bar from left). These findings show the substantial performance enhancement provided by our model with deep (three-stage) mapping-aggregating operations. Figure 3f shows the energy consumption breakdown, in comparison with conventional digital hardware. The VMM and Peripheral circuits of the system only consume 0.30 μJ and 3.84 μJ per inference, respectively, while the VMM on conventional digital hardware consumes 627.73 μJ. The distance calculation (26.92 μJ) and readout map (55.80 nJ) of our system are both on digital hardware (see Supplementary Note 1 for layer-wise energy breakdown). The overall energy consumption of our system is 31.12 μJ, compared to 654.71 μJ of the conventional digital hardware, leading to ~21.04 × energy efficiency. Finally, Fig. 3g illustrates breakdown of training complexity of our DEPLM compared to the fully trained network. The DEPLM leads to 4.05 × and 8500 × computation saving in forward and backward passes, respectively, and 13.20 × of computation reduction for updating weights. Overall, our DEPLM reduces 89.46% training complexity of the fully trained network.

Classify image as point set

The system is further evaluated on the Fashion-MNIST dataset[41] to show its effectiveness on image classification tasks, as shown in Fig. 4. Each image in the dataset is treated as a 3D point cloud $\mathbf{p_i} = (x_i, y_i, grayscale_i)$, with *grayscale* of a pixel serving as the third dimension alongside the x and y coordinates (see "Method" for data processing). For each image with 28 × 28 resolution, this leads to the creation of 784 points, which are subsequently fed into our system. Figure 4a depicts the three stages of point mapping-aggregating that produce 512 (middle) and 128 (right) points in the second and third stages undergo the transformations physically implemented on random resistive memory. The final representation vector produced by the random resistive memory-based DEPLM encoder is then classified into 10 classes by the readout layer (see Supplementary Fig. 8 for representation vectors of all samples in the dataset). Figure 4b shows the linear discriminative analysis by projecting the final representation vector into three-dimensional space. Different classes, denoted by distinct colors, are approximately mapped into distinct clusters, thanks to the random resistive memory implemented DEPLM encoder. This indicates that representation vector produced by DEPLM encoder can indeed be classified by a simple linear classifier (readout map). Figure 4c presents the L2 distances of representation vectors between all samples in the dataset. In general, diagonal submatrices exhibit brighter spots, indicating closer relationships among the intra-class representation vectors. The distances between samples from several different classes can be small, due to their intrinsic similarity. For example, the Shirt class with the lowest classification accuracy (53%), exhibits shorter distances to other clothing types like T-shirt, Pullover, and Coat. Correspondingly, the confusion matrix in Fig. 4d also demonstrates the elevated likelihood of Shift class being incorrectly classified as T-Shirt (11%), Pullover (26%), and Coat (19%), echoing their smaller inter-class feature distances (see Supplementary Fig. 9 for unnormalized confusion matrix), albeit the diagonal values are still dominating. Figure 4e compares the classification performance of our system with software methods. Our random resistive memory-based DEPLM on the Fashion-MNIST dataset achieves 77.20% accuracy, slightly lower than the software-simulated (fully trainable) counterpart attains 83.62% (91.06%), due to the hardware noise (fixed random weights). PointNet[36] and PointNet++[37] with same weight population achieve 91.34% and 90.87% accuracy, respectively, which is comparable to the trainable DEPLM (see Supplementary Table 15. Also see up-scaled simulation result on ImageNet-100 in Supplementary Table 8.) Ablation studies show that directly feeding the image point set into a global pooling layer and a readout layer only achieves 10.36% accuracy. In addition, if only a single mapping-aggregating layer is implemented, the accuracy drops to 68.9%, showing the effectiveness of our deep architecture.

Despite a slight accuracy drop compared to those software methods, our co-design demonstrates superior energy efficiency and training complexity. As shown in Fig. 4, the VMM on random resistive memory only consumes 0.48 μJ per inference, while the peripheral circuits only use 5.14 μJ. This leads to 118.52 × energy reduction compared to the VMM on conventional digital hardware (666.36 μJ). The distance calculation and software readout layer, both taken place on digital hardware, consume 38.97 μJ and 89.26 nJ, respectively (see Supplementary Note 1 for layer-wise energy breakdown). The overall energy consumption of our hybrid system is 44.68 μJ, compared to 705.42 μJ of the conventional digital system, leading to ~15.79 × energy efficiency. Figure 4g reveals the reduction in training complexity achieved with DEPLM. The necessity to compute gradients using back-propagation is limited to the linear readout layer, significantly reducing computational efforts by a factor of 8×10^3 in the backward pass. Moreover, the forward pass and weight updating computation also diminished by 2.56 × and 12.21 × , respectively. The overall training complexity of our system is 60.26 MOPs, while that of the fully trained network is 418.73 MOPs, resulting in ~85.61% reduction in training cost.

Sparse resistive array for noise robustness

We further demonstrate the benefits of weight sparsity of DEPLM in mitigating the impact of the read stochasticity (cycle-to-cycle variation) associated with nano-scale resistive memory[33,42,43]. Figure 5a shows the conductance values of 20 resistive memory cells in 30 thousand read operations, with an averaged standard deviation of 0.27 μS for these 20 devices. Figure 5b illustrates the standard deviation map over 30 thousands read operations for a dense resistive memory array (upper left) and a sparse one (upper right), and their respective distributions (bottom). It is evident that the sparse resistive memory array exhibits a lower average standard deviation since half of the cells remain un-electroformed (see Supplementary Fig. 10 for the coefficient of variation of the conductance maps). It implies a lower overall noise disturbance of sparse resistive memory array on the network, as the network's weights are mapped physically as differential pairs of two random resistive sub-arrays. Figure 5c presents the standard deviation map and the corresponding distribution of a 50 × 50 weight matrix derived from the conductance matrix of differential resistive memory pairs, where the distribution of dense (sparse) weight map is depicted in blue (orange) bars. The standard deviation of dense weights ranges between 0.32 (25% quantile) and 0.38 (75% quantile), while that of sparse weight presents a mixture of three sub-distributions. The first sub-distribution (zero-centered spike) represents the weights derived from the differential pairs of un-electroformed cells. As these weights are composed of two insulated cells, they barely suffer from cycle-to-cycle variation. The second group of weights comprises differential pairs of an insulated cell and a randomly formed cell. The third group of weights are differential pairs of both randomly formed cells, the same as that of the dense weights. The standard deviation distribution of these weights also coincides with that of dense weights, displaying the largest standard deviation among the three groups (see Supplementary Fig. 10 for coefficient of variation of the weight maps). Overall, the standard deviation of sparse weights is 0.22 on average, demonstrating an advantage over 0.36, observed on the dense weights.

We subsequently simulated the impact of resistive memory weight sparsity under various read noise levels on the performance of

https://doi.org/10.1038/s41467-025-56079-3

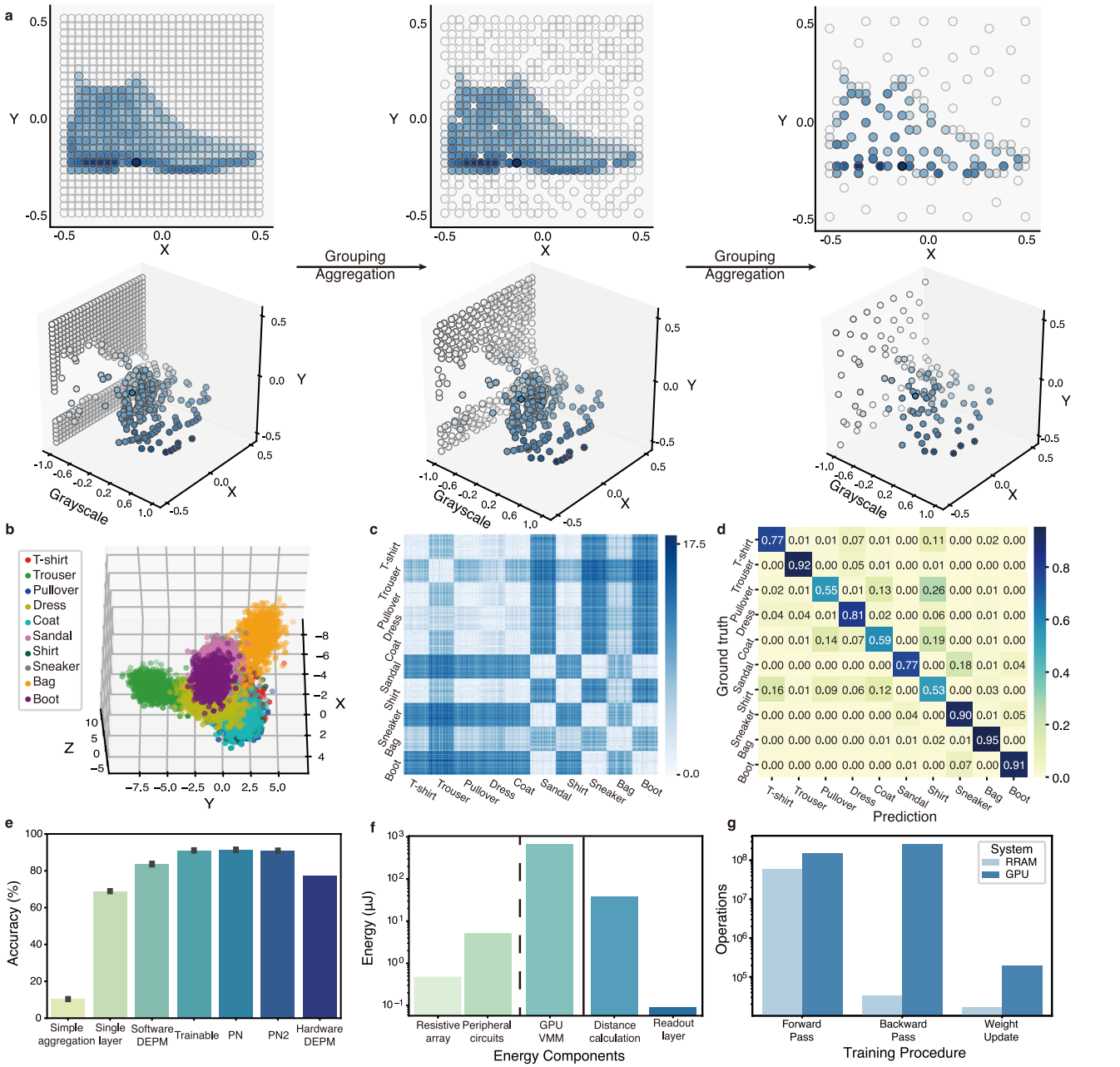

Fig. 4 | Experimental image classification with Fashion-MNIST dataset. **a** Examples from the Fashion-MNIST image dataset. Each image from the dataset is treated as a 3D point cloud by considering the pixel intensity as the third dimension besides the x and y dimensions. The top three images show the spatial projections of the point cloud of the input stage, the first and second grouping stages in the networks. The bottom three images show the corresponding 3D perspective plots. **b** Linear discriminative analysis by reducing the feature vectors generated by DEPLM to three dimensions. Different classes roughly form into separated clusters in the space. **c** Distance matrix. The L2 distances between feature vectors of all the samples in the dataset. The diagonal matrices show smaller distances within the same class. **d** Confusion matrix. **e** Accuracy comparison with other software models. The whistle represents the standard deviation. Error bar: standard deviation over 10 runs. **f** Energy breakdown of forward pass. The reduction roots on the in-memory computing with resistive memory. **g** Training complexity breakdown. The reduction is due to random and fixed weights of DEPLM.

the aforementioned three tasks, as shown in Fig. 5d–f. A general pattern shared by all three tasks indicates that moderating sparsity on resistive memory arrays with read noise would benefit the system performance. It is commonly observed in these tasks that the DEPLM with the fully dense weights is susceptible to the effects of read noise. When the noise is less than 2%, the performance of DEPLM on DVS task and the Fashion-MNIST task exhibits a noticeably large standard deviation, which is reasonable as the noise on all the DEPLM weights would cause a large accumulated deviation on multilayered VMM results. When the cycle-to-cycle noises are larger than 4%, all three tasks show a large drop in performance. The standard deviations are small due to a low average accuracy. With the increase in cell sparsity (e.g., 50%), the accuracy enhances generally on all three tasks. However, the standard deviation goes large, as the noise on the small amount of effective weight would also cause a large deviation in system output. With the extreme resistive cell sparsity (>90%), the performances on three tasks are all dropping as the amount of effective weight are too limited. To balance the performance on three tasks that

https://doi.org/10.1038/s41467-025-56079-3

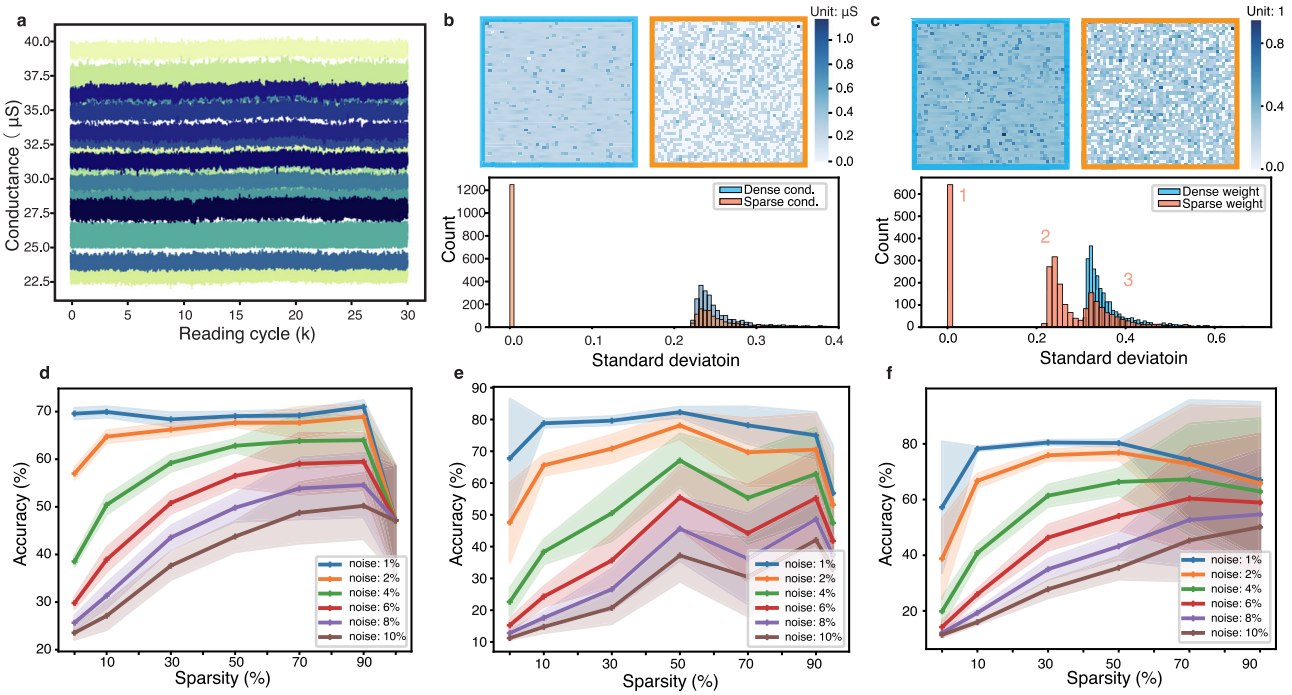

Fig. 5 | Resistive array sparsity for read stochasticity robustness. a 30 thousand reading cycles of 20 resistive memory cells. The average standard deviation is 0.27. **b** Comparison between the standard deviation map on 30 thousand reading cycles of a 50 × 50 dense conductance array (blue bounding box) and a sparse one (orange bounding box), as well as their distribution. The sparse array exhibits a smaller overall reading standard deviation. **c** Comparison between the standard deviation on 30 thousand reading cycles of a 50 × 50 dense weight map (blue bounding box) and a sparse weight map (orange bounding box), and their distribution. Simulated influence of resistive memory array sparsity under different noise levels, on mIoU of the ShapeNet 3D point cloud segmentation task (**d**), the accuracy of the DVS128 Gesture recognition task (**e**), and accuracy of Fashion-MNIST image classification task (**f**). Band: standard deviation of 10 runs simulation.

share the same memory array, we choose 50% sparsity on cells of the resistive memory array as they perform well and are stable at the experimental cycle-to-cycle read noise level.

Discussion

In this study, we present an experimental random resistive memory-based DEPLM, a hardware-software co-designed system for efficient and learning-affordable point set processing, which can be used in various edge applications such as 3D point cloud segmentation, DVS event stream classification, and image classification. Compared to the conventional machine learning models on digital hardware, our co-designed system achieves energy consumption reduction of 5.90 × , 21.04 × , and 15.79 × for the ShapeNet 3D segmentation task, DVS128 Gesture event-based event stream recognition task, and Fashion-MNIST image classification task, respectively. Moreover, it attains training cost reductions of 70.12%, 89.46%, and 85.61%, respectively, compared to conventional systems. Our work may pave the way for future efficient and affordable edge AI with multi-sensory inputs.

Methods

Fabrication of random resistive memory chips

The resistive memory array was fabricated using the 40 nm technology node and has a 1T1R structure. Each resistive memory cell was constructed between the metal 4 and metal 5 layers of the backend-of-line process, comprising a bottom electrode (BE), top electrode (TE), and transition-metal oxide dielectric layer. The BE via was patterned using photolithography and etching, filled with TaN via physical vapor deposition, and covered with a 10 nm TaN buffer layer. Subsequently, a 5 nm Ta layer was deposited and oxidized to form an 8 nm TaOx dielectric layer. Finally, a 3 nm Ta layer and 40 nm TiN layer were sequentially deposited by physical vapor deposition to form the TE. Standard logic process was used to deposit the remaining interconnection metals. The cells in the same row shared BE connections, while those in the same column shared TE connections, forming a 512 × 512 crossbar array. The 40 nm memristor chip demonstrated high yield and strong endurance performance after 30 min of post-annealing at 400 °C in a vacuum environment.

The hybrid analog-digital computing system

The hybrid analog-digital computing system consists of a 40 nm random resistive memory computing-in-memory chip and a Xilinx ZYNQ system-on-chip (SoC) integrated on a printed circuit board (PCB). The system offers parallel 64-way analog voltages for signal inputs, generated using an 8-channel digital-to-analog converter (DAC80508, TEXAS INSTRUMENTS) with 16-bit resolution, ranging from 0 V to 5 V. For signal collections, the convergence current is converted to voltages using trans-impedance amplifiers (OPA4322-Q1, TEXAS INSTRUMENTS) and read out with a 14-bit resolution analog-to-digital converter (ADS8324, TEXAS INSTRUMENTS). Both analog and digital conversions are integrated onboard. During vector-matrix multiplications, a DC voltage is applied to the RRAM chip's bit lines through a 4-channel analog multiplexer (CD4051B, TEXAS INSTRUMENTS) with an 8-bit shift register (SN74HC595, TEXAS INSTRUMENTS). The multiplication result carried by the current from the source line is converted to voltages and forwarded to the Xilinx SoC for further processing. The detailed hardware comparison with recent point cloud accelerators is discussed in Supplementary Note 4.

Grouping of the points in DEPLM

Given a point set S, we first find a subset containing P points, $S_p = \{\mathbf{p}_i, i \in \{1, \ldots, P\}\}$ that are farthest to each other using farthest point sampling (FPS) algorithm, where P is a manually set hyper-parameter These points are functioning as centers in each group. For each point in S_p, we find $k-1$ points that are nearest to it using the k nearest

neighbor (kNN) algorithm, these k points are put together as a group. By this we generate P subsets groups.

Random decoder on 3D segmentation task

The architecture of the random decoder on 3D segmentation task is shown in Supplementary Fig. 11c. The encoder part is the same as in other tasks. For the decoder, each feature propagation (FP) layer (shown in Supplementary Fig. 11b) is first interpolated according to the coordinate of its input and the corresponding encoding layer, which is then concatenated with the feature of the corresponding encoding layer. All weights in the decoder are physically mapped with the random resistive memory.

Population of random weights and weight sharing

The DELPM model backbone is tailored from PointNet series[36,37] with compressed number of set abstraction (SA) layers and parameters suitable for edge learning. Different from PointNet++ models, DEPLM is an extreme learning machine, which uses resistive memory programming stochasticity to hardware implement the random weights. This not only turns the hardware weight mapping imprecision into a parallel and low-cost random number generator but also harvests the energy efficiency of in-memory computing. Supplementary Fig. 11c shows the frozen randomly initialized blocks and trainable blocks in processing three modal data. Specifically, the encoder and the decoder in ModelNet segmentation (colored blocks in Figure R10) are randomly initialized without being optimized. Only the final single-layered task-specific heads (uncolored blocks in Supplementary Fig. 11c) are trained.

Supplementary Table 6 further lists the number of parameters being trained and frozen. For all three tasks, the number of untrained random parameters counts for more than 90% of the total model parameter count. Specifically, for the ShapeNet point cloud segmentation task, 94.16 k out of 100.01 k parameters are set as randomly fixed, counting for 94.15% of total parameter of the DEPLM. For DVS classification task, 62.47 k out of 67.59 k parameters are randomly initialized and remain fixed during training, counting for 92.60% of the total parameters. For image classification, 91.81 k out of 100.00 k parameters are randomly fixed, counting for 91.81% of the entire network.

The random weights for three tasks are mapped to the crossbar in both cross-modal and cross-layer sharing mode. See Supplementary Note 2 for mapping layout of the models and the performance comparison between different hardware weight sharing strategy.

Data representation for 3D point cloud segmentation on ShapeNet dataset

We use a common ShapeNet configuration that contains 16,874 point clouds, where each point cloud is sampled to have 2048 points. For part segmentation, the ground truth of each point cloud contains 2–5 parts, with a total of 50 part classes in the whole dataset. Point cloud in ShapeNet dataset is naturally a point set $S = \{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}\}$. The dataset is split to training set and test set with 9:1 ratio. The training set is further augmented using random scaling and random shift. The comparison with point cloud in bird eye view image format is shown in Supplementary Table 11.

Data processing on Event-based DVS128 Gesture dataset

The DVS128 Gesture dataset[40] is an event-based dataset with 10 different types of human gestures. The dataset was captured using a DVS128 camera[44] with 128 × 128 spatial resolution. We follow a similar data pre-processing scheme used by[45], that is, clipping the event camera recordings into small segments with a time window length of 0.5 S and step sizes of 0.25 S. Before feeding into the random resistive memory network, each segment of data is first denoised and 1024 events were randomly sampled. This is because the event camera data is prone to be noisy, we use a simple denoising algorithm that removes events without spatial neighbors given a time window of 0.01 S. The (x, y, t) value in the events will then be normalized into range [0, 1] and be used as the three coordinates in the point cloud. During the network training, the point cloud input is augmented by shifting the point cloud along a random offset within 10% of maximum ranges. The comparison with regular frame-based event data processing is shown in Supplementary Table 10.

Data processing on Fashion-MNIST dataset

The Fashion-MNIST dataset is an image classification dataset with 10 classes of articles of clothing. The images are in grayscale with spatial resolution 28 × 28. To convert images into point cloud representation, each pixel is treated as an individual point while its spatial coordinate (x, y) and its grayscale value $grayscale(x, y)$ are combined as the coordinates of the points (see Supplementary Table 17 for performance without $grayscale(x, y)$ as additional coordinates and Supplementary Table 18 for simulated performance with and without grayscale as coordinates on ImageNet-100 dataset). The *grayscale* is normalized to the range of [−1, 1]. The pixel coordinates (x, y) are transformed to the range of [−0.5, 0.5], by dividing both x and y by 27 and subsequently subtracting 0.5. In the training process, each image point set sample is augmented by shifting a randomly generated offset within 10% of maximum ranges. The comparison with regular grid image data processing is shown in Supplementary Table 9.

Data availability

All data that support the findings of this study are included in the main text and Supplementary Information. Processed datasets can be founded in the github repository[46]. Other data are available from the corresponding author upon request.

Code availability

The code that supports the findings within this paper and other findings of this study is available at the github repository[46].

References

1. Guo, Y. et al. Deep learning for 3d point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**, 4338–4364 (2020).
2. Gallego, G. et al. Event-based vision: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 154–180 (2020).
3. Durrant-Whyte, H. F. *Integration, Coordination and Control of Multi-sensor Robot Systems*, Vol. 36 (Springer Science & Business Media, 2012).
4. Majumder, B. D., Roy, J. K. & Padhee, S. Recent advances in multi-functional sensing technology on a perspective of multi-sensor system: a review. *IEEE Sens. J.* **19**, 1204–1214 (2018).
5. Lin, Y., Zhang, Z., Tang, H., Wang, H. & Han, S. Pointacc: efficient point cloud accelerator. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 449–461 (ACM, 2021).
6. Yu, X. et al. Point-bert: pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19313–19322 (IEEE, 2022).
7. Chen, Y., Liu, J., Zhang, X., Qi, X. & Jia, J. Voxelnext: fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21674–21683 (IEEE, 2023).
8. Liu, Z., Tang, H., Lin, Y. & Han, S. Point-voxel CNN for efficient 3D deep learning. In *Advances in Neural Information Processing Systems* 32 (Curran Associates Inc., 2019).
9. Pedram, A., Richardson, S., Horowitz, M., Galal, S. & Kvatinsky, S. Dark memory and accelerator-rich system optimization in the dark silicon era. *IEEE Des. Test.* **34**, 39–50 (2016).

10. Lin, X. et al. E2pnet: event to point cloud registration with spatio-temporal representation learning. In *Advances in Neural Information Processing Systems* 36 (Curran Associates Inc., 2024).
11. Wu, W., Qi, Z. & Fuxin, L. Pointconv: deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9621–9630 (IEEE, 2019).
12. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme learning machine: theory and applications. *Neurocomputing* **70**, 489–501 (2006).
13. Huang, G., Huang, G.-B., Song, S. & You, K. Trends in extreme learning machines: a review. *Neural Netw.* **61**, 32–48 (2015).
14. Wan, W. et al. A compute-in-memory chip based on resistive random-access memory. *Nature* **608**, 504–512 (2022).
15. Cui, J. et al. Cmos-compatible electrochemical synaptic transistor arrays for deep learning accelerators. *Nat. Electron.* **6**, 292–300 (2023).
16. Rao, M. et al. Thousands of conductance levels in memristors integrated on cmos. *Nature* **615**, 823–829 (2023).
17. Zhao, H. et al. Energy-efficient high-fidelity image reconstruction with memristor arrays for medical diagnosis. *Nat. Commun.* **14**, 2276 (2023).
18. Lanza, M. et al. Memristive technologies for data storage, computation, encryption, and radio-frequency communication. *Science* **376**, eabj9979 (2022).
19. Le Gallo, M. et al. A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nat. Electron.* **6**, 680–693 (2023).
20. Langenegger, J. et al. In-memory factorization of holographic perceptual representations. *Nat. Nanotechnol.* **18**, 479–485 (2023).
21. Li, Z. et al. Asters: adaptable threshold spike-timing neuromorphic design with twin-column reram synapses. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*. 1099–1104 (IEEE, 2022).
22. Zheng, Q. et al. Accelerating sparse attention with a reconfigurable non-volatile processing-in-memory architecture. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 1–6 (IEEE, 2023).
23. Kumar, S., Wang, X., Strachan, J. P., Yang, Y. & Lu, W. D. Dynamical memristors for higher-complexity neuromorphic computing. *Nat. Rev. Mater.* **7**, 575–591 (2022).
24. Zhu, X., Wang, Q. & Lu, W. D. Memristor networks for real-time neural activity analysis. *Nat. Commun.* **11**, 2439 (2020).
25. Mannocci, P. et al. In-memory computing with emerging memory devices: status and outlook. *APL Mach. Learn.* **1**, 010902 (2023).
26. Jing, Z., Yan, B., Yang, Y. & Huang, R. VSDCA: a voltage sensing differential column architecture based on 1T2R RRAM array for computing-in-memory accelerators. *IEEE Trans. Circuits Syst. I Regul. Pap.* **69**, 4028–4041 (2022).
27. Zhai, Y., Li, B., Yan, B. & Wang, J. Star: AN EFFICIENt softmax engine for attention model with rram crossbar. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1–2 (IEEE, 2023).
28. Zhu, K. et al. Hybrid 2d–cmos microchips for memristive applications. *Nature* **618**, 57–62 (2023).
29. Isik, B. et al. Neural network compression for noisy storage devices. *ACM Trans. Embed. Comput. Syst.* **22**, 1–29 (2023).
30. Kaul, A. et al. 3-d heterogeneous integration of RRAM-based compute-in-memORY: IMPACT OF INtegration parameters on inference accuracy. *IEEE Trans. Electron Devices* **70**, 485–492 (2022).
31. He, W. et al. Prive: efficient RRAM programming with chip verification for RRAM-based in-memory computing acceleration. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1–6 (IEEE, 2023).
32. Zhou, F. et al. Optoelectronic resistive random access memory for neuromorphic vision sensors. *Nat. Nanotechnol.* **14**, 776–782 (2019).
33. Song, M.-K. et al. Recent advances and future prospects for memristive materials, devices, and systems. *ACS Nano* **17**, 11994–12039 (2023).
34. Yao, P. et al. Fully hardware-implemented memristor convolutional neural network. *Nature* **577**, 641–646 (2020).
35. Harabi, K.-E. et al. A memristor-based Bayesian machine. *Nat. Electron.* **6**, 52–63 (2023).
36. Qi, C. R., Su, H., Mo, K. & Guibas, L. J. Pointnet: deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 652–660 (IEEE, 2017).
37. Qi, C. R., Yi, L., Su, H. & Guibas, L. J. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*. 30 (Curran Associates Inc., 2017).
38. Chang, A. X. et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. Rep. arXiv:1512.03012 [cs.GR] (Stanford University—Princeton University—Toyota Technological Institute at Chicago, 2015).
39. Wu, Z. et al. 3d shapenets: a deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1912–1920 (IEEE, 2015).
40. Amir, A. et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7243–7252 (IEEE, 2017).
41. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. Preprint at *arXiv* https://doi.org/10.48550/arXiv.1708.07747 (2017).
42. Wang, Z. et al. Resistive switching materials for information processing. *Nat. Rev. Mater.* **5**, 173–195 (2020).
43. Li, H. et al. Memristive crossbar arrays for storage and computing applications. *Adv. Intell. Syst.* **3**, 2100017 (2021).
44. Lichtsteiner, P., Posch, C. & Delbruck, T. A 128 x 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. In *2006 IEEE International Solid State Circuits Conference—Digest of Technical Papers*. 2060–2069 (IEEE, 2006).
45. Wang, Q., Zhang, Y., Yuan, J. & Lu, Y. Space-time event clouds for gesture recognition: from rgb cameras to event cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 1826–1835 (IEEE, 2019).
46. Wang, S. Code for 'random resistive memory-based extreme point learning machine for unified visual processing.' https://github.com/wangsc1912/DEPLM/tree/onpublish (2024).

Acknowledgements

This research is supported by the National Key R&D Program of China (Grant No. 2022YFB3608300), the National Natural Science Foundation of China (Grant Nos. 62122004, 62374181, 62488201), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB44000000), Beijing Natural Science Foundation (Grant No. Z210006), Hong Kong Research Grant Council (Grant Nos. 27206321, 17205922, 17212923). This research is also partially supported by ACCESS—AI Chip Center for Emerging Smart Systems, sponsored by Innovation and Technology Fund (ITF), Hong Kong SAR.

Author contributions

Z.W. and S.W. conceived the work. Z.W., D.S., S.W., Y.G., and Y.L. contributed to the design and development of the models, software and the hardware experiments. S.W., Y.G., Y.Y., B.W., H.C., and W.Z. interpreted, analyzed and presented the experimental results. Z.W., D.S., S.W., and Y.G. wrote the manuscript. All authors discussed the results and implications and commented on the manuscript at all stages.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-025-56079-3.

Correspondence and requests for materials should be addressed to Zhongrui Wang or Dashan Shang.

Peer review information *Nature Communications* thanks Cheng Wang, and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

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

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

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2025