# Supplementary Information

## Memristive singular value decomposition

Chenchen Ding (丁辰辰)[1,†], Zhengwu Liu (刘正午)[1,2†,*], Yibei Zhang (张伊蓓)[3], Can Li (李灿)[1], Jianshi Tang (唐建石)[3], Bin Gao (高滨)[3], Hao Yu (余浩)[4,*], Huaqiang Wu (吴华强)[3,*], Ngai Wong (黄毅)[1,*].

[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China;

[2]Institute of Data and Information, Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China

[3]School of Integrated Circuits, Tsinghua University, Beijing, China;

[4]School of Microelectronics, Southern University of Science and Technology, Shenzhen, China;

[†]These authors contributed equally: Chenchen Ding, Zhengwu Liu;

[*]Corresponding authors:

zwliu@eee.hku.hk (Z.L.); yuh3@sustech.edu.cn (H.Y.); wuhq@tsinghua.edu.cn (H.W.); nwong@eee.hku.hk (N.W.).

This **Supplementary Information** includes the following contents:

**Supplementary Note 3.** Analysis of bit slicing in DPM.

**Supplementary Note 4.** Analysis of vector repetition and additional input bits in VPR.

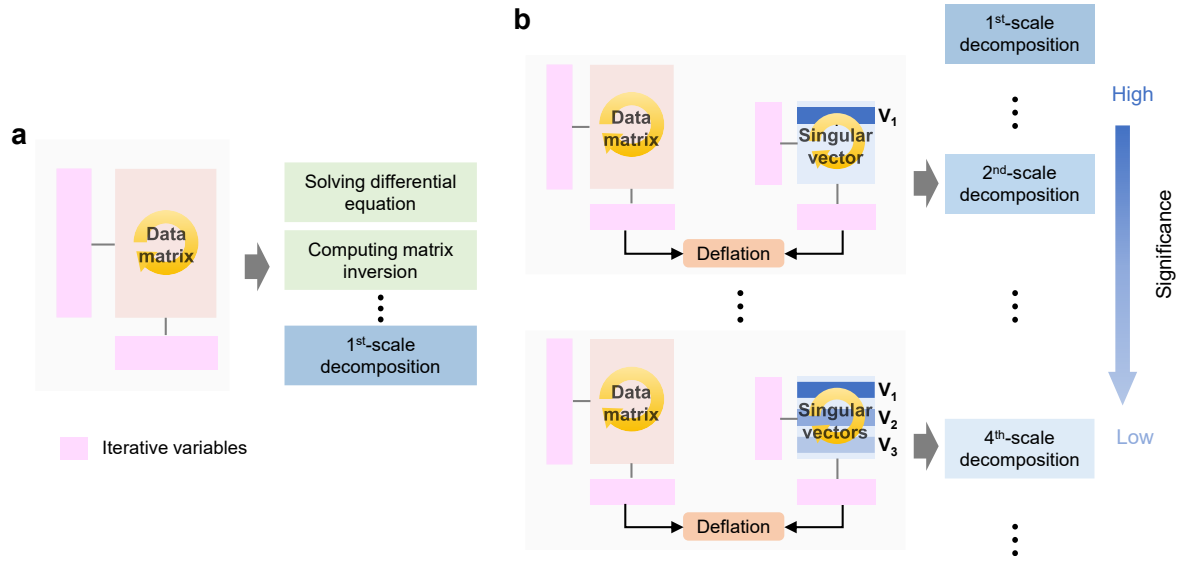
**Supplementary Note 5.** Conductance mapping with linear quantization.

**Supplementary Note 6.** Energy consumption evaluation.

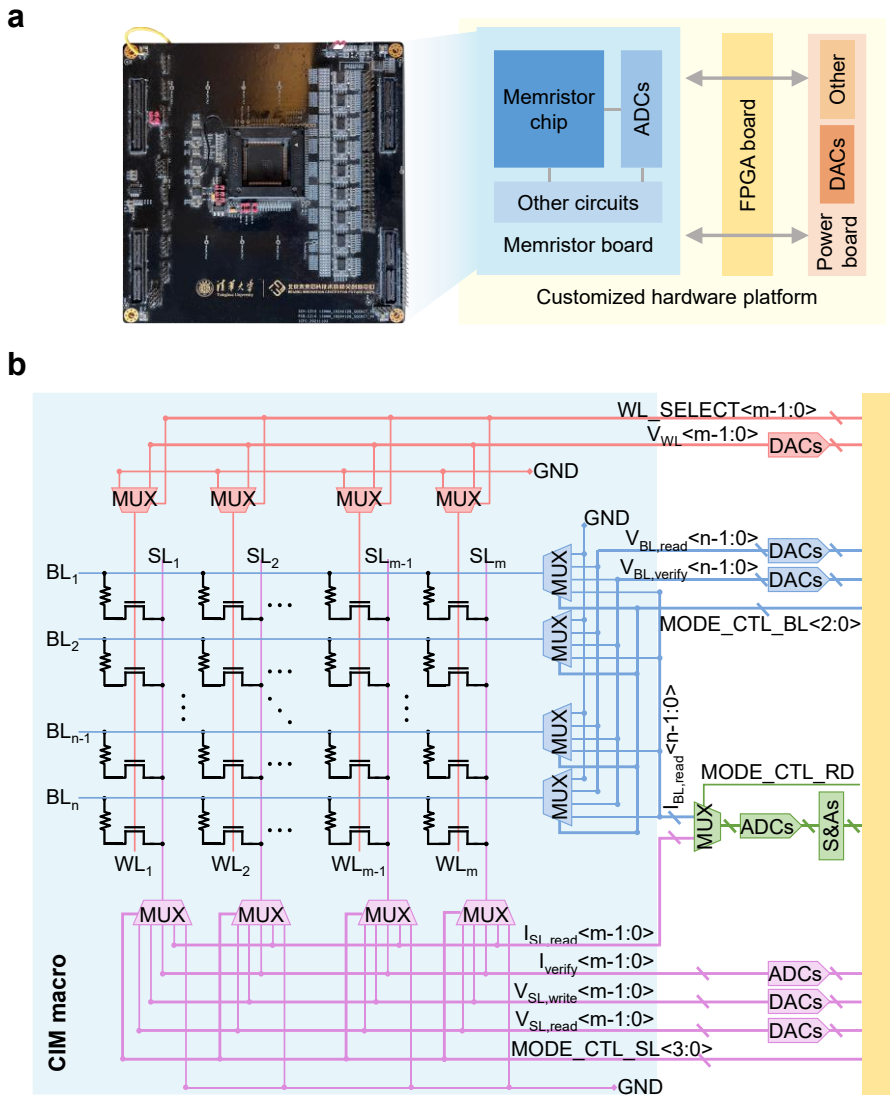
**Supplementary Note 7.** Technology scaling analysis of MSVD.

**Supplementary Note 8.** Analysis of normalized speed.

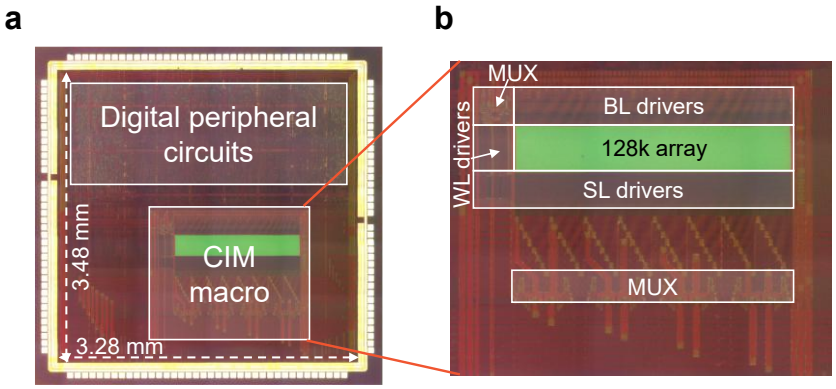
## Supplementary Figure



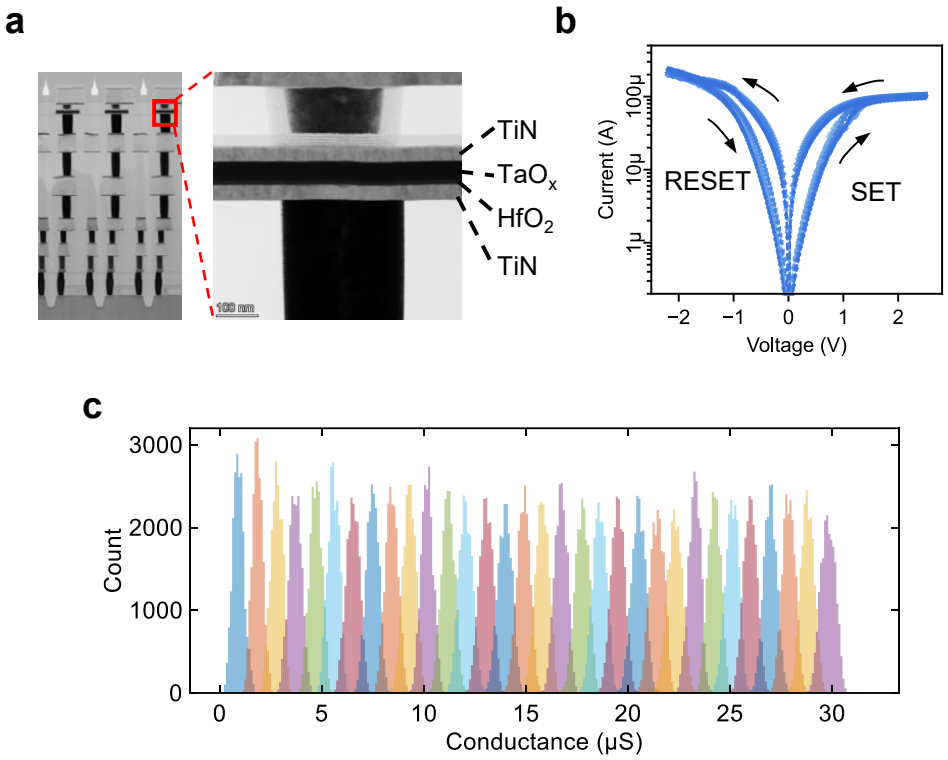

**Supplementary Figure 1. Comparison of dual-matrix iteration with single-matrix iteration. a,** Single-matrix iteration for conventional iterative algorithms such as solving differential equations and computing matrix inversions, which involve iteration on vectors within a single matrix framework. **b,** Dual-matrix iteration with deflation in SVD. SVD involves iterative coupling between the data matrix and previously determined singular vectors, where deflation progressively extracts singular components in order of decreasing significance (i.e., from large to small singular values).



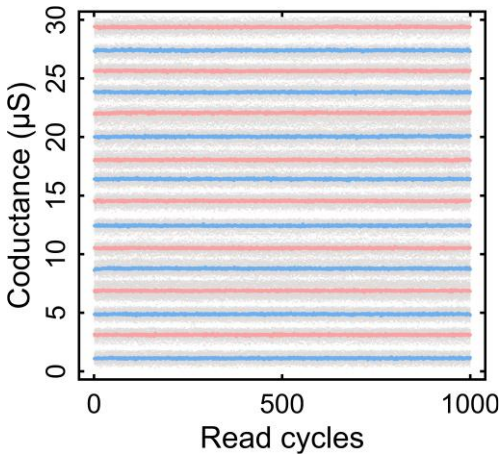

**Supplementary Figure 2. Image of the customized platform and architecture for MSVD.** **a,** Physical platform components. The platform contains three boards: memristor board, FPGA board and power board. The memristor board incorporates the memristor chip, analog-to-digital converters (ADCs) and other circuits. The FPGA (Xilinx Zynq UltraScale+ ZCU102 equipped with ARM A53 CPU) board is used for dataflow control. The power board includes digital-to-analog converters (DACs) and other circuits for powering the entire platform. BL, bit line; SL, source line; WL, word line; MUX, multiplexer; S&A: shift adder. **b,** Designed system architecture for MSVD implementation. The designed architecture combines memristor-based CIM macro for matrix computations with FPGA-based signal control and iteration management to enable complete MSVD processing. The MODE_CTL_BL/SL/RD signals are designed to facilitate operation selection and component coordination.



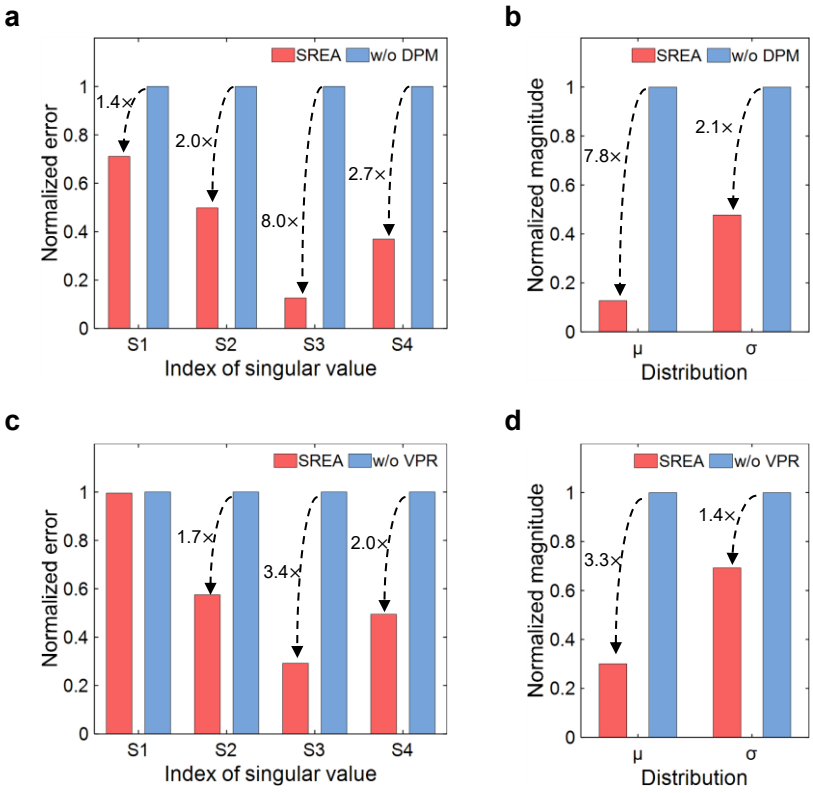

**Supplementary Figure 3. Circuit modules in the memristor chip. a,** Photograph of the 128k-cell memristor chip showing detailed locations of functional blocks. **b,** Detailed designs of compute-in-memory (CIM) macro in the chip. BL, bit line; SL, source line; WL, word line; MUX, multiplexer.



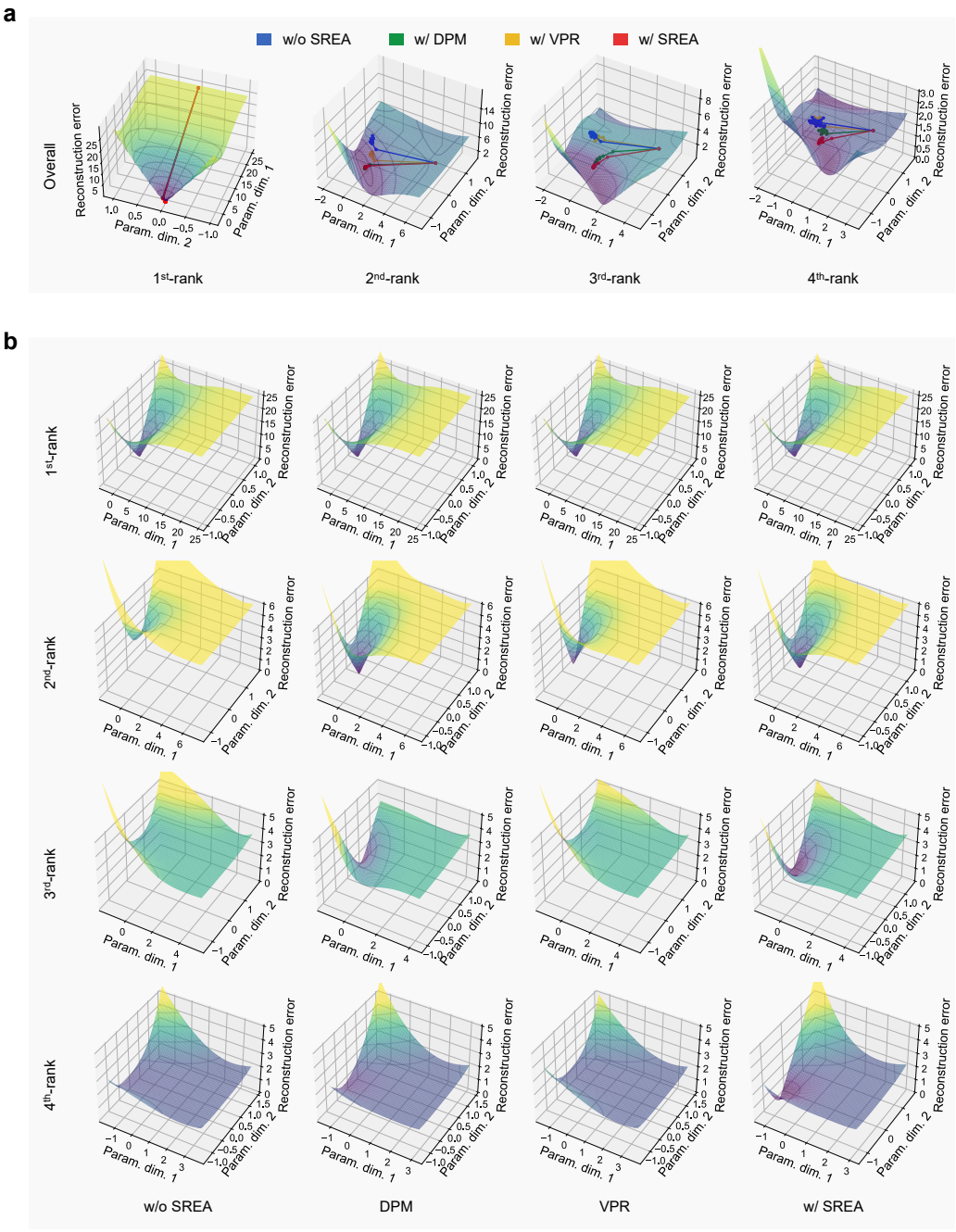

**Supplementary Figure 4. Memristor cell and its switching characteristics. a,** TEM photo of one-transistor-one-resistor cells forming the memristor array. Each cell comprises a memristor with a TiN/$TaO_x$/$HfO_2$/TiN material stack. **b,** Direct-current I-V curves during SET and RESET from 6 memristors, demonstrating excellent conductance switching characteristics. **c,** Measured conductance of memristors mapped to 32 uniformly distributed targets.



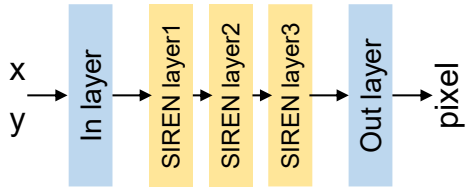

**Supplementary Figure 5. Detailed evolution of 16 conductance levels over 1000 read cycles.** Average values are plotted alongside individual measurements from 20 memristors per level, confirming high conductance stability.



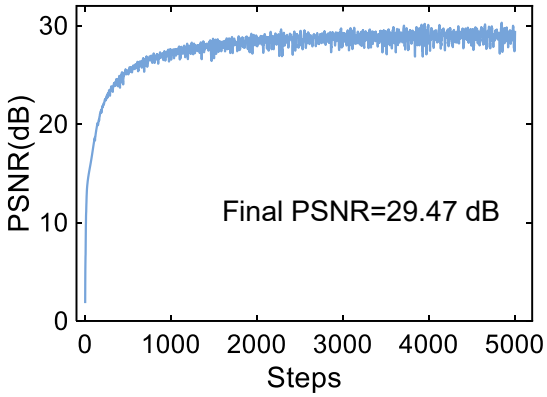

**Supplementary Figure 6. Ablation analysis of SREA components.** Individual component analysis across 20 repeated experiments evaluating separate contributions of dual-precision mapping (DPM) and vector processing refinement (VPR) within SREA. **a,** DPM effect on singular value accuracy. **b**, DPM effect on singular vector error distribution. **c,** VPR effect on singular value accuracy. **d,** VPR effect on singular vector error distribution. All panels show normalized error metrics comparing SREA with and without the respective component.



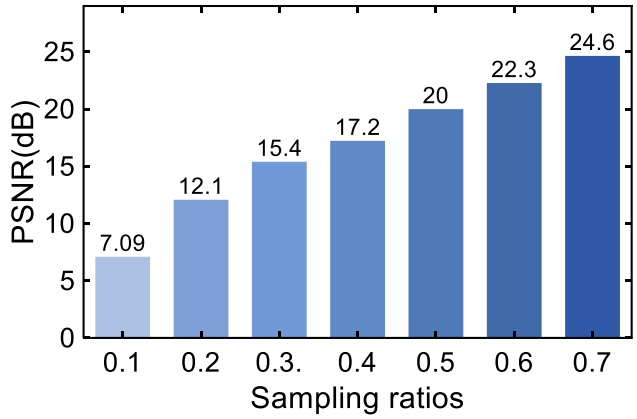

**Supplementary Figure 7. Reconstruction error landscape analysis. a,** Global landscape view showing iterative trajectories for different methods. **b,** Individual landscape views showing the progression of different methods for computation at each rank. Reconstruction error is quantified as the Frobenius norm of the difference between MSVD reconstructions and corresponding-rank ground truth.



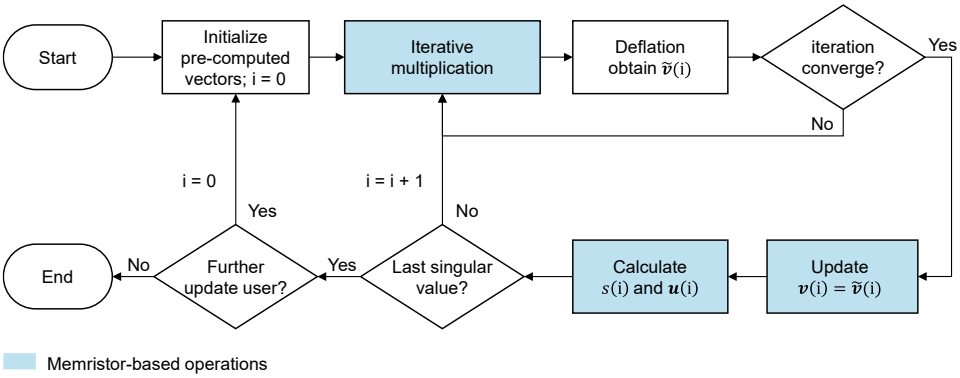

| Hidden dim | Lr | Optimizer |
|---|---|---|
| 80 | 1e-3 | Adam |




**Supplementary Figure 8. SIREN setup and PSNR with training steps.** The SIREN is set up with one input layer, one output layer and three hidden siren layers. The network is trained for 5000 steps and achieves a final PSNR of 29.47.

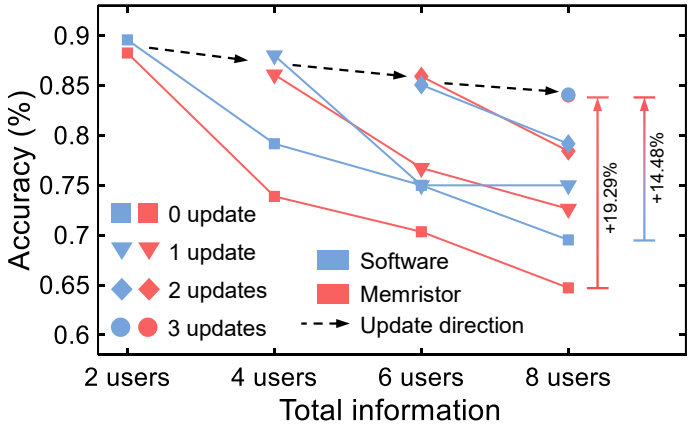



**Supplementary Figure 9. Result of compressed sensing with different sampling ratios.** Block-DCT with the block size of 16×16 is used as the sparse basis and Lasso regularization is applied as the reconstruction constraint.

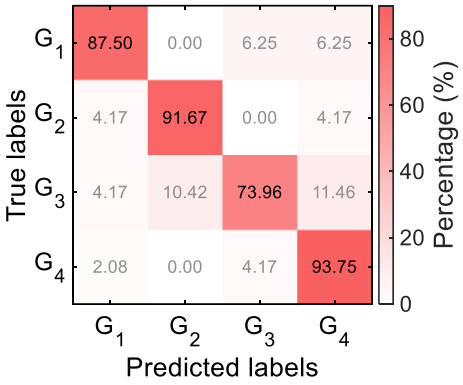



143

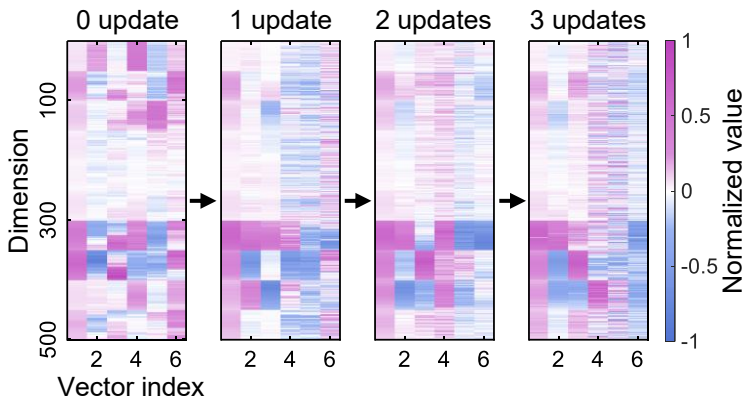
144 **Supplementary Figure 10. Flowchart of incremental MSVD for the user-scalable**
145 **recognition.** The computation process begins by specifying pre-computed vectors, then
146 iteratively computes each singular vector until convergence. The process terminates when all
147 target singular vectors are obtained and no additional user data will be incorporated.




**Supplementary Figure 11. Comparison of recognition accuracy evolution between software solutions and memristor-based solutions.**

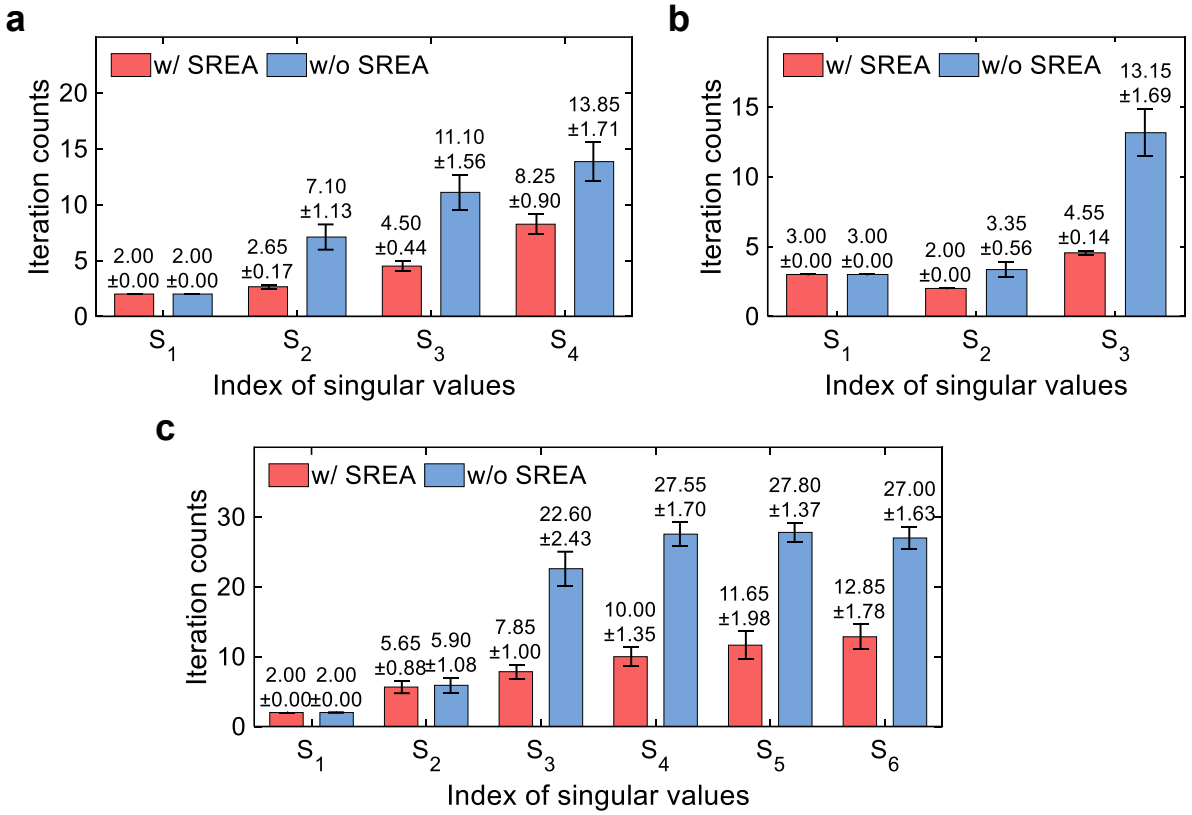



**Supplementary Figure 12. Confusion matrix of one typical result for incremental MSVD with 8-users' data.**



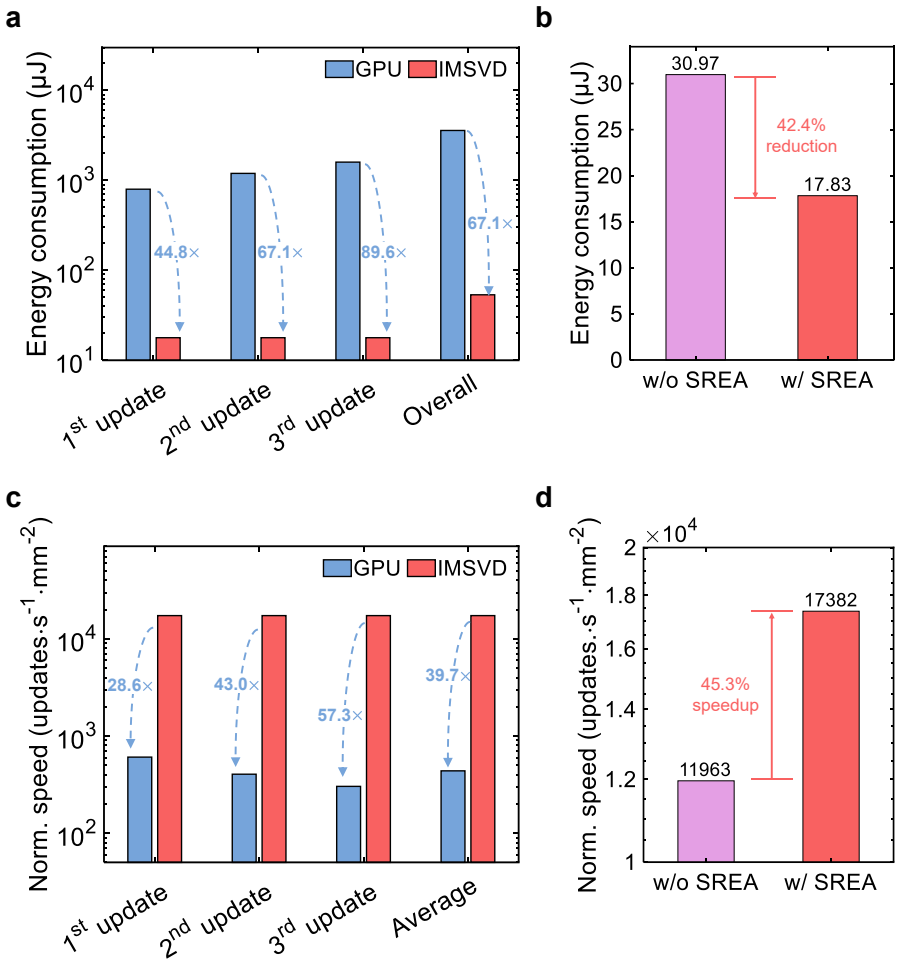

154

155 **Supplementary Figure 13. Singular vectors evolution with new data incorporation in 3**
156 **updates.**

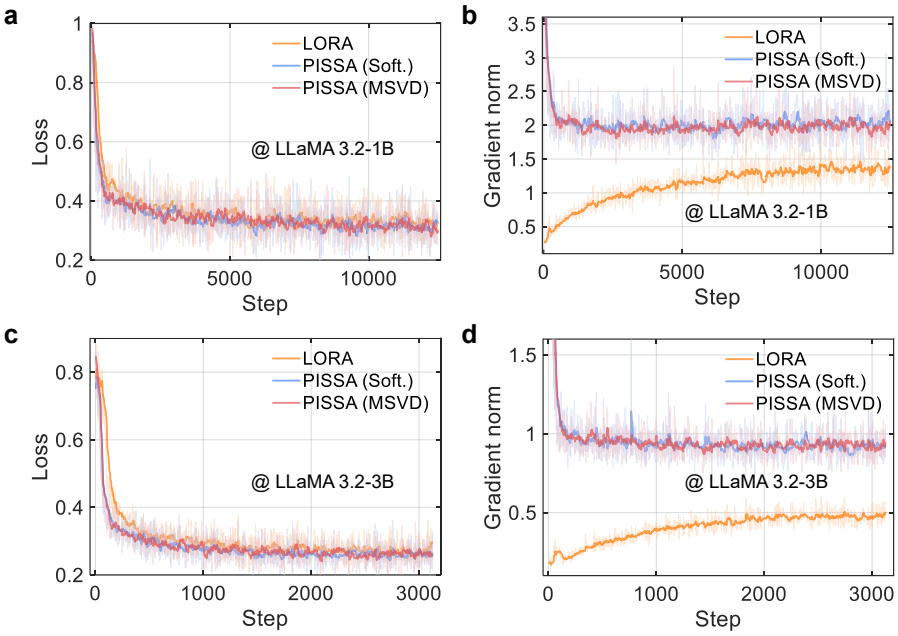



157

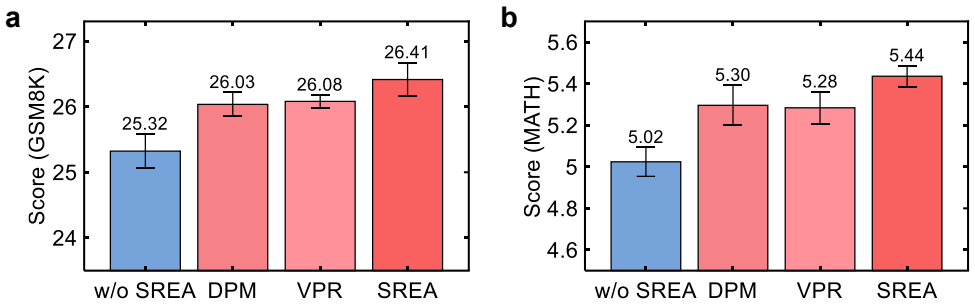
158 **Supplementary Figure 14. MSVD iteration analysis across datasets. a,** IRIS dataset. **b,**
159 COVID-19 dataset. **c,** NinaPro dataset. Results show the average number of iterations needed
160 for convergence across 20 repeated experiments. Convergence is reached when the relative
161 difference of the temporary squared singular value estimates between consecutive iterations
162 falls below 2%; a maximum limit of up to 30 iterations is imposed if convergence is not
163 achieved. MSVD with SREA demonstrates faster convergence compared to MSVD without
164 SREA across all datasets. Data are presented as mean ± standard error (n=20 independent runs).



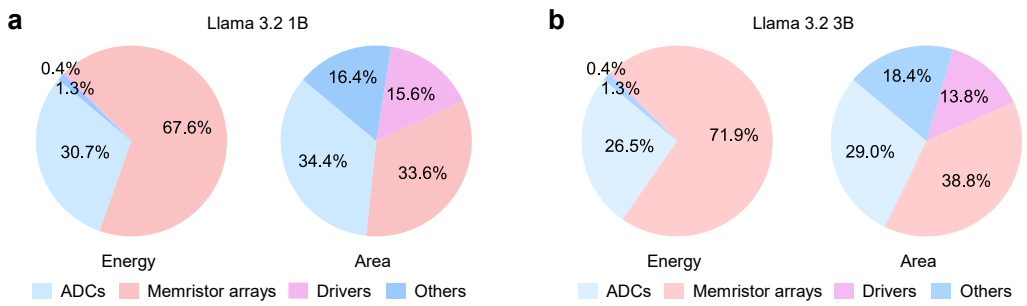

**Supplementary Figure 15. Performance comparisons of incremental implementations. a,** Energy consumption comparison between incremental MSVD with SREA and the GPU implementation across three sequential update steps. **b,** Energy consumption for each update step of incremental MSVD with and without SREA. **c,** Normalized speed comparison between incremental MSVD with SREA and the GPU implementation across three sequential update steps. **d,** Normalized speed for each update step of incremental MSVD with and without SREA.



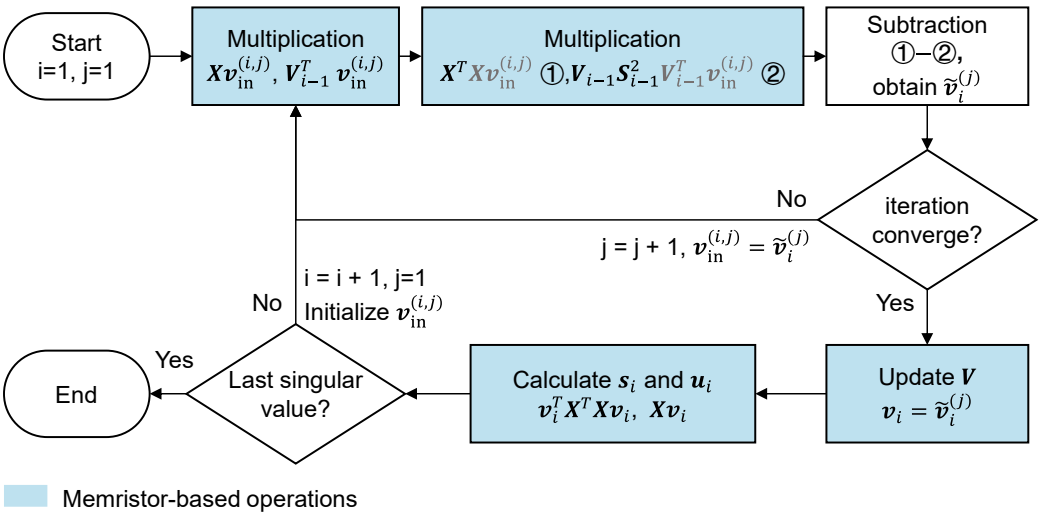

**Supplementary Figure 16. Training dynamics of LLaMA 3.2 models using different fine-tuning initialization approaches on MetaMathQA[1] dataset.** Training loss curves and gradient norms for LLaMA 3.2-1B (**a,b**) and 3.2-3B (**c,d**) models, respectively.



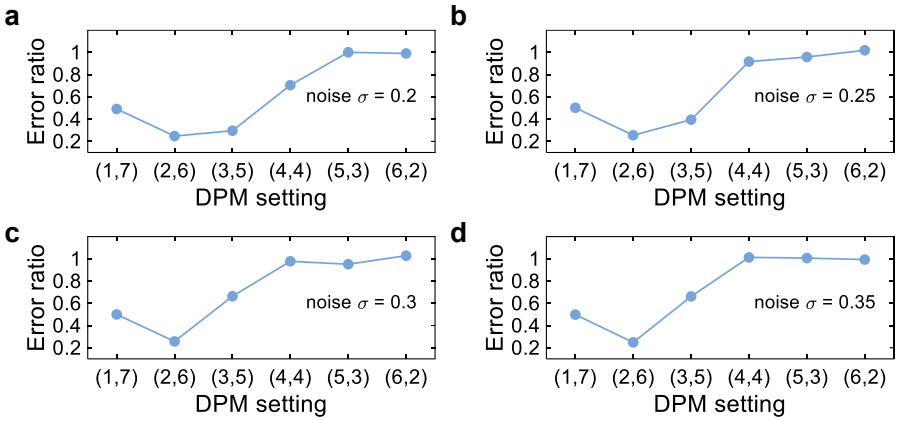

**Supplementary Figure 17. Ablation study of SREA components using LLaMA 3.2-1B.** Performance is evaluated on (**a**) the GSM8K dataset and (**b**) the MATH dataset during fine-tuning. The error bars indicate the standard error across 5 repeated experiments.




**Supplementary Figure 18. System-level energy and area breakdown for LLM weight initialization using MSVD. a, Llama 3.2 1B model.** Pie charts showing the proportion of (left) energy consumption and (right) area occupied by different hardware components. Components include ADCs, memristor arrays, Drivers, and other peripheral circuits. **b, Llama 3.2 3B model.** Energy and area breakdown for the same hardware components.

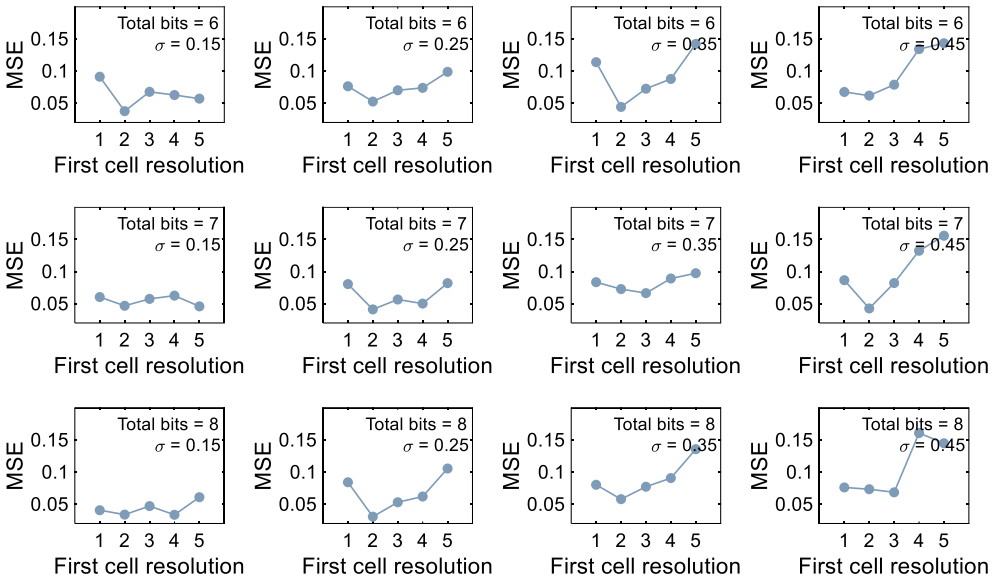



**Supplementary Figure 19. Flowchart of MSVD.** Grey parts in the equations indicate components obtained from previous calculation steps. The process terminates when all target singular vectors are obtained.

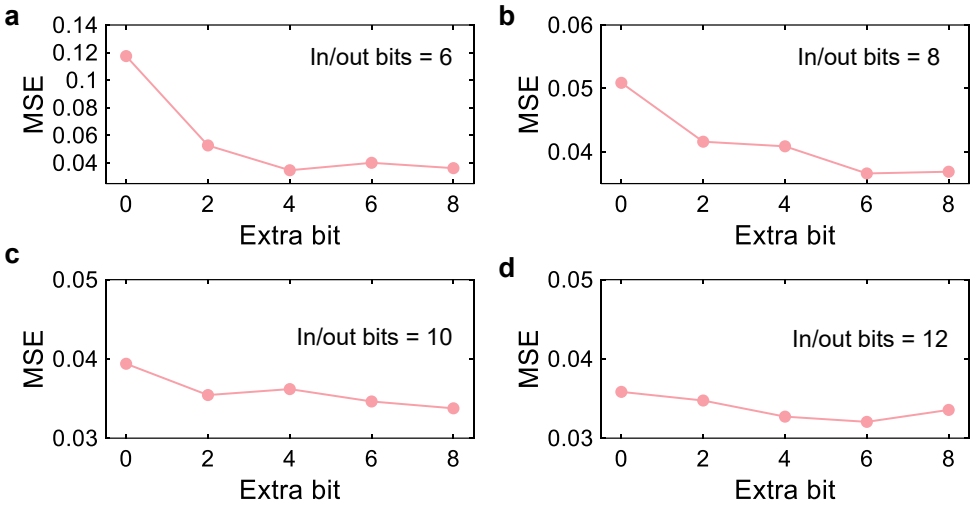



191

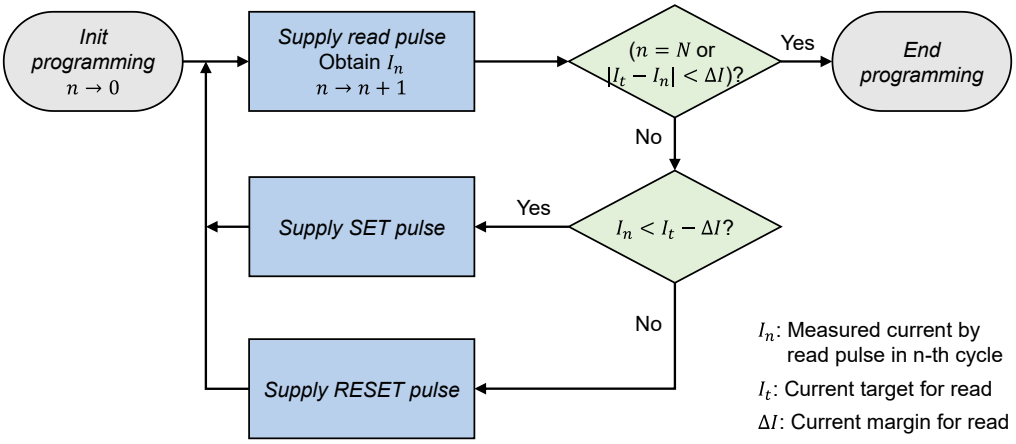
192 **Supplementary Figure 20. Ratio of VMM error with DPM to the error with direct**
193 **memristor mapping on a 128×128 memristor array.** The devices are simulated with
194 conductance variability of σ = 0.2 μS (**a**), 0.25 μS (**b**), 0.3 μS (**c**), and 0.35 μS (**d**) when target
195 maximum conductance is set to 30 μS. The input pulses are drawn from a uniform Bernoulli
196 distribution with equal probability of 0 and 1.




**Supplementary Figure 21. Error analysis of the DPM strategy on IRIS dataset with fixed total numerical resolution.** The Mean Squared Error (MSE) of the solved right singular vectors is shown across different DPM strategies under simulated noise conditions. Each experiment was repeated 10 times. The parameter σ controls the standard deviation of read noise (normal distribution) and write noise range (uniform distribution) in μS following the device behavior and write-verify characteristics[2].

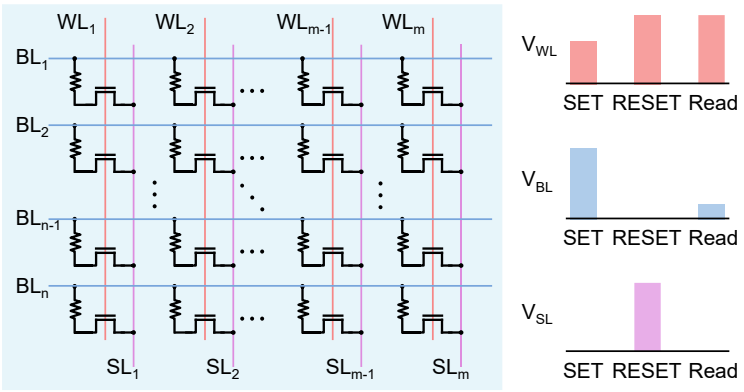



**Supplementary Figure 22. Error analysis of additional bits configurations in VPR**. Mean square error (MSE) of solved right singular vectors (RSVs) across different additional bit settings in VPR under different basic system resolution settings from (**a**) 6 bit, (**b**) 8 bit, (**c**) 10 bit to (**d**) 12 bit. Each experiment was repeated 20 times on IRIS dataset.




209

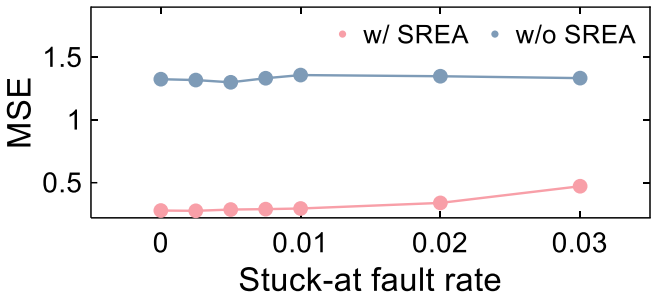
210 **Supplementary Figure 23. Workflow of programming with write and verify processes.**



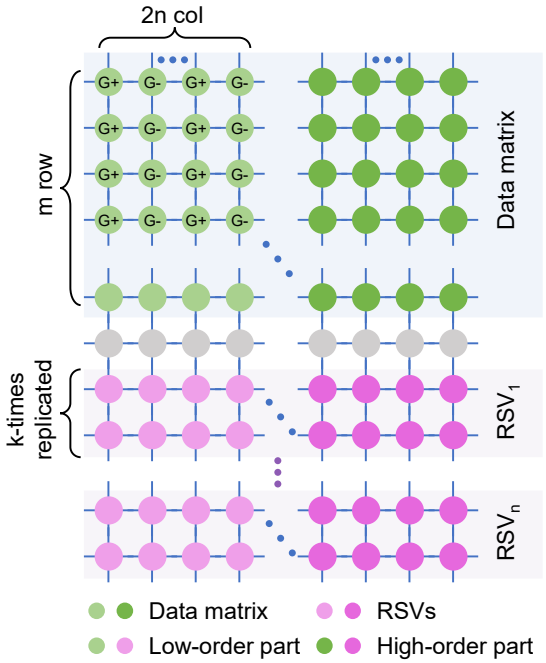

211

212 **Supplementary Figure 24. Relative voltage configurations applied to the WL, SL, and BL**
213 **of the selected cell during SET, RESET, and Read operations.**

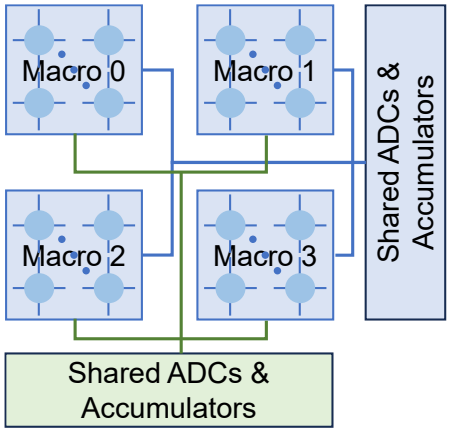
214



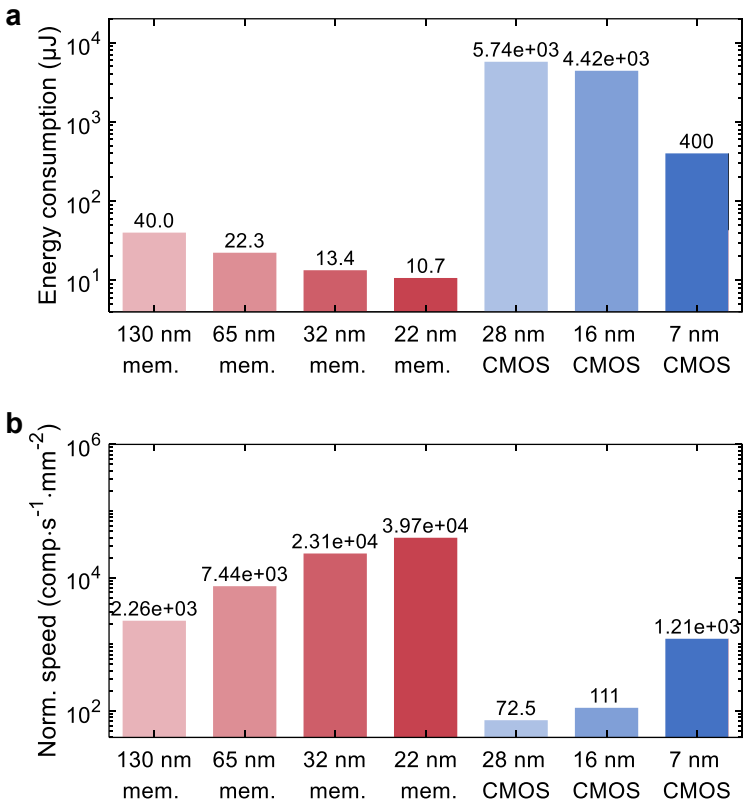

**Supplementary Figure 25. Mean squared error (MSE) of the four singular values obtained in simulation by MSVD with and without SREA on the IRIS dataset under different stuck-at fault rates.** Each simulation was repeated 20 times.



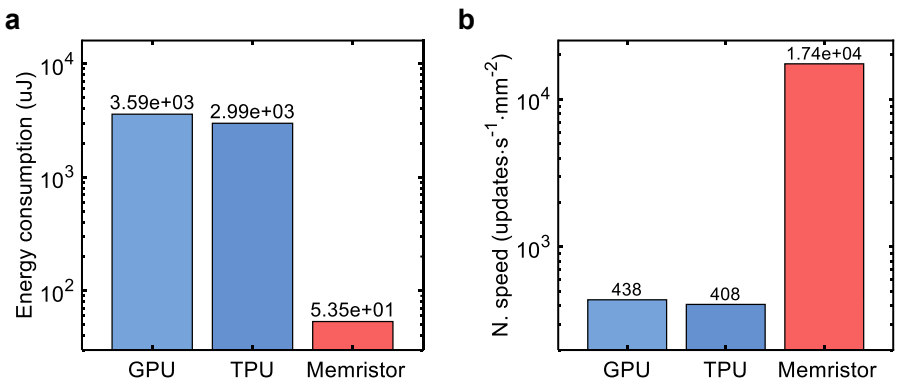

**Supplementary Figure 26. Mapping method of MSVD with SREA on the memristor array.** For a given $m \times n$ matrix, the data is mapped into positive and negative components. The DPM scheme splits the matrices into two parts, and the VPR method replicates each of the $i$ solved RSVs $k$ times on the array.



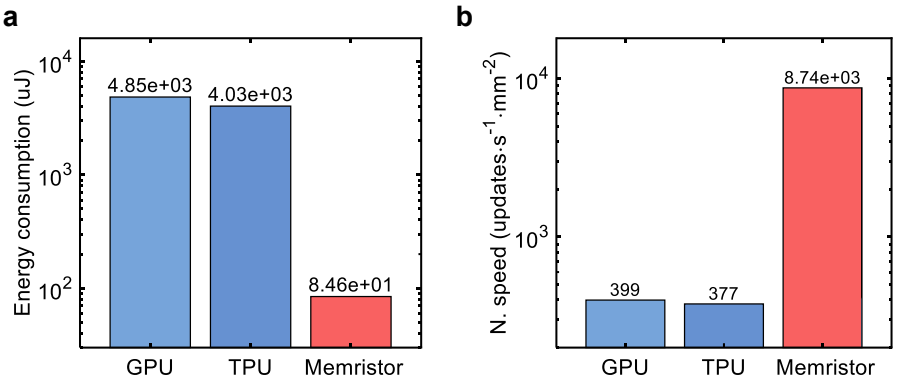

**Supplementary Figure 27. Multi-macro architecture for large-scale matrix deployment in MSVD.** Large matrices are partitioned into sub-blocks and mapped onto multiple macros. The final results are obtained through inter-macro accumulation or concatenation of partial outputs.




**Supplementary Figure 28. Benchmark of energy consumption and normalized speed of MSVD compared to the CMOS-based SVD under different technology node.** NVIDIA M40, P100 and A100 GPUs are used as 28nm CMOS, 16nm CMOS, 7nm CMOS baselines respectively.



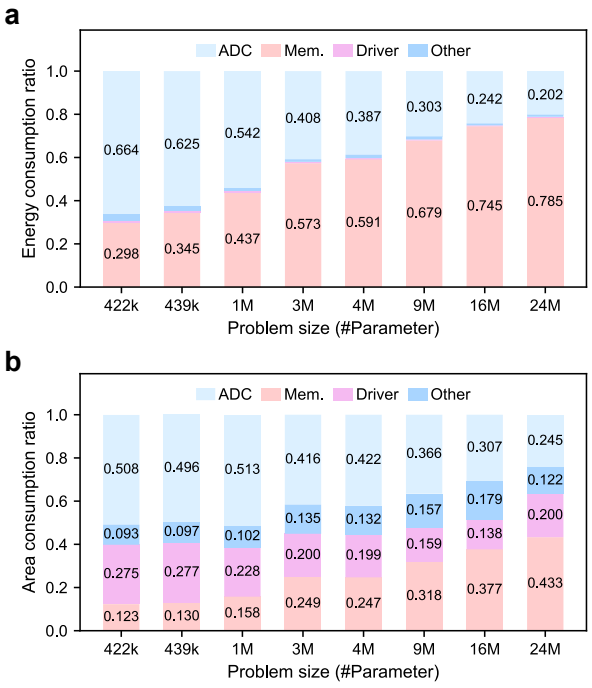

**Supplementary Figure 29. Benchmarking incremental MSVD against A100 GPU and Google TPUv3 for EMG-based gesture recognition. a,** Energy consumption during updates. **b,** Normalized speed. The TPUv3 chip area is estimated to be approximately 700 mm² based on literature[3].




240

241 **Supplementary Figure 30. Benchmarking incremental MSVD against A100 GPU and**
242 **Google TPUv3 for EEG-based sleep stage detection. a,** Energy consumption during updates.
243 **b,** Normalized speed. The TPUv3 chip area is estimated to be approximately 700 mm² based
244 on literature[3].

245




**Supplementary Figure 31. Memristor array, ADC, driver, and other peripheral circuit contributions to total system energy and area across tasks of different scales.** The 422k and 439k problem sizes correspond to the EMG-based gesture recognition and EEG-based sleep stage detection tasks, respectively, while the 1M to 24M problem sizes correspond to weight matrices from the LLaMA 3.2 1B and 3B models. **a, Energy breakdown.** Bar chart showing the proportion of total system energy consumed by the memristor arrays, ADCs, drivers, and other peripheral circuits across eight representative MSVD workloads spanning a wide range of scales. **b, Area breakdown.** Bar chart showing the proportion of total system area occupied by the memristor arrays, ADCs, drivers, and other peripheral circuits for the same eight workloads.

## Supplementary Table

**Supplementary Table 1. Comparison with some representative memristor-based systems[4-13].**

| Works | Application | Iterative solving? | Iteration property | Precision optimization method | Input scalable? | Energy efficiency* | Efficiency advancement compared to A100 GPU[#] |
|---|---|---|---|---|---|---|---|
| Nat. Electron. 2018 | Differential equation solving | Yes | Single-matrix single-scale | Uniform bit division | No | 34× (V100 GPU) | 20× |
| PNAS 2019 | Eigenvector solving | Yes | single-matrix, single-scale | Analog feedback | No | - | - |
| Sci. China Inf. Sci. 2021 | Neural network | No | - | Spatial extended allocation | No | - | - |
| IEEE Nano. Mag. 2022 | Principle component analysis | Yes | Single-matrix, two-scale | - | No | 250× (RTX8000 GPU) | 35× |
| IEEE ISCAS 2022 | Spectral centrality analysis | Yes | Single-matrix, single-scale | - | No | 15× (TPU) | 47× |
| Nat. Commun. 2023 | Image reconstruction | No | - | Quasi-analog mapping | No | 153× (V100 GPU) | 39× |
| Sci. Adv. 2023 | Differential equation solving | Yes | Single-matrix single-scale | Hybrid precision task allocation | No | 85× (ASIC, sparse) | 30× (Sparse) |
| Science 2024 | Differential equation solving | Yes | Single-matrix, single-scale | Analog slicing | No | 22× (ASIC, VMM only) | 103× (VMM only) |
| Sci. Adv. 2025 | Neural ODE solving | Yes, | Layer-level iterative | Continuous-time computing | No | 674× (GTX285 GPU) | 6.0× |
| Nat. Commun. 2025 | Neural network | No | - | Layer ensemble averaging | No | - | - |
| **This work** | **SVD** | **Yes** | **Dual-matrix, multi-scale** | **SREA (DPM+VPR)** | **Yes, incremental MSVD** | **65× (A100 GPU)** | **65×** |

*Energy efficiency advancement reported by paper

[#] Results versus other CMOS circuits are shifted by compute precision to match the NVIDIA A100 GPU comparison for a more direct comparison

**Supplementary Table 2. Comparisons of DPM and VPR to the generic bit-slicing techniques[14].**

| | Positional bit slicing (Le Gallo et al.) | DPM | Equal-fill bit-slicing (Le Gallo et al.) | VPR |
|---|---|---|---|---|
| Strategy | Uniform slice, e. g., [4,4] | Non-uniform slice, e. g., [2,6] | All component repetition | RSVs-only repetition & input-precision enhancement |
| High-order-error-amplification aware | No | Yes | -- | -- |
| Input-range aware | -- | -- | No | Yes |
| Optimization level | Operation (VMM) | Task (MSVD) | Operation (VMM) | Task (MSVD) |
| Optimization goal | Precision (If fine-grained) | Precision-cost balance (Dual cell) | Precision | Precision-cost balance (RSVs-only) |

**Supplementary Table 3. SREA settings in experiments.**

| TASK | DPM | VPR | | | Iteration limit |
|---|---|---|---|---|---|
| | | Basic in/out bits | Additional bits | Vector rpt. times | |
| IRIS | [2,5] | 8 | 6 | 4 | 20 |
| HKU-LOGO | [3,5] | 8 | 4 | 4 | 20 |
| COVID-19 | [2,5] | 8 | 8 | 4 | 20 |
| NinaPro | [3,5] | 8 | 10 | 4 | 30 |
| Sleep-EDFx | [3,5] | 8 | 6 | 16 | 15 |
| LoRA | [2,5] | 10 | 6 | 24 | 15 |

**Supplementary Table 4. Matrix dimensions and normalized singular value gap ranges across datasets.**

| Dataset | Matrix dimension | Parameter scale | Norm. singular value gap range |
|---|---|---|---|
| IRIS | 150×4 | ~100 | 0.006-0.94 |
| COVID-19 | 104×51 | ~1k | 0.03-0.90 |
| NinaPro | 864×500 | ~100k | 0.002-0.91 |

**Supplementary Table 5. Fine-tuning settings (a) and weight dimensions (b) of LLaMA 3.2 models**

**a**

| Param. name | Rank | Iteration number | LoRA alpha | Token length | Learning rate | Steps (Batch size) | Device |
|---|---|---|---|---|---|---|---|
| LLaMA-3.2-1B | 4 | 15 | 4 | 1024 | 2E-5 | 12500 (4) | A100 ×1 |
| LLaMA-3.2-3B | 4 | 15 | 4 | 1024 | 2E-5 | 3125 (8) | A100 ×2 |

**b**

| Layer | Query layer | Key layer | Value layer | Output projection | Up projection | Gate layer | Down projection |
|---|---|---|---|---|---|---|---|
| LLaMA-3.2-1B | 2048×2048 | 2048×512 | 2048×512 | 2048×2048 | 2048×8192 | 2048×8192 | 8192×2048 |
| LLaMA-3.2-3B | 3072×3072 | 3072×1024 | 3072×1024 | 3072×3072 | 3072×8192 | 3072×8192 | 8192×3072 |

**Supplementary Table 6. Energy benchmark of MSVD.**

| Operations | w/ SREA (μJ) | | w/o SREA (μJ) | |
|---|---|---|---|---|
| Update singular vectors | 0.61 | 0.20 (Write)<br>0.41 (Verify) | 4.61 | 0.33 (Write)<br>4.28 (Verify) |
| Multiplications on data matrix | 8.86 | | 17.00 | |
| Multiplications on pre-computed vectors (IMSVD only) | 4.43 | | 4.93 | |
| Multiplications on vectors to be computed | 3.58 | | 4.08 | |
| Other auxiliary operations | 0.35 | | 0.35 | |
| **Sum** | MSVD | IMSVD | MSVD | IMSVD |
| | 13.40 | 17.83 | 26.04 | 30.97 |

IMSVD: incremental MSVD

**Supplementary Table 7. Comparison of MSVD to other CMOS-based ASICs designed for decomposition applications[15-18].**

| Works | Appl. | Architecture | Benchmark tech. node (nm) | Data precision | Problem scale | Decomposition efficiency* ($Vec \cdot uJ^{-1}$) | Norm. decomposition efficiency# ($Elem \cdot uJ^{-1}$) | Precision norm. efficiency† ($Elem \cdot bit \cdot uJ^{-1}$) | Decomposition Speed* ($kVec \cdot s^{-1} \cdot mm^{-2}$) | Norm. decomposition speed# ($kElem \cdot s^{-1} \cdot mm^{-2}$) | Precision norm speed† ($kElem \cdot bit \cdot s^{-1} \cdot mm^{-2}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TCAS I 2019 | SVD | CMOS-based von Neumann | 40 | 19 | 8×8 | 3.53 | 28.2 | 536 | - | - | - |
| TCAS I 2023 | EVD | CMOS-based von Neumann | 28 | 32 | 64×64 | 0.018 | 1.15 | 36.8 | $7.4\times10^{-1}$ | $4.7\times10^{1}$ | $1.5\times10^{3}$ |
| TCAS I 2024 | SVD | CMOS-based von Neumann | 90 | 22 | 16×8×8 (Tensor) | 0.33 | 2.64 | 58.1 | $5.9\times10^{0}$ | $4.7\times10^{1}$ | $1.0\times10^{3}$ |
| Microelectron. J. 2025 | EVD | CMOS-based von Neumann | 22 | 10 | 1024×1024 | 0.0048 | 4.92 | 49.2 | $1.5\times10^{-1}$ | $1.5\times10^{2}$ | $1.5\times10^{3}$ |
| **This work** | **SVD** | **Memristor-based CIM** | **32** | **[3, 5] DPM** | **864×500** | **0.43** | **215** | **1935** | $\mathbf{1.3\times10^{2}}$ | $\mathbf{6.5\times10^{4}}$ | $\mathbf{5.9\times10^{5}}$ |

SVD: singular value decomposition; EVD: eigen value decomposition
*The decomposition efficiency and speed are evaluated using the reported energy efficiency, latency, and area.
#To provide a fair comparison, we multiply each metric by the corresponding vector size to obtain the element-wise efficiency and speed.
†To provide a fair comparison with precision, we multiply each metric by the corresponding vector size and data precision to obtain the element-wise efficiency and speed with bit length.

**Supplementary Table 8. Area analysis of MSVD implementations. a,** Area breakdown for MSVD. **b,** Area breakdown for incremental MSVD.

**a**

| MSVD | Area ($mm^2$) | |
|---|---|---|
| | With SREA | Without SREA |
| Array | 0.029 | 0.014 |
| MUX | 0.013 | 0.007 |
| ShiftAdd | 0.009 | 0.004 |
| SL drivers | 0.022 | 0.011 |
| WL drivers | 0.013 | 0.006 |
| BL drivers | 0.030 | 0.015 |
| ADCs | 0.120 | 0.172 |
| **SUM** | **0.236** | **0.229** |

**b**

| IMSVD | Area ($mm^2$) | |
|---|---|---|
| | With SREA | Without SREA |
| Array | 0.030 | 0.014 |
| MUX | 0.014 | 0.007 |
| ShiftAdd | 0.011 | 0.006 |
| SL drivers | 0.029 | 0.014 |
| WL drivers | 0.019 | 0.010 |
| BL drivers | 0.038 | 0.019 |
| ADCs | 0.171 | 0.228 |
| **SUM** | **0.313** | **0.298** |

SL: source line; WL: word line; BL: bit line; ADC: analog-to-digital converter
IMSVD: incremental MSVD

**Supplementary Table 9. Measured area of the analog macro in the fabricated 130 nm chip and its scaled area at 32 nm**

| Analog macro | Area ($mm^2$) | |
|---|---|---|
| | 130 nm | 32 nm |
| Array | 0.31 | 0.019 |
| MUX | 0.30 | 0.019 |
| SL drivers | 0.26 | 0.016 |
| WL drivers | 0.03 | 0.002 |
| BL drivers | 0.28 | 0.018 |
| **SUM** | **1.18** | **0.074** |

**Supplementary Note 1. Analysis of error landscape on IRIS dataset.**

The iterative computation process of power iteration deflation is inherently dynamic, where each step's result influences the subsequent computation, while the data matrix remains constant throughout the iteration. Unlike neural networks, where convergence is driven by loss-based weight updates[19], power iteration converges through the inherent iterative characteristics of solving for each corresponding rank. In this process, the previous iteration's result acts more like a "weight" for the next computation. Thus, we applied PCA to reduce the dimensionality of the iterative solutions (concatenating **U**, **S**, **V** obtained at each iteration) into two orthogonal directional vectors, $\mathrm{Vec}_{\mathrm{dim1}}$ and $\mathrm{Vec}_{\mathrm{dim2}}$. We then sampled along these two directions as $\alpha \times \mathrm{Vec}_{\mathrm{dim1}} + \beta \times \mathrm{Vec}_{\mathrm{dim2}}$ (where $(\alpha, \beta)$ is the sampling point in the space after PCA), and calculated the corresponding reconstruction error at each sampled point to form a 3D error landscape (**Fig. 3f** and **Supplementary Figure 7b**).

This landscape visualizes how system noise affects solution quality. During each solving iteration, the prominence of the current largest singular value relative to the residual facilitates convergence, while noise tends to "flatten" the solving process (i.e., the enhancement of the dominant singular value during computation becomes obscured by errors). The gradient of this landscape reflects the prominence of the singular value being solved for the corresponding rank in the actual solving process, as well as the impact of system noise on iteration difficulty. This provides an intuitive visualization of the difficulty in accurately solving for each rank under the current configuration and the ease of convergence. Specifically, the MSVD landscape with SREA optimization exhibits steeper gradients, indicating reduced noise influence during the iterative process and facilitating convergence.

To facilitate direct comparison of convergence paths across different configurations, which may converge along varying directional vectors, we constructed a global landscape (**Supplementary Figure. 7a**). This visualization combines overall iterative trajectories across different configurations, allowing for a direct comparation of the solution paths under different methods. Notably, the iteration with SREA optimization follows a more direct and efficient path, resulting in faster practical convergence.

**Supplementary Note 2. Analysis of minimum detectable singular values.**

For the minimum resolvable singular value, we analyze the system from the perspective of error. The dominant error sources are quantization error and device noise. For a signed mapped numerical matrix taking the [a,b] bit slicing, the quantization error can be modeled as a uniform distribution of $U\left(-\frac{M}{2^{a+b+1}}, \frac{M}{2^{a+b+1}}\right)$, leading to an overall standard deviation $\sigma_q = \frac{M}{2\sqrt{3}\cdot 2^{a+b}}$, where $M$ denotes the maximum value of the matrix entries.

Using a memristor programming range of $G_{max} = 30$ μS and an equivalent noise level of $\sigma_r = 0.3$ μS as an example, the system-level numerical error (without SREA) is approximately $\sigma_{\boldsymbol{X}} = \frac{\sigma_r M}{G_{max}}$. With SREA optimization, the error on the numerical matrix can be reduced to $\sigma_{\boldsymbol{X},\mathrm{SREA}} = \left(\frac{\sigma_r M}{G_{max}\cdot 2^{\mathrm{a}}}\right)$. Moreover, for vector operations, the numerical error can be further reduced to $\sigma_{\boldsymbol{v},\mathrm{norm,SREA}} = \left(\frac{\sigma_r M}{G_{max}\cdot 2^{\mathrm{a}}\cdot\sqrt{\#\mathrm{RPT}}}\right)$, where #RPT denotes the repeated number of mapped vectors. For practical deployment configurations (e.g., $a = 2$ or 3, and $b = 5$), we observe that for MSVD without SREA, the overall system error is dominated by device noise associated with the numerical matrix, based on the relative magnitudes of the contributing error sources. For MSVD with SREA, the impacts of device noise and quantization error are comparable in magnitude.

Here, we first analyze the case where device noise is the dominant source of system error. Consider a ground-truth numerical matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ and an error matrix $\boldsymbol{T}$. Because the properties of $\boldsymbol{X}$ affect the stability of the SVD solution, we assume that $\boldsymbol{X}$ is not ill-conditioned and that its entries follow an i.i.d. Gaussian distribution $\mathcal{N}(0, \sigma_{\mathrm{data}}^2)$. By the Gaussian extreme-value approximation, the maximum entry magnitude satisfies: $M = \sigma_{\mathrm{data}}\sqrt{2\ln(mn)} + O\left(\frac{\sigma_{\mathrm{data}}\ln(\ln(mn))}{\sqrt{\ln(mn)}}\right)$. The spectral norm[20] of $\boldsymbol{T}$ is $\|\boldsymbol{T}\|_2 = \Theta\left(\sigma_{\boldsymbol{X},\mathrm{SREA}}\left(\sqrt{m}+\sqrt{n}\right)\right)$, and for $\boldsymbol{X}$, it is $\|\boldsymbol{X}\|_2 = \Theta\left(\sigma_{\mathrm{data}}\left(\sqrt{m}+\sqrt{n}\right)\right)$. For typical problem sizes like $m, n \in [10,5000]$, we have $\sqrt{2\ln(mn)} \in [3.03,5.84]$, which can be treated as an approximate constant in common workloads. Therefore, for MSVD with SREA at practical scales, we obtain:

$$\frac{\|\boldsymbol{T}\|_2}{\|\boldsymbol{X}\|_2} = \Theta\left(\frac{\sigma_r}{G_{max}\cdot 2^{a}}\right). \quad (1)$$

Wedin's bound[21] characterizes how numerical perturbations affect the subspaces of the SVD solution. Specializing this result to the estimation of the $i$-th singular value (and its associated singular subspace), the angle between the computed vector and the true vector can be bounded as:

$$\sin\theta \leq \frac{\|\boldsymbol{T}\|_2}{\delta}, \tag{2}$$

where $\theta$ is the angle of i-th solved vector to the ground truth, $\delta$ is the local spectral gap given by $\delta = s_i - s_{i+1}$, $s_i$ is the i-th singular value. Therefore, for the solution to be meaningful, we require $\|\boldsymbol{T}\|_2 < s_i - s_{i+1}$, i.e., the perturbation magnitude must be smaller than the local spectral gap. Since $\|\boldsymbol{X}\|_2 = s_1$ by definition, we have the criterion for reliable solving:

$$\frac{s_i - s_{i+1}}{s_1} > \Theta\left(\frac{\sigma_r}{G_{max} \cdot 2^a}\right). \tag{3}$$

Analogously, for the case where quantization error is the dominant error source, a parallel derivation yields:

$$\frac{s_i - s_{i+1}}{s_1} > \Theta\left(\frac{1}{2\sqrt{3} \cdot 2^{a+b}}\right) \tag{4}$$

Since device noise and quantization error are statistically independent, the two error contributions combine in quadrature, giving a unified joint criterion:

$$\frac{s_i - s_{i+1}}{s_1} > \sqrt{\Theta\left(\frac{\sigma_r}{G_{max} \cdot 2^a}\right)^2 + \Theta\left(\frac{1}{2\sqrt{3} \cdot 2^{a+b}}\right)^2} \tag{5}$$

For a sharper quantitative bound, if the perturbation angle is required to be within a target value $\theta'$, i.e., $\sin(\theta) < \sin(\theta')$ with $\theta, \theta' \in (0°, 90°)$, the corresponding requirement on the normalized spectral gap can be expressed as

$$\frac{s_i - s_{i+1}}{s_1} > \frac{1}{\sin(\theta')} \cdot \frac{\|\boldsymbol{T}\|_2}{\|\boldsymbol{X}\|_2} \tag{6}$$

It should be noted that, although the spectral norm $\|\boldsymbol{X}\|_2$ of the data matrix can be readily obtained for a given problem, the spectral norm of the error matrix $\|\boldsymbol{T}\|_2$ varies throughout the actual computation and therefore cannot be assigned a fixed value. We thus estimate the ratio $\|\boldsymbol{T}\|_2/\|\boldsymbol{X}\|_2$ by averaging over multiple simulations to provide a representative reference.

In practice, for high-accuracy objectives such as reconstruction or low-rank approximation, the truncation point can be chosen conservatively such that $\frac{s_i - s_{i+1}}{s_1} \gg \sqrt{\Theta\left(\frac{\sigma_r}{G_{max} \cdot 2^a}\right)^2 + \Theta\left(\frac{1}{2\sqrt{3} \cdot 2^{a+b}}\right)^2}$ (e.g., 0.096 vs. $\sqrt{\Theta(0.0025)^2 + \Theta(0.0023)^2}$ for COVID-19 data low-rank approximation). Based on ten repeated simulations, we estimate that under the experimental configuration (**Supplementary Table 3**) with a memristor conductance noise of $\sigma_r = 0.3$ μS, the effective spectral norm ratio $\|\boldsymbol{T}\|_2/\|\boldsymbol{X}\|_2$ in the computation is approximately 0.0065±0.0002. To obtain reasonably accurate results, e.g., constraining the deflection angle between the last resolved singular vector and its ground truth to within 6°, a rough estimate suggests that the normalized spectral gap to be resolved may need to exceed $(\|\boldsymbol{T}\|_2/\|\boldsymbol{X}\|_2)/\sin(6°) \approx 0.062$. By contrast, for feature-extractor applications where the downstream accuracy requirement is less stringent and including additional singular vectors can effectively expand the feature dimensionality, a looser criterion may be adopted, requiring only $\frac{s_i - s_{i+1}}{s_1} > \sqrt{\Theta\left(\frac{\sigma_r}{G_{max} \cdot 2^a}\right)^2 + \Theta\left(\frac{1}{2\sqrt{3} \cdot 2^{a+b}}\right)^2}$ (e.g., 0.0085 vs. $\sqrt{\Theta(0.00125)^2 + \Theta(0.0011)^2}$ for EMG feature-extractor; the simulation results of $\|\boldsymbol{T}\|_2/\|\boldsymbol{X}\|_2$ under a memristor noise of 0.3 μS yield approximately 0.00215±2×10$^{-5}$). Correspondingly, allowing the angle between the last resolved singular vector and its ground truth to be within 15° would map to a normalized spectral gap $(\|\boldsymbol{T}\|_2/\|\boldsymbol{X}\|_2)/\sin(15°) \approx 0.0083$, which may serve as a rough reference for identifying reliably resolved components.

In summary, for a more quantitative criterion, the ratio $\|\boldsymbol{T}\|_2/\|\boldsymbol{X}\|_2$ can be estimated by modeling the non-ideal characteristics of the memristor device together with the computed first singular value; for a desired accuracy expressed as a maximum allowable deflection angle $\theta'$, the corresponding normalized spectral gap threshold is given by $(\|\boldsymbol{T}\|_2/\|\boldsymbol{X}\|_2)/\sin(\theta')$ and singular values whose normalized spectral gaps comfortably exceed this threshold can be taken as a rough criterion for reliable resolution.

**Supplementary Note 3. Analysis of bit slicing in DPM.**

Let $\boldsymbol{X} = (\boldsymbol{X}_h, \boldsymbol{X}_l) \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ denote the SVD data matrix mapped to the memristor array with DPM. For analytical simplicity, we focus on a single vector-matrix multiplication, which serves as the fundamental computation of SVD. Consider one accumulated column of $\boldsymbol{X}_h$. In each computation, the input $\boldsymbol{V}_{\mathrm{in}}$ is a binary (0/1) bit vector. Since rows with 0 input elements contribute nothing to the output, we focus on the rows where the elements equal 1. The output can then be written as:

$$\mathrm{Out} = \mathrm{Quant}\left(\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0} \boldsymbol{X}_h^{(i)}\right), \tag{7}$$

where $\mathrm{Quant}(\cdot)$denotes the analog-to-digital conversion (quantization) operation.

Due to memristor noise, $\boldsymbol{X}_h$ and $\boldsymbol{X}_l$ consist of the true values plus additive noise, i.e.,

$$\boldsymbol{X}_h = \boldsymbol{X}_{h,\mathrm{true}} + \boldsymbol{X}_{h,\mathrm{noise}};\ \boldsymbol{X}_l = \boldsymbol{X}_{l,\mathrm{true}} + \boldsymbol{X}_{l,\mathrm{noise}}. \tag{8}$$

Taking a single computation as an example, after splitting, the result can be expressed as:

$$\mathrm{Output} = \sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0} \left[\left(\boldsymbol{X}_{h,\mathrm{true}}^{(i)} + \boldsymbol{X}_{h,\mathrm{noise}}^{(i)}\right) \times 2^b + \left(\boldsymbol{X}_{l,\mathrm{true}}^{(i)} + \boldsymbol{X}_{l,\mathrm{noise}}^{(i)}\right)\right] \tag{9}$$

When the high-order bits contain noise, this noise can be amplified by the bit position and ultimately affects the result. However, considering the quantization operation, and assuming a memristor mapping upper bound of $G_{max}$, the actual high-bit error can be expressed as:

$$\mathrm{Error}_{h,\mathrm{noise}} = \mathrm{Quant}\left(\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0} \boldsymbol{X}_{h,\mathrm{noise}}^{(i)}\right) = \mathrm{Quant}\left(\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0} \frac{G_{h,noise}^{(i)} \times (2^a - 1)}{G_{max}}\right) \times 2^b, \tag{10}$$

The overall error can be expressed as:

$$\mathrm{Error}_{\mathrm{DPM,noise}} = \mathrm{Quant}\left(\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0} \frac{G_{h,noise} \times (2^a - 1)}{G_{max}}\right) \times 2^b + \mathrm{Quant}\left(\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0} \frac{G_{l,noise} \times (2^b - 1)}{G_{max}}\right), \tag{11}$$

$$\mathrm{Error}_{\mathrm{orig,noise}} = \mathrm{Quant}\left(\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0} \frac{G_{noise} \times (2^{a+b} - 1)}{G_{max}}\right). \tag{12}$$

For analytical convenience, based on measured data (**Fig. 2f**), $\boldsymbol{X}_{h,\mathrm{noise}}$ can be modeled as Gaussian noise $\mathcal{N}(0,\sigma^2)$, which is a common practice in memristor noise modeling, where $\sigma$ represents the error in the mapped values.

Notably, under otherwise fixed conditions, we can control the bit-splitting configuration to adjust $\frac{G_{h,noise}\times(2^a-1)}{G_{max}}$, such that the result of $\mathrm{Quant}\left(\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0}\frac{G_{h,noise}^{(i)}\times(2^a-1)}{G_{max}}\right)$ is mostly zero. This effectively prevents high-bit errors from affecting the low-bit computation.

Specifically, during the computation, as the minimum resolvable unit is the resolution of memristor $\mathrm{Reso.}_{\mathrm{Mem}}$, a sufficient condition for the computation to be effectively error-free can be written as

$$\mathrm{Error}=\sum_{\boldsymbol{V}_{\mathrm{in}}^{(i)}\neq 0}\boldsymbol{X}_{h,\mathrm{noise}}^{(i)}<\frac{1}{2}\times\mathrm{Reso.}_{\mathrm{Mem}}. \tag{13}$$

Since the terms $\boldsymbol{X}_{h,\mathrm{noise}}^{(i)}$ are i.i.d., if the number of nonzero input elements is $n$, then the aggregated computation error follows:

$$\mathrm{Error}\sim\mathcal{N}(0,n\sigma^2). \tag{14}$$

Under a $2\sigma$-criterion which covers approximately 95% error distribution, the noise impact can be regarded as negligible when

$$4\sqrt{n}\,\sigma<\mathrm{Reso.}_{\mathrm{Mem}}. \tag{15}$$

Assuming a device programming range in [0, $G_{max}$], the number of most-significant bits that remain effectively unaffected can be approximated as

$$\#\mathrm{Bit}=\log_2\left(\frac{G_{max}}{4\sqrt{n}\,\sigma}+1\right). \tag{16}$$

Using the 128×128 matrix as an example, the maximum required input parallelism is 128. Assuming the input pulses follow a uniform Bernoulli distribution over “0” and “1”, the expected number of nonzero row inputs is 64. For an equivalent device-noise standard deviation $\sigma\in[0.2,0.35]$ $\mu$S, we obtain $4\sqrt{n}\,\sigma\in[6.4,11.2]$ $\mu$S. With a programming $G_{max}$ of 30 $\mu$S, for an overall 8-bit representation, assigning only 2 bits to the high part makes the noise impact after output quantization to be very small, and thus prevents it from interfering with the computation of the lower bits. Alternatively, if we adopt a less conservative of about $2\sqrt{n}\,\sigma$,

then the allowable number of high bits becomes $\log_2\left(\frac{G_{max}}{2\sqrt{n}\,\sigma}+1\right)$, which relaxes the allocation to roughly 3 bits. This improves the overall representation precision while ensuring that accumulated errors in the high bits have negligible influence on the computation performed by the low-bit cells. It is worth noting that in systems where signed inputs are encoded as multiple bit-serial pulses with positive and negative components applied separately, the effective fraction of nonzero input pulses per cycle is relatively small — particularly for the more significant input bit positions. This inherently low activation sparsity allows DPM to remain effective even as the array size scales up.

Once the error in the high-order bits is controlled, the impact of low-bit noise is effectively reduced due to the decreased representation levels. With this setting, we also program the low-bit cells to a relatively high resolution to achieve higher representational fidelity in expectation. **Supplementary Figure 20** shows how the multiplication error varies with different DPM configurations for a random 128×128 matrix under various noise levels. When low resolution is employed for the higher-order part, such as using [1,7] or [2,6] partitioning, high-bit noise can be effectively controlled, leading to substantial reduction in overall error compared to the no-DPM case. For instance, with [2,6] partitioning, the error can ideally be reduced to one-quarter of the original value. For higher partitioning schemes such as [3,5], while the high-bit computation introduces some error, the majority of the error is still eliminated, resulting in effective overall precision improvement. However, for the commonly used uniform [4,4] partitioning without device awareness, the reduction in overall error compared to the no-DPM case is very limited.

We further conducted a study on the DPM strategy with simulation shown in **Supplementary Figure 21**. In each experiment, we fixed the total bit budget and systematically varied the bit resolution assigned to the first cell, in order to quantify the impact of different partitions on the SVD results.

Overall, when the first cell is allocated too few bits, the effective precision of the second cell is correspondingly constrained; as a result, the joint representational capability of the two-cell scheme cannot be substantially improved. In contrast, when the first cell is allocated too many bits (e.g., $>$ 3bits for an overall 7–8-bit representation), noise in the most-significant portion tends to dominate and can effectively mask the contribution of the least-significant portion. Compared to conventional single-cell analog mapping, the slicing mechanism in DPM doubles the array area. However, considering that the array area (post-slicing) occupies only

~12.3% of the total area (**Fig. 6e**, exemplified by the EMG-based gesture recognition task), the additional array area overhead introduced by DPM amounts to approximately 6.5% of the total. This is a worthwhile trade-off for the resulting gains in precision.

Based on these observations, for a target overall resolution of 6–8 bits, assigning 2–3 bits to the first cell provides the most favorable trade-off. We therefore adopt this configuration in our experiments.

**Supplementary Note 4. Analysis of vector repetition and additional input bits in VPR.**

We present an analysis of the effectiveness of the vector repetition and additional bits used in VPR. Given that the deflation process involves subtracting the contribution of currently solved components from the full data matrix, VPR is designed to address component residual arising during deflation with minimal computational overhead, thereby effectively enhancing the accuracy and stability of subsequent solving steps.

For the solved right singular vectors (RSVs) to be stored on array to implement deflation, it has independent Gaussian-like noise on devices with noise magnitude (standard deviation) of $\sigma$. With the $n$-times repeating and averaging approach, the equivalent readout noise magnitude can be reduced to $\sigma/\sqrt{n}$ with the law of the Standard Error of the Mean. For low-rank decomposition, the volume of solved singular vectors is typically much smaller than entire problem scale. Consequently, properly replicating RSVs to enhance numerical precision is a cost-effective optimization tailored to the specific computational characteristics of MSVD.

Regarding the allocation of additional input bits, as shown in **Eq. (4)**, large dynamic ranges of singular values expand the input vector's value range, demanding higher quantization precision. For instance, when computing the fourth singular vector for the IRIS dataset, the first three scaling factors ($s_1^2, s_2^2, s_3^2$) are 629.5, 36.09, and 11.70, respectively, creating an amplification gap of approximately 54× (629.5/11.70) between the largest and smallest scaled components. If standard quantization precision were used, the resulting quantization errors would be substantially amplified in subsequent multiplications, degrading final accuracy. VPR addresses this by allocating additional bits to enhance the precision of input vectors in the second multiplication in each power iteration loop (i.e., the output of the first multiplication scaled by singular values) on RSVs. Taking the system baseline of 8-bit input/output precision as an example, the 6 additional bits bring the effective input precision at this stage to 14 bits (8 + 6), thereby mitigating the impact of singular value dynamic ranges. This approach also helps reduce the accumulation of quantization errors during iterative computations, ultimately achieving more accurate results. An experiment on the IRIS dataset was conducted under different input/output precisions measuring the mean square error (MSE) of solved RSVs (**Supplementary Figure 22**). Each data point represents the average of 20 repeated experiments. We varied the system basic input/output precision from 6 bits to 12 bits and set the additional bit used in VPR from 0 to 8 bits to demonstrate its impact. As shown in **Supplementary Figure 22**, allocating additional bits provides effective error reduction. In

contrast, when sufficient additional bits are present, enhancing the overall input/output bit precision offers only limited improvement. Furthermore, the benefits of additional bit allocation in second multiplication on RSVs in the loop remain consistent across various input/output precisions, though these benefits decrease under high-precision conditions (>10b). Appropriate additional bits in VPR can consistently improve solution accuracy while effectively reducing the computational overhead compared to global input-output precision enhancement.

**Supplementary Note 5. Conductance mapping with linear quantization.**

For mapping the data matrix $\boldsymbol{X}$ onto the array, we first quantize $\boldsymbol{X}$. Following a standard mapping procedure, we decompose the data into positive and negative components and quantize each component to the total bit number of DPM. We determine the required rescaling factor by mapping the maximum absolute entry of $\boldsymbol{X}$ to the maximum quantization level. We then map the maximum quantized value of each slice to the maximum conductance state of the corresponding cell. For example, with a 2-bit cell and a maximum target conductance of 30 μS, the quantized value 3 is mapped to 30 μS. This establishes the numerical-to-conductance correspondence used throughout the mapping.

For vectors, the value range is in [0,1]. Accordingly, we map the upper bound "1" to the maximum quantization level, and we keep the scaling factor consistent across vector updates to ensure correct VMM operation. For larger-scale workloads (in MSVD-based LLM initialization for fine-tuning), since the expected magnitude of vector entries is relatively small, we map the maximum value of the first computed vector to 3/4 of the maximum quantization level. This improves effective representational precision while preserving sufficient dynamic range.

During computation, the scaling factors for all matrix blocks are kept fixed, and thus no additional dynamic overhead is introduced.

**Supplementary Note 6. Energy consumption evaluation.**

We evaluate the energy consumption of MSVD across three application scenarios at the 32 nm technology node. The benchmark architecture is illustrated in **Supplementary Figure 2**, including memristor arrays, ADCs, and peripheral circuits. The energy consumption of CMOS-based peripheral circuits was simulated using XPEsim[22] under 32 nm technology. The consumption of the ADCs was also projected to the 32 nm node following established technology scaling method[4,23]. Although our chip validation was performed on 130 nm technology, the energy evaluation adopts 32 nm technology to reflect the potential of our approach in advanced processes and enable fair comparison with modern GPU baselines. Software operations (normalization, vector products) were evaluated using GPU energy efficiency as reference. For all evaluations of MSVD with SREA, 8-bit ADCs[24] are used for parallel computing and 6-bit ADCs[25] handle mapping operations; without SREA, 10-bit ADCs[26] are utilized with the analog mapping paradigm[2,27]. We validated our evaluation framework by measuring the multiplication energy of the analog macro at 130 nm. For an 800 ×128 matrix (with an average cell conductance of ~10 μS) and 8-bit inputs, the multiplication energy is less than 16.3 nJ with 8-bit ADCs scaled to 130 nm. This aligns closely with our XPEsim-based benchmark result of 14.36 nJ at 130 nm node, confirming the reliability of our evaluation framework.

**EMG-based gesture recognition.** In the context of user-scalable EMG-based gesture recognition, SVD and incremental SVD constitute the most energy-intensive operations. We use three matrix configurations: 864×500 (new user information), 6×500 (vectors to be computed), and 6×500 (pre-computed singular vectors), the detailed settings of SREA are shown in **Supplementary Table 3**. With SREA, parallel computing proceeds with 15 iterations per rank; without SREA, 30 iterations are required due to its convergence characteristics (**Supplementary Figure 14**). MSVD with SREA consumes 13.40 μJ, and incremental MSVD consumes 17.83 μJ for each update (with 0.61 μJ for singular vectors updating including write and verify). For approaches without SREA, these energy consumptions are 26.04 μJ and 30.97 μJ, respectively (**Supplementary Table 6**). For user scaling from 2 to 8 users in 3 steps, incremental MSVD with SREA requires a total of 3×17.83 = 53.49 μJ. To benchmark these results, we compare with GPU baselines where the energy efficiency of A100 GPU[28] is 390 GOPS·W$^{-1}$ (TF32, using PyTorch's built-in SVD). With 156 MOPs needed, GPU consumes 400 μJ. Complete singular vector evolution for user addition requires 3.59 mJ (1.40 GOPs) using conventional approaches. Compared to conventional GPU-based approaches, MSVD

achieves 29.9× energy efficiency improvement for singular vector solving, while incremental MSVD provides 67.1× energy reduction for user addition. Against a Google TPUv3 baseline (469 GOPS·W$^{-1}$), the total TPU energy for 3 updates amounts to 2.99 mJ, yielding a 55.9× energy efficiency advantage for incremental MSVD.

**EEG-based sleep stage detection.** We benchmark energy of MSVD with SREA for EEG-based sleep stage detection at 32 nm, with settings detailed in **Supplementary Table 3**. Three matrix configurations are used: 900×500 (new data per update), 5×500 (vectors to be computed), and 5×500 (pre-computed singular vectors), each solved with 15 iterations per rank. Incremental MSVD consumes 21.14 μJ per update, totaling 84.56 μJ over four updates, with a cumulative computational load of 1.89 GOPs. The corresponding energy on an A100 GPU and TPU v3 amounts to 4.85 mJ and 4.03 mJ, respectively, representing advantages of 57.3× and 47.6× for incremental MSVD.

**MSVD initialization for low-rank adaptation.** For the LLaMA 3.2-1B model with the settings detailed in **Supplementary Tables 3, 5**, weight matrices fall into three size categories (transposed if column number is larger than row number): 2048×512, 2048×2048, and 8192×2048, each solved for 4 ranks with fixed 15 iterations per rank, with MSVD energy consumptions of 27.84 μJ, 82.28 μJ, and 247.8 μJ, and computational loads of 252 MOPs, 1.01 GOPs, and 4.03 GOPs, respectively. Accounting for the number of each matrix type per transformer layer, the total energy per layer amounts to 963.6 μJ (27.84 μJ ×2 + 82.28 μJ ×2 + 247.8 μJ ×3), with a total computational load of 14.6 GOPs. The corresponding A100 GPU energy consumption is 37.4 mJ, representing a 38.8× energy efficiency advantage for MSVD with SREA. For the LLaMA 3.2-3B model under the same SREA configuration and iteration settings, the three matrix size categories are 3072×1024, 3072×3072, and 8192×3072, with MSVD energy consumptions of 62.17 μJ, 157.6 μJ, and 352.5 μJ, and computational loads of 766 MOPs, 2.30 GOPs, and 6.07 GOPs, respectively. The total energy per layer amounts to 1.497 mJ (62.17 μJ ×2 + 157.6 μJ ×2 + 352.5 μJ ×3), with a total computational load of 24.3 GOPs. The corresponding A100 GPU energy consumption is 62.4 mJ, representing a 41.7× energy efficiency advantage for MSVD.

**Supplementary Note 7. Technology scaling analysis of MSVD.**

We utilized XPEsim and the ADC scaling method[4,23] to further analyze the scaling trends of MSVD's energy efficiency and normalized speed across different technology nodes in EMG-based gesture recognition task. The scaling of ADCs is performed based on analysis of ADC designs reported in the literature[4,23]. By controlling the Nyquist sampling rate $f_N$, effective number of bits (ENOB), we analyze how area and power consumption scale with technology node. The scaling model used follows:

$$\text{Area} = 10^{\alpha_0}\lambda^{\alpha_1}2^{\alpha_2\cdot\text{ENOB}}f_N^{\alpha_3}$$

$$\text{Power} = 10^{\beta_0}\lambda^{\beta_1}2^{\beta_2\cdot\text{ENOB}}f_N^{\beta_3}$$

Using the survey data[29] from 2005 to 2025, we extract the following coefficients for the SAR ADC employed in our benchmark: $\alpha_0$ =-0.55, $\alpha_1$ =1.07, $\alpha_2$ =0.34, $\alpha_3$ =-0.08; $\beta_0$ =-12.63, $\beta_1$=0.95, $\beta_2$=0.67, $\beta_3$=-1.18.

**Supplementary Figure 28** illustrates these metrics for the EMG-based gesture recognition task, benchmarking MSVD against the 28 nm NVIDIA M40, 16 nm P100, and 7 nm A100 GPUs. When scaled to the 22 nm node, MSVD achieves a 413× improvement in energy efficiency and a 358× speedup compared to the P100 (16 nm). Similarly, at the 32 nm node, MSVD demonstrates a 432× improvement in energy efficiency and a 321× speedup over the M40 (28 nm). Notably, even at the 130 nm node, MSVD exhibits a 143.5× advantage in energy efficiency and a 31.17× advantage in speed compared to 28 nm CMOS technology. These results consistently underscore the significant advantages of MSVD in both energy efficiency and speed over CMOS-based HPC systems.

**Supplementary Note 8. Analysis of normalized speed.**

We evaluate the normalized speed of MSVD across three application scenarios at the 32 nm technology node.

**EMG-based gesture recognition.** The computational delay of MSVD comprises two primary components: delays from vector-matrix multiplication operations and mapping during singular vector updates. Under digital control, these delays are determined by operating cycle count and cycle duration. MSVD with SREA operates with 15 iterations, yielding multiplication delays of 121.8 μs and update delays of 62.0 μs. MSVD without SREA requires 30 iterations for convergence, resulting in multiplication delays of 153.0 μs and singular vector updating delays of 127.5 μs. For incremental MSVD operating with memristor arrays for data matrix, pre-computed vectors, and vectors to be computed in parallel, the delay of operations on pre-computed vectors can be hidden within the concurrent operations on the other two matrices, introducing no additional overhead.

**Supplementary Table 8** shows the area breakdown, with peripheral circuit evaluated by XPEsim[22] under 32 nm technology. The total areas required for MSVD implementation are 0.236 $mm^2$ and 0.229 $mm^2$ with and without SREA, respectively. For incremental MSVD, the corresponding chip areas are 0.313 $mm^2$ and 0.298 $mm^2$, respectively. To validate these benchmark results, we measured the 130 nm analog macro, which occupies approximately 1.18 $mm^2$. When scaled to the 32 nm node, this projects to roughly 0.074 $mm^2$ (detailed in **Supplementary Table 9**). Considering the design redundancy in our test chip, this footprint is comparable to the benchmark results of analog macro (without ADCs) presented in **Supplementary Table 8** for a similar array size, confirming the accuracy of our area estimates.

Leveraging these delay and area metrics, the normalized solving speed of MSVD with SREA is 1 / [(121.8+62.0) μs × 0.236 $mm^2$] = 23054 computations·$s^{-1}$·$mm^{-2}$, compared to 1 / [(153.0+127.5) μs × 0.229 $mm^2$] = 15568 computations·$s^{-1}$·$mm^{-2}$ for MSVD without SREA, demonstrating the SREA's 48.1% speedup. The normalized speed of incremental MSVD is 1 / [(121.8+62.0) μs × 0.313 $mm^2$] = 17382 updates· $s^{-1}$·$mm^{-2}$ with SREA and 1 / [(153.0+127.5) μs × 0.298 $mm^2$] = 11963 updates· $s^{-1}$·$mm^{-2}$ without SREA, representing a 45.3% improvement.

Compared to the A100 GPU[28] (156 TOPS, 826 $mm^2$), the normalized speed is 1211 computations·$s^{-1}$·$mm^{-2}$ for 156 MOPs per computation. Our MSVD with SREA shows a 19.0× advantage in normalized speed. For user scalable tasks, conventional methods require solving matrices with corresponding computational loads of 311, 467, and 623 MOPs. Consequently,

the normalized speeds for the 3 updates are 607, 404, and 303 updates· $s^{-1}\cdot mm^{-2}$, respectively, with an average speed of 438 updates·$s^{-1}\cdot mm^{-2}$. As shown in **Supplementary Figure 15**, incremental MSVD achieves advantages of 28.6×, 43.0×, and 57.3× in the three update processes, respectively, resulting in an average normalized speed improvement of 39.7×. For the Google TPUv3[3] (123 TOPS, ~700 $mm^2$), the normalized speeds for the three updates are 565, 376, and 282 updates·$s^{-1}\cdot mm^{-2}$, with an average speed of 408 updates·$s^{-1}\cdot mm^{-2}$, and incremental MSVD with SREA achieves a 42.6× advantage.

**EEG-based sleep stage detection.** For incremental MSVD with SREA, with settings detailed in **Supplementary Table 3**, the required chip area is 0.324 $mm^2$ and the delay per update is 353.5 μs, yielding a normalized speed of 8744 updates·$s^{-1}\cdot mm^{-2}$. The four update processes have computational loads of 271, 406, 541, and 675 MOPs, respectively. For the A100 GPU, the corresponding normalized speeds are 697, 465, 349, and 280 updates·$s^{-1}\cdot mm^{-2}$, with an average of 448 updates·$s^{-1}\cdot mm^{-2}$, resulting in a 19.5× advantage for incremental MSVD. For the TPU v3, the normalized speeds are 648, 433, 324, and 260 updates·$s^{-1}\cdot mm^{-2}$, with an average of 416 updates·$s^{-1}\cdot mm^{-2}$, corresponding to a 21.0× advantage.

**MSVD initialization for low-rank adaptation.** For the LLaMA 3.2-1B model with the settings detailed in **Supplementary Tables 3, 5**, weight matrices fall into three size categories (transposed if column number is larger than row number): 2048×512, 2048×2048, and 8192×2048, with MSVD chip areas of 0.451 $mm^2$, 1.15 $mm^2$, and 2.94 $mm^2$, respectively. The total chip area per transformer layer is 12.0 $mm^2$ (0.451 $mm^2$ × 2 + 1.15 $mm^2$ × 2 + 2.94 $mm^2$ × 3), with a computational delay of 325 μs, yielding a normalized speed of 255 layers·$s^{-1}\cdot mm^{-2}$. For the total computational load of 14.6 GOPs per layer, the A100 GPU achieves a normalized speed of 12.9 layers·$s^{-1}\cdot mm^{-2}$, giving MSVD a 19.8× speed advantage. For the LLaMA 3.2-3B model, the three matrix size categories are 3072×1024, 3072×3072, and 8192×3072, with chip areas of 0.847 $mm^2$, 1.99 $mm^2$, and 3.84 $mm^2$, respectively. The total chip area per transformer layer is 17.2 $mm^2$ (0.847 $mm^2$ × 2 + 1.99 $mm^2$ × 2 + 3.84 $mm^2$ × 3), with the same computational delay of 325 μs, yielding a normalized speed of 179 layers·$s^{-1}\cdot mm^{-2}$. For the total computational load of 24.3 GOPs per layer, the A100 GPU achieves a normalized speed of 7.76 layers·$s^{-1}\cdot mm^{-2}$, resulting in a 23.1× speed advantage for MSVD.

## Supplementary References


1 Yu, L. *et al.* Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284* (2023).

2 Zhao, H. *et al.* Implementation of Discrete Fourier Transform using RRAM Arrays with Quasi-Analog Mapping for High-Fidelity Medical Image Reconstruction. in *2021 IEEE International Electron Devices Meeting (IEDM).* 12.14.11-12.14.14.

3 Norrie, T. *et al.* The Design Process for Google's Training Chips: TPUv2 and TPUv3. *IEEE Micro* **41**, 56-63 (2021).

4 Zidan, M. A. *et al.* A general memristor-based partial differential equation solver. *Nature Electronics* **1**, 411-420 (2018).

5 Sun, Z. *et al.* Solving matrix equations in one step with cross-point resistive arrays. *Proceedings of the National Academy of Sciences* **116**, 4123-4128 (2019).

6 Zhang, W. *et al.* Array-level boosting method with spatial extended allocation to improve the accuracy of memristor based computing-in-memory chips. *Science China Information Sciences* **64**, 160406 (2021).

7 Mannocci, P. *et al.* In-Memory Principal Component Analysis by Crosspoint Array of Resistive Switching Memory: A new hardware approach for energy-efficient data analysis in edge computing. *IEEE Nanotechnology Magazine* **16**, 4-13 (2022).

8 Korkmaz, A. *et al.* Analog Acceleration of the Power Method using Memristor Crossbars. in *2022 IEEE International Symposium on Circuits and Systems (ISCAS).* 1194-1198.

9 Zhao, H. *et al.* Energy-efficient high-fidelity image reconstruction with memristor arrays for medical diagnosis. *Nature Communications* **14**, 2276 (2023).

10 Li, J. *et al.* Sparse matrix multiplication in a record-low power self-rectifying memristor array for scientific computing. *Sci Adv* **9**, eadf7474 (2023).

11 Song, W. *et al.* Programming memristor arrays with arbitrarily high precision for analog computing. *Science* **383**, 903-910 (2024).

12 Yousuf, O. *et al.* Layer ensemble averaging for fault tolerance in memristive neural networks. *Nature Communications* **16**, 1250 (2025).

13 Chen, H. *et al.* Continuous-time digital twin with analog memristive neural ordinary differential equation solver. *Sci Adv* **11**, eadr7571 (2025).

14 Le Gallo, M. *et al.* Precision of bit slicing with in-memory computing based on analog phase-change memory crossbars. *Neuromorphic Computing and Engineering* **2**, 014009 (2022).

15 Wu, C.-H. & Tsai, P.-Y. An SVD processor based on Golub–Reinsch algorithm for MIMO precoding with adjustable precision. *IEEE Transactions on Circuits and Systems I: Regular Papers* **66**, 2572-2583 (2019).

16 Moon, S., Lee, N. & Lee, Y. A Scalable Precoding Processor for Large-Scale MU-MIMO Systems. *IEEE Transactions on Circuits and Systems I: Regular Papers* **70**, 3029-3039 (2023).

17 Tsai, T.-Y., Shen, C.-A., Wu, T.-L. & Huang, Y.-H. The algorithm and vlsi architecture of high-throughput and highly efficient tensor decomposition engine. *IEEE Transactions on Circuits and Systems I: Regular Papers* **71**, 3134-3145 (2024).

18 Huang, T. *et al.* A power-iteration-based beamformer for large-scale antenna arrays. *Microelectron J* **166**, 106868 (2025).

19 Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems* **31** (2018).

20 Golub, G. H. & Van Loan, C. F. *Matrix computations*. (JHU press, 2013).

21 Wedin, P.-Å. Perturbation bounds in connection with singular value decomposition. *BIT Numerical Mathematics* **12**, 99-111 (1972).

22 Zhang, W. *et al.* Design Guidelines of RRAM based Neural-Processing-Unit: A Joint Device-Circuit-Algorithm Analysis. in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. 1-6.

23 Verhelst, M. & Murmann, B. Area scaling analysis of CMOS ADCs. *Electronics letters* **48**, 314-315 (2012).

24 Kull, L. *et al.* A 3.1 mW 8b 1.2 GS/s Single-Channel Asynchronous SAR ADC With Alternate Comparators for Enhanced Speed in 32 nm Digital SOI CMOS. *IEEE Journal of Solid-State Circuits* **48**, 3049-3058 (2013).

25 Choo, K. D., Bell, J. & Flynn, M. P. 27.3 Area-efficient 1GS/s 6b SAR ADC with charge-injection-cell-based DAC. in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. 460-461.

26 Hao, J. *et al.* 10.2 A Single-Channel 2.6GS/s 10b Dynamic Pipelined ADC with Time-Assisted Residue Generation Scheme Achieving Intrinsic PVT Robustness. in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. 168-170.

27 Wu, W. *et al.* A Methodology to Improve Linearity of Analog RRAM for Neuromorphic Computing. in *2018 IEEE Symposium on VLSI Technology.* 103-104.

28 *NVIDIA A100 Tensor Core GPU Architecture*, <https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf> (2020).

29 *ADC Performance Survey 1997-2026*, <https://github.com/bmurmann/ADC-survey> (2026).