

Memristive singular value decomposition

Corresponding Author: Professor Ngai Wong

Parts of this Peer Review File have been redacted as indicated to remove third-party material.

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Attachments originally included by the reviewers as part of their assessment can be found at the end of this file.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The manuscript presents a memristive singular value decomposition (MSVD) framework enhanced by a selective representation enhanced architecture (SREA). Through this design, the authors achieve accuracy levels that approach software implementations across a variety of applications, including image enhancement, temporal data reconstruction, and EMG-based gesture recognition. The reported results are compelling, with 31 dB PSNR in image tasks, near-perfect correlation (0.999) for COVID-19 data reconstruction, and gesture recognition accuracy reaching 88.26%, while incremental learning improves performance by nearly 20% as additional users are introduced. Beyond accuracy, the approach demonstrates efficiency, achieving more than fifty-fold improvements in energy consumption and a thirty-seven-fold boost in normalized speed compared to GPU baselines, underscoring its practical promise.

The significance of this work lies in extending the role of memristor-based compute-in-memory architectures beyond neural networks to general-purpose decomposition algorithms. By addressing the challenges of dual-matrix iteration and deflation inherent to SVD, the authors push the field toward more versatile memristor accelerators that could benefit edge-AI and wearable systems. In contrast to earlier studies focused primarily on linear solvers or isolated matrix operations, this contribution distinguishes itself by tackling multi-scale iterative algorithms and validating them experimentally on custom hardware. As such, it adds a meaningful advance in both hardware-aware algorithm design and energy-efficient data analysis. There are several concerns that have been listed as follows:

1. The authors report large gains in energy efficiency (50.7×) and normalized speed (37×) compared to GPU implementations. While impressive, these benchmarks are performed on small and relatively low-dimensional datasets such as IRIS (150×4) and NinaPro (864×500). It is unclear whether the same magnitude of efficiency gain would be observed for real-world, large-scale matrices (e.g., >10,000 dimensions). The authors somehow consider different applications, but not very different in size, to claim scalability.
2. Image reconstruction results show PSNR = 31 dB and SSIM = 0.94, indicating moderate fidelity. However, state-of-the-art software methods regularly achieve >35 dB PSNR. This raises the question of whether the proposed memristor approach sacrifices too much accuracy for efficiency. A more direct comparison to recent algorithmic baselines in imaging would help contextualize the significance of this result.
3. Gesture recognition with incremental MSVD achieves 88.26% for 2 users and ~84% for 8 users. While the numbers are promising, they are based on NinaPro, a small, well-controlled EMG dataset. Real-world EMG signals are significantly noisier, and cross-user variability is much larger. Without evaluation on more diverse datasets or field-collected data, the generalizability of these accuracy results is uncertain.
4. The incremental MSVD framework improves performance by ~19% compared to baseline non-incremental approaches. However, the scalability is demonstrated only up to 8 users. This limited scope does not convincingly support the broader claim that MSVD is a “scalable framework.” The reviewer suggests expanding the number of users (e.g., 50+) or simulating synthetic large-scale incremental additions to demonstrate true scalability.
5. The hardware benefits are presented primarily in comparison to GPU baselines, including NVIDIA A100. GPUs are general-purpose architectures not optimized for matrix decompositions. The lack of comparisons with dedicated ASIC or FPGA accelerators limits the strength of the argument. Without such comparisons, the reviewer feels the claimed hardware efficiency could be overstated. Supplementary Table 1 provides a useful survey of prior memristor-based computing works. However, this comparison is descriptive only — no quantitative metrics (e.g., energy per iteration, error rates) are provided. Without direct side-by-side benchmarking, the claimed advantages of MSVD remain qualitative.

6. The authors demonstrate applications in three domains: imaging, temporal COVID-19 data, and gesture recognition. While these domains show diversity, they remain confined to relatively narrow use cases. Singular value decomposition has far broader importance (e.g., recommendation systems, low-rank adaptation in transformers, PCA in genomics). Expanding the application discussion to such areas would make the contribution more significant for a wider audience.
7. The authors mainly emphasize MSVD's role in edge AI and wearables. While important, the broader relevance of SVD in modern ML is overlooked. For example, SVD is foundational for LoRA (Low-Rank Adaptation) in large language models, matrix factorization in recommender systems, and dimensionality reduction in high-throughput genomics. Relating MSVD to these broader fields would increase cross-disciplinary impact.
8. The authors claim "software-comparable accuracy," citing high correlation (0.999) in COVID-19 reconstructions. While correlation is indeed strong, it is unclear whether these results generalize across multiple experimental runs, given device-level variability.
9. Supplementary Fig. 5 shows ablation analysis of dual-precision mapping (DPM) and vector processing refinement (VPR). These results confirm that both components contribute to reduced error. To prove robustness, for ablations, it is better to include more challenging conditions, such as large singular value gaps or noisy, high-dimensional data.
10. Incremental MSVD is evaluated for up to 8 users, with energy and accuracy improvements reported in Supplementary Fig. 9. While useful, this is a very limited scale. In practical multi-user settings (e.g., healthcare monitoring), dozens or hundreds of users may be encountered. The reviewer suggests scaling experiments further, whether by adding a larger scale or extending by estimation and feasibility analysis to substantiate the claim of efficient "user-scalable recognition."
11. COVID-19 data reconstruction shows a correlation of 0.999. However, this dataset is relatively clean, and reconstruction may not be very challenging. Authors can consider testing the robustness via noisy temporal data (e.g., stock market, mobility, or sensor readings with missing entries). Without such tests, the strong results may not generalize to harder temporal data.
12. Energy savings are explained by reduced iterations (15 vs 30) when using SREA (Supplementary Note 4). However, it is unclear whether the same convergence thresholds and error tolerances were used in both cases. If looser tolerances were allowed in the hardware implementation, the comparison would be biased. Reviewer suggests clearly specifying thresholds and ensuring fair comparison.
13. Supplementary Fig. 8 shows average iteration counts across datasets. While informative, only mean values are reported. Reviewer notes the absence of variance analysis across chips, trials, or different matrix initializations. Since memristors are prone to stochastic noise, such variance should be reported to ensure reproducibility and reliability.
14. Convergence plots (main text) show reduced drift when using SREA, but residual drift is still visible. The long-term stability of the method is not demonstrated — what happens after 1,000 or 10,000 iterations? Without long-term stability tests, it remains unclear whether MSVD can be used for prolonged or online learning tasks.
15. Supplementary Fig. 10 evaluates error sensitivity under additional precision bits in VPR. While this is an excellent analysis, it is entirely simulation-based. Reviewer suggests validating the predicted error reductions experimentally, since real memristor noise characteristics may deviate from the modeled distributions.
16. Incremental MSVD is compared to a naive baseline, as shown in Supplementary Fig. 9. While this demonstrates improvement, stronger baselines exist — digital incremental-SVD algorithms optimized in HPC systems. Without comparison to these, it is difficult to assess whether MSVD provides unique advantages beyond the memristor platform.
17. The choice of dual-precision mapping (2-bit + 5-bit split) is justified partly by Supplementary Note 2. However, the reasoning is somewhat ad hoc, and no sensitivity analysis is provided. Reviewer suggests testing alternative splits (e.g., 3+5, 4+4) to identify optimal configurations across datasets. Without this, the design choice appears arbitrary.
18. The authors replicate singular vectors four times to reduce residual error. Supplementary Table 4 shows that this incurs noticeable area overhead. Reviewer suggests providing a quantitative trade-off analysis: How much accuracy gain per unit area overhead? Without this, the design appears unbalanced.
19. The experimental platform integrates a memristor chip, an FPGA (ZCU102), and a power board (Supplementary Fig. 2). Energy results were estimated using 32 nm simulations for peripheral circuits, although the actual chip was 130 nm. Reviewer notes this may exaggerate energy efficiency claims, since measured results at 130 nm may differ. A sensitivity analysis across technology nodes would add credibility.
20. Many critical details (energy evaluation assumptions, bit allocation, normalized speed analysis) are placed in supplementary material. Reviewer argues that some of these (e.g., convergence thresholds, VPR precision allocation) should appear in the main text. Otherwise, readers may miss important context.
21. Supplementary Fig. 2 shows the schematic of the system. It is unusual how DACs are placed, and it seems the symbol should be reversed. MODE_CTL should be at least 3 bits to select the MUXs, but it showed a single-bit wire.

Reviewer #2

(Remarks to the Author)

The manuscript proposes a scheme for in-memory singular value decomposition (SVD) using memristive crossbars. The challenge to perform meaningful SVD with memristors is their limited precision, and therefore, the authors propose a dual-precision mapping scheme. The idea is interesting, and the experimental demonstrations appear promising. However, I feel that I cannot confidently assess the contribution because the experimental scope, mapping flow, and energy accounting are not described with clarity.

First, the authors should clarify whether all the results presented were obtained experimentally or if some of them were simulated. For experimental results, it should be clarified what exactly is done on memristors, on-chip periphery circuitry, FPGA, or elsewhere.

Several tasks (e.g., 256x256 image, and EMG task) exceed the array column count of the memristor array used experimentally. I think the authors might have used time multiplexing, which is also hinted by the schematic in Suppl Fig 2. I

also guess that doing the experiments involved massive reprogramming of the memristor array. How the tasks investigated are mapped to the memristor array is an extremely important aspect of the approach and of the experiments. It should be detailed for each task in the Methods section (matrix dimensions, what is stored in memristors, how SREA is handled experimentally, number of tiles/passes, inter-tile accumulation, what remains resident vs. streamed, iteration counts...). Only then can we understand what has been done experimentally.

The energy breakdown should state clearly what is included for memristor programming (write pulses) vs reads. Right now it is not clear whether the write pulse energy for program and verify programming is included. This is a concern as I feel that this energy might be high, even possibly dominant, if the scheme involves repeated memristor programming (which is my impression). Besides this concern, I think the methodology for energy evaluation should be more detailed.

Overall, the schematics provided in the Main articles did not help me understand the paper. I relied on text only. I recommend reworking them extensively.

Reviewer #3

(Remarks to the Author)

Dear Authors,

Thank you for the very interesting paper on MSVD. This particular topic has a number of interesting subtleties, and thus my co-reviewer and I had a large number of points of feedback. Our report is in the attached Word document.

Martin Lueker-Boden, PhD

Reviewer #4

(Remarks to the Author)

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The authors have made a substantial effort to revise the manuscript and address the concerns raised in the first round of review. The revised version shows clear improvements in methodological clarity, benchmarking scope, and reproducibility. In particular, the addition of hardware-calibrated large-scale simulations, expanded comparisons with ASIC-based implementations, and improved explanations of the system architecture and data mapping significantly strengthen the paper. The clarification of the distinction between experimentally validated components and simulated results is also appreciated, as it improves transparency.

That said, while the manuscript is now considerably stronger, several key issues remain only partially addressed. These do not invalidate the contribution, but they limit the strength of some of the claims and should be addressed more carefully before publication.

Reviewer #1 – Detailed Second-Round Evaluation (Expanded)

Comment 1: Scalability and Large-Scale Validation

The authors have made a commendable effort to address scalability by introducing large-scale experiments (e.g., LoRA/LLM-inspired workloads) using hardware-calibrated simulations. This addition demonstrates awareness of the importance of evaluating beyond small benchmark datasets. Despite this effort, the scalability claims remain supported exclusively by simulation. In memristor-based systems, large-scale behavior is strongly influenced by non-ideal factors such as wire resistance/capacitance, ADC/DAC bandwidth constraints, array tiling overhead, and cumulative device noise across iterative operations. These effects are often nonlinear and difficult to capture accurately in simulation, particularly when scaling to very large matrix sizes or deep iteration chains. As a result, the current evidence does not fully validate the claimed system-level scalability.

Actionable fix: Clearly annotate all large-scale results as simulation-based projections in both figures and text. Add a scalability analysis section that explicitly models system-level overheads, including tiling strategies, interconnect costs, and peripheral bottlenecks. Provide a breakdown of latency and energy contributions (array vs ADC/DAC vs communication) as a function of matrix size. If feasible, include at least one intermediate-scale hardware experiment to bridge the gap between small-scale validation and large-scale projections.

Comment 2: Dataset size and diversity

The manuscript has been extended to reference more complex workloads, which improves the perceived breadth of applicability.

Comment 3: Accuracy and reconstruction quality (PSNR/SSIM)

The authors acknowledge the trade-off between hardware efficiency and reconstruction accuracy, which is an important aspect of the design.

Comment 4: Gesture recognition robustness

The incremental learning results are better explained, and the framework shows promise for adaptive scenarios.

Comment 5: Incremental learning scalability

The authors extend incremental MSVD evaluation through simulation and additional explanation.

Comment 6: Benchmarking beyond GPU baselines

The addition of ASIC-based comparisons and technology scaling is a strong and important improvement. However, these comparisons rely on cross-technology extrapolation and modeling assumptions, particularly for analog/mixed-signal components such as ADCs. Differences in precision, workload configuration, and convergence criteria may also affect fairness.

Actionable fix: Clearly document all assumptions used in scaling analysis, including voltage, frequency, ADC resolution, and memory access patterns. Provide a sensitivity analysis showing how results vary with key parameters (e.g., ADC energy, precision). Where possible, include a matched-precision comparison to ensure fairness across platforms.

Comment 7: Quantitative comparison with prior memristor and mixed-signal works

The manuscript now includes a broader survey of related work, which improves context. The comparison remains qualitative and does not allow for rigorous assessment of relative performance. Without normalization, differences in technology node, array size, and precision obscure the true contribution.

Actionable fix: Construct a normalized comparison table including: energy per operation (or per SVD iteration), precision, convergence tolerance, array size, and technology node. Add a “comparison assumptions” column to clarify differences across works. If possible, use the authors’ modeling framework to re-estimate one or two prior works under comparable conditions, enabling a more meaningful comparison.

Comment 8: Novelty claim

The revised positioning is appropriate and technically accurate. No further changes required.

Comment 9: Broader impact and applicability claims

The manuscript now connects MSVD to applications such as LoRA and general ML pipelines.

These claims implicitly assume that (i) precision requirements, (ii) matrix sizes, and (iii) memory bandwidth constraints in those domains are compatible with the proposed architecture. This is non-trivial—e.g., LoRA often involves higher precision and tight integration with training loops.

Actionable fix: Add a feasibility analysis paragraph specifying: required precision, expected matrix sizes, and bandwidth needs for one representative application (e.g., LoRA). Clearly label such sections as “Projected applicability” and include a short list of technical gaps to be addressed (e.g., training-time updates, gradient support, higher precision accumulation).

Comment 10: Extension to other decompositions (PCA, QR, etc.)

The extension is mentioned but not technically substantiated. Algorithms like QR (e.g., Householder, Givens rotations) and tensor decompositions require different dataflows, orthogonality constraints, and often tighter numerical stability than SVD via power iteration/deflation.

Actionable fix: Provide a mapping table: for each decomposition (PCA, QR), specify (i) required primitive operations, (ii) whether current hardware supports them, and (iii) what modifications are needed. Include a short algorithm-to-hardware mapping example (even pseudocode) showing how one additional decomposition would be implemented on the proposed architecture.

Comment 11: Statistical rigor and robustness

Still insufficiently addressed. Memristor variability (programming noise, drift, cycle-to-cycle variation) can alter convergence trajectories. Reporting only mean values obscures stability and reliability.

Actionable fix: Report mean $\pm$ standard deviation over ≥ 20 runs for key metrics (PSNR, accuracy, iterations). Include seed-controlled repeated experiments to isolate algorithmic vs device-induced variance. Add a noise injection study (e.g., Gaussian or measured device noise) to quantify sensitivity of convergence and final accuracy.

Comment 12: Ablation study generality

Ablation confirms component importance but under limited conditions. The relative benefit of DPM and VPR likely depends on matrix condition number, singular value spectrum, and noise level.

Actionable fix: Extend ablation to at least one ill-conditioned matrix and one high-dimensional synthetic dataset. Report how performance gains vary with condition number and noise amplitude. Add a plot of error vs iteration with/without each component across these conditions.

Comment 13: Incremental MSVD scaling behavior

Expanded explanation but still lacks system-level analysis. Incremental updates can incur rank growth, re-orthogonalization costs, and memory overhead. Stability may degrade with many updates.

Actionable fix: Provide complexity analysis: time and memory as a function of number of users/data blocks. Report rank growth behavior and whether truncation is applied. Include a stability experiment showing error vs number of incremental updates (e.g., $2 \rightarrow 32$ users, even if partially simulated).

Comment 14: Temporal dataset difficulty
Justification provided.

Comment 15: Convergence criteria and energy fairness

Iteration counts clarified, but stopping criteria still ambiguous. Energy comparisons are only meaningful if final error tolerances are matched across platforms.

Actionable fix: Explicitly state stopping conditions (e.g., $\|A - USV^T\|/\|A\| < \epsilon$ or singular vector change $< \delta$). Provide a table showing final error vs energy for each platform at matched ϵ . If tolerances differ, include a Pareto curve (energy vs error) to enable fair comparison.

Comment 16: Device variability and robustness

Device mismatch and drift can degrade orthogonality and convergence.

Actionable fix: Include either (i) measured variability statistics across devices or (ii) Monte Carlo simulations using realistic device distributions. Show impact on final singular values/vectors and convergence speed. If mitigation is used (e.g., redundancy, recalibration), quantify its benefit and overhead.

Comment 17: Long-term convergence stability

Lacks experimental evidence but if possible add one. Iterative methods may accumulate error over long runs; drift can cause divergence or slow convergence.

Actionable fix: Provide a long-horizon experiment (e.g., 500–1000 iterations or repeated SVD runs). Plot error vs iteration over extended runs. Discuss drift compensation strategies (e.g., periodic re-normalization, recalibration).

Comment 18: Simulation vs experimental validation (precision scaling)

Precision scaling is central to claimed efficiency; simulation may not capture real quantization and nonlinearity effects.

Actionable fix: Clearly label all plots as “Measured” vs “Simulated” in captions. If possible include a discrepancy analysis between simulation and measurement for a shared configuration.

Comment 19: Framing and scope of claims

Limitations acknowledged.

Comment 20: Strength of baseline comparisons

Improved with ASIC comparisons, but still incomplete. Lack of comparison to optimized digital SVD (e.g., FPGA/HPC libraries) limits competitiveness assessment.

Actionable fix: Add at least one strong digital baseline (e.g., FPGA implementation or optimized CPU/GPU library with matched precision/tolerance). If not feasible, include a careful discussion of expected performance relative to such baselines and why direct comparison is challenging.

Comment 21: Reproducibility and accessibility

Evaluation: Code release commitment is positive. Custom hardware limits reproducibility; without models, results cannot be independently verified.

Actionable fix: Release a behavioral simulator or calibrated model (e.g., Python/Matlab) that reproduces key results. Provide example scripts and datasets to replicate figures. Document hardware abstraction (API or interface) so others can map algorithms onto the architecture. If possible, a video of the setup and real-time demo of result would be interesting to contain.

Reviewer #2

(Remarks to the Author)

The authors have substantially overhauled the paper, and I find the revised manuscript much more convincing.

The Figures are greatly improved. Most of the necessary information is now present. However, I still do not think the clarity is ideal, particularly with regard to my concerns about the mapping. The relevant details can be recovered, but the presentation still feels somewhat like a puzzle for the reader.

Regarding the energy benchmarks, some aspects could still be debated, but the methodology is now clearly described, and I consider it appropriate for an early-stage study of this type.

Overall, I think the manuscript can be published, although I would recommend further improvements in readability.

Reviewer #3

(Remarks to the Author)

(Please see my original review for general comments on the significance and merits of this paper.)

This second draft neatly addresses the concerns raised in our original review, particularly the additions to the supplementary material, which brought in some great clarity. I also enjoyed reading the responses to the other reviewers. Thank you.

The supplementary notes 1 and 2 are particularly helpful in answering many of my previous technical comments, as are the flowcharts, and the SREA mapping diagram in Supplementary Figure 25.

Regarding the error analysis in supplementary note 2, it is fairly unsatisfying that the bounds for the limits of spectral resolution are expressed in terms of $O(\text{constant})$. Without knowing the constant of proportionality for this big-O expression, the inequalities are only useful as they provide roughly guidelines or heuristics. For instance one might ask whether a hypothetical user should believe a particular singular value, and as such the answer is unclear. If there is anything the authors can do put in actual numerical limits (perhaps within some probability?) the extra numerical rigor would make this result more practical. However, I do not feel this is a pre-requisite publication.

As an editorial note (to both the authors and editors), there is at least one typo in a figure (in supplementary Figure 22 the top blue box in this flowchart reads "Suppling read pulse", which is clearly a typo.) These kinds of errors are easy to overlook when relying on spell check.

Version 2:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The authors have made substantial efforts throughout the revision process to address the reviewers' comments and significantly improve the manuscript. The revised version demonstrates notable improvements in methodological clarity, benchmarking rigor, scalability discussion, statistical reporting, and overall presentation quality.

In particular, the authors provided clearer distinctions between measured and simulation-based results, expanded the benchmarking framework with ASIC-level comparisons and normalized metrics, strengthened the scalability and robustness analyses, and improved the supplementary materials and mapping explanations. The additional discussions on device variability, convergence behavior, reproducibility, and projected applicability further strengthen the technical depth and transparency of the work.

While some limitations inherent to large-scale memristor-based system evaluation remain, the authors have addressed the concerns raised in a technically reasonable and scientifically transparent manner. The manuscript is now in good condition and presents a meaningful contribution to memristor-based compute-in-memory systems and hardware acceleration for matrix decomposition.

I believe the paper is suitable for publication in its current form.

Reviewer #2

(Remarks to the Author)

The authors have improved the clarity of the manuscript. The paper remains somewhat difficult to read, but I believe the revision is sufficient and the manuscript can be accepted for publication.

Reviewer #3

(Remarks to the Author)

Thank you for the edits.

Regarding my Comment 2 in the previous round: Thank you for the clarifications. I agree that an analytic analysis of the spectral resolution leaves contains application- and memory device-specific inputs that are challenging to capture in an analytic form. Thus I appreciate the example of a monte-carlo approach based on Supp. Note 3, and the worked examples which have been added to the revised Supplementary Note 2. The resulting procedure, though involved, at least gives the hypothetical user a method to follow when evaluating whether a system like this would be suitable for their application.

Regarding Comment 3: Thank you for addressing the editorial errors.

Open Access This Peer Review File is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, 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 license, and indicate if changes were made.

In cases where reviewers are anonymous, credit should be given to 'Anonymous Referee' and the source.

The images or other third party material in this Peer Review File are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license 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 license, visit <https://creativecommons.org/licenses/by/4.0/>

Response letter to reviewers' comments

Reviewer #1 (Remarks to the Author):

General comments:

The manuscript presents a memristive singular value decomposition (MSVD) framework enhanced by a selective representation enhanced architecture (SREA). Through this design, the authors achieve accuracy levels that approach software implementations across a variety of applications, including image enhancement, temporal data reconstruction, and EMG-based gesture recognition. The reported results are compelling, with 31 dB PSNR in image tasks, near-perfect correlation (0.999) for COVID-19 data reconstruction, and gesture recognition accuracy reaching 88.26%, while incremental learning improves performance by nearly 20% as additional users are introduced. Beyond accuracy, the approach demonstrates efficiency, achieving more than fifty-fold improvements in energy consumption and a thirty-seven-fold boost in normalized speed compared to GPU baselines, underscoring its practical promise.

The significance of this work lies in extending the role of memristor-based compute-in-memory architectures beyond neural networks to general-purpose decomposition algorithms. By addressing the challenges of dual-matrix iteration and deflation inherent to SVD, the authors push the field toward more versatile memristor accelerators that could benefit edge-AI and wearable systems. In contrast to earlier studies focused primarily on linear solvers or isolated matrix operations, this contribution distinguishes itself by tackling multi-scale iterative algorithms and validating them experimentally on custom hardware. As such, it adds a meaningful advance in both hardware-aware algorithm design and energy-efficient data analysis.

Response:

We sincerely thank you for the thorough and encouraging evaluation. We are glad that the significance and novelty of our contributions have been well recognized, particularly in advancing memristor-based compute-in-memory architectures beyond traditional neural networks toward versatile, general-purpose decomposition algorithms. Guided by your specific comments, we have made substantial revisions to the manuscript, with detailed responses provided below.

Comment #1:

There are several concerns that have been listed as follows:

1. The authors report large gains in energy efficiency (50.7×) and normalized speed (37×) compared to GPU implementations. While impressive, these benchmarks are performed on small and relatively low-dimensional datasets such as IRIS (150×4) and NinaPro (864×500). It is unclear whether the same magnitude of efficiency gain would be observed for real-world, large-scale matrices (e.g., >10,000 dimensions). The authors somehow consider different applications, but not very different in size, to claim scalability.

Response:

Thank you for the valuable questions. In the original manuscript, we evaluated a diverse set of tasks with parameter scales ranging from $\sim 10^2$ to $\sim 10^5$, which covers a broad spectrum of edge-task settings. To further address the concern about scalability to substantially larger matrices/models, we additionally performed hardware-calibrated simulation on large-model fine-tuning tasks with parameter scales on the order of $\sim 10^9$, demonstrating the feasibility of MSVD in this regime. During simulation, we employ a hardware-calibrated framework that incorporates a noise-aware memristor model and peripheral circuit models—including quantization and shift-and-add operations—to emulate a larger memristor-based computing system.

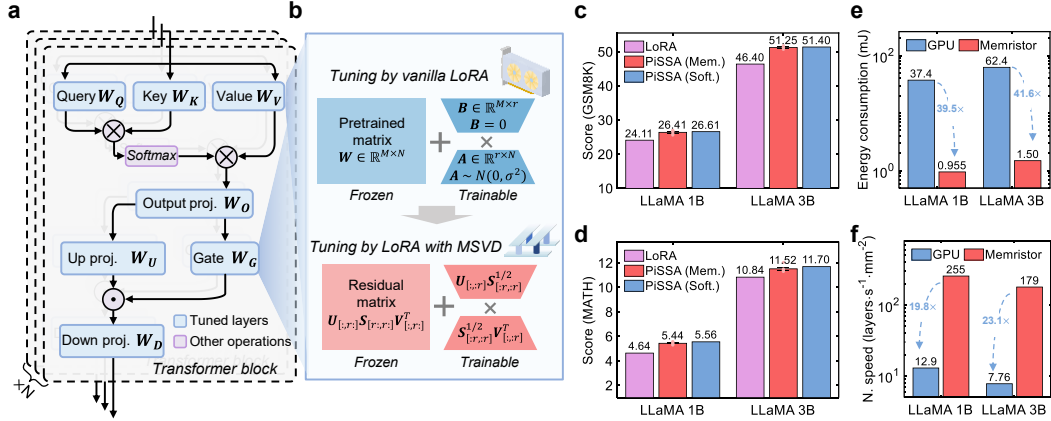


Fig. R1. Large language model low-rank adaptation with MSVD-based initialization. **a**, Architecture of transformer layer in LLaMA 3.2 models. **b**, PiSSA initialization using MSVD for low-rank adaptation (LoRA, red), compared to the vanilla LoRA (blue). **c,d**, Comparison of fine-tuning performance using different approaches on LLaMA 3.2 1B and 3B models across GSM8K and MATH datasets. The error bars indicate the standard error across 5 repeated experiments. **e,f**, Comparison of energy efficiency (**e**) and normalized speed (**f**) for PiSSA initialization on memristor versus GPU.

Table R1. Fine-tuning settings (a) and weight matrix dimensions (b) of LLaMA 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 $\times$ 1
LLaMA-3.2-3B	4	15	4	1024	2E-5	3125 (8)	A100 $\times$ 2

b

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

Specifically, we added LoRA-based experiments on LLaMA 3.2-1B and LLaMA 3.2-3B. Our implementation follows the recently popular PiSSA method¹. Conventional LoRA parameterizes the adaptation using a low-rank matrix structure, thereby producing low-rank updates during training. In contrast, PiSSA explicitly performs an SVD-based initialization of the LoRA weights, with the aim of accelerating training and improving optimization quality. As shown in Fig. R1, we implement the SVD required for PiSSA using memristors (MSVD). Specifically, we decompose the weights intended for fine-tuning, such as the linear-layer weights in the attention and MLP modules, to obtain the initialization factors required for tuning.

As illustrated in **Figs. R1a,b**, the target weight matrix is decomposed via SVD: the singular vectors corresponding to the dominant ranks are extracted and used to initialize the two low-rank LoRA matrices, while the remaining components are retained in the original (non-tuned) weight matrix. The detailed training configuration is provided in Table R1. Importantly, the hardware-calibrated simulation encompasses all weight matrices across every transformer layer, with matrix dimensions ranging from 2048 $\times$ 512 to 8192 $\times$ 3072 (**Table R1**). While this falls slightly below the reviewer’s suggested threshold of 10000, we note that 8192 represents the standard hidden dimension of mainstream open-source LLMs, and the consistent efficiency advantages observed across matrix sizes from 2048 to 8192 suggest that the gains are robust across this practically relevant scale range.

Fig. R2 reports the training gradients and loss curves for LLaMA 3.2-1B and LLaMA 3.2-3B under different methods on MetaMathQA dataset. The results show that MSVD with SREA, when used to realize PiSSA, achieves training behavior comparable to the software baseline, and is consistently more effective than vanilla LoRA.

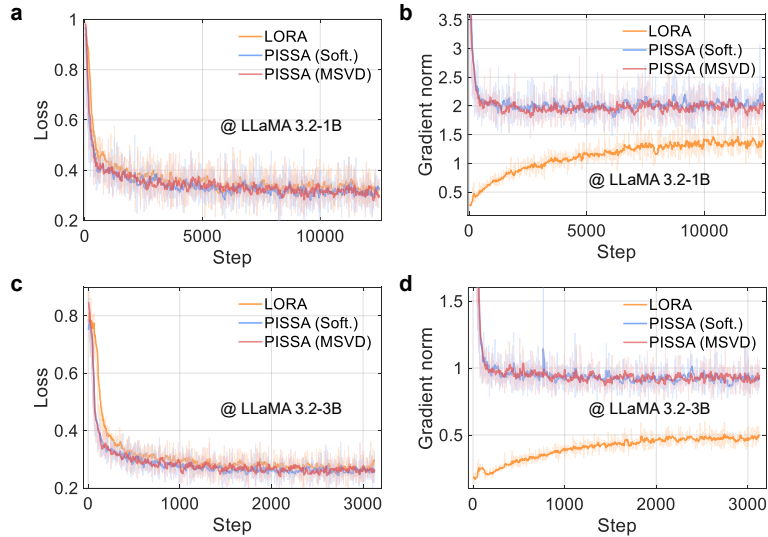


Fig. R2. Training dynamics of LLaMA 3.2 models using different fine-tuning initialization approaches on MetaMathQA dataset. Training loss curves and gradient norms for LLaMA 3.2-1B (a,b) and 3.2-3B (c,d) models, respectively.

Figs. R1c, R1d further compare performance on the GSM8K and MATH datasets among software-based PiSSA, PiSSA implemented with MSVD, and vanilla LoRA. The memristor-based PiSSA variants outperform vanilla LoRA, and PiSSA realized by MSVD with SREA achieves accuracy comparable to the software implementation. **Figs. R1e, R1f** present the benchmarks of energy efficiency and normalized speed. Compared with an NVIDIA A100 GPU, PiSSA with the memristor-based implementation achieves an energy-efficiency advantage of $39.5\times$ and $41.6\times$ for LLaMA 3.2-1B and LLaMA 3.2-3B, respectively, and a speed advantage of $19.8\times$ and $23.1\times$.

Furthermore, we would like to clarify an update regarding our overall benchmark evaluation. To provide a more rigorous and realistic comparison, we have further refined our benchmark modeling and adopted a more efficient A100 tensor core performance as baseline. Consequently, the previously reported $50.7\times$ energy efficiency and $37\times$ speedup for the smaller-scale tasks have been updated to $29.9\times$ and $19.0\times$, respectively. Despite comparing against this much stronger GPU baseline, our memristor-based implementation for large-scale LLM fine-tuning still achieves remarkable energy efficiency advantages ($39.5\times$ and $41.6\times$) and speedups ($19.8\times$ and $23.1\times$). These consistent, order-of-magnitude improvements across both small and large-scale tasks firmly validate the scalability and real-world potential of the proposed MSVD approach. We have updated the corresponding data and descriptions in the revised manuscript.

Overall, these experiments demonstrate the effectiveness of MSVD and SREA on substantially larger-scale tasks.

[1] Meng, F., Wang, Z. & Zhang, M. PiSSA: principal singular values and singular vectors adaptation of large language models. in Proceedings of the 38th International Conference on Neural Information Processing Systems. Article 3846 (Curran Associates Inc.).

Revisions:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the content in **Introduction** on page 2 as: “Recently, SVD-based biomedical information processing^{1,4} and large language model (LLM) adaptation have attracted great attention^{15,16}.”; on pages 3-4 as: “Furthermore, we validated the potential of MSVD in larger-scale applications through LLM tuning with MSVD-based initialization. Specifically, for the LLaMA 3.2-3B model, the MSVD-based method outperformed vanilla LoRA by 4.85 points (a 10.45% relative improvement) on the GSM8K⁵³ dataset and 0.68 points (a 6.27% relative improvement) on the MATH⁵⁴ dataset.”; on page 4 as: “Additionally, for LLaMA 3.2-3B model initialization, MSVD achieves $41.6\times$ higher energy efficiency and $23.1\times$ faster speed compared to CMOS-based hardware.”
2. We added the section “**LLM fine-tuning with MSVD initialization**” on pages 11-12 as: “

LLM fine-tuning with MSVD initialization

To further investigate the scalability of MSVD and SREA in large-scale applications, we evaluated their performance on the parameter-efficient fine-tuning of LLMs through simulation. As the scale of language models continues to grow, low-rank adaptation¹³

has emerged as a crucial technique, which mitigates prohibitive training costs by freezing the dense weights and introducing small, trainable low-rank matrices to capture task-specific updates. Crucially, initializing these matrices via SVD^{15,16} explicitly extracts the principal components of the pre-trained weights, effectively accelerating convergence and enhancing final performance. Along with the rapid development of memristor-based language model fine-tuning accelerators⁶⁵, executing efficient SVD directly on memristors has become highly critical to further optimize the initialization overhead and streamline the entire fine-tuning pipeline.

Here, we employed a simulation incorporating memristor variations and peripheral circuit behaviors to demonstrate principal singular values and singular vectors adaptation¹⁵ (PiSSA), a representative low-rank adaptation method that we specifically initialize with MSVD (**Figs. 7a,b**). In this approach, all static weights within the model (with dimensions such as 2048×2048 and 2048×512 , detailed in **Supplementary Table 5**) underwent rank-4 decomposition. The resulting matrices U , S , and V were then utilized to construct the branches for subsequent fine-tuning (see **Methods** for details). We applied MSVD-based PiSSA, with SREA (configurations detailed in **Supplementary Table 3**), to fine-tune LLaMA 3.2 models⁶⁶ (1B and 3B) on the MetaMathQA⁶⁷ dataset for two and one epoch, respectively, and then evaluated them on the GSM8K⁵³ and MATH⁵⁴ datasets. The loss curves and gradient norms (**Supplementary Figure 16**) during fine-tuning indicate that MSVD-based PiSSA effectively accelerates fine-tuning progress compared to vanilla LoRA, while maintaining optimization dynamics that closely align with the software-based PiSSA baseline. Regarding mathematical problem-solving capabilities, on the GSM8K (**Fig. 7c**) and MATH (**Fig. 7d**) benchmarks, MSVD-based PiSSA achieves results comparable to its software counterpart (e.g., 26.41 vs. 26.61 and 51.25 vs. 51.40 on GSM8K for 1B and 3B, respectively). Moreover, it consistently outperforms vanilla LoRA, improving the GSM8K score from 24.11 to 26.41 (a 9.5% relative gain) for the 1B model and from 46.40 to 51.25 (a 10.5% relative gain) for the 3B model. **Supplementary Figure 17** presents an ablation study of SREA on LLaMA 3.2-1B. The introduction of SREA yields a performance improvement of 1.09 points (a 4.3% relative gain) on the GSM8K dataset and 0.42 points (an 8.3% relative gain) on the MATH dataset, further confirming the efficacy of SREA in large-scale tasks. **Figs. 7e,f** illustrate the energy efficiency and normalized speed of MSVD-based initialization compared to GPUs. At the 32nm technology node, the memristor-based system achieves a $41.6 \times$ advantage in energy efficiency and a $23.1 \times$ speedup relative to GPUs. These results collectively demonstrate the scalability of MSVD to larger workloads and its feasibility for practical LLM applications."

3. We added section "MSVD initialization for low-rank adaptation" in **Methods** on pages 21-22: "

MSVD initialization for low-rank adaptation

Low-rank adaptation techniques typically partition the pre-trained model weights into two distinct components: a frozen (untrainable) residual matrix $\in \mathbb{R}^{M \times N}$, and two trainable low-rank branches, denoted as Branch_A $\in \mathbb{R}^{r \times N}$ and Branch_B $\in \mathbb{R}^{M \times r}$, where r represents the target low rank.

In our approach, we utilize MSVD to initialize these components. Specifically, we apply MSVD to the static weights in each transformer layer of LLMs, including the query, key, and value layers, the output-projection, up-projection, down-projection, and the gate layers, as illustrated in **Fig. 7a**. For each weight matrix, we perform a rank-4 MSVD, executing 15 iterations per rank. The resulting decomposition factors are then used to construct the PiSSA weight initialization. The trainable low-rank branches are

formed as:

$$\text{Branch}_A = \mathbf{S}_{[:,r]}^{1/2} \mathbf{V}_{[:,r]}^T, \text{Branch}_B = \mathbf{U}_{[:,r]} \mathbf{S}_{[:,r]}^{1/2} \quad (16)$$

where $\mathbf{S}$, $\mathbf{V}$, $\mathbf{U}$ represent solved singular values, right singular vectors and left singular vectors, respectively. Concurrently, the frozen residual matrix is obtained by subtracting the low-rank approximation from the original pre-trained weight matrix $\mathbf{W}$:

$$\text{Mat}_{\text{res}} = \mathbf{W} - \mathbf{U}_{[:,r]} \mathbf{S}_{[:,r]} \mathbf{V}_{[:,r]}^T \quad (17)$$

Detailed parameter configurations for fine-tuning, along with the specific weight dimensions of transformer layers are provided in **Supplementary Table 5.**"

4. In **Introduction**, on page 4, we added: "Additionally, for LLaMA 3.2-3B model initialization, MSVD achieves $41.6\times$ higher energy efficiency and $23.1\times$ faster speed compared to CMOS-based hardware."

5. On pages 18, we added: "For EEG-based sleep stage detection task and low-rank adaptation with MSVD initialization, we utilize a hardware-calibrated simulation framework that incorporates a noise-aware memristor model and peripheral circuit models (including quantization and shift-and-add operations) to emulate a larger memristor-based computing system. During these experiments, the device conductance mapping range is set from 0 to 30 μS . The device read and write variations are modeled as Gaussian-distributed noise based on our actual device measurements (**Fig. 2g**) to ensure fidelity."

6. In **Supplementary Note 6**, we added: "**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 244.8 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 954.5 μJ ($27.84 \mu\text{J} \times 2 + 82.28 \mu\text{J} \times 2 + 244.8 \mu\text{J} \times 3$), with a total computational load of 14.6 GOPs. The corresponding A100 GPU energy consumption is 37.4 mJ, representing a $39.2\times$ 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 ($62.17 \mu\text{J} \times 2 + 157.6 \mu\text{J} \times 2 + 352.5 \mu\text{J} \times 3$), with a total computational load of 24.3 GOPs. The corresponding A100 GPU energy consumption is 62.4 mJ, representing a $41.7\times$ energy efficiency advantage for MSVD."

7. In **Supplementary Note 8**, we added: "**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 ($0.451 \text{mm}^2 \times 2 + 1.15 \text{mm}^2 \times 2 + 2.94 \text{mm}^2 \times 3$), with a computational delay of 325 μs , yielding a normalized speed of 255 $\text{layers}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$. For the total computational load of 14.6 GOPs per layer, the A100 GPU achieves a normalized speed of 12.9 $\text{layers}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$, giving MSVD a $19.8\times$ 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 ($0.847 \text{mm}^2 \times 2 + 1.99 \text{mm}^2 \times 2$

+ $3.84 \text{ mm}^2 \times 3$), with the same computational delay of $325 \text{ } \mu\text{s}$, yielding a normalized speed of $179 \text{ layers} \cdot \text{s}^{-1} \cdot \text{mm}^{-2}$. For the total computational load of 24.3 GOPs per layer, the A100 GPU achieves a normalized speed of $7.76 \text{ layers} \cdot \text{s}^{-1} \cdot \text{mm}^{-2}$, resulting in a $23.1\times$ speed advantage for MSVD.”

8. We added **Fig. R1** as **Fig. 7** in manuscript.
9. We added **Fig. R2** as **Supplementary Figure 16**.
10. We added **Table R1** as **Supplementary Table 5**.

Comment #2

2. Image reconstruction results show PSNR = 31 dB and SSIM = 0.94, indicating moderate fidelity. However, state-of-the-art software methods regularly achieve >35 dB PSNR. This raises the question of whether the proposed memristor approach sacrifices too much accuracy for efficiency. A more direct comparison to recent algorithmic baselines in imaging would help contextualize the significance of this result.

Response:

Thank you for this important question. We would like to clarify that the >35 dB PSNR figures cited from state-of-the-art software methods are typically achieved under unconstrained, full-precision software environments, whereas our MSVD operates under the fundamental precision constraints of memristive hardware. A more meaningful comparison is therefore against methods operating under comparable computational constraints. To this end, we have added comparisons to two representative baselines: (i) a neural implicit image representation method (SIREN¹) and (ii) a compressed sensing method.

SIREN¹ is a neural network architecture that uses periodic sine functions as activation functions, enabling effective representation of complex signals and strong image reconstruction performance. For a fair comparison, we configured a SIREN network with a parameter count comparable to the data scale of our MSVD-based HKU logo reconstruction experiment, with SIREN layer dimensions of 80. Training employed the Adam optimizer with a learning rate of 1×10^{-3} for up to 5,000 steps, ultimately achieving a reconstruction PSNR of 29.47 dB (**Fig. R3**). Notably, SIREN requires thousands of iterative training steps to produce reconstruction results, making such computational overhead prohibitive for edge deployment.

Additionally, we provide a comparison against compressed sensing. We employed Block-DCT with a block size of 16×16 as the sparse basis and performed image compression at varying sampling ratios, with Lasso regularization applied as the reconstruction constraint. As shown in **Fig. R4**, at a sampling ratio comparable to our experiments (sampling ratio = 0.3), the recovery PSNR was only 15.4 dB, while a sampling ratio of 0.5 yielded 20 dB. Moreover, compressed sensing requires computationally intensive reconstruction procedures, whereas MSVD achieves recovery through a single matrix multiplication.

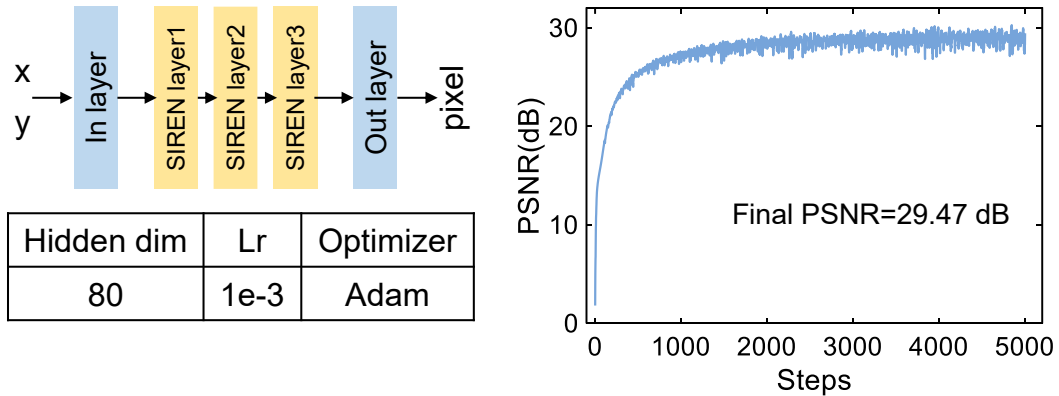


Fig. R3. SIREN setup and PSNR with training steps. The SIREN is set up with one input layer, one output layer and three hidden siren layers. The network is trained for 5000 steps and achieves a final PSNR of 29.47.

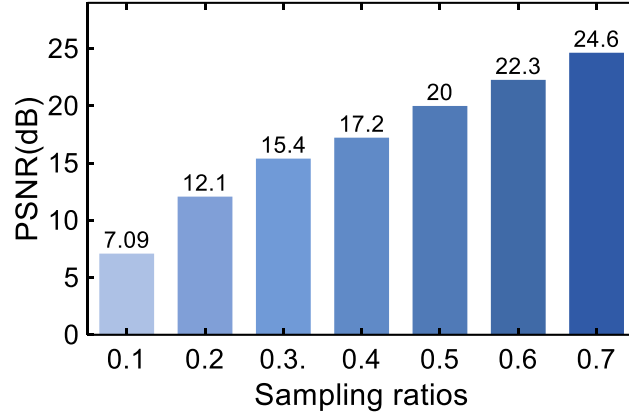


Fig. R4. Result of compressed sensing by block-DCT approach with different sampling ratios.

Taken together, under comparable parameter and compression constraints, our MSVD achieves a reconstruction quality that is highly competitive with these baselines. Furthermore, MSVD accomplishes this with significantly lower computational overhead, requiring only a single matrix multiplication for recovery, demonstrating that it provides a practical and efficient solution for edge computing applications. Looking ahead, the reconstruction quality of MSVD can be further improved through finer-grained DPM and increased VPR redundancy, as well as ongoing optimization of device characteristics.

[1] Sitzmann, V., Martel, J., Bergman, A., Lindell, D. & Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33, 7462-7473 (2020).

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On pages 7-8, we added: “Compared to neural network-based⁵⁸ and compressed sensing⁵⁹ reconstruction methods (**Supplementary Figures 8, 9**), MSVD-based low-rank reconstruction offers edge-friendly efficiency by avoiding the large-scale parameters and extensive training of neural networks, as well as the complex reconstruction algorithms associated with compressed sensing. For applications demanding higher reconstruction accuracy, system performance can be further enhanced through finer-grained DPM and increased VPR redundancy. Moreover, as memristor device characteristics continue to improve, the overall performance of the MSVD-based framework is expected to advance accordingly.”
2. We added **Fig. R3** as **Supplementary Figure 8**.
3. We added **Fig. R4** as **Supplementary Figure 9**.

Comment #3

3. Gesture recognition with incremental MSVD achieves 88.26% for 2 users and ~84% for 8 users. While the numbers are promising, they are based on NinaPro, a small, well-controlled EMG dataset. Real-world EMG signals are significantly noisier, and cross-user variability is much larger. Without evaluation on more diverse datasets or field-collected data, the generalizability of these accuracy results is uncertain.

Response:

Thank you for this important comment. While obtaining additional diverse EMG field data remains challenging, we address the reviewer's concern on generalizability by validating our approach on EEG-based sleep stage detection using the Sleep-EDFx dataset, a field-collected, cross-subject dataset with substantial real-world variability that provides a more rigorous test of cross-subject generalization.

This subset contains electroencephalogram (EEG) signals recorded from different subjects at home using cassette-tape recorders during wakefulness, rapid eye movement (REM) sleep, and deep sleep (non-REM stage 3), providing real-world field-collected data with inherent environmental noise and substantial cross-subject variability. In the hardware-calibrated simulation framework, we incorporated noise levels measured from actual devices ($\sigma_{\text{write}} = 0.27 \mu\text{S}$, $\sigma_{\text{read}} = 0.21 \mu\text{S}$). The signals processed by short-time Fourier transform were sent to the memristor-based system for feature extraction via MSVD/incremental MSVD, and the extracted features were subsequently classified using logistic regression (**Fig. R5a**).

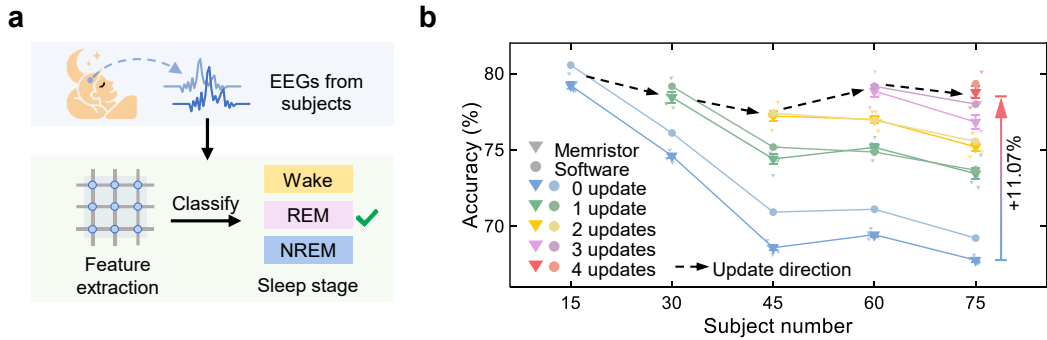


Fig. R5. Subject-scalable EEG-based sleep stage detection with incremental MSVD. a, Schematic of sleep stage detection with MSVD and incremental MSVD. **b,** Recognition accuracy evolution as subject number increases from 15 to 75. The error bars indicate the standard error across 5 repetitions.

To evaluate scalability and cross-subject generalization, we incrementally increased the training and testing data from 15 to 75 subjects over five successive updates. At each update, incremental MSVD was used to obtain the feature extractor for all current subjects and to update the classifier. Each data matrix and incremental matrix had dimensions of 900×500 , containing information from 15 subjects. MSVD/incremental MSVD yielded 500×5 right singular vectors (RSVs) for feature extraction. The SREA configuration is detailed in **Table R2**.

Table R2. DPM and VPR settings in EEG-based sleep stage detection.

TASK	DPM	VPR	
		Additional bits	Repeated vectors
Sleep-EDFx	[3,5]	6	16

Fig. R5 presents the incremental learning results, comparing memristor-based and software implementations at each stage. After four rounds of subject updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline. Meanwhile, the memristor-based system demonstrated performance comparable to software implementation, validating its effectiveness on diverse, real-world data with substantial cross-subject variability.

Taken together, these results directly address the three concerns raised: the use of field-collected EEG data captures inherent environmental noise, the cross-subject design spanning 75 subjects reflects substantial inter-subject variability, and the scale of the dataset demonstrates generalizability beyond small, well-controlled benchmarks.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. In Introduction on page 2, we added: “Recently, SVD-based biomedical information processing^{1,4} and large language model (LLM) adaptation have attracted great attention^{15,16}.”; on page 3, we added: “For edge learning applications, we developed an incremental MSVD method for user-scalable gesture recognition and sleep stage detection using biomedical signals. The incremental approach maintained robust performance, surpassing non-incremental approaches by 19.29% and 11.07% in accuracy across in these two tasks, respectively.”
2. We added the context on page 9: “We further explored the scalability of incremental MSVD through EEG-based sleep state detection with a larger cohort of 75 subjects, using a hardware-calibrated simulation framework (**Fig. 5g** and **Methods**). This experiment leveraged the Sleep-EDFx dataset⁶³, which contains EEG signals recorded from subjects during wakefulness, rapid eye movement (REM) sleep, and deep sleep (non-REM stage 3). In our experiment, EEG signals processed by short-time Fourier transform were fed into the memristor-based system for MSVD/incremental MSVD feature extraction, followed by logistic regression classification. Similar to EMG-based experiment, at each update, incremental MSVD refined the feature extractor to incorporate all current subjects and updated the classifier accordingly. Both the original data matrix and incremental updated matrices had dimensions of 900×500, each containing information from 15 subjects. MSVD/incremental MSVD yielded RSVs with dimensions of 500×5 (solved with 15 iterations per vector) for feature extraction. After four steps of subject updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline, while the memristor-based system demonstrated performance comparable to software implementation (**Fig. 5f**).”
3. We added benchmarks on page 18: “Additionally, energy benchmarking performed on incremental MSVD for EEG-based sleep stage detection demonstrates a 57.3× advantage (84.6 μJ vs. 4852 μJ) over the A100 GPU and a 47.7× advantage (84.6 μJ vs. 4035 μJ) over the Google TPUv3 (**Supplementary Figure 29**).”; on page 19: “Additionally, in the context of EEG-based sleep stage detection, incremental MSVD demonstrates a 21.0× average speedup (8744 updates·s⁻¹·mm⁻² vs. 416 updates·s⁻¹·mm⁻²) relative to the TPUv3 (**Supplementary Figure 29**).”
4. We revised the **Figs. 5f,b** and their captions as follows: “

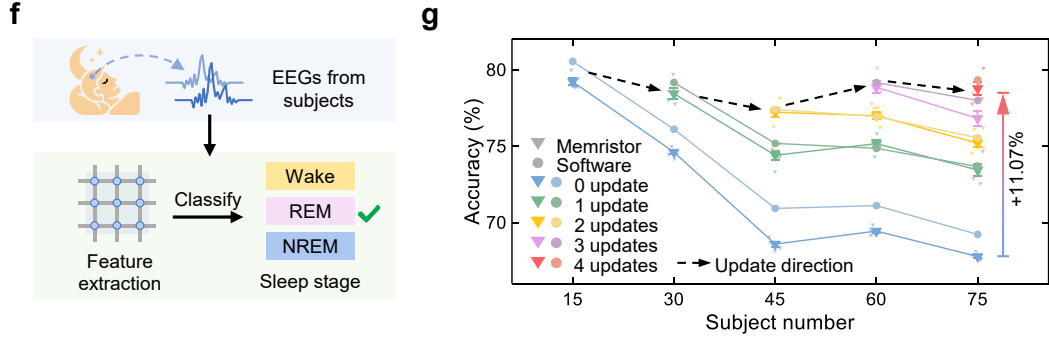


Fig. 5 | User-scalable recognition with incremental MSVD. ... f, Schematic of EEG-based subject-scalable sleep stage detection with incremental MSVD. g, Detection accuracy evolution as subject number increases from 15 to 75. The error bars indicate the standard error across 5 repetitions.”

5. In **Supplementary Note 6**, we added: “**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.”

6. In **Supplementary Note 8**, we added: “**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 $\text{updates}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$. 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 $\text{updates}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$, with an average of 448 $\text{updates}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$, resulting in a 19.5× advantage for incremental MSVD. For the TPU v3, the normalized speeds are 648, 433, 324, and 260 $\text{updates}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$, with an average of 416 $\text{updates}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$, corresponding to a 21.0× advantage.”

7. We added the content in **Table R2** into **Supplementary Table 3**.

Comment #4

4. The incremental MSVD framework improves performance by ~19% compared to baseline non-incremental approaches. However, the scalability is demonstrated only up to 8 users. This limited scope does not convincingly support the broader claim that MSVD is a “scalable framework.” The reviewer suggests expanding the number of users (e.g., 50+) or simulating synthetic large-scale incremental additions to demonstrate true scalability.

Response:

We appreciate this feedback on scalability validation. As detailed in our response to Comment #3, we have conducted additional experiments using the Sleep-EDFx EEG dataset, incrementally scaling from 15 to 75 subjects over five successive updates. This directly addresses the reviewer’s suggestion of demonstrating scalability with 50+ users. The SREA configuration used in this experiment is detailed in **Table R3**. In the simulation, we incorporated noise levels measured from actual ($\sigma_{\text{write}} = 0.27 \mu\text{S}$, $\sigma_{\text{read}} = 0.21 \mu\text{S}$). After four rounds of incremental updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline, while maintaining performance comparable to the software implementation (**Fig. R6**). These results establish that incremental MSVD scales efficiently to 75 subjects while sustaining robust recognition performance, directly validating the scalability claims in the manuscript.

Table R3. DPM and VPR settings in EEG-based sleep stage detection.

TASK	DPM	VPR	
		Additional bits	Repeated vectors
Sleep-EDFx	[3,5]	6	16

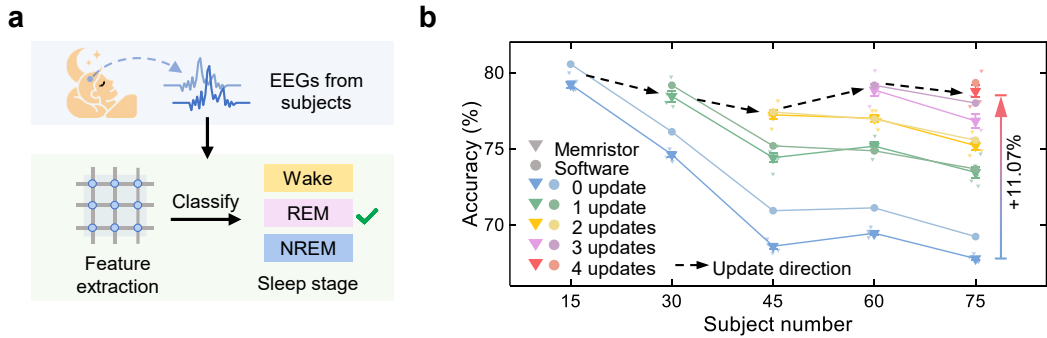


Fig. R6 (Fig. R5). Subject-scalable EEG-based sleep stage detection with incremental MSVD. a, Schematic of sleep stage detection with MSVD and incremental MSVD. **b,** Recognition accuracy evolution as subject number increases from 15 to 75. The error bars indicate the standard error across 5 repetitions.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On page 8, we revised the subsection “**Incremental learning with MSVD**” as: “This has led to its integration in wearable medical devices for analyzing physiological signals, such as electromyogram (EMG) and electroencephalogram (EEG). As the user or subject base expands, these systems must classify increasingly diverse signal patterns.

However, physiological signals vary significantly between individuals^{60,61}, with the same underlying activity producing distinct signal patterns across different users. This variability complicates the application of feature extractors and classifiers trained on existing users to new users.”

2. We added the context on page 9: “We further explored the scalability of incremental MSVD through EEG-based sleep state detection with a larger cohort of 75 subjects, using a hardware-calibrated simulation framework (**Fig. 5g** and **Methods**). This experiment leveraged the Sleep-EDFx dataset⁶³, which contains EEG signals recorded from subjects during wakefulness, rapid eye movement (REM) sleep, and deep sleep (non-REM stage 3). In our experiment, EEG signals processed by short-time Fourier transform were fed into the memristor-based system for MSVD/incremental MSVD feature extraction, followed by logistic regression classification. Similar to EMG-based experiment, at each update, incremental MSVD refined the feature extractor to incorporate all current subjects and updated the classifier accordingly. Both the original data matrix and incremental updated matrices had dimensions of 900×500 , each containing information from 15 subjects. MSVD/incremental MSVD yielded RSVs with dimensions of 500×5 (solved with 15 iterations per vector) for feature extraction. The SREA configurations for MSVD and incremental MSVD are detailed in **Supplementary Table 3**. After four steps of subject updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline, while the memristor-based system demonstrated performance comparable to software implementation (**Fig. 5f**).”

3. We added benchmarks on page 18: “Additionally, energy benchmarking performed on incremental MSVD for EEG-based sleep stage detection demonstrates a $57.3 \times$ advantage ($84.6 \mu\text{J}$ vs. $4852 \mu\text{J}$) over the A100 GPU and a $47.7 \times$ advantage ($84.6 \mu\text{J}$ vs. $4035 \mu\text{J}$) over the Google TPUv3 (**Supplementary Figure 29**).”; on page 19: “Additionally, in the context of EEG-based sleep stage detection, incremental MSVD demonstrates a $21.0 \times$ average speedup ($8744 \text{ updates} \cdot \text{s}^{-1} \cdot \text{mm}^{-2}$ vs. $416 \text{ updates} \cdot \text{s}^{-1} \cdot \text{mm}^{-2}$) relative to the TPUv3 (**Supplementary Figure 29**).”

4. We revised the **Fig. 5** as mentioned before.

5. We added the content in **Table R3** into **Supplementary Table 3**.

Comment #5

5. The hardware benefits are presented primarily in comparison to GPU baselines, including NVIDIA A100. GPUs are general-purpose architectures not optimized for matrix decompositions. The lack of comparisons with dedicated ASIC or FPGA accelerators limits the strength of the argument. Without such comparisons, the reviewer feels the claimed hardware efficiency could be overstated. Supplementary Table 1 provides a useful survey of prior memristor-based computing works. However, this comparison is descriptive only — no quantitative metrics (e.g., energy per iteration, error rates) are provided. Without direct side-by-side benchmarking, the claimed advantages of MSVD remain qualitative.

Response:

Thank you for this suggestion about more comprehensive comparisons. Regarding comparisons to dedicated hardware accelerators, we have now included quantitative comparisons with state-of-the-art ASIC implementations in **Table R4**. Compared to current state-of-the-art ASIC implementations for SVD and eigenvalue decomposition (EVD), MSVD demonstrates significant advantages in normalized energy efficiency ($215 \text{ Elem} \cdot \mu\text{J}^{-1}$ vs. $28.2 \text{ Elem} \cdot \mu\text{J}^{-1}$) and throughput ($6.5 \times 10^4 \text{ Elem} \cdot \text{s}^{-1}$ vs. $1.5 \times 10^2 \text{ Elem} \cdot \text{s}^{-1}$). Furthermore, relative to CMOS-based SVD implementations, our incremental MSVD demonstration processed matrices with total dimensions of 3456×500 ($4 \times 864 \times 500$), exhibiting superior scalability. Regarding FPGA-based accelerators, we note that publicly available FPGA implementations specifically designed for SVD or EVD decomposition are limited in the literature, and ASIC designs represent a more rigorous and representative benchmark for dedicated hardware comparison.

Table R4. Comparison of MSVD to other CMOS-based von Neumann ASIC designed for decomposition applications¹⁻⁴.

Works	Application	Architecture	Benchmark tech. node (nm)	Problem scale	Decomposition efficiency* (Vec·uJ ⁻¹)	Norm. decomposition efficiency# (Elem·uJ ⁻¹)	Decomposition Speed* (kVec·s ⁻¹ ·mm ⁻²)	Norm. decomposition speed# (kElem·s ⁻¹ ·mm ⁻²)
TCAS I 2019	SVD	CMOS-based von Neumann	40	8×8	3.53	28.2	-	-
TCAS I 2023	EVD	CMOS-based von Neumann	28	64×64	0.018	1.15	7.4×10^{-1}	4.7×10^1
TCAS I 2024	SVD	CMOS-based von Neumann	90	16×8×8 (Tensor)	0.33	2.64	5.9×10^0	4.7×10^1
Microelectron. J. 2025	EVD	CMOS-based von Neumann	22	1024×1024	0.0048	4.92	1.5×10^{-1}	1.5×10^2
This work	SVD	Memristor-based CIM	32	864×500	0.43	215	1.3×10^2	6.5×10^4

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 address the reviewer’s concern regarding quantitative benchmarking, we have updated original **Supplementary Table 1** with energy efficiency comparisons against the benchmark results reported by other memristor-based works (as shown in **Table R5**). Then we normalizing each result to the A100 GPU baseline at the corresponding precision level to show an apple-to-apple comparison. The results show that our work uniquely combines multi-level iterative decomposition capability with outstanding energy efficiency.

Table R5. Comparison with some representative memristor-based systems

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

- [1] 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).
- [2] 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). <https://doi.org/10.1109/TCSI.2023.3268314>
- [3] 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).
- [4] Huang, T. et al. A power-iteration-based beamformer for large-scale antenna arrays. *Microelectron J* 166, 106868 (2025). <https://doi.org/https://doi.org/10.1016/j.mejo.2025.106868>

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On page 18, we added: “**Supplementary Table 7** further presents an energy efficiency comparison against other CMOS-based decomposition ASICs; despite variations in task scale, the advantage of MSVD remains evident.”
2. On page 19, we added “Additionally, regarding the speed comparison against CMOS-based ASICs, as shown in **Supplementary Table 7**, MSVD exhibits a notable lead, despite the difference in problem size.”
3. We added **Table R4** as **Supplementary Table 7**.
4. We replaced **Supplementary Table 1** with **Table R5**.

Comment #6

6. The authors demonstrate applications in three domains: imaging, temporal COVID-19 data, and gesture recognition. While these domains show diversity, they remain confined to relatively narrow use cases. Singular value decomposition has far broader importance (e.g., recommendation systems, low-rank adaptation in transformers, PCA in genomics). Expanding the application discussion to such areas would make the contribution more significant for a wider audience.

Response:

We appreciate this suggestion and have significantly expanded our evaluation to include two additional application domains: health monitoring via EEG-based sleep stage detection and low-rank adaptation for large language models (LLMs), covering both edge-based feature extraction tasks and training tasks in larger-scale computing systems, as shown in **Fig. R7**, **Fig. R8**, and **Fig. R9**. We note that while the reviewer also suggested recommendation systems and PCA in genomics, we selected these two domains as the most representative expansions: LLM low-rank adaptation is among the most impactful current applications of SVD in large-scale AI, and EEG-based health monitoring addresses biomedical signal processing with substantial cross-subject variability and real-world noise. In these experiments, MSVD/incremental MSVD with SREA achieved software-comparable performance while demonstrating significant advantages in energy efficiency and speed compared to CMOS-based high-performance computing (HPC) systems (**Fig. R8**, **Fig. R9**). Consequently, our demonstrated applications now encompass image reconstruction, temporal data processing, biomedical feature extraction, and large language model adaptation, spanning diverse data modalities and application scales from edge devices to large-scale AI systems.

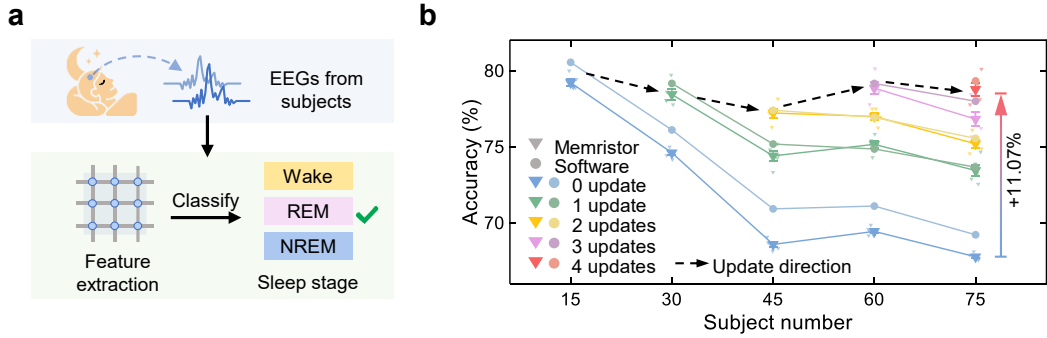


Fig. R7 (Fig. R5). Subject-scalable EEG-based sleep stage detection with incremental MSVD. a, Schematic of sleep stage detection with MSVD and incremental MSVD. **b**, Recognition accuracy evolution as subject number increases from 15 to 75. The error bars indicate the standard error across 5 repetitions.

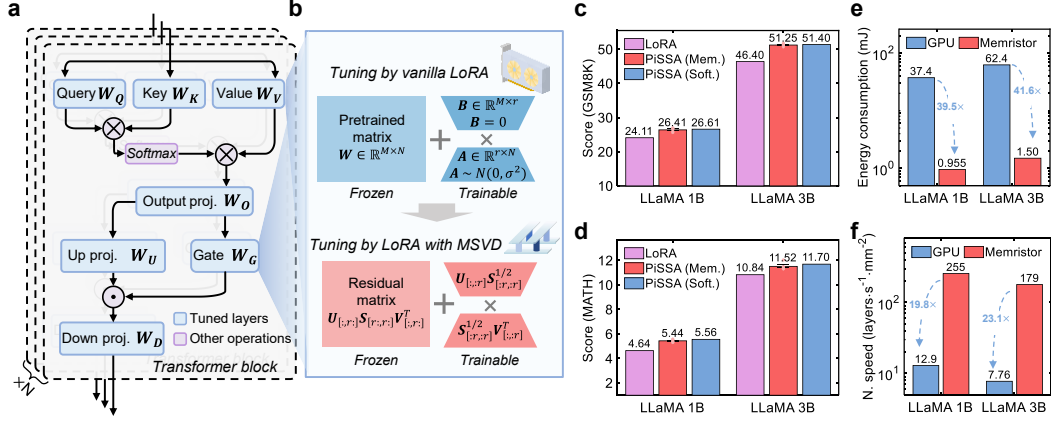


Fig. R8 (Fig. R1). Large language model low-rank adaptation with MSVD-based initialization. **a**, Architecture of transformer layer in LLaMA 3.2 models. **b**, PiSSA initialization using MSVD for low-rank adaptation (LoRA, red), compared to the vanilla LoRA (blue). **c,d**, Comparison of fine-tuning performance using different approaches on LLaMA 3.2 1B and 3B models across GSM8K and MATH datasets. The error bars indicate the standard error across 5 repeated experiments. **e,f**, Comparison of energy efficiency (**e**) and normalized speed (**f**) for PiSSA initialization on memristor versus GPU.

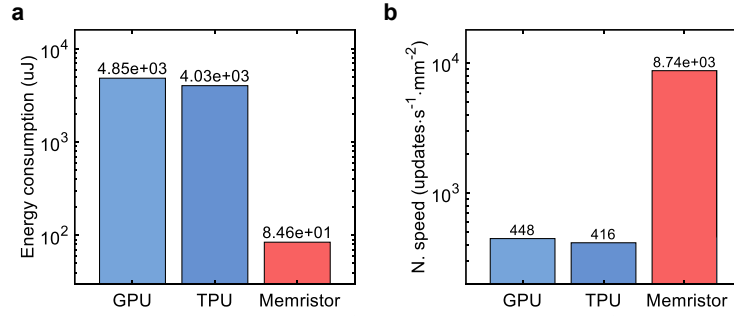


Fig. R9. Benchmarking incremental MSVD against A100 GPU and Google TPUv3 for EEG-based sleep stage detection. **a**, Energy consumption during updates. **b**, Normalized speed.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the context on page 9: “We further explored the scalability of incremental MSVD through EEG-based sleep state detection with a larger cohort of 75 subjects, using a hardware-calibrated simulation framework (Fig. 5g and Methods). This experiment leveraged the Sleep-EDFx dataset⁶³, which contains EEG signals recorded from subjects during wakefulness, rapid eye movement (REM) sleep, and deep sleep (non-REM stage 3). In our experiment, EEG signals processed by short-time Fourier transform were fed into the memristor-based system for MSVD/incremental MSVD feature extraction, followed by logistic regression classification. Similar to EMG-based experiment, at each update, incremental MSVD refined the feature extractor to incorporate all current subjects and updated the classifier accordingly. Both the original data matrix and incremental updated matrices had dimensions of 900×500, each containing information from 15 subjects. MSVD/incremental MSVD yielded RSVs with dimensions of 500×5 (solved with 15 iterations per vector) for feature extraction. The SREA configurations for MSVD and incremental MSVD are detailed

in **Supplementary Table 3**. After four steps of subject updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline, while the memristor-based system demonstrated performance comparable to software implementation (**Fig. 5f**)."

2. We added benchmarks on page 20: "Additionally, energy benchmarking performed on incremental MSVD for EEG-based sleep stage detection demonstrates a $57.3\times$ advantage ($84.6\ \mu\text{J}$ vs. $4852\ \mu\text{J}$) over the A100 GPU and a $47.7\times$ advantage ($84.6\ \mu\text{J}$ vs. $4035\ \mu\text{J}$) over the Google TPUv3 (**Supplementary Figure 29**)."; on page 19: "Additionally, in the context of EEG-based sleep stage detection, incremental MSVD demonstrates a $21.0\times$ average speedup ($8744\ \text{updates}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$ vs. $416\ \text{updates}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$) relative to the TPUv3 (**Supplementary Figure 29**)."

3. We added the section "**LLM fine-tuning with MSVD initialization**" on pages 11-12 as: "

LLM fine-tuning with MSVD initialization

To further investigate the scalability of MSVD and SREA in large-scale applications, we evaluated their performance on the parameter-efficient fine-tuning of LLMs through simulation. As the scale of language models continues to grow, low-rank adaptation¹³ has emerged as a crucial technique, which mitigates prohibitive training costs by freezing the dense weights and introducing small, trainable low-rank matrices to capture task-specific updates. Crucially, initializing these matrices via SVD^{15,16} explicitly extracts the principal components of the pre-trained weights, effectively accelerating convergence and enhancing final performance. Along with the rapid development of memristor-based language model fine-tuning accelerators⁶⁵, executing efficient SVD directly on memristors has become highly critical to further optimize the initialization overhead and streamline the entire fine-tuning pipeline.

Here, we employed a simulation incorporating memristor variations and peripheral circuit behaviors to demonstrate principal singular values and singular vectors adaptation¹⁵ (PiSSA), a representative low-rank adaptation method that we specifically initialize with MSVD (**Figs. 7a,b**). In this approach, all static weights within the model (with dimensions such as 2048×2048 and 2048×512 , detailed in **Supplementary Table 5**) underwent rank-4 decomposition. The resulting matrices **U**, **S**, and **V** were then utilized to construct the branches for subsequent fine-tuning (see **Methods** for details). We applied MSVD-based PiSSA, with SREA (configurations detailed in **Supplementary Table 3**), to fine-tune LLaMA 3.2 models⁶⁶ (1B and 3B) on the MetaMathQA⁶⁷ dataset for two and one epoch, respectively, and then evaluated them on the GSM8K⁵³ and MATH⁵⁴ datasets. The loss curves and gradient norms (**Supplementary Figure 16**) during fine-tuning indicate that MSVD-based PiSSA effectively accelerates fine-tuning progress compared to vanilla LoRA, while maintaining optimization dynamics that closely align with the software-based PiSSA baseline. Regarding mathematical problem-solving capabilities, on the GSM8K (**Fig. 7c**) and MATH (**Fig. 7d**) benchmarks, MSVD-based PiSSA achieves results comparable to its software counterpart (e.g., 26.41 vs. 26.61 and 51.25 vs. 51.40 on GSM8K for 1B and 3B, respectively). Moreover, it consistently outperforms vanilla LoRA, improving the GSM8K score from 24.11 to 26.41 (a 9.5% relative gain) for the 1B model and from 46.40 to 51.25 (a 10.5% relative gain) for the 3B model. **Supplementary Figure 17** presents an ablation study of SREA on LLaMA 3.2-1B. The introduction of SREA yields a performance improvement of 1.09 points (a 4.3% relative gain) on the GSM8K dataset and 0.42 points (an 8.3% relative gain) on the MATH dataset, further confirming the efficacy of SREA in large-scale tasks. **Figs. 7e,f** illustrate the energy efficiency and normalized speed of MSVD-based initialization compared to

GPUs. At the 32nm technology node, the memristor-based system achieves a $41.6\times$ advantage in energy efficiency and a $23.1\times$ speedup relative to GPUs. These results collectively demonstrate the scalability of MSVD to larger workloads and its feasibility for practical LLM applications.”

4. We added section “**MSVD initialization for low-rank adaptation**” in **Methods** on pages 21-22: “

MSVD initialization for low-rank adaptation

Low-rank adaptation techniques typically partition the pre-trained model weights into two distinct components: a frozen (untrainable) residual matrix $\in \mathbb{R}^{M \times N}$, and two trainable low-rank branches, denoted as Branch_A $\in \mathbb{R}^{r \times N}$ and Branch_B $\in \mathbb{R}^{M \times r}$, where r represents the target low rank.

In our approach, we utilize MSVD to initialize these components. Specifically, we apply MSVD to the static weights in each transformer layer of LLMs, including the query, key, and value layers, the output-projection, up-projection, down-projection, and the gate layers, as illustrated in **Fig. 7a**. For each weight matrix, we perform a rank-4 MSVD, executing 15 iterations per rank. The resulting decomposition factors are then used to construct the PiSSA weight initialization. The trainable low-rank branches are formed as:

$$\text{Branch}_A = \mathbf{S}_{[:,r]}^{1/2} \mathbf{V}_{[:,r]}^T, \text{Branch}_B = \mathbf{U}_{[:,r]} \mathbf{S}_{[:,r]}^{1/2}, \quad (16)$$

where $\mathbf{S}$, $\mathbf{V}$, $\mathbf{U}$ represent solved singular values, right singular vectors and left singular vectors, respectively. Concurrently, the frozen residual matrix is obtained by subtracting the low-rank approximation from the original pre-trained weight matrix $\mathbf{W}$:

$$\text{Mat}_{\text{res}} = \mathbf{W} - \mathbf{U}_{[:,r]} \mathbf{S}_{[:,r]} \mathbf{V}_{[:,r]}^T \quad (17)$$

Detailed parameter configurations for fine-tuning, along with the specific weight dimensions of transformer layers are provided in **Supplementary Table 5.**”

5. In **Introduction**, on page 4, we added: “Additionally, for LLaMA 3.2-3B model initialization, MSVD achieves $41.6\times$ higher energy efficiency and $23.1\times$ faster speed compared to CMOS-based hardware.”

6. We revised **Figs. 5f,g** as mentioned before.

7. We added **Fig. R8** as **Fig. 7**.

8. We added **Fig. R9** as **Supplementary Figure 29**.

9. In **Discussion** section on page 11, we revised the context as: “Experimental demonstrations across image reconstruction, temporal data analysis, biomedical signal processing and efficient LLM fine-tuning show that our approach achieves software-comparable accuracy.”

Comment #7

7. The authors mainly emphasize MSVD's role in edge AI and wearables. While important, the broader relevance of SVD in modern ML is overlooked. For example, SVD is foundational for LoRA (Low-Rank Adaptation) in large language models, matrix factorization in recommender systems, and dimensionality reduction in high-throughput genomics. Relating MSVD to these broader fields would increase cross-disciplinary impact.

Response:

Thank you for this insightful suggestion regarding the broader applicability of MSVD beyond edge AI and wearables. Following the reviewer's suggestion, we have expanded our work to demonstrate MSVD's opportunity in low-rank adaption of large language models.

Specifically, we added LoRA-based experiments on LLaMA 3.2-1B and LLaMA 3.2-3B. Our implementation follows the recently popular PiSSA method. Conventional LoRA parameterizes the adaptation using a low-rank matrix structure, thereby producing low-rank updates during training. In contrast, PiSSA explicitly performs an SVD-based initialization of the LoRA weights, with the aim of accelerating training and improving optimization quality. As shown in **Fig. R10**, we implement the SVD required for PiSSA using memristors (MSVD). Specifically, we decompose the weights targeted for fine-tuning (e.g., linear-layer weights in the attention and MLP modules), thereby obtaining the initialization factors for tuning.

As illustrated in **Fig. R10**, the target weight matrix is decomposed via SVD: the singular vectors corresponding to the dominant ranks are extracted and used to initialize the two low-rank LoRA matrices, while the remaining components are retained in the original (non-tuned) weight matrix. The results show that MSVD with SREA, when used to realize PiSSA, achieves training behavior comparable to the software baseline, and is consistently more effective than MSVD without SREA and vanilla LoRA.

Figs. R10c,d further compare performance on the GSM8K and MATH datasets among software-based PiSSA, PiSSA implemented with MSVD, and vanilla LoRA. The memristor-based PiSSA variants outperform vanilla LoRA, and PiSSA realized by MSVD with SREA achieves accuracy comparable to the software implementation. **Figs. R10e,f** present the benchmarks of energy-efficiency and normalized speed. Compared with an NVIDIA A100 GPU, PiSSA with the memristor-based implementation for LLaMA 3.2-3B achieves an energy-efficiency advantage of 41.6× and a speed advantage of 23.1×.

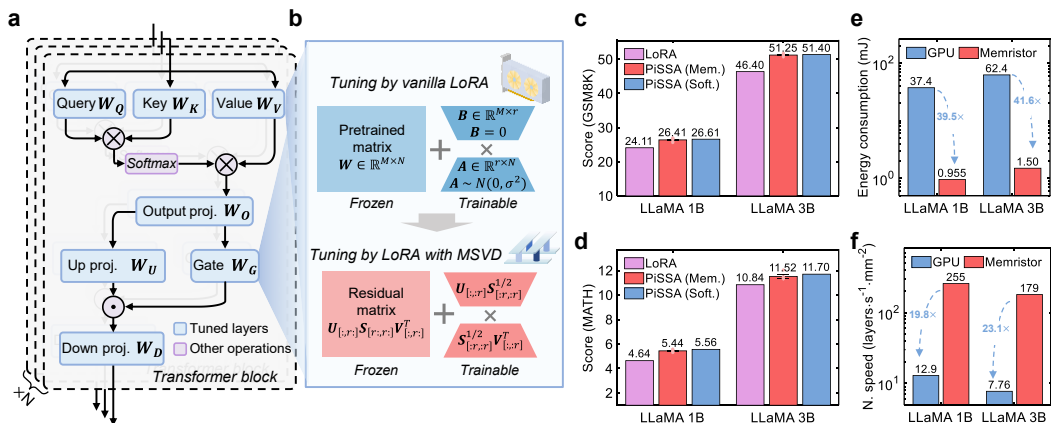


Fig. R10 (Fig. R1). Large language model low-rank adaptation with MSVD-based

initialization. a, Architecture of transformer layer in LLaMA 3.2 models. **b**, PiSSA initialization using MSVD for low-rank adaptation (LoRA, red), compared to the vanilla LoRA (blue). **c,d**, Comparison of fine-tuning performance using different approaches on LLaMA 3.2 1B and 3B models across GSM8K and MATH datasets. **e,f**, Comparison of energy efficiency (**e**) and normalized speed (**f**) for PiSSA initialization on memristor versus GPU.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the section “**LLM fine-tuning with MSVD initialization**” on pages 11-12 as: “

LLM fine-tuning with MSVD initialization

To further investigate the scalability of MSVD and SREA in large-scale applications, we evaluated their performance on the parameter-efficient fine-tuning of LLMs through simulation. As the scale of language models continues to grow, low-rank adaptation¹³ has emerged as a crucial technique, which mitigates prohibitive training costs by freezing the dense weights and introducing small, trainable low-rank matrices to capture task-specific updates. Crucially, initializing these matrices via SVD^{15,16} explicitly extracts the principal components of the pre-trained weights, effectively accelerating convergence and enhancing final performance. Along with the rapid development of memristor-based language model fine-tuning accelerators⁶⁵, executing efficient SVD directly on memristors has become highly critical to further optimize the initialization overhead and streamline the entire fine-tuning pipeline.

Here, we employed a simulation incorporating memristor variations and peripheral circuit behaviors to demonstrate principal singular values and singular vectors adaptation¹⁵ (PiSSA), a representative low-rank adaptation method that we specifically initialize with MSVD (**Figs. 7a,b**). In this approach, all static weights within the model (with dimensions such as 2048×2048 and 2048×512 , detailed in **Supplementary Table 5**) underwent rank-4 decomposition. The resulting matrices **U**, **S**, and **V** were then utilized to construct the branches for subsequent fine-tuning (see **Methods** for details). We applied MSVD-based PiSSA, with SREA (configurations detailed in **Supplementary Table 3**), to fine-tune LLaMA 3.2 models⁶⁶ (1B and 3B) on the MetaMathQA⁶⁷ dataset for two and one epoch, respectively, and then evaluated them on the GSM8K⁵³ and MATH⁵⁴ datasets. The loss curves and gradient norms (**Supplementary Figure 16**) during fine-tuning indicate that MSVD-based PiSSA effectively accelerates fine-tuning progress compared to vanilla LoRA, while maintaining optimization dynamics that closely align with the software-based PiSSA baseline. Regarding mathematical problem-solving capabilities, on the GSM8K (**Fig. 7c**) and MATH (**Fig. 7d**) benchmarks, MSVD-based PiSSA achieves results comparable to its software counterpart (e.g., 26.41 vs. 26.61 and 51.25 vs. 51.40 on GSM8K for 1B and 3B, respectively). Moreover, it consistently outperforms vanilla LoRA, improving the GSM8K score from 24.11 to 26.41 (a 9.5% relative gain) for the 1B model and from 46.40 to 51.25 (a 10.5% relative gain) for the 3B model. **Supplementary Figure 17** presents an ablation study of SREA on LLaMA 3.2-1B. The introduction of SREA yields a performance improvement of 1.09 points (a 4.3% relative gain) on the GSM8K dataset and 0.42 points (an 8.3% relative gain) on the MATH dataset, further confirming the efficacy of SREA in large-scale tasks. **Figs. 7e,f** illustrate the energy efficiency and normalized speed of MSVD-based initialization compared to

GPUs. At the 32nm technology node, the memristor-based system achieves a $41.6\times$ advantage in energy efficiency and a $23.1\times$ speedup relative to GPUs. These results collectively demonstrate the scalability of MSVD to larger workloads and its feasibility for practical LLM applications.”

2. We added section “**Simulation of MSVD initialization for low-rank adaptation**” in **Methods** on pages 19-20: “

MSVD initialization for low-rank adaptation

Low-rank adaptation techniques typically partition the pre-trained model weights into two distinct components: a frozen (untrainable) residual matrix $\in \mathbb{R}^{M \times N}$, and two trainable low-rank branches, denoted as $\text{Branch}_A \in \mathbb{R}^{r \times N}$ and $\text{Branch}_B \in \mathbb{R}^{M \times r}$, where r represents the target low rank.

In our approach, we utilize MSVD to initialize these components. Specifically, we apply MSVD to the static weights in each transformer layer of LLMs, including the query, key, and value layers, the output-projection, up-projection, down-projection, and the gate layers, as illustrated in **Fig. 7a**. For each weight matrix, we perform a rank-4 MSVD, executing 15 iterations per rank. The resulting decomposition factors are then used to construct the PiSSA weight initialization. The trainable low-rank branches are formed as:

$$\text{Branch}_A = \mathbf{S}_{[:,r]}^{1/2} \mathbf{V}_{[:,r]}^T, \text{Branch}_B = \mathbf{U}_{[:,r]} \mathbf{S}_{[:,r]}^{1/2} \quad (16)$$

where $\mathbf{S}$, $\mathbf{V}$, $\mathbf{U}$ represent solved singular values, right singular vectors and left singular vectors, respectively. Concurrently, the frozen residual matrix is obtained by subtracting the low-rank approximation from the original pre-trained weight matrix $\mathbf{W}$:

$$\text{Mat}_{\text{res}} = \mathbf{W} - \mathbf{U}_{[:,r]} \mathbf{S}_{[:,r]} \mathbf{V}_{[:,r]}^T \quad (17)$$

Detailed parameter configurations for fine-tuning, along with the specific weight dimensions of transformer layers are provided in **Supplementary Table 5.**”

3. In **Introduction**, on page 4, we added: “Additionally, for LLaMA 3.2-3B model initialization, MSVD achieves $41.6\times$ higher energy efficiency and $23.1\times$ faster speed compared to CMOS-based hardware.”

4. We added **Fig. R10** as **Fig. 7** in manuscript.

5. In **Discussion** section on page 11, we revised the context as: “Experimental demonstrations across image reconstruction, temporal data analysis, biomedical signal processing and efficient LLM fine-tuning show that our approach achieves software-comparable accuracy.”

Comment #8

8. The authors claim “software-comparable accuracy,” citing high correlation (0.999) in COVID-19 reconstructions. While correlation is indeed strong, it is unclear whether these results generalize across multiple experimental runs, given device-level variability.

Response:

Thank you for raising this question. We clarify that 20 independent experimental runs were performed for the COVID-19 reconstruction task. We have now revised **Fig. 4f** to include error bars (standard error) for each result, explicitly showing the variability across multiple runs. As shown in **Fig. R11**, MSVD with SREA demonstrates improved performance stability in reconstruction results compared to MSVD without SREA.

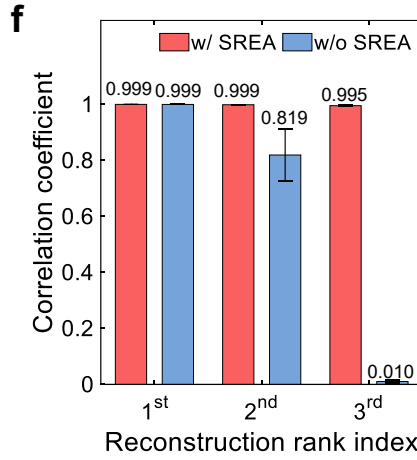


Fig. R11 (Fig. 4f). Comparison of correlation coefficients between MSVD with SREA and MSVD without SREA for 1st-, 2nd-, and 3rd-rank reconstructed matrices of the dataset⁵², using software SVD solutions as reference. The error bars indicate the standard error across 20 repetitions.

Revision:

To illustrate these points, we have made the following revision in the manuscript:

1. We updated **Fig. 4f** with **Fig.R11**.

Comment #9

9. Supplementary Fig. 5 shows ablation analysis of dual-precision mapping (DPM) and vector processing refinement (VPR). These results confirm that both components contribute to reduced error. To prove robustness, for ablations, it is better to include more challenging conditions, such as large singular value gaps or noisy, high-dimensional data.

Response:

We conducted simulation-based ablation experiments on the larger-scale LLaMA 3.2-1B model fine-tuning task, with evaluations performed on both the MATH dataset and the GSM8K dataset. These two tasks represent relatively challenging task types. The LLaMA 3.2-1B model has an overall parameter scale on the order of $\sim 10^9$. Each experiment was repeated five times. As shown in **Fig. R12**, the SREA components remain highly effective even for large-scale, challenging tasks. On the GSM8K dataset (**Fig. R12a**), DPM and VPR contributed average score improvements of 1.04 and 0.89, respectively. Similarly, on the MATH dataset (**Fig. R12b**) with the LLaMA 3.2-1B model, they yielded gains of 0.28 and 0.26, respectively. Overall, the introduction of SREA improved the average score by 1.09 points (a 4.3% relative increase) on GSM8K and by 0.42 points (an 8.4% relative increase) on MATH compared to the MSVD baseline. These results demonstrate that both DPM and VPR maintain their efficacy even under challenging conditions involving high-dimensional data and complex tasks.

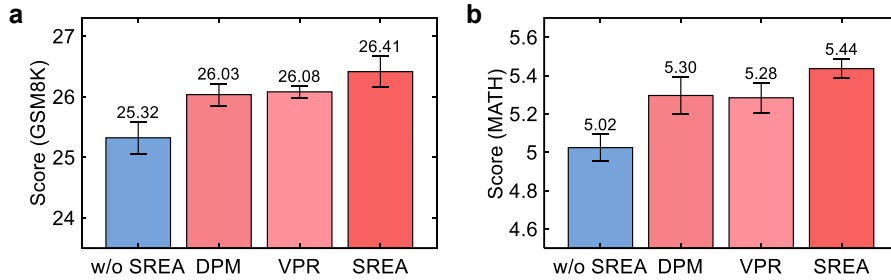


Fig. R12. Ablation study of SREA for MSVD on the LLM fine-tuning task. LLaMA 3.2-1B is used as the base model and fine-tuned with MSVD-based PiSSA. The fine-tuned models are evaluated on the GSM8K dataset (a) and the MATH dataset (b). Error bars indicate the standard error across 5 repeated experiments.

Response:

To illustrate these points, we have made the following revisions in the manuscript:

1. On page 11, we added: “**Supplementary Figure 17** presents an ablation study of SREA on LLaMA 3.2-1B. The introduction of SREA yields a performance improvement of 1.09 points (a 4.3% relative gain) on the GSM8K dataset and 0.42 points (an 8.3% relative gain) on the MATH dataset, further confirming the efficacy of SREA in large-scale tasks.”
2. We added **Fig. R12** as **Supplementary Figure 17**.

Comment #10

10. Incremental MSVD is evaluated for up to 8 users, with energy and accuracy improvements reported in Supplementary Fig. 9. While useful, this is a very limited scale. In practical multi-user settings (e.g., healthcare monitoring), dozens or hundreds of users may be encountered. The reviewer suggests scaling experiments further, whether by adding a larger scale or extending by estimation and feasibility analysis to substantiate the claim of efficient “user-scalable recognition.”

Response:

We appreciate your valuable feedback regarding scalability validation. To directly address this concern, we conducted additional experiments with simulation demonstrating incremental MSVD’s scalability with a substantially larger user base using a subset of the Sleep-EDFx dataset (**Fig. R13a**)—a practical healthcare monitoring scenario. This subset contains EEG signals recorded from different subjects at home using cassette-tape recorders during wakefulness, rapid eye movement (REM) sleep, and deep sleep (non-REM stage 3), providing real-world field-collected data with inherent environmental noise and substantial cross-subject variability.

We incrementally scaled the subject number from 15 to 75 over five successive updates, substantially expanding the user base compared to the original experiment (2 to 8 subjects). STFT-processed signals were fed into the memristor-based system for MSVD/incremental MSVD feature extraction, followed by logistic regression classification. At each update, incremental MSVD obtained the feature extractor for all current subjects and updated the classifier. Each original data matrix and incremental matrix had dimensions of 900×500 , containing information from 15 subjects. MSVD/incremental MSVD yielded 500×5 right singular vectors (RSVs) for feature extraction. In the simulation, we incorporated noise levels measured from actual devices (writing $\sigma = 0.27 \mu\text{S}$, reading $\sigma = 0.21 \mu\text{S}$).

Fig. R13b presents the incremental learning results, comparing memristor-based and software implementations at each stage. After four rounds of subject updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline, while the memristor-based system demonstrated performance comparable to software implementation. **Fig. R13c** demonstrates energy consumption and normalized speed of incremental MSVD comparing with CMOS-based HPC systems. Compared to A100 GPU and Google TPUv3 (a VMM-optimized ASIC), the memristor-based system achieves at least $47.6\times$ energy efficiency improvement and $19.5\times$ speed enhancement. As the incremental update operates only on newly added data matrices rather than the full accumulated dataset, computational overhead scales linearly with the number of new subjects per stage, suggesting that the approach remains tractable for even larger user populations in the hundreds.

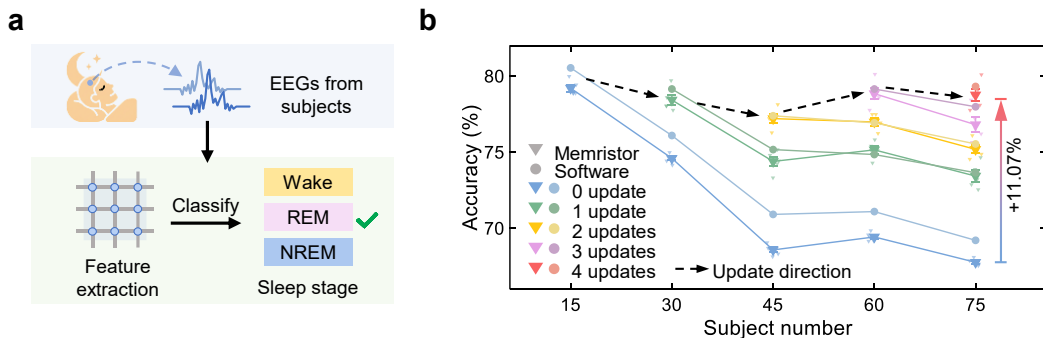


Fig. R13 (Fig. R5). Subject-scalable EEG-based sleep stage detection with

incremental MSVD. **a**, Schematic of sleep stage detection with MSVD and incremental MSVD. **b**, Recognition accuracy evolution as subject number increases from 15 to 75. The error bars indicate the standard error across 5 repetitions.

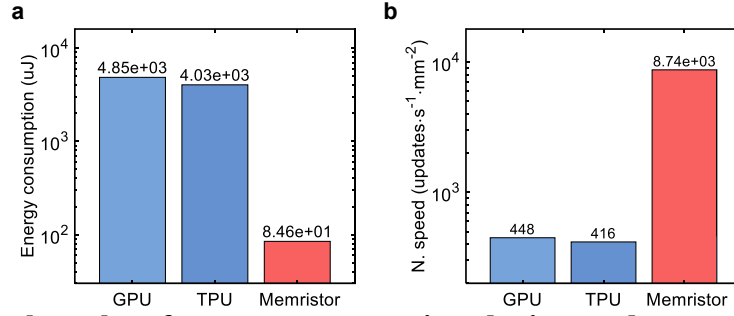


Fig. R14. Benchmarks of energy consumption during updates and normalized speed of incremental MSVD in EEG-based sleep stage detection comparing with A100 GPU and Google TPUv3.

Response:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the context on page 9: “We further explored the scalability of incremental MSVD through EEG-based sleep state detection with a larger cohort of 75 subjects, using a hardware-calibrated simulation framework (**Fig. 5g** and **Methods**). This experiment leveraged the Sleep-EDFx dataset⁶³, which contains EEG signals recorded from subjects during wakefulness, rapid eye movement (REM) sleep, and deep sleep (non-REM stage 3). In our experiment, EEG signals processed by short-time Fourier transform were fed into the memristor-based system for MSVD/incremental MSVD feature extraction, followed by logistic regression classification. Similar to EMG-based experiment, at each update, incremental MSVD refined the feature extractor to incorporate all current subjects and updated the classifier accordingly. Both the original data matrix and incremental updated matrices had dimensions of 900×500, each containing information from 15 subjects. MSVD/incremental MSVD yielded RSVs with dimensions of 500×5 (solved with 15 iterations per vector) for feature extraction. The SREA configurations for MSVD and incremental MSVD are detailed in **Supplementary Table 3**. After four steps of subject updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline, while the memristor-based system demonstrated performance comparable to software implementation (**Fig. 5f**).”
2. We added benchmarks on page 18: “Additionally, energy benchmarking performed on incremental MSVD for EEG-based sleep stage detection demonstrates a 57.3× advantage (84.6 μJ vs. 4852 μJ) over the A100 GPU and a 47.7× advantage (84.6 μJ vs. 4035 μJ) over the Google TPUv3 (**Supplementary Figure 29**).”; on page 19: “Additionally, in the context of EEG-based sleep stage detection, incremental MSVD demonstrates a 21.0× average speedup (8744 updates·s⁻¹·mm⁻² vs. 416 updates·s⁻¹·mm⁻²) relative to the TPUv3 (**Supplementary Figure 29**).”
3. We added **Fig. 5f** and **Fig. 5g** as indicated before.
4. We added **Fig. R14** as **Supplementary Figure 29**.

Comment #11

11. COVID-19 data reconstruction shows a correlation of 0.999. However, this dataset is relatively clean, and reconstruction may not be very challenging. Authors can consider testing the robustness via noisy temporal data (e.g., stock market, mobility, or sensor readings with missing entries). Without such tests, the strong results may not generalize to harder temporal data.

Response:

Thank you for this valuable suggestion. We would like to respectfully clarify that our paper already includes evaluations on challenging real-world temporal data beyond the COVID-19 dataset. The NinaPro dataset used in our gesture recognition experiments (**Fig. 5**) consists of real-world temporal EMG signals recorded from human subjects during hand movements, which contain inherent physiological noise, inter-subject variability, and temporal dynamics, demonstrating MSVD's capability to handle noisy temporal sensor data in practice.

Following the reviewer's suggestion, we have added larger experiment using the Sleep-EDFx dataset for EEG-based sleep stage detection (**Fig. R15**). This dataset contains EEG signals from 75 subjects recorded at home using cassette-tape recorders, representing realistic field-collected temporal data with substantial environmental noise, and signal quality variations.

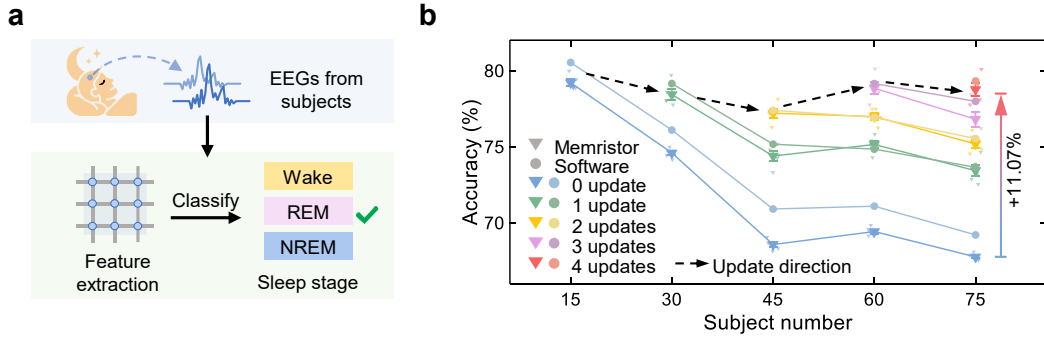


Fig. R15 (Fig. R5). Subject-scalable EEG-based sleep stage detection with incremental MSVD. a, Schematic of sleep stage detection with MSVD and incremental MSVD. **b,** Recognition accuracy evolution as subject number increases from 15 to 75. The error bars indicate the standard error across 5 repetitions.

As shown in **Fig. R16**, the memristor-based system achieves software-comparable performance on both temporal data tasks, effectively processing these noisy temporal signals despite device-level stochastic noise. The consistent performance across two distinct temporal modalities (EMG and EEG) with different noise characteristics confirms MSVD's robustness for temporal data processing and demonstrates generalization beyond the clean COVID-19 reconstruction task.

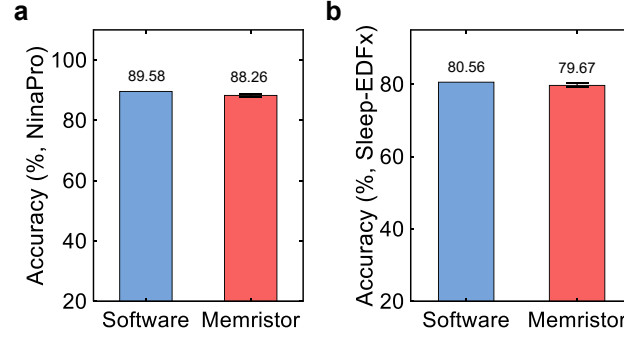


Fig. R16. Performances of MSVD-based classification method on EMG-based (a), and EEG-based (b) temporal data. The error bars indicate the standard error.

Response:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the context on pages 9-10: “We further explored the scalability of incremental MSVD through EEG-based sleep state detection with a larger cohort of 75 subjects, using a hardware-calibrated simulation framework (**Fig. 5g** and **Methods**). This experiment leveraged the Sleep-EDFx dataset⁶³, which contains EEG signals recorded from subjects during wakefulness, rapid eye movement (REM) sleep, and deep sleep (non-REM stage 3). In our experiment, EEG signals processed by short-time Fourier transform were fed into the memristor-based system for MSVD/incremental MSVD feature extraction, followed by logistic regression classification. Similar to EMG-based experiment, at each update, incremental MSVD refined the feature extractor to incorporate all current subjects and updated the classifier accordingly. Both the original data matrix and incremental updated matrices had dimensions of 900×500 , each containing information from 15 subjects. MSVD/incremental MSVD yielded RSVs with dimensions of 500×5 (solved with 15 iterations per vector) for feature extraction. The SREA configurations for MSVD and incremental MSVD are detailed in **Supplementary Table 3**. After four steps of subject updates, the incremental MSVD-based method achieved a 11.07% performance improvement compared to the non-updated baseline, while the memristor-based system demonstrated performance comparable to software implementation (**Fig. 5f**).”
2. We added benchmarks on page 21: “Additionally, energy benchmarking performed on incremental MSVD for EEG-based sleep stage detection demonstrates a $57.3 \times$ advantage ($84.6 \mu\text{J}$ vs. $4852 \mu\text{J}$) over the A100 GPU and a $47.7 \times$ advantage ($84.6 \mu\text{J}$ vs. $4035 \mu\text{J}$) over the Google TPUv3 (**Supplementary Figure 29**).”; on page 19: “Additionally, in the context of EEG-based sleep stage detection, incremental MSVD demonstrates a $21.0 \times$ average speedup ($8744 \text{ updates} \cdot \text{s}^{-1} \cdot \text{mm}^{-2}$ vs. $416 \text{ updates} \cdot \text{s}^{-1} \cdot \text{mm}^{-2}$) relative to the TPUv3 (**Supplementary Figure 29**).”
3. We added **Fig. 5f** and **Fig. 5g**.
4. We added **Fig. R16** as **Supplementary Figure 29**.

Comment #12

12. Energy savings are explained by reduced iterations (15 vs 30) when using SREA (Supplementary Note 4). However, it is unclear whether the same convergence thresholds and error tolerances were used in both cases. If looser tolerances were allowed in the hardware implementation, the comparison would be biased. Reviewer suggests clearly specifying thresholds and ensuring fair comparison.

Response:

Thank you for your comment. In our experiments, for both MSVD with and without SREA, identical convergence thresholds were applied: convergence was reached when the relative difference between consecutive iterations fell below 2%, with a maximum iteration limit of 30 set for both cases to ensure fair benchmarking.

Fig. R17 demonstrates that for the gesture recognition task, the average number of iterations to convergence was 12.85 with SREA and 27.00 without SREA under these identical criteria. In subsequent experiments, to ensure convergence while providing reasonable benchmarks, we conservatively set the iteration counts to 15 for MSVD with SREA and 30 for MSVD without SREA. These values exceed the observed average convergence requirements, ensuring that both implementations reach the same convergence threshold without bias. This confirms that the reported energy savings stem from the algorithmic efficiency of SREA rather than relaxed error tolerances.

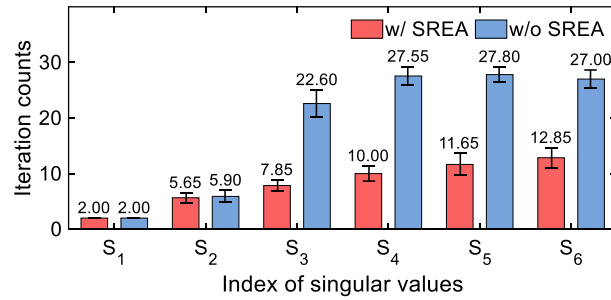


Fig. R17. MSVD iteration analysis on NinaPro dataset. Results show the average number of iterations needed for convergence across 20 repeated experiments, and convergence is reached when the relative difference between consecutive iterations falls below 2%. MSVD with SREA demonstrates faster convergence compared to MSVD without SREA across all datasets. The error bars indicate the standard error.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the **Supplementary Figure 14** with **Fig. R17**.
2. We revised sentences on page 9 as: “**Fig. 6a** illustrates the relationship between the number of iterations required for convergence and normalized singular value gaps across three datasets (IRIS, COVID-19, and NinaPro), where the convergence criterion is defined as a relative difference of less than 2% between consecutive solutions.”

Comment #13

13. Supplementary Fig. 8 shows average iteration counts across datasets. While informative, only mean values are reported. Reviewer notes the absence of variance analysis across chips, trials, or different matrix initializations. Since memristors are prone to stochastic noise, such variance should be reported to ensure reproducibility and reliability.

Response:

Thank you for this suggestion. We conducted the experiments on the same set of memristor chip but repeated for several times to reflect the impact of variance caused by initialization and iterative computation noise. We have revised original **Supplementary Fig. 8** to include error bars showing the standard error that characterize the variability of memristor-based system in multiple tries. As shown in **Fig. R18**, MSVD with SREA not only converges faster but also demonstrates better stability in iteration counts. This demonstrates the effective optimization of SREA on the iterative solution process.

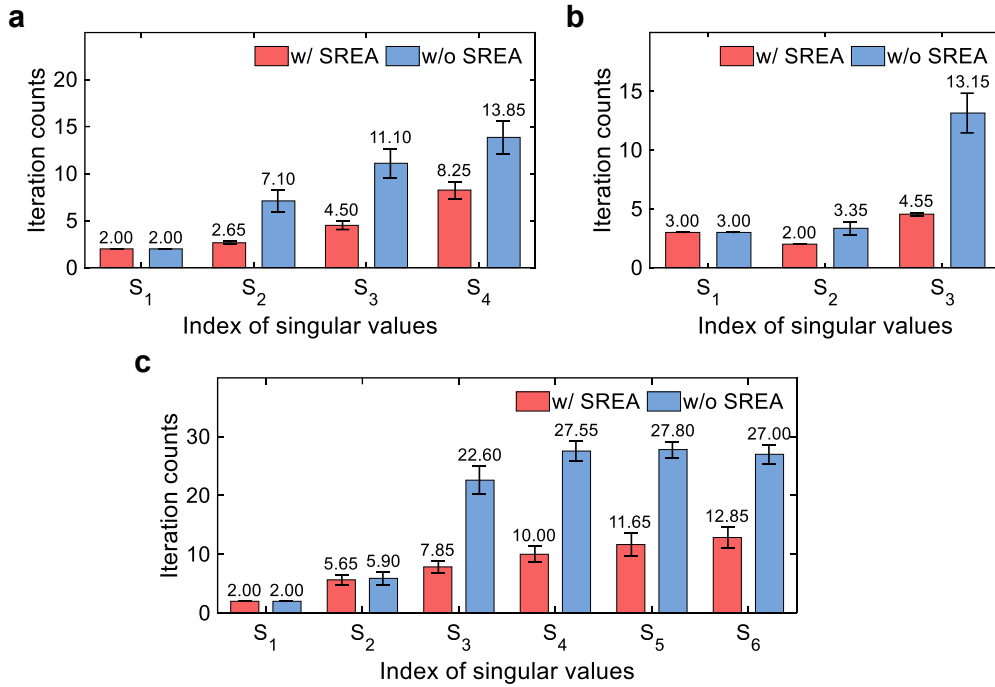


Fig. R18. MSVD iteration analysis across datasets. **a**, IRIS dataset. **b**, COVID-19 dataset. **c**, NinaPro dataset. Results show the average number of iterations needed for convergence across 20 repeated experiments, and convergence is reached when the relative difference between consecutive iterations falls below 2%; a maximum limit of up to 30 iterations is imposed if convergence is not achieved. MSVD with SREA demonstrates faster convergence compared to MSVD without SREA across all datasets. The error bars indicate the standard error.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised **Supplementary Figure 14** with **Fig. R18**.
2. We revised sentences on page 10 as: “**Fig. 6a** shows the relationship between convergence iterations and singular value gaps across three datasets (IRIS, COVID-19,

and NinaPro). The convergence criterion is set to a relative solution difference of less than 2% between consecutive iterations.”

Comment #14

14. Convergence plots (main text) show reduced drift when using SREA, but residual drift is still visible. The long-term stability of the method is not demonstrated — what happens after 1,000 or 10,000 iterations? Without long-term stability tests, it remains unclear whether MSVD can be used for prolonged or online learning tasks.

Response:

Thank you for this question regarding long-term stability. For individual device read operations, the values remain relatively stable during iterative processes. As shown in **Fig. R19 (Fig. 2f)**, conductance values are stable over 1,000 read operations, indicating that MSVD has the potential for prolonged learning at the device level.

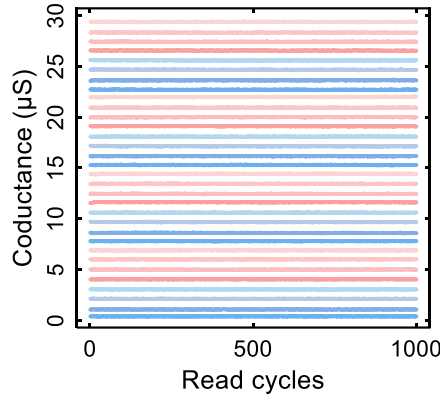


Fig. R19 (Fig. 2f). Average conductance evolutions of 32 levels over 1000 read cycles, showing good conductance stability. Each level includes data from 100 memristors.

Additionally, according to **Fig. 20**, experiments with low-rank matrices typically converge within a small number of iterations (e.g., 15 iterations). From the perspectives of computational economy and effectiveness, we employ iteration truncation strategies (e.g., limiting to 30 iterations). For online learning tasks, the introduction of new data typically initiates a new round of computation rather than simply increasing iteration count (e.g., incremental MSVD). In such cases, effective convergence truncation, combined with finer-grained DPM and more redundant VPR, can effectively address the challenges of continuously introduced new data. Furthermore, for typical edge computing tasks such as classification, a certain degree of error is often acceptable. And the results from several different types of applications in our manuscript demonstrate that errors (contains the error caused by drift) are controlled in reasonable ranges with our designed SREA.

On the other hand, the EMG-based gesture recognition and newly added EEG-based sleep stage detection tasks demonstrate multi-stage solving processes, directly addressing the online learning scenarios raised by the reviewer. In these tasks, multiple rounds of multi-level iterations are performed with cumulative iteration counts approaching 400. Nevertheless, the memristor-based system achieves software-comparable accuracy and successfully implements incremental classification in both tasks, validating MSVD's practical value for online learning applications.

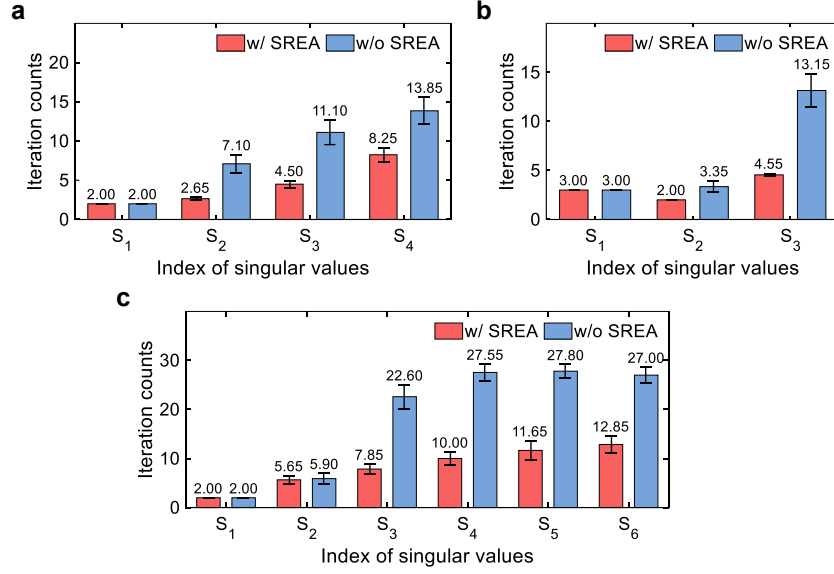


Fig. R20. MSVD iteration analysis across datasets. **a**, IRIS dataset. **b**, COVID-19 dataset. **c**, NinaPro dataset. Results show the average number of iterations needed for convergence across 20 repeated experiments, and convergence is reached when the relative difference between consecutive iterations falls below 2%; a maximum limit of up to 30 iterations is imposed if convergence is not achieved. MSVD with SREA demonstrates faster convergence compared to MSVD without SREA across all datasets. The error bars indicate the standard error.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised **Fig. 2f** with **Fig. R19**, in which the data points have been reduced in size to better reveal the underlying fluctuations.
2. We revised **Supplementary Figure 14** with **Fig. R20**.

Comment #15

15. Supplementary Fig. 10 evaluates error sensitivity under additional precision bits in VPR. While this is an excellent analysis, it is entirely simulation-based. Reviewer suggests validating the predicted error reductions experimentally, since real memristor noise characteristics may deviate from the modeled distributions.

Response:

Thank you for this suggestion. We have updated this experiment to use real devices, and the corresponding results are shown in **Fig. R21**. 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. The results demonstrate that additional precision bits in VPR remain effective under realistic device noise conditions.

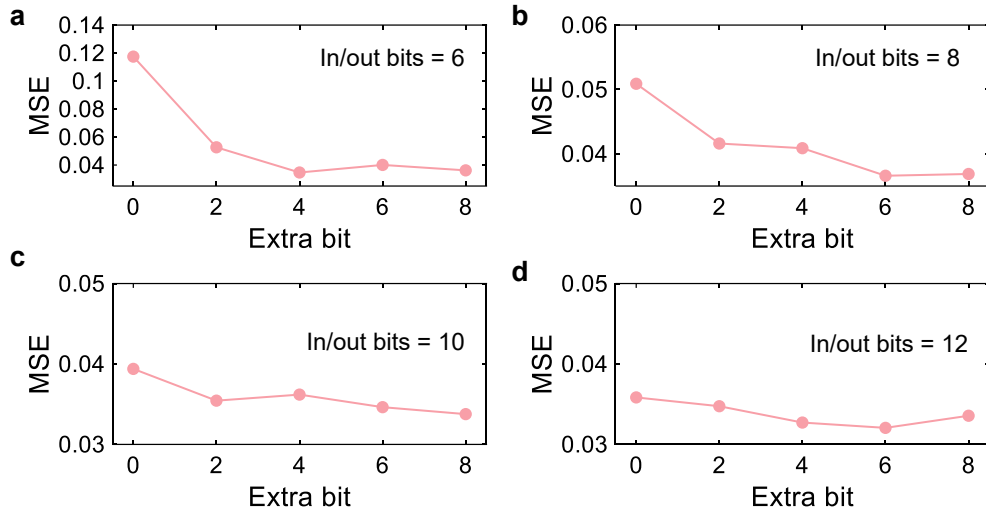


Fig. R21. Error analysis of additional bits configurations in VPR. Mean square error (MSE) of solved right singular vectors (RSVs) across different additional bit settings in VPR under different basic system resolution settings from (a) 6 bit, (b) 8 bit, (c) 10 bit to (d) 12 bit. Each experiment was repeated 20 times on IRIS dataset.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We replaced the original **Supplementary Figure 21** with **Fig. R21**.
2. We revised the sentences in **Supplementary Note 4** on page 46: “An experiment on the IRIS dataset was conducted under different input/output precisions measuring the mean square error (MSE) of solved RSVs (**Supplementary Figure 21**). Each data point represents the average of 20 repeated experiments. We varied the 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 21**, 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.”

Comment #16

16. Incremental MSVD is compared to a naive baseline, as shown in Supplementary Fig. 9. While this demonstrates improvement, stronger baselines exist — digital incremental-SVD algorithms optimized in HPC systems. Without comparison to these, it is difficult to assess whether MSVD provides unique advantages beyond the memristor platform.

Response:

Thank you for this suggestion. We have reviewed major GPU- and ASIC-compatible open-source libraries for HPC systems, including PyTorch, TensorFlow, and JAX, and have not found operators directly optimized for incremental SVD. To our knowledge, existing implementations of incremental SVD are primarily CPU-based. Considering that without systematic optimization, operators in current HPC systems struggle to achieve ideal computational efficiency, we compared our approach with well-optimized SVD operators in professional libraries. Since power iteration-deflation-based low-rank SVD computations primarily consist of matrix multiplications, we have conducted comparisons with more optimized HPC systems and operational modes following your suggestion.

First, we changed our comparison baseline to the tensor core of the A100 GPU, which is specifically optimized for matrix computation characteristics. Additionally, we further benchmarked against Google's TPU v3, an HPC ASIC specifically designed for large-scale matrix multiplications. **Fig. R22** presents the results for EMG-based gesture recognition, showing that compared to TPU v3, incremental MSVD achieves 54.5× advantage in energy efficiency and 42.9× advantage in speed.

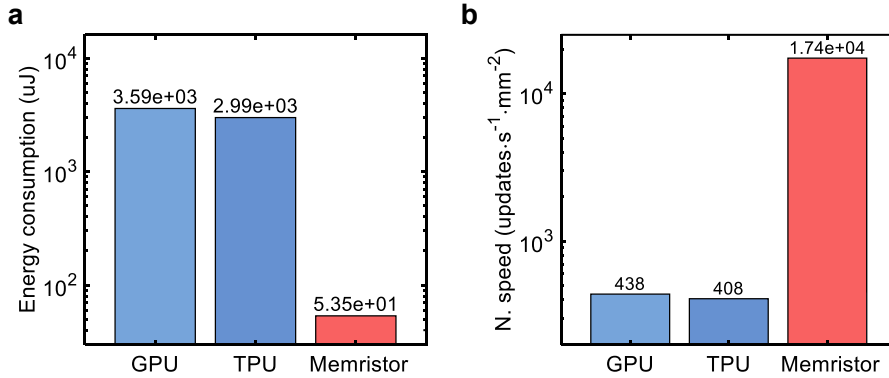


Fig. R22. Benchmarking incremental MSVD against A100 GPU and Google TPuv3 for EMG-based gesture recognition. **a**, Energy consumption during updates. **b**, Normalized speed. The TPuv3 chip area is estimated to be approximately 700 mm² based on literature¹.

Fig. R23 further shows the results for EEG-based sleep stage detection. Compared to TPU v3, incremental MSVD achieves 47.7× advantage in energy efficiency and 21.0× advantage in speed.

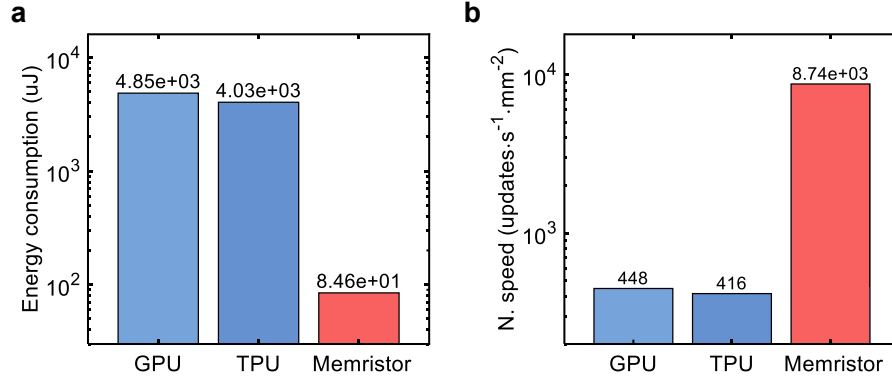


Fig. R23. Benchmarking incremental MSVD against A100 GPU and Google TPUv3 for EEG-based sleep stage detection. a, Energy consumption during updates. **b,** Normalized speed. The TPUv3 chip area is estimated to be approximately 700 mm² based on literature¹.

[1] Norrie, T. et al. The Design Process for Google's Training Chips: TPUv2 and TPUv3. IEEE Micro 41, 56-63 (2021). <https://doi.org/10.1109/MM.2021.3058217>

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added **Fig. R22** as **Supplementary Figure 28**.
2. We added **Fig. R23** as **Supplementary Figure 29**.
3. On page 20, we added: “Compared to an application-specific integrated circuit (ASIC) designed for high performance computing (Google TPUv3), incremental MSVD achieves a 54.5× overall energy efficiency advantage (**Supplementary Figure 28**). **Supplementary Table 5** further presents an energy efficiency comparison against other CMOS-based decomposition ASICs; despite variations in task scale, the advantage of MSVD remains evident. Additionally, energy benchmarking performed on incremental MSVD for EEG-based sleep stage detection demonstrates a 57.3× advantage (84.6 μJ vs. 4852 μJ) over the A100 GPU and a 47.7× advantage (84.6 μJ vs. 4035 μJ) over the Google TPUv3 (**Supplementary Figure 29**).”
4. On page 21, we added: “Compared to Google TPUv3 (~700 mm² estimated die size⁶⁵), incremental MSVD exhibits a 42.6× average speed improvement (17382 updates·s⁻¹·mm⁻² vs. 408 updates·s⁻¹·mm⁻², **Supplementary Figure 28**). Additionally, in the context of EEG-based sleep stage detection, incremental MSVD demonstrates a 21.0× average speedup (8744 updates·s⁻¹·mm⁻² vs. 416 updates·s⁻¹·mm⁻²) relative to the TPUv3 (**Supplementary Figure 29**).”

Comment #17

17. The choice of dual-precision mapping (2-bit + 5-bit split) is justified partly by Supplementary Note 2. However, the reasoning is somewhat ad hoc, and no sensitivity analysis is provided. Reviewer suggests testing alternative splits (e.g., 3+5, 4+4) to identify optimal configurations across datasets. Without this, the design choice appears arbitrary.

Response:

Thank you for this question. We conducted a simulation-based ablation study on the bit-partition strategy on IRIS dataset shown in **Fig. R24**. 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.

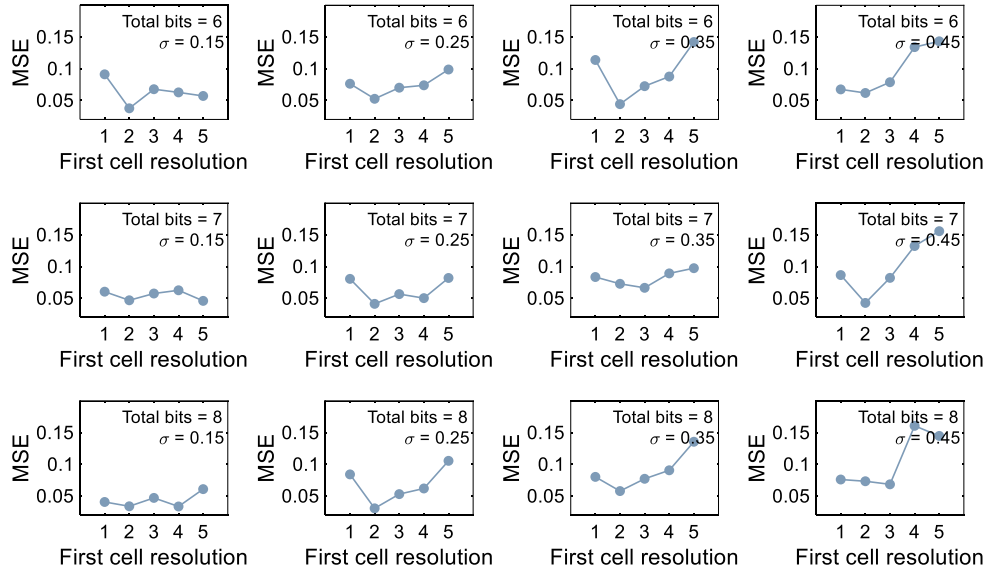


Fig. R24. Error analysis of DPM strategy on IRIS dataset with fixed total numerical resolution. Mean square error of solved right singular vectors across different DPM strategy. Each experiment was repeated 10 times.

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., > 3 bits 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.

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.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On page 15, we added: “The detailed DPM setting in different experiments is shown in **Supplementary Table 3**. Simulation-based studies and analysis, detailed in **Supplementary Figures 19, 20** and **Note 3**, further demonstrate the validity and effectiveness of the DPM.”
2. We added **Supplementary Note 3 “Analysis of bit slicing in DPM”** as: “**Supplementary Note 3. Analysis of bit slicing in DPM.**”

Let $\mathbf{X} = (\mathbf{X}_h, \mathbf{X}_l) \in \mathbb{R}^{M \times N}$ 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 $\mathbf{X}_h$. In each computation, the input $\mathbf{V}_{\text{in}}$ 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.....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.”

3. We added **Fig. R24** as **Supplementary Figure 20**.

Comment #18

18. The authors replicate singular vectors four times to reduce residual error. Supplementary Table 4 shows that this incurs noticeable area overhead. Reviewer suggests providing a quantitative trade-off analysis: How much accuracy gain per unit area overhead? Without this, the design appears unbalanced.

Response:

Thank you for this comment. We need to clarify that the area increase primarily comes from two sources: the area required for DPM slicing and the area for replicating singular vectors in VPR. For low-rank applications, the data volume of singular vectors is much smaller than that of the data matrix; therefore, 4× replication does not introduce significant additional area overhead as shown in **Fig. R25**. For numerical errors following a Gaussian distribution $N(\mu, \sigma^2)$, averaging after multiple replications transforms the error distribution to $N(\mu, \sigma^2/n)$. For example, with 4× replication, σ is reduced to 1/2 of the original value.

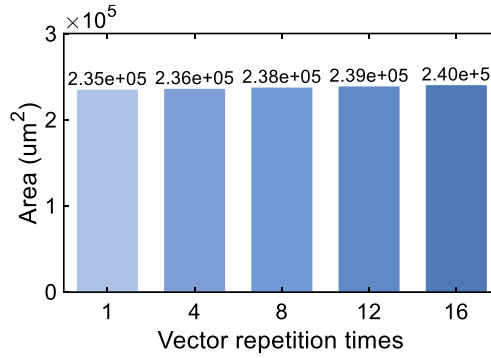


Fig. R25. Benchmarks of MSVD area across different RSV repetition times.

On the other hand, the main additional array area overhead comes from DPM. For DPM with bit partitioning [a, b], the first cell uses low resolution (e.g., a=2 bits) to represent the most significant bits, ensuring negligible errors during analog-to-digital conversion in the computation output, while the second cell represents the remaining value. This reduces the actual numerical error range to $1/2^a$ of the original value under the same device noise level (e.g., 1/4 for a=2). Combined with the VPR effect, these two mechanisms significantly reduce errors. Therefore, for multi-stage iterative tasks with high precision requirements, this approach introduces approximately 1× additional array area (5-15% of total area) while reducing numerical precision errors to below 1/4 in ideal cases (using [2,5] division as an example), making this trade-off worthwhile.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the analysis of VPR in **Supplementary Note 4** as: “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 σ . 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.”

2. We added this analysis of area with the analysis of DPM settings in **Supplementary Note 3 “Analysis of bit slicing in DPM”** as:

“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. 6g**, exemplified by the EMG-based gesture recognition task), the additional array area introduced by DPM amounts to approximately 6.2% of the total. This is a worthwhile trade-off for the resulting gains in precision.”

Comment #19

19. The experimental platform integrates a memristor chip, an FPGA (ZCU102), and a power board (Supplementary Fig. 2). Energy results were estimated using 32 nm simulations for peripheral circuits, although the actual chip was 130 nm. Reviewer notes this may exaggerate energy efficiency claims, since measured results at 130 nm may differ. A sensitivity analysis across technology nodes would add credibility.

Response:

Thank you for this suggestion. Following your suggestion, we have benchmarked the energy efficiency and speed of MSVD at different technology nodes in the EMG-based gesture recognition task. As shown **Fig. R26**, compared to CMOS-based SVD, MSVD demonstrates significant advantages in both energy efficiency and speed across different technology nodes. Compared to 28nm CMOS (NVIDIA M40 GPU), MSVD under the 130nm technology node shows a $143.5\times$ advantage in energy efficiency and a $31.17\times$ advantage in speed. Furthermore, at the 22nm node, MSVD achieves $413\times$ energy efficiency improvement and $358\times$ speedup compared to the NVIDIA P100 (16nm). These results consistently demonstrate the significant energy efficiency and speed advantages of MSVD comparing to the CMOS-based systems across different technology nodes.

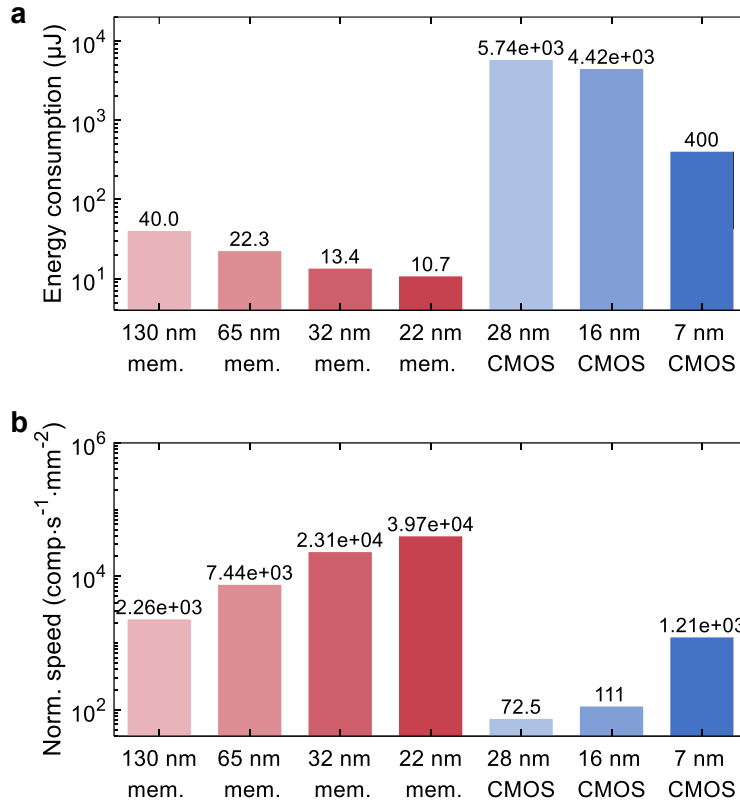


Fig. R26. Benchmark of energy consumption and normalized speed of MSVD compared to the CMOS-based SVD under different technology node. NVIDIA M40, P100 and A100 GPUs are used as 28nm CMOS, 16nm CMOS, 7nm CMOS baselines respectively.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added technology scaling analysis in supplementary information as:

“Supplementary Note 7. Technology scaling analysis of MSVD

We utilized XPEsim and the ADC scaling method^{3,22} to further analyze the scaling trends of MSVD’s energy efficiency and normalized speed across different technology nodes. **Supplementary Figure 27** 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.”

2. We added **Fig. R26** as **Supplementary Figure 27**.

Comment #20

20. Many critical details (energy evaluation assumptions, bit allocation, normalized speed analysis) are placed in supplementary material. Reviewer argues that some of these (e.g., convergence thresholds, VPR precision allocation) should appear in the main text. Otherwise, readers may miss important context.

Response:

We appreciate your suggestion to improve the visibility of experimental details. In response, we have incorporated the SREA settings and iteration numbers into the Benchmark subsection of the **Methods**. Additionally, we have explicitly defined the convergence thresholds used for evaluating iteration numbers within the main text to complement the supplementary figures. Furthermore, to ensure readers have immediate access to the necessary context, we have integrated details regarding data mapping and experimental implementation directly into the main manuscript.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On pages 17-18, we added: “We evaluated energy consumptions using verified designs and XPEsim⁶⁴ under mature memristor embedded 32 nm technology. For the EMG-based gesture recognition task, the system was configured with a [3,5] DPM division, 4-times vector repetition and 10 additional input bits for VPR alongside basic 8-bit input. Furthermore, 15 iterations are considered for each rank decomposition.”
2. On page 9, we added: “**Fig. 6a** shows the relationship between convergence iterations and singular value gaps across three datasets (IRIS, COVID-19, and NinaPro). The convergence criterion is set to a relative solution difference of less than 2% between consecutive iterations.”
3. We added the subsection “**Data mapping for MSVD and experiment configuration**” in **Methods** on page 15: “

Data mapping for MSVD and experiment configuration

A write–verify scheme is employed to program the conductance of the memristor devices, as illustrated in **Supplementary Figure 22**. The linearly quantized data (**Supplementary Note 5**) are programmed by applying SET/RESET voltage pulses ranging from 0.5 V to 2 V, followed by a read verification step at 0.2 V. This write–verify cycle is repeated until the read current falls within the target threshold range. **Supplementary Figure 23** details the relative voltage configurations applied to the WL, SL, and BL of the selected cell during the SET, RESET, and read operations, respectively. To accommodate both positive and negative matrix elements, a differential mapping scheme is adopted. Benefiting from the high yield of our devices (~99.99%), the impact of stuck-at faults on MSVD is minimal (as analyzed in **Supplementary Figure 24**), eliminating the need for complex fault-tolerant mapping strategies.

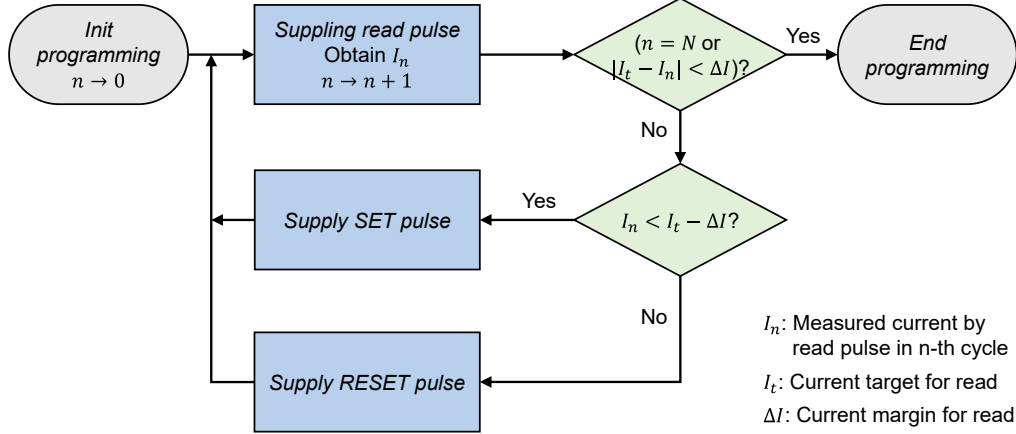
Supplementary Figure 25 illustrates how the data matrix and the computed RSVs are mapped onto a single macro for implementing MSVD with SREA. The DPM scheme partitions the matrices into two segments, and the VPR method replicates the RSVs on the array. Detailed SREA configurations for each experiment are provided in **Supplementary Table 3**. Constrained by the scale of our testing platform, for the image low-rank reconstruction and EMG-based gesture recognition tasks, we employed a reprogramming strategy to acquire the device conductances of the full matrix. We then conducted experiments by simulating the larger-scale peripheral circuits based on these measurements. Furthermore, we propose a multi-macro architecture (shown in **Supplementary Figure 26**), which serves as a reference for larger-scale deployments

utilizing multiple macros.

For EEG-based sleep stage detection task and low-rank adaptation with MSVD initialization, we utilize a hardware-calibrated simulation framework that incorporates a noise-aware memristor model and peripheral circuit models (including quantization and shift-and-add operations) to emulate a larger memristor-based computing system. During these experiments, the device conductance mapping range is set from 0 to 30 μS . The device read and write variations are modeled as Gaussian-distributed noise based on our actual device measurements (Fig. 2g) to ensure fidelity.”

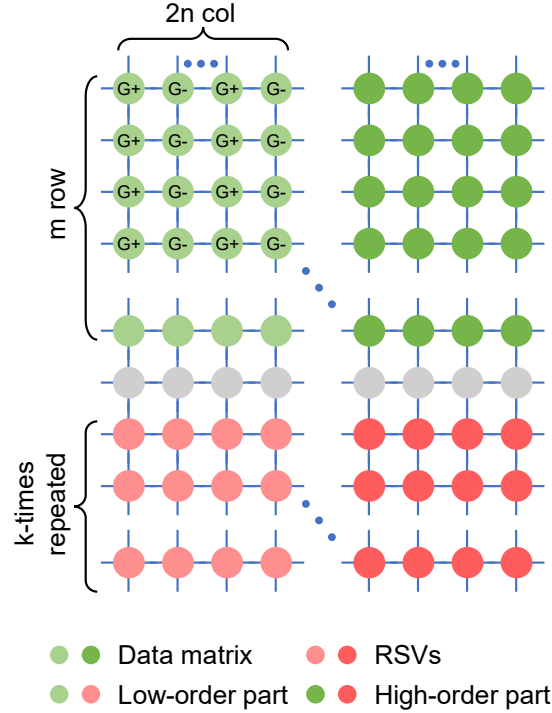
4. We revised **Supplementary Figure 14** as mentioned before.

5. We added **Supplementary Figure 22** as: “



Supplementary Figure 22. Workflow of programming with write and verify processes.”

6. We added **Supplementary Figure 25** as: “



Supplementary Figure 25. 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 the RSVs k times on the array.”

Comment #21

21. Supplementary Fig. 2 shows the schematic of the system. It is unusual how DACs are placed, and it seems the symbol should be reversed. MODE_CTL should be at least 3 bits to select the MUXs, but it showed a single-bit wire.

Response:

Thank you for this question. These DACs are placed on the Power board, where they work together with voltage sources to generate the voltages required for memristor crossbar read and write operations. We sincerely apologize for the errors in the **Supplementary Fig. 2**. We need to clarify that the DACs are positioned on the Power board for power supply, the ADCs are positioned on the memristor board for actual readout, and the MODE_CTL signal indeed requires multiple bits for control. We have revised **Supplementary Fig. 2b** and updated the caption of **Supplementary Figure 2** as shown **Fig. R27**: “

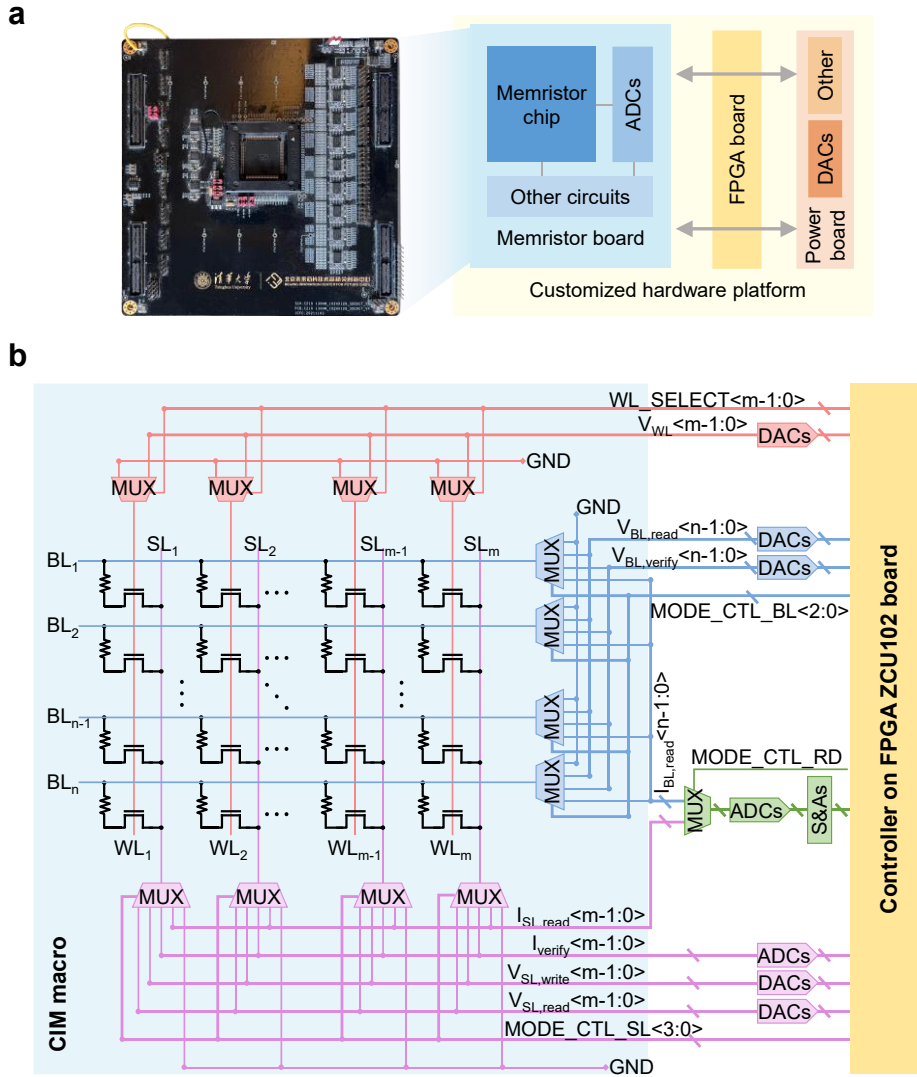


Fig. R27. Image of the customized platform and architecture for MSVD. a, Physical platform components. The platform contains three boards: memristor board, FPGA board and power board. The memristor board incorporates the memristor chip, analog-to-digital converters (ADCs) and other circuits. The FPGA (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.”

Revision:

To illustrate these points, we have made the following revision in the manuscript:

1. We revised **Supplementary Figure 2** as **Fig. R27**.

Reviewer #2**General Comments:**

The manuscript proposes a scheme for in-memory singular value decomposition (SVD) using memristive crossbars. The challenge to perform meaningful SVD with memristors is their limited precision, and therefore, the authors propose a dual-precision mapping scheme. The idea is interesting, and the experimental demonstrations appear promising. However, I feel that I cannot confidently assess the contribution because the experimental scope, mapping flow, and energy accounting are not described with clarity.

Response:

We sincerely thank you for the constructive feedback and for recognizing the promise of our experimental demonstrations. We fully acknowledge that the descriptions of the experimental scope, mapping flow, and energy accounting were insufficiently detailed in the original manuscript. In the revised version, we have expanded the methodology section to describe the mapping procedures step by step, clarified the experimental scope, and provided a more detailed breakdown of the energy evaluation methodology. We hope these revisions have adequately addressed your concerns. Detailed responses to each specific comment are provided below.

Comment #1

First, the authors should clarify whether all the results presented were obtained experimentally or if some of them were simulated. For experimental results, it should be clarified what exactly is done on memristors, on-chip periphery circuitry, FPGA, or elsewhere.

Response:

We appreciate the reviewer's suggestion to clarify the experimental implementation. The MSVD computation comprises VMMs and several auxiliary operations, including element-wise vector multiplication/subtraction, normalization, and convergence detection. For VMMs, the output data are obtained from measurements of on-chip memristor devices via on-board ADCs, directly incorporating the non-idealities of real memristors, which is a key focus of this study. The auxiliary operations, including shift-add, normalization, element-wise vector multiplication (for vector scaling in power iteration), subtraction (for deflation), and convergence detection, are implemented on the FPGA and CPU.

Specifically, for MSVD-based image reconstruction and MSVD/incremental MSVD-based EMG gesture recognition, the required weight matrices slightly exceed the physical array size. To address this, we adopted a reprogramming strategy and measured the devices to cover the full weight matrix. These experiments were then conducted by simulating the corresponding large-scale peripheral circuits (for control, shift-add, and element-wise operations) based on these measurements.

For the newly added experiments, including EEG-based sleep stage detection with an expanded subject cohort and low-rank adaptation on LLaMA 3.2-1B/3B models with MSVD initialization, we employed a hardware-calibrated simulation framework that incorporates a noise-aware memristor model and peripheral circuit models, including quantization and shift-and-add operations, to emulate a larger memristor-based computing system. Device conductance values and noise characteristics were modeled based on our actual device measurements to ensure simulation fidelity, while system-level inference was performed via simulation.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On pages 15-16, we added section “**Data mapping for MSVD and experiment configuration**” in **Methods** as: “

Data mapping for MSVD and experiment configuration

A write–verify scheme is employed to program the conductance of the memristor devices, as illustrated in **Supplementary Figure 22**. The linearly quantized data (**Supplementary Note 5**) are programmed by applying SET/RESET voltage pulses ranging from 0.5 V to 2 V, followed by a read verification step at 0.2 V. This write–verify cycle is repeated until the read current falls within the target threshold range. **Supplementary Figure 23** details the relative voltage configurations applied to the WL, SL, and BL of the selected cell during the SET, RESET, and read operations, respectively. To accommodate both positive and negative matrix elements, a differential mapping scheme is adopted. Benefiting from the high yield of our devices (~99.99%), the impact of stuck-at faults on MSVD is minimal (as analyzed in **Supplementary Figure 24**), eliminating the need for complex fault-tolerant mapping strategies. **Supplementary Figure 25** illustrates how the data matrix and the computed RSVs are mapped onto a single macro for implementing MSVD with SREA. The DPM scheme partitions the matrices into two segments, and the VPR method replicates the RSVs on

the array. Detailed SREA configurations for each experiment are provided in **Supplementary Table 3**. Constrained by the scale of our testing platform, for the image low-rank reconstruction and EMG-based gesture recognition tasks, we employed a reprogramming strategy to acquire the device conductances of the full matrix. We then conducted experiments by simulating the larger-scale peripheral circuits based on these measurements. Furthermore, we propose a multi-macro architecture (shown in **Supplementary Figure 26**), which serves as a reference for larger-scale deployments utilizing multiple macros.

For EEG-based sleep stage detection task and low-rank adaptation with MSVD initialization, we utilize a hardware-calibrated simulation framework that incorporates a noise-aware memristor model and peripheral circuit models (including quantization and shift-and-add operations) to emulate a larger memristor-based computing system. During these experiments, the device conductance mapping range is set from 0 to 30 μS . The device read and write variations are modeled as Gaussian-distributed noise based on our actual device measurements (**Fig. 2g**) to ensure fidelity.”

2. On page 14 in Methods, we added: “The flowchart of the MSVD process is illustrated in **Supplementary Figure 18**. During computation, the data matrix $\mathbf{X}$ and the computed RSVs ($\mathbf{V}$) are stored on the memristor crossbar arrays to perform the multiplications. The auxiliary operations, including element-wise vector multiplication (for $\mathbf{S}^2$ scaling of RSV inputs), subtraction, normalization (for deflation), and convergence detection, are implemented on the FPGA and CPU. At the beginning of each power iteration-deflation stage, the iterative processes on both matrices ($\mathbf{X}$ and $\mathbf{V}$) start with the same randomly initialized input vector $\tilde{\mathbf{v}}_i = \mathbf{v}_{\text{in}}$, and the iterative vector-matrix multiplications described in **Eq. (4)** are then performed in situ on the arrays. To prevent excessive iterations, the maximum iteration count is restricted to fewer than 30. The LSVs ($\mathbf{U}$) are obtained via matrix–vector multiplication with the data matrix as expressed in **Eq. (6)** and therefore do not require additional on-array storage.”

Comment #2:

Several tasks (e.g., 256x256 image, and EMG task) exceed the array column count of the memristor array used experimentally. I think the authors might have used time multiplexing, which is also hinted by the schematic in Suppl Fig 2. I also guess that doing the experiments involved massive reprogramming of the memristor array.

How the tasks investigated are mapped to the memristor array is an extremely important aspect of the approach and of the experiments. It should be detailed for each task in the Methods section (matrix dimensions, what is stored in memristors, how SREA is handled experimentally, number of tiles/passes, inter-tile accumulation, what remains resident vs. streamed, iteration counts...). Only then can we understand what has been done experimentally.

Response:

Thank you for this valuable question. For the image reconstruction task and EMG task, deploying these tasks on a single chip does involve time-multiplexing operations. This approach is a standard practice when mapping multiple data blocks or neural network layers onto the same array. Also, as these two experimental scales exceed the capacity of our single macro, constrained by the testing platform and the limited array size, we employed a reprogramming strategy to acquire the memristor conductance values for the full matrix. It is important to note that in systems with larger array scales or multi-tile integration, this time-multiplexing overhead can be avoided.

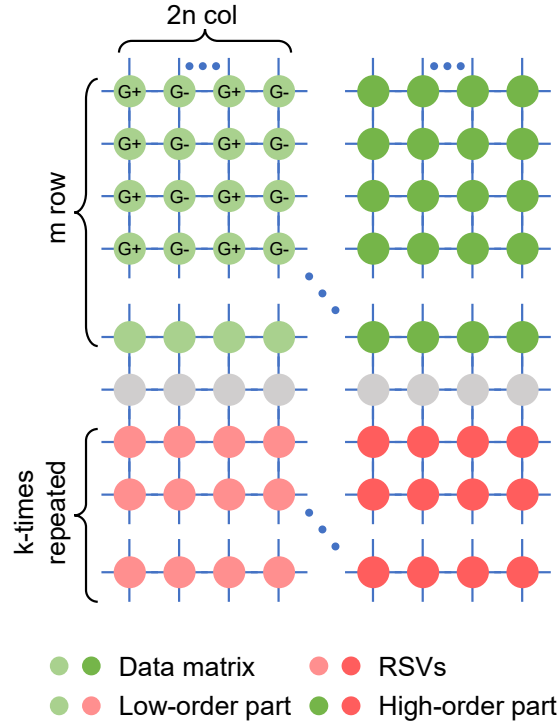


Fig. R28. Mapping method of MSVD with SREA on the memristor array. For a given $m \times n$ matrix, the data is mapped into positive and negative components. The DPM scheme splits the matrices into two parts, and the VPR method replicates the RSVs k times on the array.

The matrix dimensions for the IRIS (150x4), image reconstruction (256x256), COVID-19 (104x51) tasks were already presented in their respective sections within the **Results** of the original manuscript. To maintain consistency, we have now

supplemented the corresponding **Results** paragraphs with the matrix dimensions for EMG-based gesture recognition (864×500 , which were previously illustrated in original **Fig. 5** and the supplementary information). Here we provide additional details on how MSVD with SREA is mapped onto the array, as shown in **Fig. R28**. The memristor chip used in our experiments consists of a single tile. For the data mapping method, the values are mapped onto the array using positive and negative representations, and the DPM scheme splits them into two matrices. The VPR strategy further repeatedly maps RSVs onto the crossbar.

To clarify what remains resident versus what is streamed, two matrices are required during the computation process: the data matrix and the RSV matrix. The data matrix remains completely static (resident on the array), whereas the RSV matrix is quasi-static, updating only one vector after multiple computational iterations. Furthermore, the matrices are iteratively multiplied by a randomly initialized dynamic vector (streamed as voltage inputs) until a stable solution is reached. During experiments, inputs to the array are digitally encoded as a sequence of 1-bit voltage pulses (0V representing 0 and 0.2V representing 1), where the i -th pulse ($i = 1, 2, \dots, n$) carries a positional weight of $2^{(i-1)}$, collectively representing an n -bit input value.

Additionally, we have newly included simulation-based experiments on incremental learning using the Sleep-EDFx dataset (900×500) and MSVD-initialization-based LLM fine-tuning with the weight across various scales (e.g., 2048×2048 and 2048×512 , detailed in **Table R7**). We provide a multi-macro deployment architecture as shown in **Fig. R29**, which serves as a reference for large-scale deployments utilizing multiple macros if the scale of a single macro is not sufficient. Large-scale data and vectors can be partitioned and stored across multiple macros, and through accumulation/concatenation of outputs from macros (i.e., the inter-tile accumulation mentioned by the reviewer), larger-scale MSVD can be realized.

The iteration counts for convergence on datasets are specified in **Fig. R30** (**Supplementary Figure 14**). In these experiments, the maximum iteration count is restricted to fewer than 30; if convergence is not achieved by the limitation, the current iteration terminates. For the newly added experiments on the Sleep-EDFx dataset and LLM models, the iteration count was fixed at 15 per vector. We have also listed the specific SREA configurations for each experiment in **Table R6**.

Per your suggestion, all the aforementioned details regarding matrix dimensions, mapping strategies, resident/streamed data, and iteration counts have been explicitly integrated into the **Methods** section and the Supplementary Information of the revised manuscript.

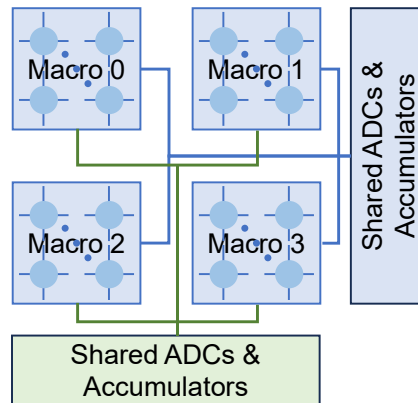


Fig. R29 The multi-macro architecture for deploying large matrix for MSVD.

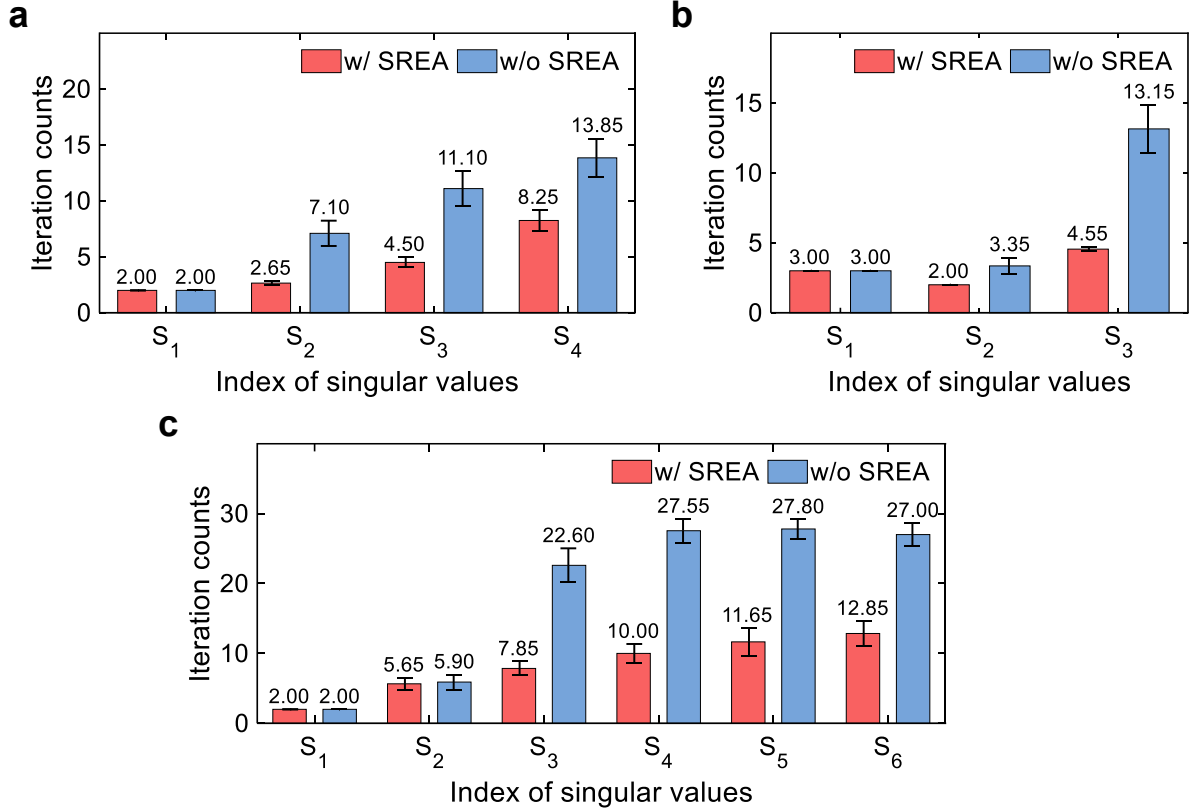


Fig. R30 (Fig. R20). MSVD iteration analysis across datasets. **a**, IRIS dataset. **b**, COVID-19 dataset. **c**, NinaPro dataset. Results show the average number of iterations needed for convergence across 20 repeated experiments. Convergence is reached when the relative difference between consecutive iterations falls below 2%; a maximum limit of up to 30 iterations is imposed if convergence is not achieved. MSVD with SREA demonstrates faster convergence compared to MSVD without SREA across all datasets. The error bars indicate the standard error.

Table R6. 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

Table R7. 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

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

- On page 9, we added: “In each step of adding a user, an 864×500 EMG signal matrix containing only the new user’s data was incorporated to update the feature extractors (with dimensions of 500×6) via incremental MSVD..... MSVD with SREA (configurations detailed in **Supplementary Table 3**) achieves software-comparable performance.”
- On page 6, we added: “SREA was configured with a [2, 5] bit slicing for DPM; 6 additional input bits (with a system standard 8-bit input precision), and 4-times vector repetition for VPR.”; on page 7, we revised: “We demonstrated the effectiveness of MSVD with SREA (see **Supplementary Table 3** for detailed configurations) for image enhancement using a 256×256 HKU logo (**Fig. 4a**).”; on page 7, we revised: “MSVD with SREA (configurations detailed in **Supplementary Table 3**) achieves high reconstruction accuracy with mean normalized errors...”; on page 9, we added: “Both the original data matrix and incremental updated matrices had dimensions of 900×500 , each containing information from 15 subjects. MSVD/incremental MSVD yielded RSVs with dimensions of 500×5 (solved with 15 iterations per vector) for feature extraction. The SREA configurations for MSVD and incremental MSVD are detailed in **Supplementary Table 3**.”; on page 12, we added: “In this approach, all static weights within the model (with dimensions such as 2048×2048 and 2048×512 , detailed in **Supplementary Table 5**) underwent rank-4 decomposition..... We applied MSVD-based PiSSA, with SREA (configurations detailed in **Supplementary Table 3**), to fine-tune LLaMA 3.2 models⁶⁶ (1B and 3B) on the MetaMathQA⁶⁷ dataset...”.
- On pages 17-18, we added section “**Data mapping for MSVD and experiment configuration**” as:

“Data mapping for MSVD and experiment configuration

A write–verify scheme is employed to program the conductance of the memristor devices, as illustrated in **Supplementary Figure 22**. The linearly quantized data (**Supplementary Note 5**) are programmed by applying SET/RESET voltage pulses ranging from 0.5 V to 2 V, followed by a read verification step at 0.2 V. This write–verify cycle is repeated until the read current falls within the target threshold range. **Supplementary Figure 23** details the relative voltage configurations applied to the WL, SL, and BL of the selected cell during the SET, RESET, and read operations, respectively. To accommodate both positive and negative matrix elements, a differential mapping scheme is adopted. Benefiting from the high yield of our devices ($\sim 99.99\%$), the impact of stuck-at faults on MSVD is minimal (as analyzed in **Supplementary Figure 24**), eliminating the need for complex fault-tolerant mapping strategies. **Supplementary Figure 25** illustrates how the data matrix and the computed RSVs are mapped onto a single macro for implementing MSVD with SREA. The DPM scheme partitions the matrices into two segments, and the VPR method replicates the RSVs on

the array. Detailed SREA configurations for each experiment are provided in **Supplementary Table 3**. Constrained by the scale of our testing platform, for the image low-rank reconstruction and EMG-based gesture recognition tasks, we employed a reprogramming strategy to acquire the device conductances of the full matrix. We then conducted experiments by simulating the larger-scale peripheral circuits based on these measurements. Furthermore, we propose a multi-macro architecture (shown in **Supplementary Figure 26**), which serves as a reference for larger-scale deployments utilizing multiple macros.

For EEG-based sleep stage detection task and low-rank adaptation with MSVD initialization, we utilize a hardware-calibrated simulation framework that incorporates a noise-aware memristor model and peripheral circuit models (including quantization and shift-and-add operations) to emulate a larger memristor-based computing system. During these experiments, the device conductance mapping range is set from 0 to 30 μS . The device read and write variations are modeled as Gaussian-distributed noise based on our actual device measurements (**Fig. 2g**) to ensure fidelity.”

4. We added **Fig. R28** as **Supplementary Figure 25**.
5. We added **Fig. R29** as **Supplementary Figure 26**.
6. We added **Fig. R30** as **Supplementary Figure 14**.
7. We added **Table R6** as **Supplementary Table 3**.
8. We added **Table R7** as **Supplementary Table 8**.

Comment #3:

The energy breakdown should state clearly what is included for memristor programming (write pulses) vs reads. Right now it is not clear whether the write pulse energy for program and verify programming is included. This is a concern as I feel that this energy might be high, even possibly dominant, if the scheme involves repeated memristor programming (which is my impression). Besides this concern, I think the methodology for energy evaluation should be more detailed.

Response:

Thank you for this comment. Our programming operation includes both write and verify stages, following the workflow shown in **Fig. R31**. During the write stage, we apply positive or negative (SET/RESET) pulses with a pulse width of 50 ns and an amplitude in the range of 0.5–2 V. During the verify stage, we use a 50 ns, 0.2 V pulse to read out the device conductance.

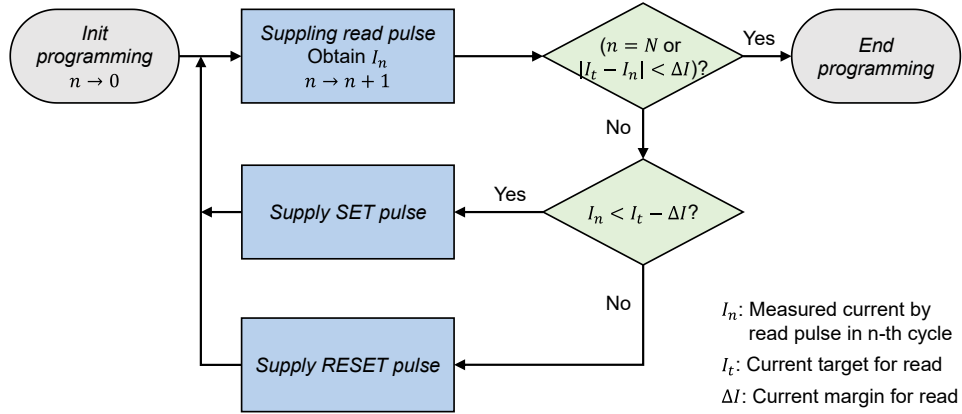


Fig. R31. Workflow of programming with write and verify processes.

Table R8. Energy benchmark of MSVD.

Operations	w/ SREA (μJ)		w/o SREA (μJ)	
Update singular vectors	0.61	0.20 (Write)	4.61	0.33 (Write)
		0.41 (Verify)		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

In our benchmark in original **Supplementary Table 3**, the energy for updating singular vectors (vector programming) includes both write and verify energy. Following your suggestion, we have further separated the programming energy into write and verify components in the supplementary materials. As shown in **Table R8**, for MSVD with SREA, updating singular vectors consumes 0.2 μJ for write and 0.41 μJ for verify. For MSVD without SREA using the analog mapping approach, the corresponding energy consumptions are 0.33 μJ and 4.28 μJ, respectively. The verify energy exceeds

the write energy because the verify stage requires readout through ADCs in addition to the array and peripheral driving circuits, whereas the write stage only involves the array and peripheral driving circuits without ADC involvement.

Regarding the concern about repeated programming potentially dominating the total energy consumption: the total programming energy for singular vector updating (write + verify = 0.61 μJ with SREA) represents only a small fraction of the total MSVD computation energy (13.40 μJ), and therefore does not dominate the overall energy budget. This is because, in low-rank SVD, the number of parameters in the vectors to be updated is very small relative to the computational scale of the power iteration-deflation process, and the system performs many iterative multiplications on the large data matrix before programming a single singular vector to the array. It is also worth noting that the programming energy of the data matrix itself is not included in the per-computation energy benchmark, as the data matrix is programmed onto the array only once and remains resident throughout all subsequent iterative computations. For consistency, the equivalent memory-loading energy for initializing the data matrix in the GPU baseline is likewise excluded. We note that the GPU memory-loading energy is expected to exceed the one-time memristor programming energy, meaning that including both would further favor our approach.

Regarding the methodology for energy evaluation, we have now provided full details in **Supplementary Note 6**. Briefly, energy consumption is evaluated with the benchmark architecture detailed in **Supplementary Figure 2**, which includes memristor arrays, ADCs, and peripheral circuits. The energy benchmark is conducted based on this large-array architecture, where the full data matrix is resident on the array throughout computation and no reprogramming of the data matrix is required. It should be noted that this benchmark architecture differs from the physical chip validation described in General Comment #2: in the prototype, reprogramming was necessitated by the limited array size of the chip, whereas the energy evaluation reflects the potential of the approach in a system-level deployment with appropriately sized arrays. The CMOS-based peripheral circuit energy was simulated using XPEsim under 32 nm technology. ADC energy was projected to the 32 nm node following established technology scaling methods^{1,2}. Although our chip validation was performed on 130 nm technology, the energy evaluation uses 32 nm technology to reflect the potential of our approach in advanced processes and enable fair comparison with modern GPU baselines.

For the EMG-based gesture recognition task (NinaPro dataset), the system was configured with a [3,5] DPM division, 4-times vector repetition, and 10 additional input bits for VPR alongside basic 8-bit input, with 15 iterations considered for each rank decomposition. Under these settings, a single MSVD computation consumes 13.40 μJ , while one incremental MSVD update consumes 17.83 μJ (including 0.61 μJ for singular vector programming). For approaches without SREA, these values are 26.04 μJ and 30.97 μJ , respectively. The reliability of the XPEsim-based evaluation framework was further confirmed by a direct comparison with measured multiplication energy on our fabricated 130 nm chip: for an 800 $\times$ 128 array with 8-bit inputs, the measured energy is less than 16.3 nJ, closely matching the XPEsim-projected value of 14.36 nJ at the same technology node. We have updated related **Methods** sections. Full details are provided in **Supplementary Note 6** and **Supplementary Table 6**.

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

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

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On pages 17-18, we added section “**Data mapping for MSVD and experiment configuration**” as:

“Data mapping for MSVD and experiment configuration

A write–verify scheme is employed to program the conductance of the memristor devices, as illustrated in **Supplementary Figure 22**. The linearly quantized data (**Supplementary Note 5**) are programmed by applying SET/RESET voltage pulses ranging from 0.5 V to 2 V, followed by a read verification step at 0.2 V. This write–verify cycle is repeated until the read current falls within the target threshold range.

Supplementary Figure 23 details the relative voltage configurations applied to the WL, SL, and BL of the selected cell during the SET, RESET, and read operations, respectively. To accommodate both positive and negative matrix elements, a differential mapping scheme is adopted. Benefiting from the high yield of our devices (~99.99%), the impact of stuck-at faults on MSVD is minimal (as analyzed in **Supplementary Figure 24**), eliminating the need for complex fault-tolerant mapping strategies.

Supplementary Figure 25 illustrates how the data matrix and the computed RSVs are mapped onto a single macro for implementing MSVD with SREA. The DPM scheme partitions the matrices into two segments, and the VPR method replicates the RSVs on the array. Detailed SREA configurations for each experiment are provided in **Supplementary Table 3**. Constrained by the scale of our testing platform, for the image low-rank reconstruction and EMG-based gesture recognition tasks, we employed a reprogramming strategy to acquire the device conductances of the full matrix. We then conducted experiments by simulating the larger-scale peripheral circuits based on these measurements. Furthermore, we propose a multi-macro architecture (shown in **Supplementary Figure 26**), which serves as a reference for larger-scale deployments utilizing multiple macros.

For EEG-based sleep stage detection task and low-rank adaptation with MSVD initialization, we utilize a hardware-calibrated simulation framework that incorporates a noise-aware memristor model and peripheral circuit models (including quantization and shift-and-add operations) to emulate a larger memristor-based computing system. During these experiments, the device conductance mapping range is set from 0 to 30 μS . The device read and write variations are modeled as Gaussian-distributed noise based on our actual device measurements (**Fig. 2g**) to ensure fidelity.”

2. We added **Fig. R31** as **Supplementary Figure 22**.
3. We added **Table R8** as **Supplementary Table 6**.

Comment #4:

Overall, the schematics provided in the Main articles did not help me understand the paper. I relied on text only. I recommend reworking them extensively.

General Comment #4

Overall, the schematics provided in the Main articles did not help me understand the paper. I relied on text only. I recommend reworking them extensively.

Response:

We appreciate your feedback regarding the clarity of the schematics. Following your recommendation, we have extensively reworked the main schematics in **Figs. 2a,d** as shown in **Fig. R32**. We have explicitly refined the visual representation of the iteration process, clearly illustrating the step-by-step operations and the specific variables/contents being solved at each stage and indicate the component of DPM and VPR forming SREA.

To further elucidate the technical details without overcrowding the main text, we have introduced highly detailed supplementary figures. These include the comprehensive workflow of the MSVD algorithm accompanied by the governing equations (**Fig. R33**), as well as the explicit mapping strategies of the MSVD on the memristor arrays (**Figs. R34, 35**).

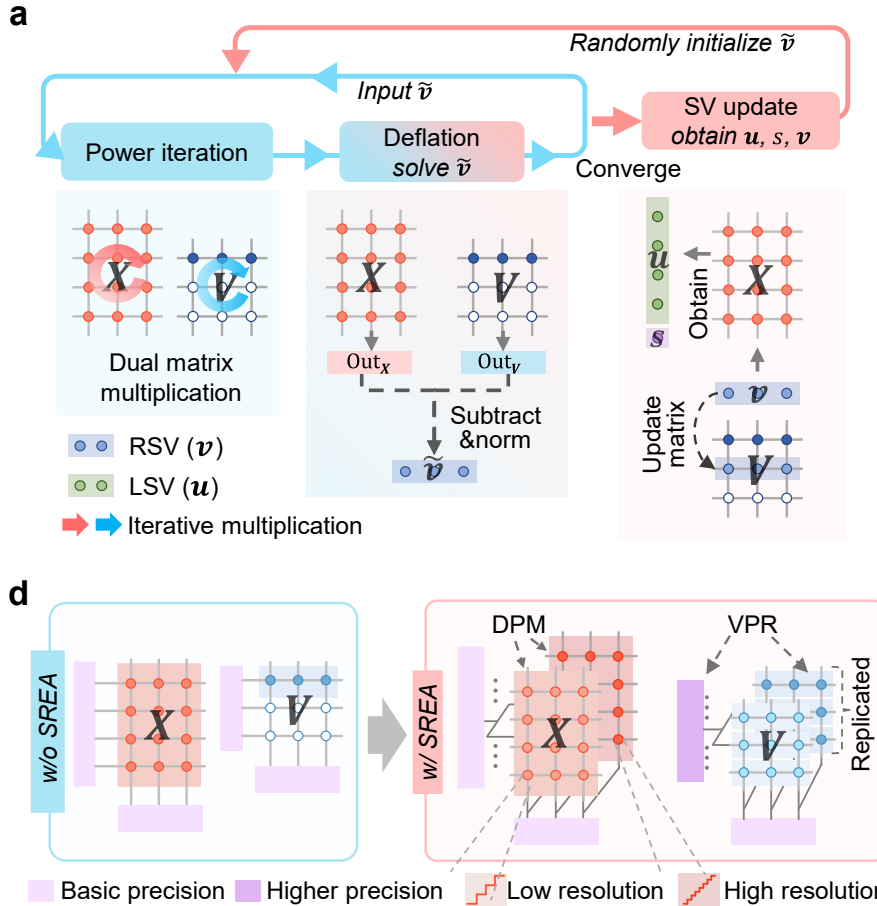


Fig. R32 (Figs. 2a,d). **a**, Schematic and data flow of the MSVD for calculating singular vectors (SVs). RSV (v) and LSV (u) finally form the matrices V and U , respectively. **d**, Designed SREA for MSVD, which transforms computational results from impractical to practical through dual-precision mapping (DPM) and vector processing refinement (VPR). S_1, S_2 : the first and the second singular values.

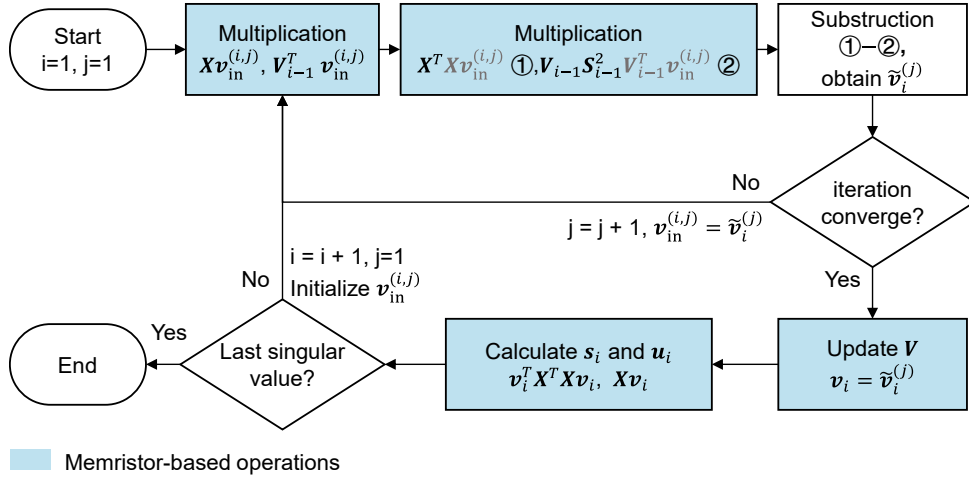


Fig. R33. Flowchart of MSVD. Grey parts in the equations indicate components obtained from previous calculation steps. The process terminates when all target singular vectors are obtained.

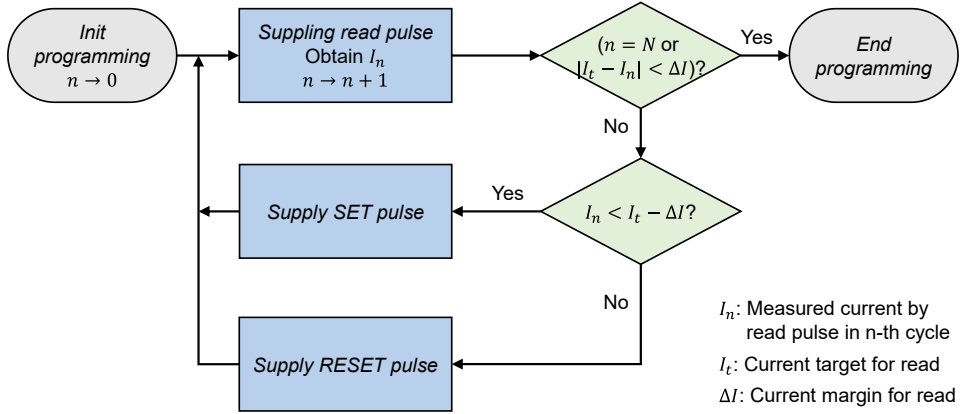


Fig. R34. Workflow of programming with write and verify processes.

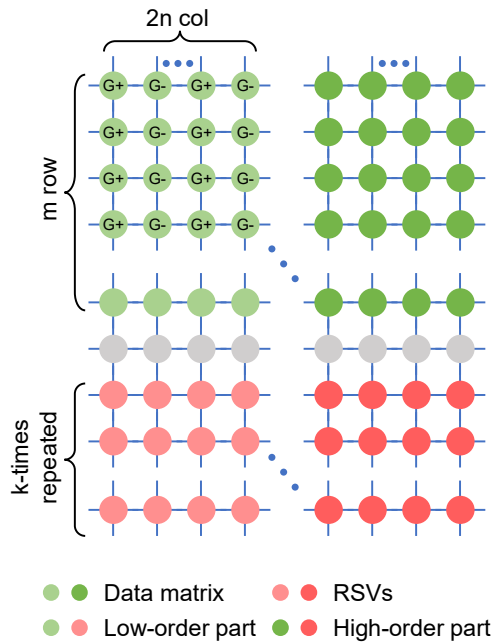


Fig. R35. Mapping method of MSVD with SREA on the memristor array. For a given $m \times n$ matrix, the data is mapped into positive and negative components. The DPM scheme splits the matrices into two parts, and the VPR method replicates the RSVs k times on the array.

We have also substantially revised the content in both the **Results** and **Methods** sections. Specifically, we have detailed the step-by-step execution of power iteration-deflation approach of MSVD and incremental MSVD in both **Results** sections and **Methods** sections. We also give a more comprehensive explanation of SREA and more detailed system workflow with the VMM operating, and further added one subsection titled “**Data mapping for MSVD and experiment configuration**” to show the data mapping method and experiment configurations.

We hope these comprehensive updates now clearly convey the algorithmic and hardware details, and that these revisions fully address your concerns.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the **Result** section on page 4 as: “SVD decomposes the data matrix X into three matrices ... As illustrated in Fig. 2a, the computation for each singular vector is executed through a iterative loop. Starting with a randomly initialized voltage vector ($\tilde{v}$), the process performs dual matrix multiplications on ... we can calculate the complete matrices U , S , and V . See **Methods** for details.”
2. We revised the **Methods** section on pages 12-13 as: “The SVD of a data matrix $X \in \mathbb{R}^{M \times N}$ can be mathematically defined as $X = USV^T$... The flowchart of the MSVD process is illustrated in **Supplementary Figure 18**. During computation, the data matrix X and the computed RSVs (V) are stored on the memristor crossbar arrays to perform the multiplications. At the beginning of each power iteration-deflation stage, the iterative processes on both matrices (X and V) start with the same randomly initialized input vector $\tilde{v}_i = v_{in}$, and the iterative vector-matrix multiplications described in Eq. (4) are then performed in situ on the arrays. To prevent excessive iterations, the maximum iteration count is restricted to fewer than 30. The LSVs (U) are obtained via matrix–vector multiplication with the data matrix as expressed in Eq. (6) and therefore do not require additional on-array storage.”
3. We revised the **Methods** section on pages 13-14 as: “Incremental MSVD is designed to obtain the updated RSVs for the tasks with incremental added data matrices. Building upon the implemented MSVD ... When new data arrives, it is mapped to the data array, and the previously solved in-memory RSVs (V_{exist}) are used to represent existing data and calculate updated RSVs. During the continuous solving progress, both V_{exist} and V are stored on the array. In each new round of incremental MSVD (e.g., when updating user data matrix), the previously computed V' serves as the new V_{exist} , while the newly solved RSVs are overwritten onto the region that originally stored the previous V_{exist} .”
4. On pages 15-16, we added section “**Data mapping for MSVD and experiment configuration**” as:

“Data mapping for MSVD and experiment configuration

A write–verify scheme is employed to program the conductance of the memristor devices, as illustrated in **Supplementary Figure 22**. The linearly quantized data

(**Supplementary Note 4**) are programmed by applying SET/RESET voltage pulses ranging from 0.5 V to 2 V, followed by a read verification step at 0.2 V. This write–verify cycle is repeated until the read current falls within the target threshold range. **Supplementary Figure 23** details the relative voltage configurations applied to the WL, SL, and BL of the selected cell during the SET, RESET, and read operations, respectively. To accommodate both positive and negative matrix elements, a differential mapping scheme is adopted. Benefiting from the high yield of our devices (~99.99%), the impact of stuck-at faults on MSVD is minimal (as analyzed in **Supplementary Figure 24**), eliminating the need for complex fault-tolerant mapping strategies.

Supplementary Figure 25 illustrates how the data matrix and the computed RSVs are mapped onto a single tile for implementing MSVD with SREA. **Supplementary Table 3** lists the settings of SREA in each experiment and **Supplementary Note 5** provides an analysis of the minimum resolvable singular value by MSVD with noise. Constrained by the scale of our testing platform, for the image low-rank reconstruction and EMG-based gesture recognition tasks, we employed a reprogramming strategy to acquire the device conductances of the full matrix. We then conducted experiments by simulating the larger-scale peripheral circuits based on these measurements. Furthermore, we propose a multi-macro architecture (shown in **Supplementary Figure 26**), which serves as a reference for larger-scale deployments utilizing multiple macros. For EEG-based sleep stage detection task and low-rank adoption with MSVD initialization, we utilize a hardware-calibrated simulation framework to simulate larger memristor-based system. During these experiments, the device conductance mapping range is set from 0 to 30 μS . The device read and write variations are modeled as Gaussian-distributed noise based on our actual device measurements (**Fig. 2g**) to ensure fidelity.”

5. We revised the Figs. 2a,d as Fig. R32.
6. We added Fig. R33 as Supplementary Figure 18.
7. We added Fig. R34 as Supplementary Figure 22.
8. We added Fig. R35 as Supplementary Figure 25.

Reviewer #3

Dear Authors,

Thank you for the very interesting paper on MSVD. This particular topic has a number of interesting subtleties, and thus my co-reviewer and I had a large number of points of feedback. Our report is in the attached Word document.

Martin Lueker-Boden, PhD

Response:

Thank you for the detailed and constructive feedback provided. Your enthusiasm for the topic and thorough examination of our work have greatly helped us improve the manuscript. Our point-by-point responses to the specific comments from you and Reviewer #4 are provided jointly below.

Reviewer #4

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Response:

Thank you for the time and effort invested in co-reviewing this manuscript, and we hope our revisions adequately address the concerns raised.

Reviewer #3 & #4

General Comments:

This paper aims to perform Singular-Value Decomposition (SVD) using memristive arrays, hence their proposed method: Memristive SVD, or MSVD. The authors make use of the iterative power method for extracting the largest singular value of the matrix (along with the corresponding left and right eigenvectors). These modes can be projected (or “deflated”) out of the original matrix and the process can then be repeated to extract successive singular value modes. This algorithm is well-established, used for many applications, and does appear well-suited for application in this setting. The authors also introduce a pair of innovations to deal with errors which they refer to collectively as selective representation enhancement architecture, or SREA. This application is important, yet the paper itself needs further revision before the reader can truly assess the value of the techniques described here. As a reviewer I was tremendously enthusiastic about this paper, as SVD is a tremendously important tool for machine learning and data analytics in general. So, I was intrigued to learn about the author's methods and learn more about the possibilities they present. I was even more interested to assess the limitations of the approach here. How broadly applicable is it? This reviewer is truly curious! Unfortunately, there are many places where I and my co-reviewer were left unable to deeply explore our questions about the authors' methods, because the descriptions of the methods lack sufficient detail to really quantify the strengths or limitations of their techniques. In general, the paper could benefit greatly from some reorganization for readability. It also lacks sufficient detail as to how the techniques are employed. Finally, the paper shares the limitations of many in-memory compute papers in that the comparison to the state of the art is presented in a very one-sided fashion, yet the paper does not provide enough detail to allow the astute reader to check their claims. The review here is therefore by necessity incomplete. An updated revision would be required to clarify the methods here, and then a full review could be made to dig into the finer points of the paper. We have made an effort however to point out as much as we can see.

Response:

We sincerely appreciate your enthusiasm for our work and your recognition of the critical importance of our hardware-accelerated SVD in machine learning and data analytics. We are deeply encouraged by your strong interest in our MSVD framework and the SREA techniques.

In response to your suggestions, we have reorganized the manuscript, especially in the Methods section, to improve readability, and have added more comprehensive experimental details including the MSVD implementation and the SREA configurations. We have also significantly expanded and strengthened our benchmarking methodology, providing a more comprehensive and transparent comparison against a wider range of baselines across various technology nodes and hardware platforms, and validated the reliability of our benchmark model by cross-referencing it with measurements from the physical chip. Furthermore, we have introduced a theoretical analysis of SREA and compared it with existing bit-slicing methods, as well as a deeper mathematical derivation of the feasible solution bounds for MSVD to further clarify its suitable working range.

We hope these revisions provide the clarity and depth required for a thorough assessment of our work. Please see our point-by-point responses below.

Minor Comment #1:

Minor Comments:

1. In Supplementary Material, the equation on line 132 is incomplete, could just be left as S^2 .

Response:

Thank you for this suggestion. We have revised the text on line 132 and moved the content of original **Supplementary Note 1** into **Methods** section in main text.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We have revised the subsection **“Implementation of MSVD with memristor arrays”** in **Methods** section on pages 12-13.

Minor Comment #2

2. The insets in Figure 4d,e are not very clear – at least not without zooming in a lot. Could the figures be adjusted so that they (along with the text) are bigger?

Response:

Thank you for this helpful suggestion. We have increased the inset sizes and enlarged the font in **Figs. 4d,e** to improve clarity and readability.

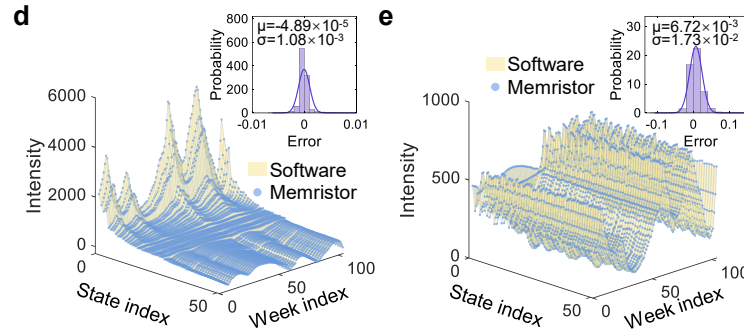


Fig. R36. Revised Fig. 4d,e. d,e, 1st- (d) and 2nd-rank (e) reconstructions of the COVID-19 dataset⁴⁹ using MSVD with SREA.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised **Fig. 4d** and **Fig. 4e** as **Fig. R36d** and **Fig. R36e**.

Minor comment #3

3. It would be quite helpful for the community interested in this work if the authors could release code for the main MSVD algorithm with SREA for one or few of the datasets presented in this paper. The hardware platform could be replaced with arbitrary devices following some normal distribution, perhaps one calibrated to Supplementary Figure 4c.

Response:

Thank you for this suggestion. We fully agree on the importance of making our work accessible to the community. The codes that support the findings in this study will be deposited in a publicly accessible repository (https://github.com/Dchenc12/memristive_svd) upon publication of the manuscript, and will include an implementation of MSVD with SREA demonstrated on a representative dataset, with the modeled memristor variations. We have updated the **Code Availability** statement accordingly.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We updated our **Code Availability** statement as: “[The codes that support the findings of this study will be available at GitHub \(https://github.com/Dchenc12/memristive_svd\)](https://github.com/Dchenc12/memristive_svd) upon publication or from the corresponding authors upon request.”

Minor comment #4

4. In Figure 2f, 2g, how were the 100 devices chosen? For 2f, could the marker size be decreased? In its current state, it is difficult to see minor fluctuations clearly.

Response:

Thank you for this question. The 100 devices presented in **Figs. 2f** and **2g** were selected at random. Following the reviewer's suggestion, we have reduced the marker size in **Fig. 2f** to better visualize the data points.

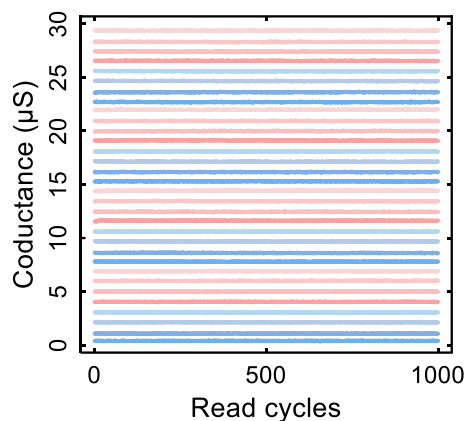


Fig. R37. Average conductance evolutions of 32 levels over 1000 read cycles, showing good conductance stability. Each level includes data from 100 memristors.

To clarify, **Fig. 4f** displays the average conductance evolution across 100 memristors per level, and therefore individual device fluctuations are inherently smoothed out. To provide a clearer view of the individual scatter, we have additionally presented the measured conductance data points from 20 devices alongside their mean values for 16 representative conductance levels in **Fig. R37**, where only 16 levels are shown to avoid visual overcrowding.

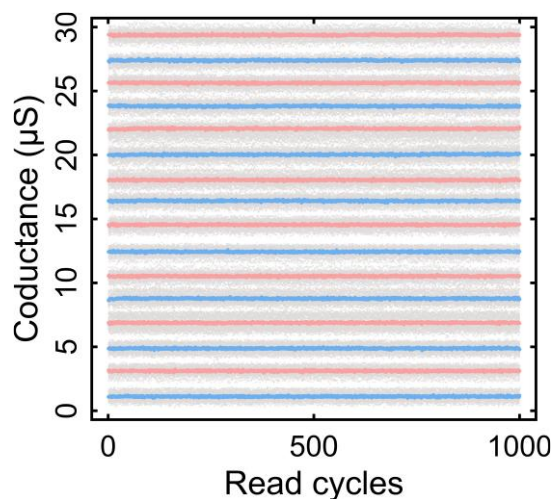


Fig. R38. Evolution of 16 conductance levels over 1000 read cycles. Average values are superimposed with individual read-outs from 20 memristors per level.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised **Fig. 4f** as **Fig. R37**;
2. We added **Fig. R38** as **Supplementary Figure 5**.

Major Comments:

Comment #1

1. General Readability: This paper satisfies the detailed guidelines for the structure of Nature Communications papers with Results and Methods sections. However, in the case of this paper these guidelines have led to a paper that is very hard to process. One simple solution would be to define terms clearly as soon as they are introduced. For instance, the terms “dual-precision mapping” and “vector processing refinement” are used without definition in the Results section or reference to the Methods section. Also results for PSNR and SSIM are unfortunately introduced before the author's definition is given in methods.

a. Recommendation 1: Introduce the full terms and acronyms in the Results section “Dual-Precision Mapping (DPM)” “Vector Processing Refinement (VPM)”, and refer the reader to the Methods section the first time these are introduced.

b. Recommendation 2: Write out the full names of the PSNR and SSIM metrics when they are first introduced and refer the reader to the methods section (as is commonly done in many of the references for this paper).

More broadly, the authors are strongly encouraged to look closely at the manuscript for linear presentation of their material. As it stands the manuscript requires several reads of the Results, Methods, and Supplemental sections before a coherent picture of the work becomes clear. Again, this is largely an artifact of the format for this journal, but it is particularly difficult with this paper, which has several unique definitions, and no fewer than three example applications.

Response:

Thank you for the valuable suggestions to improve the manuscript's readability.

Regarding Recommendation 1: In the Results section, upon the first mention of “Dual-Precision Mapping” and “Vector Processing Refinement”, we have now provided their full names and abbreviations (DPM and VPR), along with a reference to the Methods section to help readers locate the technical details.

Regarding Recommendation 2: In the “**Low-rank approximation via MSVD**” subsection of the **Results**, we have added the full names and briefly explained the utility of the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). We have also included citations pointing to the Methods section for their specific mathematical definitions.

More broadly, we have reorganized the **Methods** section to follow a more logical sequence, and have added introductory descriptions throughout the manuscript to ensure key concepts are explained upon first introduction, reducing the need for the reader to navigate back and forth between sections.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On page 5, we revised sentence as: “To address these challenges and enable high-fidelity MSVD, we propose SREA (Fig. 2d), incorporating dual-precision mapping (DPM, see details in **Methods**) and vector processing refinement (VPR, see details in **Methods**).”
2. On page 7, we added sentence as: “To quantify the reconstruction quality, we employed peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM), as detailed in **Methods**.”
3. On page 13, we revised subsection title as: “**Dual-precision mapping (DPM)**”; on

page 13, revised subsection title as: “**Vector processing refinement (VPR)**”

4. We reorganized Methods section as follows: “**Implementation of MSVD with memristor arrays**”, “**Dual-precision mapping (DPM)**”, “**Vector processing refinement (VPR)**”, “**Customized memristor hardware platform and architecture**”, “**Data mapping for MSVD and experiment configuration**”, “**Performance metrics for low-rank approximation on image and temporal data**”, “**Incremental MSVD for gesture recognition**”, “**Benchmarks of energy efficiency**”, “**Benchmarks of normalized speed**”, “**MSVD initialization for low-rank adaptation**”.
5. We have substantially revised the subsection “**Design of MSVD with SREA**” in the **Results** section (pages 4–5).

Comment #2

2. Presentation of the Iterative Power Method. There are two sections to describe the iterative power method, one in the main Methods section (starting in line 290) and another in Supplementary Note 1. It is the opinion of this reviewer that the interpretation of the Methods treatment does not stand well on its own, as the text does not give the reader a good sense of the method. These paragraphs in the supplemental section could be potentially replaced with Supplemental note 1 for more clarity (though Methods section Equation (3) would need to be retained to complete the discussion). These two sections could be combined for a net saving of space and clearer presentation, if the word- and line limits allow.

Response:

Thank you for this suggestion. We have merged the two sections accordingly, incorporating the content of original **Supplementary Note 1** into the main Methods section. **Supplementary Eq. (3)** has been retained and moved into the main text as **Eq. (2)**. The section of 'Implementation of MSVD with memristor arrays' has also been revised to improve overall readability.

Revision:

To illustrate these points, we have made the following revision in the manuscript:

1. We revised the section “**Implementation of MSVD with memristor arrays**” as:

“The SVD of a data matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ can be mathematically defined as $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{M \times M}$, $\mathbf{V} \in \mathbb{R}^{N \times N}$ and $\mathbf{S} \in \mathbb{R}^{M \times N}$ are the matrices of LSVs, RSVs and singular values, respectively. Consequently, the data covariance matrix $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{N \times N}$ can be expressed as $\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{S}^2\mathbf{V}^T$.

Since the columns of $\mathbf{V}$ (denoted as $\mathbf{v}_i$) form an orthonormal basis for $\mathbb{R}^N$, any randomly initialized input vector $\mathbf{v}_{in}$ can be represented as a linear combination of these basis vectors:

$$\mathbf{v}_{in} = c_1\mathbf{v}_1 + \dots + c_n\mathbf{v}_n, c_1 \neq 0 \quad (1)$$

where c_i are scalar coefficients. Thus, the iteratively multiplying $\mathbf{X}^T\mathbf{X}$ by $\mathbf{v}_{in}$ for k times yields:

$$\begin{aligned} (\mathbf{X}^T\mathbf{X})^k\mathbf{v}_{in} &= c_1s_1^{2k}\mathbf{v}_1 + c_2s_2^{2k}\mathbf{v}_2 + \dots + c_Ns_N^{2k}\mathbf{v}_N \\ &= s_1^{2k} \left(c_1\mathbf{v}_1 + c_2\left(\frac{s_2}{s_1}\right)^{2k}\mathbf{v}_2 + \dots + c_n\left(\frac{s_n}{s_1}\right)^{2k}\mathbf{v}_n \right) \end{aligned} \quad (2)$$

where s_i is the i -th diagonal value in matrix $\mathbf{S}$, and $s_1 > s_2 > \dots > s_N > 0$. As $k \rightarrow \infty$, the terms $\left(\frac{s_i}{s_1}\right)^{2k}$ for $i \geq 2$ approach 0. ...”

Comment #3

3. Presentation on the use of the arrays in the iterative power method. The use of the array for computing the singular vectors needs to be fleshed out more as this is given only light treatment in both the main paper and the supplementals, and understanding these details is of critical importance for the careful reading of this paper. However, almost all of these operational details are left uncertain. Here is what the reader can gather (but is not made explicit)

a. The Array is used to calculate iterative powers of $A=XTX$ (from supplemental note 1), and

b. The Array has 32 fixed conductance states (from supplemental figure 4) This implies that there are peripheral circuits that operate on a voltage drive, and that the matrix X is mapped to conductance from its original units, and the random input vectors are mapped to units of voltage at the start of the iterative power algorithm. Then a scaling factor, call it "Rscale" (as it has units of resistance), is applied to the output currents before it is applied back to the inputs of X (presumably in the other direction to realize the necessary transpose). How is this scaling factor chosen? Is it fixed throughout the algorithm, or is it adaptively chosen to maximize signal to noise? If it is adaptively chosen, is the power for the of this additional processing included in the power budget? The pipeline for actually performing these operations should be added as a Supplemental note. Several of the following comments will return to this issue of definition several times.

Response:

Thank you for this question. To further clarify the computational workflow, we have added a flowchart of the MSVD process to the supplementary information, as shown in **Fig. R39**.

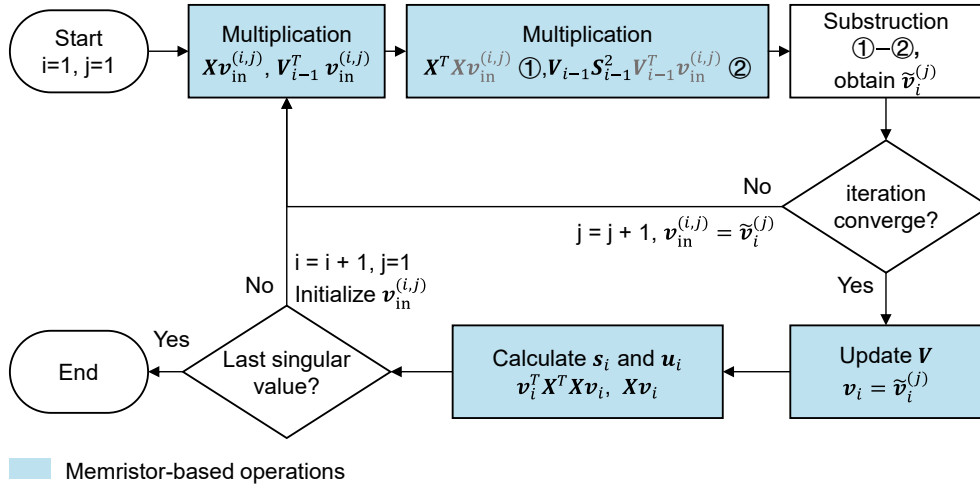


Fig. R39. Flowchart of MSVD. Grey parts in the equations indicate components obtained from previous calculation steps. The process terminates when all target singular vectors are obtained.

Regarding the transpose operation, with the designed architecture in **Supplementary Figure 2**, the transpose vector-matrix multiplication can be realized by applying inputs through the source lines (SL) and reading out through the bit lines (BL) via the ADCs, without requiring additional hardware.

For mapping the data matrix X onto the array, we first quantize 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 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 newly added LLM decomposition experiments), 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. Since the scaling factors are kept fixed throughout computation, no adaptive rescaling is performed and no additional power overhead is incurred beyond what is already accounted for in the power budget.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the section “**Conductance mapping with linear quantization**” in the supplementary information as:

“For mapping the data matrix X onto the array, we first quantize 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 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.”

2. We added **Fig. R39** as **Supplementary Figure 18**.

Comment #4

4. General comments on SREA (DPM and VPR): DPM and VPR could both be referred to as varieties of bit-slicing as defined by Le Gallo et al. "Precision of bit slicing with in-memory computing based on analog phase-change memory crossbars" *Neuromorphic Computing and Engineering*, Volume 2, No. 1 (2022). In light of this paper, DPM could be referred to as Positional Bit-Slicing, while VPR can be referred to as Equal-Fill Bit-Slicing. If the authors agree that DPM and VPR are indeed reflective of this analysis, they should make use of this reference and the common terms used therein. Doing so will not only connect the author's work to the common nomenclature for these techniques it could also potentially save critical word-count and give the authors the opportunity to leverage the error propagation analyses in these works, which actually give this reviewer some concern about the broader benefit of these techniques, as I point out in some of the following comments. If there are distinctions to be made between the bit-slicing approaches used here and one's used in the Le Gallo (2022) analysis, then a note of how the author's new bit-slicing approaches are unique would be a useful enhancement to the present paper.

Response:

Thank you for this valuable question. For DPM, our approach differs from the direct digital (positional) bit-slicing scheme proposed by Le Gallo *et al.* As illustrated by Equations (2) and (4) in Le Gallo et al.'s paper (**Fig. R40**), positional slicing is a uniform splitting strategy (e.g., splitting an 8-bit value into two 4-bit parts). Under such uniform slicing, computations on the high-order bits can easily introduce errors, which are subsequently amplified by their positional weights, ultimately preventing the overall error from decreasing as expected. Le Gallo et al. noted this as a limitation of positional bit slicing, and this is also one of the key issues that our DPM is designed to address.

[Figure Redacted]

Fig. R40. Excerpt from Le Gallo et al. illustrating the digital (positional) bit-slicing scheme¹.

When bit-encoded data is mapped onto the array, the final results are obtained through quantization following analog multiplication. Consequently, as long as the analog errors do not exceed the quantization threshold, the digitized output remains unaffected. Leveraging this property, DPM maps the high-order part to fewer target conductance levels, which effectively enlarges the quantization threshold. This makes the high-order bits more robust against analog noise, thereby mitigating high-order error amplification (a detailed analysis is provided in the response to the next question). With the high-order error amplification tightly controlled, we then map the low-order bits with a higher resolution, thereby enhancing the overall precision of numerical representation. These operations allow the system to achieve meaningful computation accuracy and improved iterative stability, reducing numerical drifting during power iterations. Furthermore, compared to overly fine-grained slicing strategies (e.g., SLC-style bit slicing) that lead to excessive resource overhead, DPM substantially improves accuracy with minimal additional cost. We have provided a comprehensive comparison in **Table R9**. We believe that DPM differs from positional slicing not merely in its non-uniform slicing approach, but fundamentally in its design philosophy. DPM accounts for the interplay between device noise and quantization thresholds, is inherently aware of high-order-error amplification, and is specifically tailored to counteract numerical drift in MSVD iterations to achieve a precision-cost balance—factors that are not considered in standard positional slicing.

Table R9. Comparisons of DPM and VPR to the generic bit-slicing techniques.

	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)

For VPR, it consists of two components: (1) repeated mapping of RSVs and (2) additional-bit application for the second multiplication in each power iteration step on RSVs. For the repeated-mapping component, the underlying principle is similar to the equal-fill bit-slicing strategy discussed by Le Gallo et al. However, in our work, VPR is not applied as a uniform repeated-mapping scheme to all matrices. Instead, we apply it selectively to the key RSVs, ensuring that the deflated components in the MSVD computation are highly accurate, which effectively minimizes the component residual after deflation. Furthermore, the additional-bit application in VPR is designed to accommodate the fact that input vectors to the RSVs are scaled to a wide dynamic range by $\mathbf{S}^2$; thus, increasing the bit width mitigates the impact of input quantization (as shown in **Supplementary Figure 10**). Taken together, these two techniques specifically optimize the computation of RSV-related terms required in MSVD, reducing the component residual during the deflation process and thereby ensuring the accurate multi-stage solving of MSVD. A comparison between VPR and Equal-Fill Bit-Slicing is also presented in **Table R8**. It is worth noting that, compared to applying repetition across the entire system, VPR achieves a far better precision-cost balance. Therefore, we believe VPR should not be directly categorized as equal-fill bit-slicing.

In summary, both DPM and VPR are specialized optimizations tailored

specifically for MSVD. They are designed to address the numerical drifting and component residual in MSVD's iteration and deflation steps, achieving high-precision reconstruction through targeted, low-overhead interventions rather than generic bit-slicing techniques. We have added Le Gallo et al. (2022) to the reference list of the revised manuscript.

[1] 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).

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the “**Analysis of bit slicing in DPM**” as **Supplementary Note 3**: “**Supplementary Note 2. Analysis of bit slicing in DPM.** Let the SVD data matrix mapped to the memristor array to be $\mathbf{X} = (\mathbf{X}_h, \mathbf{X}_l)$ and $\mathbf{X} \in \mathbb{R}^{m \times n}$. For analytical simplicity, we focus on a single vector-matrix multiplication, which serves as the fundamental computation of SVD. Consider one accumulated column of $\mathbf{X}_h$. In each computation, the input $\mathbf{V}_{in}$ 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 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.”
2. We included the paper by Le Gallo et al. as **Reference [51]** in the revised main text.
3. On page 5, we added: “While generic bit-slicing techniques have been developed for memristor-based vector-matrix multiplications⁵¹, MSVD requires specialized precision management. DPM utilizes a high-order-error-aware strategy to mitigate the error amplification commonly observed in uniform positional bit-slicing, without introducing excessive overhead. Concurrently, VPR precisely addresses the expanded input dynamic range and the consequently magnified errors from memristor variations specific to RSVs, rather than relying on global equal-fill bit-slicing. Ultimately, by elevating the optimization focus to the system-level iterative task of MSVD, SREA achieves a precision-cost balance (**Supplementary Table 2**), making efficient MSVD practically viable.”

Comment #5

5. Limits of DPM?: This method is described in line 313, and illustrated in Figure 2d. The concept is to use two arrays to enhance the precision of values of $\mathbf{X}$. It is a form of Positional Bit slicing (to use the nomenclature from Le Gallo et al (2022), see Comment 4). However, it is unclear how this improves precision in practice if the arrays used to store the high-order and low-order arrays have similar noise. If the two arrays have similar noise magnitude of noise, then the noise in the high-order array will get amplified in equation 7 along with the high-order signal when the high-order and low-order signals are merged. This could easily result in noise that drowns out the extra precision from the low-order parts (an issue that the authors do not address). The resolution to this problem is not clear at all from the text, and frankly it puts some doubt as to the utility of this technique here. However, the author's data suggests that it is still useful, but how then does it actually work? Please include a formal error propagation analysis as a supplemental to explain to the reader why noise in the high-order array is not detrimental to DPM.

Response:

Thank you for this valuable question. For conventional bit-slicing, for example, the 8-bit case in Le Gallo *et al.* where the value is uniformly split into a 4-bit high-order part plus a 4-bit low-order part. When the error occurs in the computation of the high-order 4 bits, it can be amplified by the positional weight and preventing the overall error to decrease. And for the normal mapping method for memristor to 4-bit resolution, this error can often happen. Le Gallo *et al.* pointed out that this is an inherent issue of positional bit-slicing, and it is also one of the key limitations that our proposed DPM is designed to address.

Following reviewers' suggestion, here we outline error propagation with DPM. Let $\mathbf{X} = (\mathbf{X}_h, \mathbf{X}_l) \in \mathbb{R}^{M \times N}$ 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 $\mathbf{X}_h$. In each computation, the input $\mathbf{V}_{in}$ 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:

$$\text{Out} = \text{Quant} \left(\sum_{v_{in}^{(i)} \neq 0} \mathbf{X}_h^{(i)} \right),$$

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

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

$$\mathbf{X}_h = \mathbf{X}_{h,\text{true}} + \mathbf{X}_{h,\text{noise}}; \quad \mathbf{X}_l = \mathbf{X}_{l,\text{true}} + \mathbf{X}_{l,\text{noise}}.$$

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

$$\text{Output} = \sum_{v_{in}^{(i)} \neq 0} \left[(\mathbf{X}_{h,\text{true}}^{(i)} + \mathbf{X}_{h,\text{noise}}^{(i)}) \times 2^b + (\mathbf{X}_{l,\text{true}}^{(i)} + \mathbf{X}_{l,\text{noise}}^{(i)}) \right]$$

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:

$$\text{Error}_{h,\text{noise}} = \text{Quant}\left(\sum_{v_{\text{in}}^{(i)} \neq 0} \mathbf{X}_{h,\text{noise}}^{(i)}\right) = \text{Quant}\left(\sum_{v_{\text{in}}^{(i)} \neq 0} \frac{G_{h,\text{noise}}^{(i)} \times (2^a - 1)}{G_{\text{max}}}\right) \times 2^b$$

where $\text{Quant}(\cdot)$ denotes the analog-to-digital conversion (quantization) operation. And the overall error can be expressed as:

$$\begin{aligned} \text{Error}_{\text{DPM},\text{noise}} &= \text{Quant}\left(\sum_{v_{\text{in}}^{(i)} \neq 0} \frac{G_{h,\text{noise}} \times (2^a - 1)}{G_{\text{max}}}\right) \times 2^b \\ &\quad + \text{Quant}\left(\sum_{v_{\text{in}}^{(i)} \neq 0} \frac{G_{l,\text{noise}} \times (2^b - 1)}{G_{\text{max}}}\right) \\ \text{Error}_{\text{orig},\text{noise}} &= \text{Quant}\left(\sum_{v_{\text{in}}^{(i)} \neq 0} \frac{G_{\text{noise}} \times (2^{a+b} - 1)}{G_{\text{max}}}\right) \end{aligned}$$

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

Notably, under otherwise fixed conditions, we can control the bit-splitting configuration to adjust $\frac{G_{h,\text{noise}} \times (2^a - 1)}{G_{\text{max}}}$, such that the result of

$\text{Quant}\left(\sum_{v_{\text{in}}^{(i)} \neq 0} \frac{G_{h,\text{noise}}^{(i)} \times (2^a - 1)}{G_{\text{max}}}\right)$ 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 Reso_{Mem} , a sufficient condition for the computation to be effectively error-free can be written as

$$\text{Error} = \sum_{v_{\text{in}}^{(i)} \neq 0} \mathbf{X}_{h,\text{noise}}^{(i)} < \frac{1}{2} \times \text{Reso}_{\text{Mem}}.$$

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

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

Under a 2σ -criterion which includes 95% error distribution, the noise impact can be regarded as negligible when

$$4\sqrt{n} \sigma < \text{Reso}_{\text{Mem}}.$$

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

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

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 $\{0, 1\}$, the expected number of nonzero row inputs is 64. For an equivalent device-noise standard deviation $\sigma \in [0.2, 0.35] \mu\text{S}$, we obtain $4\sqrt{n} \sigma \in [6.4, 11.2] \mu\text{S}$. With a programming G_{max} of $30 \mu\text{S}$, 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 bound about $2\sqrt{n}\sigma$, then the allowable number of high bits become $\log_2 \left(\frac{G_{max}}{2\sqrt{n}\sigma} + 1 \right)$, 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.

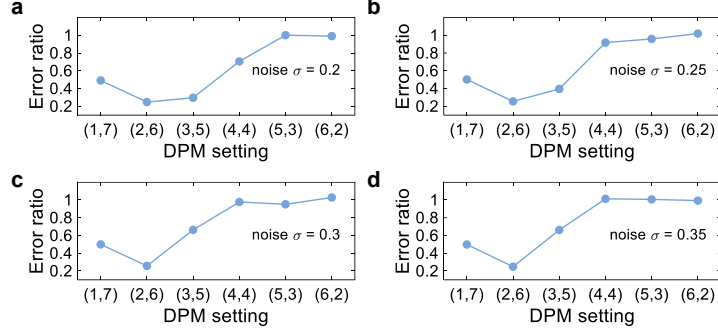


Fig. R41. Ratio of VMM error with DPM to the error with direct memristor mapping on a 128×128 memristor array. The devices are simulated with conductance variability of $\sigma = 0.2 \mu\text{S}$ (a), $0.25 \mu\text{S}$ (b), $0.3 \mu\text{S}$ (c), and $0.35 \mu\text{S}$ (d) when target maximum conductance is set to $30 \mu\text{S}$. The input pulses are drawn from a uniform Bernoulli distribution with equal probability of 0 and 1.

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. **Fig. R41** 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 high-order-error awareness, the reduction in overall error compared to the no-DPM case is very limited.

We further conducted an ablation study on the DPM strategy with simulation shown in **Fig. R42**. 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., > 3 bits 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.

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.

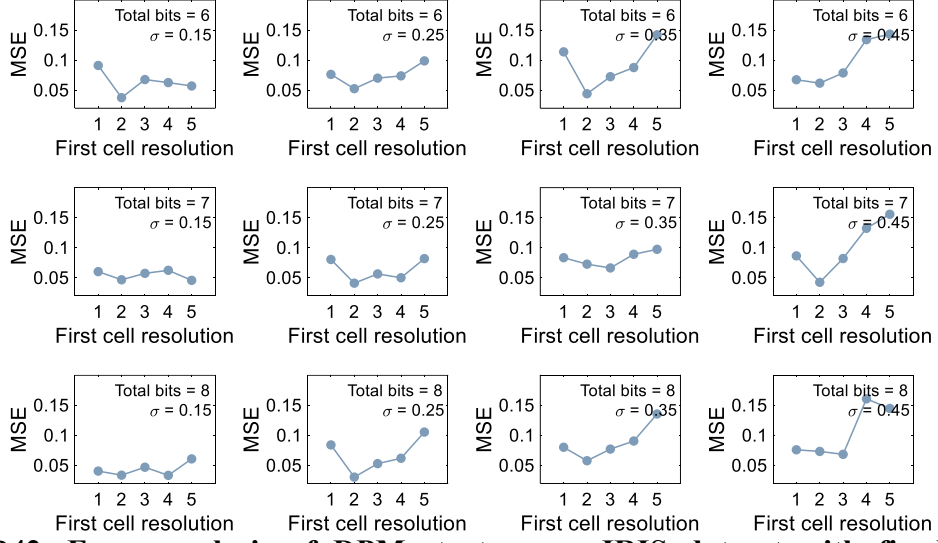


Fig. R42. Error analysis of DPM strategy on IRIS dataset with fixed total numerical resolution. Mean square error of solved right singular vectors across different DPM strategy. Each experiment was repeated 10 times.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added the analysis in this response into supplementary information as **Supplementary Note 2**.
2. We added **Fig. R41** into supplementary information as **Supplementary Figure 19**.
3. We added **Fig. R42** into supplementary information as **Supplementary Figure 20**.

Comment #6

6. Implementation details for VPR: The definition of VPR is not very clear, the use of equal fill bit-slicing (Comment 4) is appropriate if these vectors are stored as analog quantities. However the author then makes the statement that for VPR, the RSV's are “typically replicated 4-times with 6-additional input bits” (line 348). What is the significance of "6 additional input bits"? Additional with respect to what? A close reading of this sentence indicates that the inputs to the V array are digitally encoded, even though the V-array itself is an analog vector, but the phrasing here is nonetheless still confusing to the reader.

Response:

Thank you for this question. We would like to first clarify that in VPR, the RSV values are stored using DPM (e.g., a [2,5] bit configuration across two cells), and the inputs to the array are digitally encoded as a sequence of 1-bit voltage pulses (0V representing 0 and 0.2V representing 1), where the i -th pulse ($i = 1, 2, \dots, n$) carries a positional weight of $2^{(i-1)}$, collectively representing an n -bit input value. As discussed in our response to Comment #4, VPR should not be directly categorized as equal-fill bit-slicing, as it consists of two components: selective repeated mapping of RSVs (rather than uniform repetition across the entire system) and additional input bit enhancement for the second multiplication of power iteration on RSVs.

The baseline input/output resolution of the entire system is 8 bits, meaning input vectors are fed into the array through 8 voltage pulses ($n = 8$). The 6 additional input bits mentioned in the manuscript are relative to this baseline resolution, meaning the total input precision for this particular computation is $8 + 6 = 14$ bits. The reason for increasing precision at this step is that the large dynamic range of square of singular values scales the output of the first VMM operation, substantially expanding the value range of the input to the second VMM operation in each power iteration loop for RSV. 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. These factors are applied to scale the outputs of the first VMM. This creates a substantial amplification gap of approximately $54 \times (629.5/11.70)$ between the largest and smallest scaled components. If the basic quantization precision were used, the quantization errors introduced at this stage would be easily amplified in subsequent multiplication, degrading the final accuracy. To mitigate this, we apply a higher-bit quantization (14 bits) to these scaled values.

We have revised the description of VPR in the manuscript to clarify these implementation details.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On page 15, we added: “The system digitally encodes input data into a temporal sequence of 50 ns 1-bit voltage pulses (0.2V representing ‘1’ and 0V representing ‘0’), which are applied to the array to perform vector-matrix multiplication.”
2. We revised the **Results** on page 4 as: “To address these challenges and enable high-fidelity MSVD, we propose SREA (Fig. 2d), incorporating dual-precision mapping and vector processing refinement (DPM and VPR, see details in **Methods**). DPM partitions the data into high-order and low-order components (e.g., splitting an 8-bit value into a 2-bit high-order and a 6-bit low-order combination). By allocating the high-order part

to low-resolution memristor states, DPM ensures the physical robustness of critical bits, preventing deviations of high-order part being amplified by their large positional weights. Concurrently, mapping the low-order part to high-resolution states preserves fine-grained numerical precision. This mapping strategy mitigates numerical drift and provides a robust foundation for precise iterations. Complementarily, VPR mitigates the error amplification during deflation. Since RSV inputs possess a large dynamic range as scaled by square of singular values, VPR assigns higher precision to these specific inputs (e.g., applying 12-bit precision to these inputs within a standard 8-bit system precision) to prevent quantization errors from being amplified in subsequent multiplications. Furthermore, to counter memristor variations, the mapped RSVs are replicated across the array, and their outputs are averaged. This refinement drastically reduces the residual components.”

3. In **Supplementary Note 4**, we added: “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 σ . 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.”

4. In **Supplementary Note 4**, we added: “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 $54\times$ ($629.5/11.70$) 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 ($8 + 6$) ...”

Comment #7

7. Deflation: How does the deflation operation work in MSVD? Where are the LSVs (U vectors) stored? Fundamentally: How does storing analog values for V save energy? This step is not clear from the current text. In line 146 of the supplemental, we see that Xdefl is calculated from previously calculated LSV's and RSV's. However, it is not clear from the text what role the V-array plays in the calculation of Xdefl. What vector is passed through the V array? Where does it come from? Are the U vectors explicitly stored elsewhere, or recomputed from the V vector somehow?

Response:

Thank you for this question. Deflation is implemented as a subtraction between the outputs of the data array and the V array after each round of power iteration. This operation removes the contribution of the previously solved components from the data matrix, thereby enabling the computation of the subsequent rank. Regarding U , it is not stored on the memristor array. It is computed during the iterative process but does not need to be directly accessed in subsequent computations. To further clarify the MSVD computational workflow, we have added a detailed flowchart in **Fig. R43**.

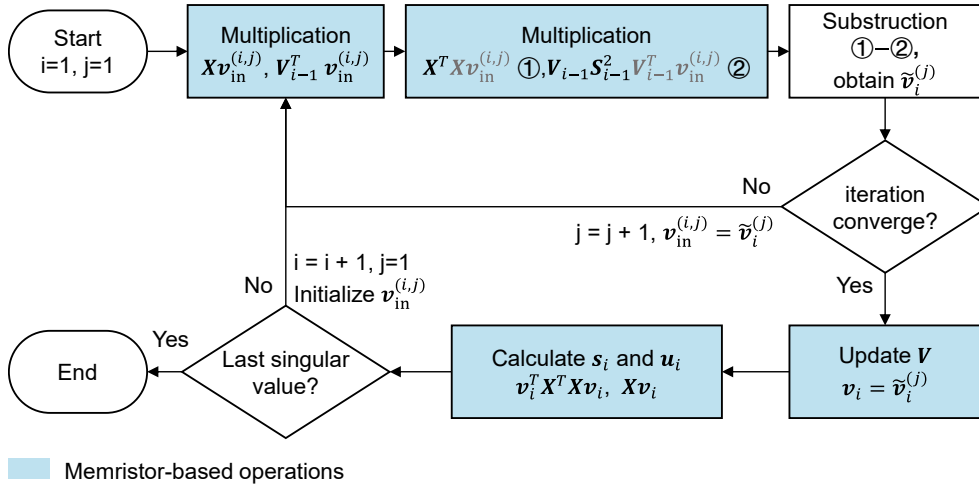


Fig. R43. Flowchart of MSVD. Grey parts in the equations indicate components obtained from previous calculation steps. The process terminates when all target singular vectors are obtained.

Storing V on the array incurs additional energy. In our benchmark (**Supplementary Note 6** and **Supplementary Table 6**), we report the extra energy required for each vector updating, i.e., the energy to program the new vector into the memristor array (**Table R10**). With this design, the iterative operations associated with V can be computed on the memristor array, thereby reducing the solving energy cost. V array is used for computing the deflation component to support the continuous solving. As shown in original **Eq. (2)**, when computing the i -th singular value and singular vector ($i > 1$), the contribution associated with the previously computed singular value and singular vectors must be removed (deflated).

Table R10. Energy breakdown of MSVD

Operations	w/ SREA (μJ)		w/o SREA (μJ)	
Update singular vectors	0.61		4.61	
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

At the start of each power-iteration round, the input applied to the V array has the same dimensionality as that applied to the data matrix. The input for the first iteration is a randomly initialized vector, whereas the input for each subsequent iteration is the output obtained after deflation in the previous round (i.e., the solved template RSV). During computation, U is obtained according to **Eq. (4)** in the original manuscript by applying the solved vector once as the input to the data matrix.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised benchmark in **Supplementary Table 6** with **Table R10**.
2. On page 13, we added: “During computation, the data matrix X and the computed RSVs (V) are stored on the memristor crossbar arrays. At the beginning of each deflation stage, the iterative process is initiated using the same randomly initialized input vector v_{in} for both the data matrix and the RSVs, and the iterative calculations described in **Eq. (4)** are then performed in situ on the arrays. The LSVs (U) are obtained via matrix–vector multiplication with the data matrix as expressed in **Eq. (6)**, and therefore do not require additional on-array storage.”
3. We added **Fig. R43** as **Supplementary Figure 18**
4. On page 4, we revised: “As illustrated in **Fig. 2a**, the computation for each singular vector is executed through an iterative loop (marked in blue). Starting with a randomly initialized input voltage vector ($\tilde{v}$), the process performs dual matrix multiplications on two memristor subarrays: one storing the data matrix (X) and the other storing the previously solved RSVs (V). The multiplications on data matrix executes one step of power iteration, while the multiplications on previously solved RSVs calculates the components to be deflated. The multiplication output from RSVs is then subtracted from the output of data matrix to update $\tilde{v}$, effectively completing the deflation to isolate the target singular vector. These operations form one iteration step of the loop. Once the loop converges or stops, the newly solved RSV (v , obtained from converged $\tilde{v}$) is updated into V for subsequent power iteration and deflation steps (illustrated in red loop in **Fig. 2a**). The corresponding singular value (s) and LSV (u) are then derived through multiplications involving the newly solved RSV and the data matrix. By repeating this process, we can calculate the complete matrices U , S , and V . See **Methods** for details.”

Comment #8

8. Minimum detectable singular vectors: The authors give a lot of examples in the text of applications where the methods here have shown reasonable fidelity, over a wide range of singular values. However, it seems that there is something being left implicit about the power iteration scheme in Supplemental Note 1, Notably the choice of Rscale (Mentioned in Comment 3, left implicit in the text). For instance, looking at equation 3 if after deflation the value of the largest remaining eigenvector is less than 1. Note that we can consider $\mathbf{X}$ to be dimensionless if we take into account the factor Rscale. Given that the details behind the choice of Rscale are omitted from the current manuscript, it's hard to say how noise and errors propagate from iteration to iteration. Once the selection criteria for Rscale is defined, one can do another error-propagation analysis to potentially put a lower bound on the minimum detectable singular values, given typical amplitudes for the cell noise, relative to the maximum weight value in DPM. This error and noise analysis, which is strongly related to the error analysis requested in Comment 5, along with the noise figures for the chip at hand, would really enhance the theoretical foundations for this technique.

Response:

Thank you for this valuable question. Regarding the rescale factor used in data matrix mapping, as explained in our response to Question 3, we determine a fixed rescale factor for each matrix based on the maximum absolute value of the mapped matrix and the upper bound of the quantized numeric range.

For the minimum resolvable value, we analyze the system from the perspectives 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 $\sigma_q = \frac{M}{2\sqrt{3} \cdot 2^{a+b}}$ and normalized standard deviation $\sigma_{q,\text{norm}} = \frac{1}{2\sqrt{3} \cdot 2^{a+b}}$, where M denotes the maximum value of the matrix entries.

Using a memristor programming range of $G_{\text{max}} = 30 \mu\text{S}$ and an equivalent noise level of $\sigma_r = 0.3 \mu\text{S}$ as an example, the system-level numerical error (without SREA) is approximately $\sigma_X = \frac{\sigma_r M}{G_{\text{max}}}$. With SREA optimization, the error on the numerical matrix can be reduced to $\sigma_{X,\text{SREA}} = \left(\frac{\sigma_r M}{G_{\text{max}} \cdot 2^a}\right)$. Moreover, for vector operations, the numerical error can be further reduced to $\sigma_{v,\text{norm,SREA}} = \left(\frac{\sigma_r M}{G_{\text{max}} \cdot 2^a \cdot \sqrt{\#\text{RPT}}}\right)$, where $\#\text{RPT}$ denotes the repeated number of mapped vectors. For practical deployment configurations (e.g., $a = 2$ or 3 , and $b = 5$), we observe that 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. We consider a ground-truth numerical matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ and an error matrix $\mathbf{T}$. Because the properties of $\mathbf{X}$ affect the stability of the SVD solution, we assume that $\mathbf{X}$ is not ill-conditioned and that its entries follow an i.i.d. Gaussian distribution $\mathcal{N}(0, \sigma_{\text{data}}^2)$. By the Gaussian extreme-value approximation, the maximum entry magnitude satisfies: $M = \sigma_{\text{data}} \sqrt{2 \ln(mn)} + O\left(\frac{\sigma_{\text{data}} \ln(\ln(mn))}{\sqrt{\ln(mn)}}\right)$. The spectral

norm¹ of $\mathbf{T}$ is $\|\mathbf{T}\|_2 = O\left(\sigma_{\mathbf{X},\text{SREA}}(\sqrt{m} + \sqrt{n})\right)$, and for $\mathbf{X}$, it is $\|\mathbf{X}\|_2 = O\left(\sigma_{\text{data}}(\sqrt{m} + \sqrt{n})\right)$. 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 can obtain

$$\frac{\|\mathbf{T}\|_2}{\|\mathbf{X}\|_2} = O\left(\frac{\sigma_r}{G_{\max} \cdot 2^a}\right).$$

Wedin's bound² characterizes how numerical perturbations affect the subspaces of solution in SVD. Specializing this result to the estimation of the i -th singular value (and its associated singular subspace), the angle between the computed vector in the space and the true vector can be bounded as:

$$\sin\theta \leq \frac{\|\mathbf{T}\|_2}{\delta},$$

where θ is the angle of i -th solved vector to the ground truth, δ is the spectral gap of left components which has $\delta = s_i - s_{i+1}$, s_i is the i -th singular value. Furthermore, in the power iteration scheme, the error does not accumulate unboundedly across iterations. The convergence rate of power iteration is governed by the ratio $\frac{s_{i+1}^2}{s_1^2}$; when the spectral gap is sufficiently large, each iteration contracts the angular error toward the true singular vector, ensuring that the per-iteration noise contributions do not compound to destabilize the solution. Therefore, for the solution to be meaningful, we require $\|\mathbf{T}\|_2 < s_i - s_{i+1}$, i.e., the perturbation magnitude must be smaller than the local spectral gap. As we replace the $\|\mathbf{X}\|_2$ with its first singular value s_1 , we have the criterion of reliable solving:

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

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

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

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

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

In practice, for high-accuracy objectives such as reconstruction or low-rank approximation, the truncation point can be chosen conservatively such that $\frac{s_i - s_{i+1}}{s_1} \gg$

$\sqrt{O\left(\frac{\sigma_r}{G_{\max} \cdot 2^a}\right)^2 + O\left(\frac{1}{2\sqrt{3} \cdot 2^{a+b}}\right)^2}$ (e.g., 0.096 vs. $\sqrt{O(0.0025)^2 + O(0.0023)^2}$ for COVID-19 data low-rank approximation). 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 $\frac{s_i - s_{i+1}}{s_1} > \sqrt{O\left(\frac{\sigma_r}{G_{\max} \cdot 2^a}\right)^2 + O\left(\frac{1}{2\sqrt{3} \cdot 2^{a+b}}\right)^2}$ (e.g., 0.0085 vs. $\sqrt{O(0.00125)^2 + O(0.0011)^2}$ for EMG feature-extractor).

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

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

Revision:

To illustrate these points, we have made the following revision in the manuscript:

1. We added this analysis into supplementary information as “**Analysis of minimum detectable singular values**”.

Comment #9

9. Use of the terms Accuracy vs. Precision: Figure 3a indicates the use standard errors (standard deviation of the experiments run), and 3b and 3c then compares distribution of these errors. The tightening of the standard error (which reflects repeatability of successive experiments) should be referred to as increases in the “precision” of the system. However, in line 145, these are listed as improvements in “accuracy”. This should either be listed in the text as improvements to precision, or what would be more interesting: Figures 3b and 3c should be redone to reflect actual improvements in accuracy (i.e. difference from ground truth). The distinction between accuracy and precision is important and the authors should review the document for any repeats of such typos.

Response:

Thank you for this question. In **Fig. 3a**, we report the mean and standard deviation of the repeated measurements and mark the position of the ground truth (software) value. This presentation allows readers to assess accuracy by visually comparing the mean against the ground truth, and to assess precision from the standard deviation across repeated runs.

We also clarify that the “error” reported in **Fig. 3b** and **Fig. 3c** is computed as the difference between our results and the software SVD ground truth. Specifically, **Fig. 3b** compares the mean absolute error of the singular values obtained by MSVD across repeated trials, with and without SREA, thereby highlighting the improvement in accuracy. **Fig. 3c** shows the distribution of singular-vector errors (i.e., deviations from the ground truth) across repeated experiments, which reflects both accuracy (how close the distribution is to zero) and precision (how concentrated the distribution is). In the original manuscript, we mainly discussed σ , which indeed relates to the system’s precision. We have now expanded the description in the text as: “**Fig. 3c** displays RSV numerical error distributions. Vectors solved with SREA have an error standard deviation ($\sigma=0.192$) that is half that of those without SREA ($\sigma=0.415$), suggesting improved precision achieved with SREA.” to make the interpretation of these plots clearer to readers.

In addition, we have reviewed similar statements elsewhere in the manuscript. For example, in lines 177-180: “The decomposition using the first and second singular vectors reveals varying pandemic trends across states (**Figs. 4d,e**). MSVD with SREA achieves high reconstruction accuracy with mean normalized errors of $\mu=-4.89\times10^{-5}$ for the first component and $\mu=6.72\times10^{-3}$ for the second component.” In such cases, the wording is intended to emphasize the deviation of the computed results from the ground-truth values, a usage we consider appropriate.

Revision:

To illustrate these points, we have made the following revision in the manuscript: We revised the sentences on page 6: “**Fig. 3c displays RSV numerical error distributions. Vectors solved with SREA have an error standard deviation ($\sigma=0.192$) that is half that of those without SREA ($\sigma=0.415$), suggesting improved precision achieved with SREA.**”

Comment #10

10. Benchmarks of energy efficiency and normalized speed: A major promise of neuromorphic compute is an increase in the power performance relative to the conventional state of the art. This paper claims a 50x power advantage for MSVD w/SREA over an A100 GPU and a 37x advantage in computing speed. Though these numbers are both potentially underselling power of the GPUs. Looking at supplemental reference 17, it would appear that the authors are taking the 78 Peak TFLOPs for non-tensor FP16 operations and dividing this by the 400W consumed by an A100. This seems reasonable, were it not for the fact that vector operations such as SVD are prime candidates for tensor memory processing. If SVD uses mostly tensor cores then the energy efficiency goes up 4x to 312 TFLOPs, which is a significant decrease bringing the energy efficiency boost down to approximately 13x, which is still high. If the authors strongly believe that the Tensor operations are not employed here, then some argument is in order. For instance, the strongest argument would come from profiling of the actual SVD tasks as performed on an A100. Meanwhile, in terms of area efficiency, the authors are considering the entire area of the GPU die rather than the active CMOS area occupied by the FP16 cores used in this. Is the MSVD area quoted also a whole chip estimate? Please confirm that the area figure of merit for “required area” for MSVD and the A100 is the same.

Response:

Thank you for this valuable suggestion. We utilized PyTorch’s (PyTorch 2.3.0 with CUDA version 12.1) low-rank SVD API to test SVD on an A100 GPU, and employed NVIDIA Nsight Systems to analyze actual hardware utilization. The results revealed that the A100 GPU operates in TF32 tensor core mode during SVD execution (with no support for 16-bit implementation). Therefore, we agree that comparing against tensor core performance is appropriate. To provide a more realistic comparison reflecting practical working scenarios, we have updated the baseline to A100 TF32 Tensor Core performance and revised all related benchmark results accordingly. As shown in **Fig. R44**, compared to the A100 TF32, MSVD demonstrates a 29.9× energy efficiency advantage, while Incremental MSVD achieves a 67.1× energy efficiency advantage.

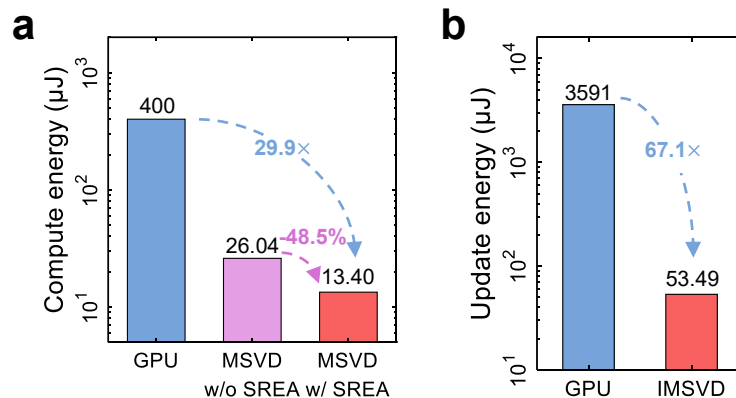


Fig. R44, Energy efficiency of (a) MSVD and (b) incremental MSVD versus GPU for singular value decomposition and 3-update user data decomposition.

The A100 chip area comprises multiple components, including cache banks, memory controllers, streaming multiprocessors (SMs), on-chip crossbar ports, and

others. While NVIDIA has not disclosed the specific area breakdown of individual modules, we believe that tensor cores and other compute cores likely constitute a substantial portion of the total area. Furthermore, all GPU computations inherently depend on cache access, instruction scheduling, and memory operations. Therefore, when operating at peak computational throughput, the majority of the GPU area is actively engaged.

In our benchmark, we evaluated the area of the computational and control modules required for MSVD, which collectively perform the necessary computations. While this area typically does not encompass the entire macro chip, in mature macro designs, these functional modules occupy nearly the entire chip area. For GPU area comparison, since we utilized the GPU's peak performance when calculating normalized speed, we believe that under this operating condition, most of the chip area is essential for supporting computation. Therefore, using the total GPU chip area as the benchmark is appropriate, and excluding the small fraction of inactive area would not significantly impact the benchmark results. This approach is also consistent with commonly adopted benchmarking methods in recent publications^{1,2}. Accordingly, we have updated the normalized speed comparison as shown in **Fig. R45**.

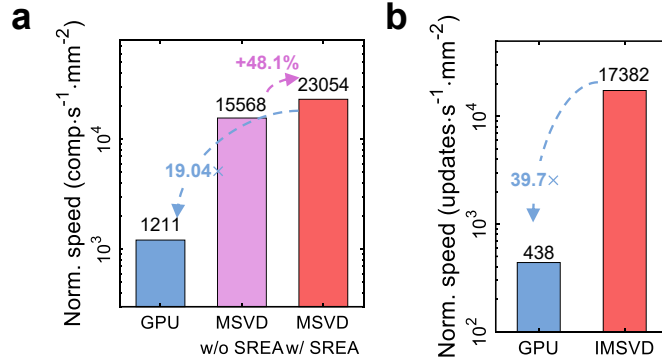


Fig. R45. Normalized speed of (a) MSVD and (b) incremental MSVD versus GPU for singular value decomposition and 3-update user data decomposition.

[1] Xu, M. et al. Efficient modeling of ionic and electronic interactions by a resistive memory-based reservoir graph neural network. *Nature Computational Science* 5, 1178-1191 (2025).

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

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised all the benchmark results compared with A100 GPU accordingly.
2. We revised the **Fig. 6** with **Figs. R44,45**.

Comment #11

11. Micrographs: Please add dimensions to the chip micrographs in Supplementary Figure 3. The “area required for MSVD implementation” of this chip is cited in the text as 0.199 to 0.243 mm² which seems much smaller than this micrograph. Looking at the number of bond pads and assuming a 70um pad size the chip appears to be about approximately 2.5mm on a side(?)

Response:

Thank you for this question. The values 0.199 and 0.243 were obtained from the hardware footprints required by MSVD and incremental MSVD, respectively, using the established circuit-level simulation tool XPESim under 32 nm technology-node. The chip used in our experiments was fabricated in 130 nm. For the convenience of testing and applications, the chip layout (e.g., array, drivers and multiplexers) were implemented with area redundancy, which incurs an area overhead than what would be expected in a mature, production-oriented tape-out. Following your suggestion, we added the dimensions to the chip micrographs as shown in **Fig. R46**.

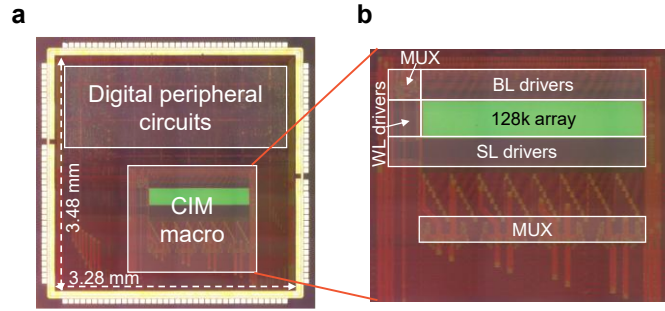


Fig. R46. Circuit modules in the memristor chip. **a**, Photograph of the 128k-cell memristor chip showing detailed locations of functional blocks. **b**, Detailed designs of compute-in-memory (CIM) macro in the chip. BL, bit line; SL, source line; WL, word line; MUX, multiplexer.

In **Table R11**, we have added the measured area of the analog macro of our 130 nm, 128K memristor chip, and further scaled this area to the 32 nm technology node, to provide a direct comparison with **Supplementary Table 8**.

Table R11. 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

As shown in **Table R11**, the area measured from the fabricated chip, when scaled to the 32 nm technology node and compared at a similar array size, is comparable to the benchmark results reported in **Supplementary Table 8**. We note that with layout-

level optimization, the area could be further reduced. Therefore, we believe the results in **Supplementary Table 8** remain a meaningful and representative reference.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the **Supplementary Figure 3** with **Fig. R46**.
2. We added **Table R11** as **Supplementary Table 9**.
3. In the original **Supplementary Note 8**, we added the content as: “To validate these benchmark results, we measured the 130 nm analog macro, which occupies approximately 1.18 mm². When scaled to the 32 nm node, this projects to roughly 0.074 mm² (detailed in **Supplementary Table 7**). 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 6** for a similar array size, confirming the accuracy of our area estimates.”

Comment #12

12. Completeness of noise models and actual chip power in Supplementary Note 3: It is clear that comparing the power of an A100 GPU to a chip fabricated in a 130nm process is not a fair comparison, and so some scaling is in order. Though why do the authors choose 32nm to model the chip power, are there any tools available to you which can give estimates of power in 7nm, the same process as the A100? Also, it would be reassuring to have a complete power model at both 130nm and 32nm, and to also cite experimental power measurements for the 130nm device as operating in the lab. Completeness of power modelling is sometimes challenging to get right, and so having this comparison between the actual chip and the expected power 130nm consumption would be a strong confirmation of the completeness of the power model.

Response:

Thank you for this valuable question regarding the completeness of our power model. Regarding the choice of technology node, the 32nm/28nm node has been demonstrated to fabricate RRAM with high yield for compute-in-memory (CIM) applications^{1,2}. Similar technology nodes are also commonly adopted in recent memristor-based system evaluations^{3,4,5}. Therefore, we scaled our implementation to 32nm for comparison with advanced-node GPUs.

Furthermore, to provide a more comprehensive analysis, we have extended our benchmarking framework to support evaluation across a range of technology nodes. We employed XPESim and ADC scaling methodologies^{6,7} to scale the array, peripheral circuits, and ADCs across different technology nodes. Previously, the ADCs were not uniformly scaled to 32 nm. We have corrected this in the updated benchmark results presented in **Table R12** and **Table R13**.

Table R12. Energy breakdown of MSVD and incremental MSVD

Operations	w/ SREA (μ J)		w/o SREA (μ J)	
Update singular vectors	0.61		4.61	
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

As shown in **Fig. R47**, we benchmarked the memristor array scaled to more technology nodes. At the 7nm node, MSVD achieves 61.7 $\times$ energy efficiency improvement and 155.4 $\times$ speedup compared to the A100 GPU. We note that published data for advanced-node (<20nm) ADCs and memristor-based CIM remain limited, which may affect the reliability of 7nm benchmark result.

Following the reviewer's suggestion, we benchmarked MSVD against GPUs at similar technology nodes to ensure fair comparison, and also evaluated MSVD performance at the 130nm technology node. At the 22nm node, MSVD achieves 413 $\times$ energy efficiency improvement and 358 $\times$ speedup compared to the P100 (16nm). At 32nm, MSVD achieves 432 $\times$ energy efficiency improvement and 321 $\times$ speedup compared to the M40 (28nm). Notably, even at the original 130nm node, the MSVD achieves 10 $\times$ energy efficiency improvement and 1.9 $\times$ speedup compared to the 7nm

A100 GPU. These results consistently demonstrate the significant energy efficiency and speed advantages of MSVD comparing to the CMOS-based HPC systems.

Table R13. Area breakdown of MSVD and 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

IMSV	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
IMSV: incremental MSV

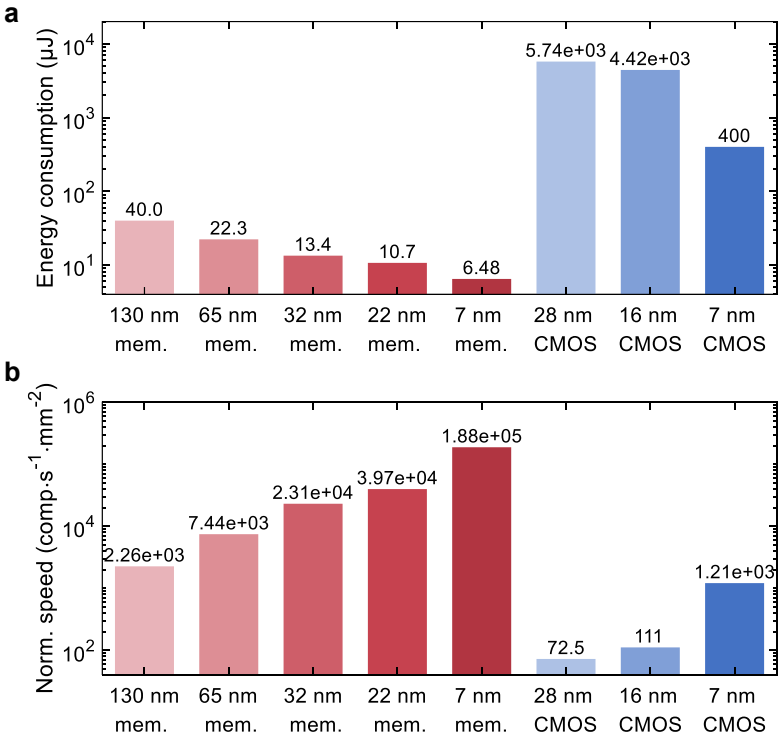


Fig. R47. Benchmark of energy consumption and normalized speed of MSVD compared to the CMOS-based SVD under different technology node. NVIDIA M40, P100 and A100 GPUs are used as 28nm CMOS, 16nm CMOS, 7nm CMOS

baselines respectively.

Following the reviewer's suggestion, here we provide additional experimental validation at 130nm to confirm the completeness of our power model. The system multiplication energy consumption is formed by the consumptions of analog macro and ADCs. We measured the analog macro energy consumption for computing an 800×128 matrix-vector multiplication (with average cell conductance $\sim 10 \mu\text{S}$) with 8-bit inputs to be less than 6.4 nJ. For ADCs, we adopted measured energy data from previously published work; when scaled to 130nm, the ADC energy consumption is approximately 9.9 nJ. Thus, the total energy for one 8-bit MAC operation is less than 16.3 nJ. In comparison, our tool benchmarks the corresponding energy at 14.36 nJ. The close agreement between experimental measurements (< 16.3 nJ) and simulated results (14.36 nJ) confirms the validity and completeness of our benchmark model.

- [1] Yao, P. et al. A 28 nm RRAM-Based 81.1 TOPS/mm²/bit Compute-In-Memory Macro with Uniform and Linear 64 Read Channels under 512 4-bit Inputs. in 2024 IEEE European Solid-State Electronics Research Conference (ESSERC). 577-580.
- [2] Zhang, J. et al. A 28nm 4Mb Embedded RRAM IP with Record-High Endurance of 107 Cycles and 10 Years@125°C Retention through Reliability-Enhanced Design-Technology Co-Optimization. in 2024 IEEE International Electron Devices Meeting (IEDM). 1-4.
- [3] Li, J. et al. Sparse matrix multiplication in a record-low power self-rectifying memristor array for scientific computing. Sci Adv 9, eadf7474 (2023).
- [4] Yu, R. et al. A full-stack memristor-based computation-in-memory system with software-hardware co-development. Nature Communications 16, 2123 (2025).
- [5] Lin, N. et al. Resistive memory-based zero-shot liquid state machine for multimodal event data learning. Nature Computational Science 5, 37-47 (2025).
- [6] Zidan, M. A. et al. A general memristor-based partial differential equation solver. Nature Electronics 1, 411-420 (2018).
- [7] Verhelst, M. & Murmann, B. Area scaling analysis of CMOS ADCs. Electronics letters 48, 314-315 (2012).

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised all the benchmark results to keep all the components in memristor-based system are benchmarked in the same node.
2. We revised the **Supplementary Table 6** with **Table R12**.
3. We revised the **Supplementary Table 8** with **Table R13**.
4. We added **Fig. R47** as **Supplementary Figure 27**.
5. We added technology scaling analysis in supplementary materials as: “

Supplementary Note 7. Technology scaling analysis of MSVD.

We utilized XPEsim and the ADC scaling method^{3,22} to further analyze the scaling trends of MSVD's energy efficiency and normalized speed across different technology nodes. **Supplementary Figure 27** illustrates these metrics for the EMG-based gesture recognition task, benchmarking MSVD against the 28 nm M40, 16 nm P100, and 7 nm A100 GPUs. When scaled to the 22 nm node, MSVD achieves a $413\times$ improvement in energy efficiency and a $358\times$ speedup compared to the P100 (16 nm). Similarly, at the 32 nm node, MSVD demonstrates a $432\times$ improvement in energy efficiency and a $321\times$ 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.”

6. In **Supplementary Note 6**, we added the content as: “We measured the multiplication energy consumption of analog macro at the 130 nm node. For an 800×128 matrix (with an average cell conductance of ~10 μ S) 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.”

Comment #13

13. Definition of “Global” vs. “Individual” landscape views: The convergence plots in Figure 6 are fascinating though they could use some more explanation. Firstly, the role of stochastic error in these figures is hard to assess. In panel 6a, the impact of DPM and VPR are made clear by presenting differences in the actual trajectory tracked through the loss what appears to be loss trajectories for the MSVD landscape. However in 6b, the reconstruction errors themselves have different shapes to reflect the different loss trajectories (if it helps to explain these, there are a number of works on loss trajectories such as <https://arxiv.org/pdf/1712.09913>), however it is not clear to this reviewer how the presence of stochastic noise (and the noise management approaches in SREA) lead to different shapes for the loss surfaces in Figure 6b. Could you please explain?

Response:

Thank you for raising this point. We also thank the reviewer for recommending the work on loss landscape visualization (arXiv:1712.09913), which we have cited in our revised version. 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. This differs fundamentally from neural networks: the iterative process does not converge through loss-driven weight updates, but rather 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, and we aim to intuitively visualize the landscape of these “weights” during the solving process. For **Figure 3f** and **Supplementary Figure 6b**, we applied PCA to reduce the dimensionality of the iterative solutions (concatenating U, S, V obtained at each iteration) to two dimensions, thereby obtaining two orthogonal directional vectors, Vec_{dim1} and Vec_{dim2} . We then sampled along these two directions as $\alpha \times \text{Vec}_{\text{dim1}} + \beta \times \text{Vec}_{\text{dim2}}$ (where (α, β) 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.

While this methodology is analogous to that presented in arXiv:1712.09913, our application differs in that we explicitly incorporate the effect of hardware noise. Here, we aim to demonstrate how noise affects solution quality under different configurations by visualizing the error landscape. During each solving iteration, the prominence of the current largest singular value relative to the residual facilitates convergence, System noise reduces the signal-to-noise ratio of the target singular value at each iteration step, causing the dominant eigenvalue to become less distinguishable from the background residual; this suppresses the sharpness of the error gradient and produces a flatter landscape. SREA addresses this by actively estimating and compensating for the noise contribution at each iteration, thereby preserving the relative prominence of the target singular value—which in turn maintains steeper gradients in the reconstructed error landscape. The gradient of this landscape therefore reflects both the prominence of the singular value being solved for the corresponding rank and the degree to which system noise impedes the iterative process, providing an intuitive visualization of convergence difficulty under each configuration.

Regarding why different configurations yield differently shaped landscapes in **Supplementary Figure 6b**: each configuration’s iterative trajectory traverses a distinct subspace of the solution space, and PCA applied to each trajectory independently yields different principal directions (Vec_{dim1} , Vec_{dim2}). The sampled landscape is defined relative to these configuration-specific principal directions and therefore captures a

different local geometry of the error surface for each case, resulting in visually distinct landscape shapes across configurations. Specifically, the MSVD landscape with SREA optimization exhibits steeper gradients, indicating reduced noise influence during the iterative process and facilitating convergence.

However, because different configurations converge along different directions, direct comparison of path directionality is not intuitive with this approach. To address this, we constructed a global landscape and iteration path analysis by combining the overall iterative trajectories across different configurations as shown in **Supplementary Figure 6a**. This figure enables readers to directly compare the solution paths under different methods. Notably, the iteration with SREA optimization follows a more direct and efficient path, resulting in faster practical convergence.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added analysis of the landscape into the supplementary information as “**Analysis of error landscape on IRIS dataset**”.

Comment #14

14. Cell switching characteristics: In Supplementary Figure 4, cell switching characteristics are shown.

a. IV curves: How are these measured? Why has a single device been chosen? Ideally, the authors should report IV curves across multiple devices to show the overall distribution of the array (keeping the measurement methodology fixed).

b. Conductance histograms: These should be further explained. How are these extracted? What is the write and read methodology? What portion of devices in the array are being used for these results (for example, are different portions of the array being tuned to different target conductance states and then read back)? Is there any filtering being done (for example, were there any shorts or stuck devices that had to be avoided?). I understand that this is an algorithmic demonstration, but explaining device properties across the array would paint a better picture of what would be needed at a device-level for recreating the algorithm's performance on the same benchmarks in some other prototyping system.

Response:

Thank you for this valuable question. For question (a), the I–V curves were obtained by applying a sweeping voltage between the top and bottom electrodes of the memristors and measuring the resulting current. Operationally, we enable the target column by selecting the corresponding WL, and use the BL as row input to address an individual device.

During the SET operation, we apply an increasing voltage from 0 to 2.5 V across the selected BL and SL ($V_{BL,sel} - V_{SL,sel}$) with a compliance current of 100 μ A, and then sweep the voltage back down from 2.5 V to 0 V. The current flowing into the selected BL is recorded throughout the sweep. During the RESET operation, we apply an increasing voltage from 0 to 2.2 V across the selected SL and BL ($V_{SL,sel} - V_{BL,sel}$) with a compliance current of 220 μ A, and then sweep the voltage back down from 2.2 V to 0 V. And the current throughout the sweep is recorded.

Following your suggestion, we have plotted DC I–V curves from six devices located at different positions on the chip (**Fig. R48, Fig. R49b**). Although devices at different locations exhibit **variation**, the measurements show that all of them provide a good ON/OFF ratio and uniform, stable SET/RESET behaviors, indicating that they can be programmed to multilevel states with good accuracy.

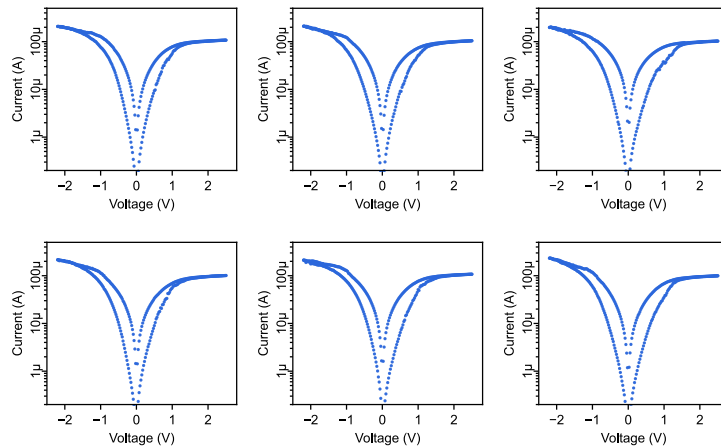


Fig. R48. DCIV curves of 6 1T1R memristor cells from different locations.

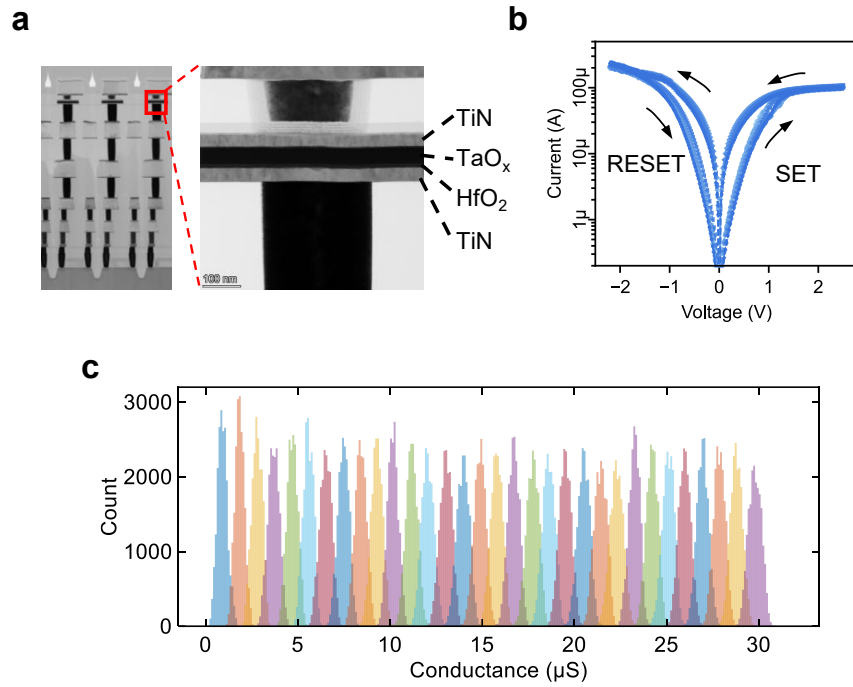


Fig. R49 (Supplementary Figure 4). Memristor cell and its switching characteristics. **a**, TEM photo of one-transistor-one-resistor cells forming the memristor array. Each cell comprises a memristor with a TiN/TaO_x/HfO₂/TiN material stack. **b**, Direct-current I-V curves during SET and RESET from 6 memristors, demonstrating excellent conductance switching characteristics. **c**, Measured conductance of memristors mapped to 32 uniformly distributed targets.

Regarding question (b) for **Fig. R49c**, in the experiment, we set 32 target conductance levels. For each level, we randomly sampled 100 devices and programmed them to the target level using a write-verify procedure. We then measured the actual read conductance of each programmed device by applying the read voltage 200 times and recording the corresponding conductance values.

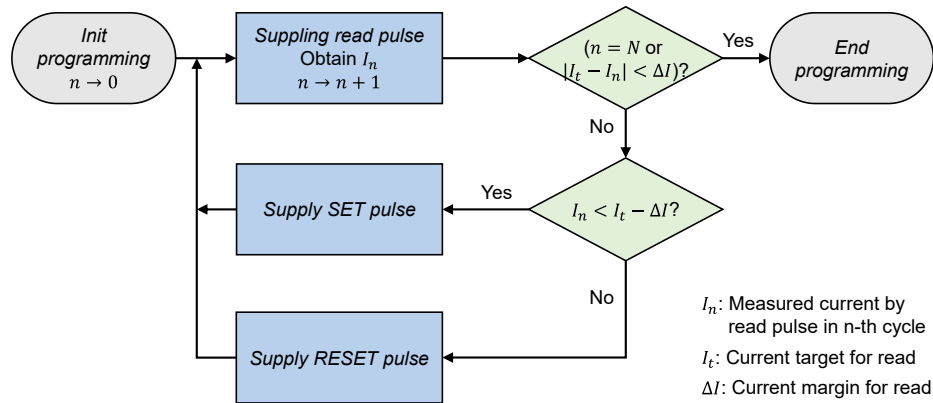


Fig. R50. Workflow of programming with write and verify processes.

Fig. R50 illustrates the workflow for cell programming. During the write stage, we apply positive or negative (SET/RESET) pulses with a pulse width of 50 ns and an amplitude in the range of 0.5–2 V. During the read (verify) stages, we use a 50 ns, 0.2 V pulse to read out the device conductance. For visualization, we group the

measurements from different devices programmed to the same target conductance level. Specifically, for each target level we aggregate 200×100 conductance readouts (200 repeated reads for each of 100 devices) and plot the resulting histogram. Different colors are used to distinguish the distributions corresponding to the 32 target levels. In our experiments, we did not observe shorted or stuck devices; therefore, no filtering was required to exclude such devices from the reported measurements.

Together, the I–V characterization across six spatially distributed devices and the conductance histograms across 32 target levels collectively demonstrate that the array exhibits consistent switching behavior, a well-defined ON/OFF ratio, and sufficient multilevel programmability to support the algorithm. The write–verify procedure with 50 ns pulses and iterative amplitude adjustment is the key enabling mechanism for achieving accurate conductance placement. We hope that these characterization results and the documented programming methodology provide a useful reference for researchers seeking to reproduce or adapt the algorithm on other memristive hardware platforms.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the **Supplementary Figure 4** with **Fig. R49**.
2. We added **Fig. R50** as **Supplementary Figure 22**.
3. On page 17, we added: “A write–verify scheme is employed to program the conductance of the memristor devices, as illustrated in **Supplementary Figure 22**. The linearly quantized data (**Supplementary Note 5**) are programmed by applying SET/RESET voltage pulses ranging from 0.5 V to 2 V, followed by a read verification step at 0.2 V. This write–verify cycle is repeated until the read current falls within the target threshold range. **Supplementary Figure 23** details the relative voltage configurations applied to the WL, SL, and BL of the selected cell during the SET, RESET, and read operations, respectively. To accommodate both positive and negative matrix elements, a differential mapping scheme is adopted. Benefiting from the high yield of our devices (~99.99%), the impact of stuck-at faults on MSVD is minimal (as analyzed in **Supplementary Figure 24**), eliminating the need for complex fault-tolerant mapping strategies”

Comment #15

15. Hard Faults: In Figure 2 and surrounding text, the authors discuss how component residuals are a problem caused by device-level conductance mapping deviations. However, the deviations being reported/observed do not include hard faults (such as stuck or broken devices), which are typically a significant problem with any computation involving memristors. It would be great if the authors could add additional commentary on this. Think about questions along these lines: a. Were there no such devices in the fabricated array? i. If yes, could some fabrication-level details be added? ii. If no, was there explicit consideration or enforcement done to avoid such devices? b. How does MSVD's performance change if there were hard faults participating? Is SREA more tolerant to them compared to w/o SREA?

Response:

Thank you for raising this question. We did not observe apparent stuck-at faults during our experiments. No explicit screening or exclusion of devices was performed prior to the experiments; the absence of stuck-at faults is attributed to the intrinsically high yield ($\sim 99.99\%$) of the fabricated array. As we presented in the manuscript, each 1T1R cell comprises a memristor with a TiN/TaO_x/HfO₂/TiN material stack and an access transistor under 130nm technology node. The layer composition includes an 8 nm HfO₂ resistive switching layer deposited by atomic layer deposition, a 45 nm sputtered TaO_x thermal enhancement layer, and 30 nm sputtered TiN top/bottom electrodes (**Fig. R51**). Here we further added a TEM photo of the fabricated memristor cells in the **Supplementary Figure 4** and replaced original **Supplementary Figure 4a**.

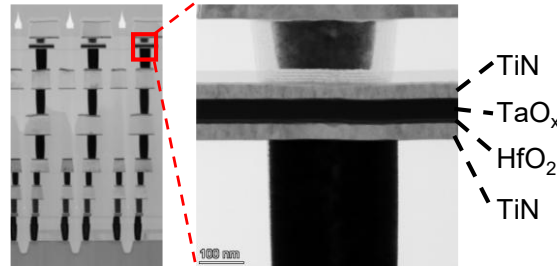


Fig. R51. TEM photo of one-transistor-one-resistor cells forming the memristor array. Each cell comprises a memristor with a TiN/TaO_x/HfO₂/TiN material stack.

On the other hand, we introduced random stuck-at faults into the data matrix through simulations on IRIS dataset and repeated the experiments 20 times. **Fig. R52** compares the mean squared error (MSE) of the 4 singular values obtained by MSVD with SREA and without SREA under different stuck-at fault rates. As shown in the figure, when the stuck-at fault rate remains below 1%, the solution error of MSVD with SREA stays relatively stable. Even when the stuck-at fault rate increases to 3%, the error can still be effectively controlled compared to MSVD without SREA. SREA suppresses hardware noise and brings the system into a precise operating regime where reconstruction accuracy is maintained at a high level; it is precisely within this regime that device faults produce meaningful and quantifiable degradation. Without SREA, the system operates under a noise floor that already dominates the error budget, such that the added impact of stuck-at faults becomes indistinguishable from the existing noise-induced error. This contrast underscores the value of SREA: by actively managing noise, it not only improves baseline accuracy but also preserves the system's sensitivity to

fault-induced degradation—making fault-tolerance a tractable and meaningful design consideration. Overall, even under relatively high stuck-at fault rates, MSVD with SREA can effectively control the influence of such non-idealities. Furthermore, with the typical array yield $\sim 99.99\%$, the impact of stuck-at fault is minimal. Therefore, we conclude that the impact of stuck-at faults is negligible in practice.

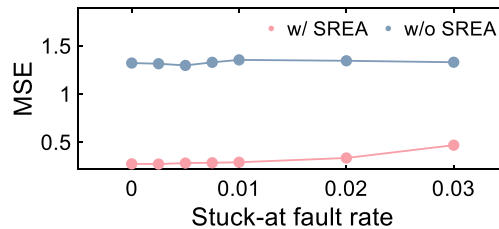


Fig. R52. Mean squared error (MSE) of the four singular values obtained by MSVD with and without SREA under different stuck-at fault rates. Each experiment was repeated 20 times.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. On page 15, we added: “Benefiting from the high yield of our devices ($\sim 99.99\%$), the impact of stuck-at faults on MSVD is minimal (as analyzed in **Supplementary Figure 24**), eliminating the need for complex fault-tolerant mapping strategies.”
2. We added **Fig. R51** in **Supplementary Figure 4**.
3. We added **Fig. R52** as **Supplementary Figure 24**.
4. We revised sentences in page 16 as: “... and 30 nm sputtered TiN top/bottom electrodes (shown in transmission electron microscope image in **Supplementary Figure 4a**)”

Comment #16

16. Correlations: In Figure 4f related text in the results section, the overall methodology for computing correlations is unclear. It would be helpful to know how these are being computed. Precisely which vectors are being compared to obtain these correlation values? If the analysis here is specific to this data set, please add a footnote here to the appropriate citations. Additionally, it would also help if the authors could comment on why SREA produces more correlated decompositions against the software SVD baseline compared to the non-SREA version (just to bring everything together).

Response:

Thank you for this question. We computed the correlations using the rank- i reconstructed matrices obtained from MSVD and from the software SVD baseline, in order to assess decomposition quality at each rank. Specifically, for a given rank i , we form the reconstruction as $\mathbf{X}_i = \mathbf{U}_i \mathbf{S}_i \mathbf{V}_i^T$, and then compute the correlation coefficient (with the equation shown in **Methods**) between $\mathbf{X}_i$ produced by MSVD (hardware) and $\mathbf{X}_i$ produced by software SVD (reference). This analysis is performed for the COVID-19 dataset⁴⁹. Following your suggestion, we have revised the figure caption and added the corresponding citation accordingly: “**d**, **e**, 1st- (**d**) and 2nd-rank (**e**) reconstructions of the COVID-19 dataset⁵² using MSVD with SREA. **f**, Comparison of correlation coefficients between MSVD with SREA and MSVD without SREA for 1st-, 2nd-, and 3rd-rank reconstructed matrices of the dataset⁵², using software SVD solutions as reference.”

We use the software SVD results as the ground truth and compare against both MSVD with SREA and MSVD without SREA. MSVD with SREA yields higher correlations because SREA addresses the major challenges in MSVD (numerical drifting and component residuals in the power iteration-deflation procedure), thereby improving the accuracy of the recovered components across ranks and leading to consistently better reconstructions.

Revision:

1. We revised the **Fig. 4** as described above.
2. We revised the sentence on page 7 as: “An analysis of correlation coefficients (defined in **Methods**) between reconstructed i -th rank matrices and the software ground truth shows that SREA-enhanced reconstructions maintain coefficients over 0.999, significantly outperforming MSVD without SREA for higher-rank reconstructions (**Fig. 4f**). This performance arises as SREA mitigates the numerical drifting and component residuals inherent in the power iteration-deflation procedure, thereby improving reconstruction accuracy on each solved rank”

Comment #17

17. Incremental MSVD: For incremental MSVD (specifically for results in Figure 5), I have a few questions. The manuscript would likely benefit from an added discussion on this aspect/clarification of Supplementary Figure 7 in the main text.

a. Going from the 0-update model with 2 users to the 1-update model with 4 users, I assume the 0-update model was used as a baseline and allowed to update based on Equations 11 and 12. If this is the case, then how is information from 2 additional users incorporated when only a single update is allowed? If this is not the case, then what is?

b. I am unclear on whether the linear regression classifier must also change with increasing # of allowed updates.

c. What are the sources of errors for Figure 5e experiments? Have the authors looked at source-wise contributions i.e. how much of it is noise from devices or the system vs. the algorithms? Do the 15 repeats use the same sets of devices?

Response:

Thank you for this question. Regarding question (a), as indicated in the Eqs. (13, 14), the information from the newly added users (user_3 and user_4) is assembled into X_{new} , while the right singular vectors (RSVs) obtained from the initial two-user (user_1 and user_2) data are denoted as V_{exist} and stored on the array. Here, a “single update” means completing one incremental update step, i.e., one full incremental MSVD solve based on V_{exist} and X_{new} . Using the NinaPro experiments as an example, one update involves solving for six new singular vectors. The updated RSVs, V' , provide a low-rank representation that incorporates information from all users (both the existing users and the newly added users). The resulting V' is then used for feature extraction and can also serve as the new V_{exist} for subsequent incremental updates.

Regarding question (b), after each Incremental MSVD update, we additionally update the regressor so that classification can be performed effectively. In our implementation, the regressor has a size of 7×4 and is obtained via an one-shot computation from the extracted features and labels (rather than via deep learning), i.e.,

$$\text{Mat}_{\text{regressor}} = (F^T F)^{-1} F^T Y$$

where F denotes the feature matrix and Y the label matrix. This step is computationally lightweight compared with incremental SVD.

Regarding question (c), as shown in Fig. R53, we have added a comparison against a software implementation to disentangle and quantify the impacts of the hardware system versus the algorithm itself.

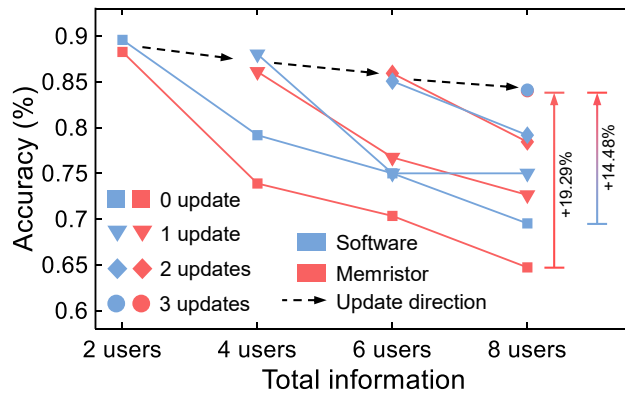


Fig. R53. Comparison of recognition accuracy evolution between software

solutions and memristor-based solutions.

Classification tasks typically exhibit tolerance to noise and moderate feature drift. Therefore, during successive user updates, incremental MSVD can still effectively capture the dominant components of the data, leading to classification performance comparable to the software baseline. In a small number of cases, noise perturbations may shift samples near the decision boundary, which can occasionally result in a slight increase in accuracy.

In contrast, when the features are not updated, incremental MSVD's approximation errors can reduce the non-dominant components shared between new and existing users compared with the software result. As the number of users increases, this mismatch may lead to a larger accuracy loss.

Overall, by using the software 0-update result as a baseline, we can quantitatively isolate the error contributions. For the 8-user scenario (shown in Fig. R53), the accuracy improves from 69.53% (software 0-update) to 84.01% (memristor-based 3-updates), representing a **14.48%** accuracy recovery. This 14.48% gap primarily reflects the algorithmic improvement gained by incorporating the missing features of the newly added users. Meanwhile, in the purely memristor-based setting, the total accuracy recovery from 0-update to 3-updates is **19.29%**. By comparing these two recovery margins ($19.29\% - 14.48\% \approx 4.8\%$), we can infer that approximately **4.8%** of the initial accuracy loss is explicitly attributable to memristor device noise, with the remaining loss due to the algorithmic absence of new user features.

Finally, the results reported from the 15 repeats were obtained using the same set of devices.

Response:

To illustrate these points, we have made the following revisions in the manuscript:

1. We added **Fig. R53** as **Supplementary Figure 11**.
2. On page 9, we added: “**Supplementary Figure 11** presents a comparison between the incremental MSVD-based classification during the user updating and SVD-based approach on software, where SVD is performed on the cumulative dataset at each corresponding stage. The results demonstrate that incremental MSVD not only restores the accuracy compromised by missing new-users' features but also mitigates the impact of device noise on non-dominant feature components, ultimately achieving software-comparable performance after user updating.”
3. We revised the sentences on page 8 as: “The complete system operates in two phases. During training, MSVD decompose the training signals to obtaining singular vectors as feature extractor (**Fig. 5a**), then a linear regression classifier is trained with the extracted feature.”
4. On page 19, we added: “During the continuous solving progress, V_{exist} and V are stored on the array. In each new round of incremental MSVD (when updating users), the previously computed V serves as the new V_{exist} , while the newly solved RSVs are overwritten onto the region originally storing last V_{exist} .”

Comment #18

18. Dimensions of singular values & gaps: In figure 6a, the authors present comparisons of the singular values across the three applications specified. However, it is important to note that in most cases the datasets under study have dimensioned data, and comparisons of magnitudes of the singular values (and hence a comparison of the gaps) are meaningless unless the datasets are first normalized to dimensionless quantities. For example, the famous Iris data set consists of linear spatial dimensions (petal and sepal lengths and widths, etc.). If the original Iris data set is expressed in units of mm, the singular values, s_2 , could change by a factor of 100 or even 1,000,000 if the units of this data set are converted to cm or m. Such a change of units would not change the spectral structure of the data, though it does make comparisons between disparate datasets problematic. This uncertainty of normalization when doing comparisons should be eliminated by expressing all gaps normalized to the first singular value for the same dataset. (i.e. each gap should be divided by the corresponding value of s_1). This also gives a more honest assessment of the capability of the MSVD, because in the end each dataset has to be renormalized to units of conductivity before they can be mapped into the X array, and so a normalization process is already implicitly done as part of the conductivity mapping.

Response:

Thank you for this important comment. We agree that singular values are dimensional quantities, and their magnitudes are directly tied to the physical units of the data. Following the reviewer's suggestion, we now use a normalized, dimensionless metric to present and compare the solution process across different tasks. We use the normalized singular gaps, defined as: $\text{Gap}_{i,\text{norm}} = (s_i^2 - s_{i+1}^2)/s_1^2$, where $i \leq n$ (n = maximum matrix rank), $s_{n+1} = 0$, and s_i represents the i -th singular value. We further note that, as you points out, the conductivity mapping process already implicitly involves a normalization step, which makes the use of dimensionless normalized gaps a natural and physically consistent choice for comparing performance across datasets.

Based on this definition, we have revised the original Fig. 6a as Fig. R54 and updated the original Supplementary Table 2 as Table R14.

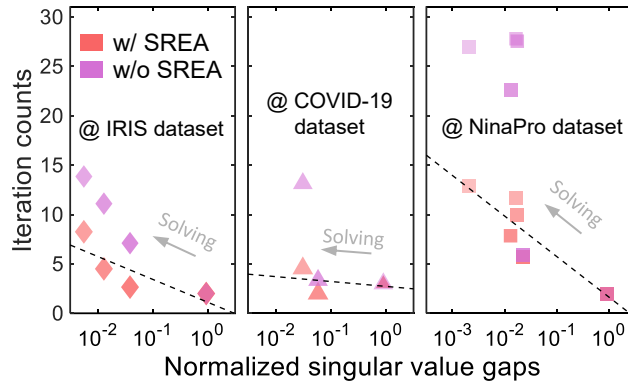


Fig. R54 (Fig. 6a). Convergence iterations versus normalized singular value gaps across datasets. The normalized gap for rank i is defined as $\text{Gap}_{i,\text{norm}} = (s_i^2 - s_{i+1}^2)/s_1^2$, where $i \leq n$ (n = maximum matrix rank), $s_{n+1} = 0$, and s_i represents the i -th singular value.

Table R14. 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

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the **Fig. 6** with **Fig. R54** as: “

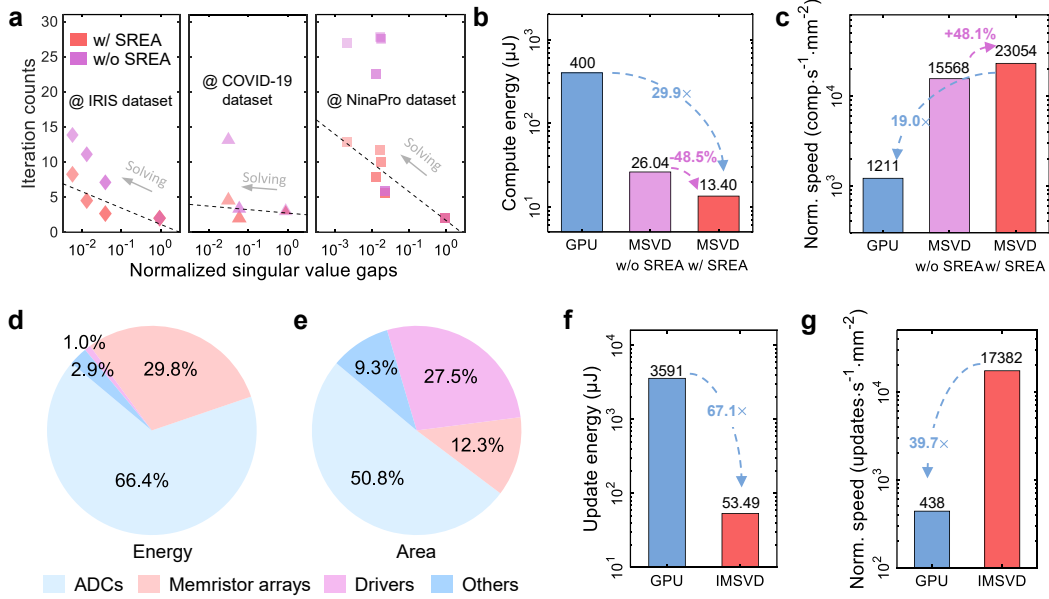


Fig. 6 | Convergence analysis and performance benchmarks of MSVD. **a**, Convergence iterations versus normalized singular value gaps across datasets. The normalized gap for rank i is defined as $\text{Gap}_{i,\text{norm}} = (s_i^2 - s_{i+1}^2)/s_1^2$, where $i \leq n$ (n = maximum matrix rank), $s_{n+1} = 0$, and s_i represents the i -th singular value. Black dashed lines indicate linear regression fits. Arrows show solution progression direction. **b,c**, Energy efficiency (**b**) and normalized speed (**c**) comparison of MSVD with SREA compared to GPU and MSVD without SREA for singular vector computation. Comp: computation. **d,e**, Compute energy breakdown (**d**) and chip area breakdown (**e**) of MSVD. ADC: analog-digital converter. **f,g**, Energy efficiency (**f**) and speed performance (**g**) of incremental MSVD for user addition scenarios.”

2. We revised the **Supplementary Table 4** with **Table R14**.

3. We revised the sentences on page 10 as: “The results reveal a generally linear trend between the number of convergence iterations and the logarithm of normalized singular value gaps across matrices of varying dimensions and normalized gap ranges from 10^0 to 10^{-3} (**Supplementary Table 4**), demonstrating the scalability of MSVD with SREA.”

Response letter to reviewers' comments

Reviewer #1 (Remarks to the Author):

The authors have made a substantial effort to revise the manuscript and address the concerns raised in the first round of review. The revised version shows clear improvements in methodological clarity, benchmarking scope, and reproducibility. In particular, the addition of hardware-calibrated large-scale simulations, expanded comparisons with ASIC-based implementations, and improved explanations of the system architecture and data mapping significantly strengthen the paper. The clarification of the distinction between experimentally validated components and simulated results is also appreciated, as it improves transparency. That said, while the manuscript is now considerably stronger, several key issues remain only partially addressed. These do not invalidate the contribution, but they limit the strength of some of the claims and should be addressed more carefully before publication.

Response:

Thank you for the careful re-evaluation and for recognizing the substantial improvements in methodological clarity, benchmarking scope, and reproducibility across the revised manuscript.

We appreciate your constructive feedback and respond to the remaining concerns point-by-point below. We believe these revisions further strengthen the paper.

Reviewer #1 – Detailed Second-Round Evaluation (Expanded)

Comment 1: Scalability and Large-Scale Validation

The authors have made a commendable effort to address scalability by introducing large-scale experiments (e.g., LoRA/LLM-inspired workloads) using hardware-calibrated simulations. This addition demonstrates awareness of the importance of evaluating beyond small benchmark datasets. Despite this effort, the scalability claims remain supported exclusively by simulation. In memristor-based systems, large-scale behavior is strongly influenced by non-ideal factors such as wire resistance/capacitance, ADC/DAC bandwidth constraints, array tiling overhead, and cumulative device noise across iterative operations. These effects are often nonlinear and difficult to capture accurately in simulation, particularly when scaling to very large matrix sizes or deep iteration chains. As a result, the current evidence does not fully validate the claimed system-level scalability.

Actionable fix: Clearly annotate all large-scale results as simulation-based projections in both figures and text. Add a scalability analysis section that explicitly models system-level overheads, including tiling strategies, interconnect costs, and peripheral bottlenecks. Provide a breakdown of latency and energy contributions (array vs ADC/DAC vs communication) as a function of matrix size. If feasible, include at least one intermediate-scale hardware experiment to bridge the gap between small-scale validation and large-scale projections.

Response:

Thank you for the feedback on scalability validation. We have made the following clarifications and additions.

Regarding simulation annotation, the original manuscript stated that LLM-scale evaluations use hardware-calibrated simulation with modeled memristor noise and peripheral circuit behavior (lines 524-530). Following the reviewer’s suggestion for greater clarity, we have now added explicit “simulation” annotations to **Fig. 7** captions.

To directly address the reviewer’s request for a breakdown of energy and area contributions, we have added two new supplementary figures. **Fig. R1** provides detailed pie charts showing the proportion of energy consumption and area occupied by different hardware components (ADCs, memristor arrays, drivers, and other peripheral circuits) for Llama 3.2 1B and 3B model initialization. **Fig. R2** extends this analysis across eight representative workloads of increasing scale, showing how the component-wise breakdown of energy and area evolves with problem size.

Drawing on our benchmarks across EMG processing, EEG processing, and LLM model initialization, these results help illustrate how peripheral overheads evolve with matrix size and iteration depth, and provide additional component-level support for the scalability discussion.

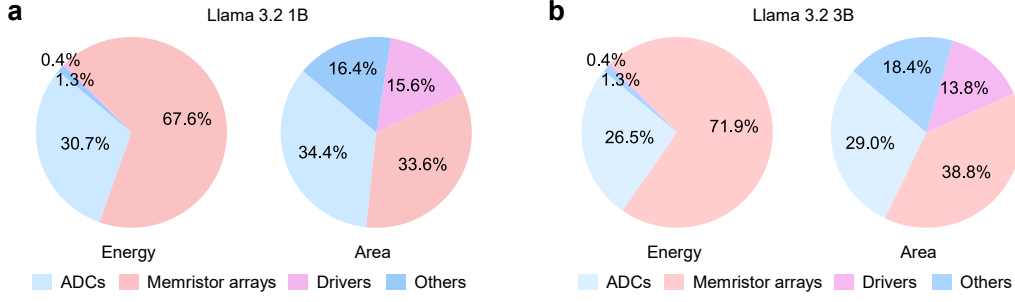


Figure R1. System-level energy and area breakdown for LLM weight initialization using MSVD. a, Llama 3.2 1B model. Pie charts showing the proportion of (left) energy consumption and (right) area occupied by different hardware components. Components include ADCs, memristor arrays, Drivers, and other peripheral circuits. **b, Llama 3.2 3B model.** Energy and area breakdown for the same hardware components.

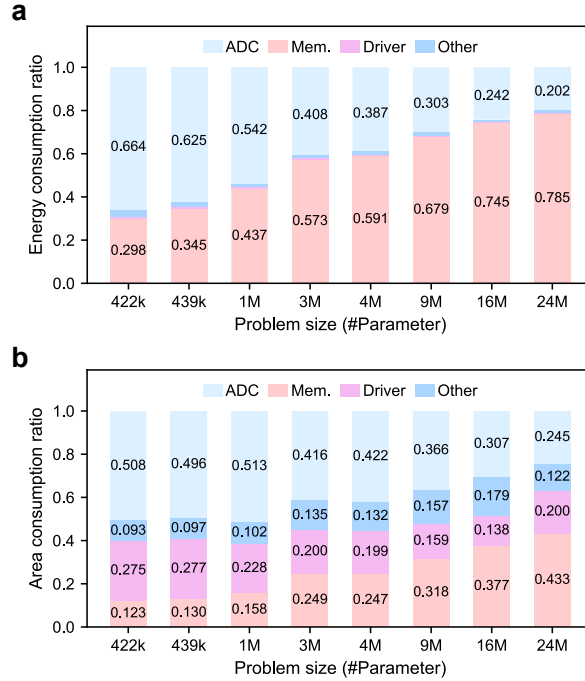


Figure R2. Memristor array, ADC, driver, and peripheral circuit contributions to total system energy and area across tasks of different scales. The 422k and 439k problem sizes correspond to the EMG-based gesture recognition and EEG-based sleep stage detection tasks, respectively, while the 1M to 24M 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.

Regarding the validation methodology, our approach bridges hardware measurements and simulation as follows. The hardware experiments on our fabricated

arrays validated the core MSVD algorithm on matrices up to 864×500 with measured device noise characteristics ($\sigma=0.27\mu\text{S}$ for mapping variations, $\sigma=0.21\mu\text{S}$ for reading fluctuations, as shown in **Fig. 2g** and reported in lines 256-259). The larger-scale simulations are explicit extensions of these measured baselines, using the same measured device statistics combined with circuit-level models of ADC behavior that were validated against our peripheral circuit measurements. Moreover, across matrices of varying dimensions, the number of convergence iterations scales linearly with the logarithm of the normalized singular value gap (**Fig. 6a, Supplementary Table 4**), demonstrating the scalability of MSVD. We believe this three-tier validation approach provides a rigorous basis for our scalability claims: measured hardware validation on fabricated arrays, hardware-calibrated simulation incorporating dominant non-idealities, and convergence analysis. We have been transparent throughout about where simulation-based extrapolation is used.

Revisions:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the captions **Fig. 7** with explicit “simulation” annotations.
2. We added **Figs. R1, R2** as the **Supplementary Figures 18, 31**.
3. We revised the method on page 12 as: “... see **Supplementary Figure 18** for the system-level energy and area breakdown of MSVD-based initialization on the LLaMA 3.2 1B and 3B models ...”; on page 21: “The energy breakdown of each system module across tasks of different scales is further shown in **Supplementary Figure 31** ... the area breakdown of each system module across tasks of different scales is shown in **Supplementary Figure 31** ...”.

Comment 2: Dataset size and diversity

The manuscript has been extended to reference more complex workloads, which improves the perceived breadth of applicability.

Response:

Thank you for acknowledging the expanded workload diversity in the revised manuscript.

Comment 3: Accuracy and reconstruction quality (PSNR/SSIM)

The authors acknowledge the trade-off between hardware efficiency and reconstruction accuracy, which is an important aspect of the design.

Response:

Thank you for recognizing this important design consideration.

Comment 4: Gesture recognition robustness

The incremental learning results are better explained, and the framework shows promise for adaptive scenarios.

Response:

Thank you for recognizing the potential of our framework in adaptive scenarios.

Comment 5: Incremental learning scalability

The authors extend incremental MSVD evaluation through simulation and additional explanation.

Response:

Thank you for acknowledging the extended incremental MSVD evaluation.

Comment 6: Benchmarking beyond GPU baselines

The addition of ASIC-based comparisons and technology scaling is a strong and important improvement. However, these comparisons rely on cross-technology extrapolation and modeling assumptions, particularly for analog/mixed-signal components such as ADCs. Differences in precision, workload configuration, and convergence criteria may also affect fairness.

Actionable fix: Clearly document all assumptions used in scaling analysis, including voltage, frequency, ADC resolution, and memory access patterns. Provide a sensitivity analysis showing how results vary with key parameters (e.g., ADC energy, precision). Where possible, include a matched-precision comparison to ensure fairness across platforms.

Response:

Thank you for recognizing the added ASIC comparisons. We address your concerns regarding cross-platform fairness and modeling assumptions.

The technology-node scaling analysis (**Supplementary Figure 28**) follows the established methodology of Verhelst & Murmann [1] and Zidan et al. [2], which derives ADC scaling relations from hundreds of measured data points. We have now explicitly documented the relations in **Supplementary Note 7**, with the key parameters including area, energy consumption, sampling frequency, and effective number of bits. Regarding memory access patterns, we note that our in-memory computing architecture fundamentally differs from von Neumann systems: data resides in the crossbar arrays and is processed in place, eliminating the memory access bottleneck.

As digital ASIC implementations typically operate at higher numerical precision than our analog system, we have added **Table R1** (revised **Supplementary Table 7**) to provide a precision-normalized comparison. We normalize energy and speed by the data precision for fairer cross-platform assessment. Under this normalization, MSVD maintains substantial efficiency advantages, confirming that the reported gains reflect genuine architectural benefits rather than precision mismatch.

Table R1. Comparison of MSVD to other CMOS-based ASICs designed for decomposition applications

Works	Appl.	Architecture	Benchmark tech. node (nm)	Data precision	Problem scale	Decomposition efficiency* (Vec-uJ ⁻¹)	Norm. decomposition efficiency# (Elem-uJ ⁻¹)	Precision norm. efficiency† (Elem-bit-uJ ⁻¹)	Decomposition Speed* (kVec-s ⁻¹ -mm ⁻²)	Norm. decomposition speed# (kElem-s ⁻¹ -mm ⁻²)	Precision norm speed† (kElem-bit-s ⁻¹ -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	7.4×10 ⁻¹	4.7×10 ¹	1.5×10 ³
TCAS I 2024	SVD	CMOS-based von Neumann	90	22	16×8×8 (Tensor)	0.33	2.64	58.1	5.9×10 ⁰	4.7×10 ¹	1.0×10 ³
Microelectron. J. 2025	EVD	CMOS-based von Neumann	22	10	1024×1024	0.0048	4.92	49.2	1.5×10 ⁻¹	1.5×10 ²	1.5×10 ³
This work	SVD	Memristor-based CIM	32	[3, 5] DPM	864×500	0.43	215	1935	1.3×10²	6.5×10⁴	5.9×10⁵

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.

Beyond the precision-normalized comparison, our conclusions are supported by multiple independent validation approaches. The hardware experiments on fabricated devices validate the algorithm under real device noise; the theoretical error analysis derives a closed-form criterion that explicitly links solvability to key system parameters (e.g., device noise and resolution), characterizing parameter sensitivity analytically rather than only empirically; and the hardware-calibrated simulations use measured device statistics for larger problem sizes. This combination provides robustness beyond what parameter sweeps alone could achieve. Additionally, the ADC scaling model is based on statistical fitting across diverse designs [1,2], already accounting for design-space variation.

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

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

Revisions:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the **Supplementary Table 7** as **Table R1**.
2. We revised the **Supplementary Note 7** as: “The scaling of ADCs is performed based on statistical analysis of ADC designs reported in the literature ... we extract the following coefficients for the SAR ADC employed in our benchmark: $\alpha_0=-0.55$, $\alpha_1=1.07$, $\alpha_2=0.34$, $\alpha_3=-0.08$; $\beta_0=-12.63$, $\beta_1=0.95$, $\beta_2=0.67$, $\beta_3=-1.18$.”

Comment 7: Quantitative comparison with prior memristor and mixed-signal works

The manuscript now includes a broader survey of related work, which improves context. The comparison remains qualitative and does not allow for rigorous assessment of relative performance. Without normalization, differences in technology node, array size, and precision obscure the true contribution.

Actionable fix: Construct a normalized comparison table including: energy per operation (or per SVD iteration), precision, convergence tolerance, array size, and technology node. Add a “comparison assumptions” column to clarify differences across works. If possible, use the authors’ modelling framework to re-estimate one or two prior works under comparable conditions, enabling a more meaningful comparison.

Response:

Thank you for your suggestions. We respectfully note that the previous-round manuscript already provides quantitative and normalized comparisons in two complementary tables, addressing the core concerns raised here.

First, **Supplementary Table 1** compares MSVD with prior memristor-based computing systems. Because these works target fundamentally different tasks (image processing, graph analytics, neural networks) with distinct system architectures, task-specific operation-level metrics are not applicable for cross-work comparison. We therefore normalize all systems by their energy efficiency relative to GPU baselines on their respective tasks, enabling fair comparison across diverse application domains.

Second, **Supplementary Table 7** provides a rigorous comparison with ASIC implementations of matrix decomposition. In the previous round revision, we reported element normalized speed ($\text{kElem}\cdot\text{s}^{-1}\cdot\text{mm}^{-2}$) and normalized efficiency ($\text{Elem}\cdot\mu\text{J}^{-1}$), enabling direct cross-platform assessment. We now further added the precision normalized comparisons in the revised table.

Regarding “energy per SVD iteration” specifically: to our knowledge, no prior memristor work has demonstrated hardware SVD, making memristor-to-memristor comparison on this metric unviable. The most controlled comparison available is therefore the comparisons of MSVD with and without SREA on the same memristor platform as shown in manuscript, which isolates the algorithmic contribution under identical hardware conditions.

Finally, we note that re-estimating prior works under our framework would require detailed circuit-level parameters and device characteristics that are not available in those publications, making rigorous back-projection infeasible within the scope of this revision.

Revisions:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the **Supplementary Table 7** as **Table R1**.

Comment 8: Novelty claim

The revised positioning is appropriate and technically accurate. No further changes required.

Response:

Thank you for this confirmation.

Comment 9: Broader impact and applicability claims

The manuscript now connects MSVD to applications such as LoRA and general ML pipelines.

These claims implicitly assume that (i) precision requirements, (ii) matrix sizes, and (iii) memory bandwidth constraints in those domains are compatible with the proposed architecture. This is non-trivial—e.g., LoRA often involves higher precision and tight integration with training loops.

Actionable fix: Add a feasibility analysis paragraph specifying: required precision, expected matrix sizes, and bandwidth needs for one representative application (e.g., LoRA). Clearly label such sections as “Projected applicability” and include a short list of technical gaps to be addressed (e.g., training-time updates, gradient support, higher precision accumulation).

Response

Thank you for this comment. We respectfully note that the concern appears to conflate MSVD’s role in LoRA initialization with in-loop training requirements. We clarify this distinction below.

MSVD is used solely for one-time weight initialization under the PiSSA scheme and is not invoked inside the training loop. This distinction is critical. The precision, bandwidth, and integration requirements raised by the reviewer (including gradient support, training-time updates, and high-precision accumulation) apply to in-loop training rather than to one-shot initialization.

Because MSVD only needs to produce a low-rank factorization once, before training begins, the precision and bandwidth constraints are substantially relaxed compared to in-loop training. The argument that “LoRA requires high precision” therefore does not directly transfer to our initialization use case.

The feasibility parameters requested (matrix sizes and precision configurations) are already documented in **Supplementary Table 3** of the manuscript.

Broader questions of memristor-based training (including gradient support, weight updates, and training-time LoRA adaptation) are outside the scope of this work. Prior studies have addressed training on specialized hardware [1], and MSVD is complementary. It provides efficient initialization that those frameworks can directly leverage.

[1] Prabhu, K. et al. MINOTAUR: A Posit-Based 0.42–0.50-TOPS/W Edge Transformer Inference and Training Accelerator. IEEE Journal of Solid-State Circuits 60, 1311-1323 (2025).

Comment 10: Extension to other decompositions (PCA, QR, etc.)

The extension is mentioned but not technically substantiated. Algorithms like QR (e.g., Householder, Givens rotations) and tensor decompositions require different dataflows, orthogonality constraints, and often tighter numerical stability than SVD via power iteration/deflation.

Actionable fix: Provide a mapping table: for each decomposition (PCA, QR), specify (i) required primitive operations, (ii) whether current hardware supports them, and (iii) what modifications are needed. Include a short algorithm-to-hardware mapping example (even pseudocode) showing how one additional decomposition would be implemented on the proposed architecture.

Response

Thank you for this comment. We would like to clarify the scope of this work.

As indicated in the title, this manuscript demonstrates memristor-based singular value decomposition (SVD). The algorithm used is power iteration with deflation, chosen because its core operation, matrix-vector multiplication, maps naturally onto the memristor crossbar architecture.

The manuscript does not discuss or claim support for other matrix decompositions such as QR or tensor factorizations. As the reviewer correctly notes, algorithms like QR (via Householder or Givens rotations) have fundamentally different dataflows, orthogonality constraints, and numerical stability requirements. A rigorous treatment of those decompositions would require separate algorithmic and architectural studies.

The present work focuses specifically on demonstrating efficient SVD computation on memristor arrays and its application across image reconstruction, temporal data analysis, biomedical signal processing and low-rank neural network adaptation. Extensions to other decomposition methods are beyond the scope of this manuscript.

Comment 11: Statistical rigor and robustness

Still insufficiently addressed. Memristor variability (programming noise, drift, cycle-to-cycle variation) can alter convergence trajectories. Reporting only mean values obscures stability and reliability.

Actionable fix: Report mean $\pm$ standard deviation over ≥ 20 runs for key metrics (PSNR, accuracy, iterations). Include seed-controlled repeated experiments to isolate algorithmic vs device-induced variance. Add a noise injection study (e.g., Gaussian or measured device noise) to quantify sensitivity of convergence and final accuracy.

Response:

Thank you for this comment. We respectfully note that the requested analysis is already provided in the previous-round manuscript, and we clarify the relevant details below.

Error bars reflecting run-to-run variation under device noise are included in the key experimental figures (**Figs. 3a, 4f, 5d,e,g, 7c,d**). We have now additionally annotated the corresponding mean $\pm$ standard error values directly on the figures reporting key metrics, in order to remain consistent with the statistical reporting convention used elsewhere in the manuscript while conveying information equivalent to standard deviation (with the number of repetitions n reported in each caption).

Regarding noise sensitivity, **Supplementary Fig. 20** presents a noise injection study quantifying solution accuracy under varying Gaussian noise levels on the IRIS dataset. **Supplementary Fig. 24** demonstrates MSVD stability under different stuck-at-fault rates. These controlled noise injection experiments characterize the sensitivity of convergence and final accuracy to device perturbations.

Our hardware experiments inherently incorporate device non-idealities, including write variation, read fluctuation, and cycle-to-cycle variability. The reported metrics and error bars therefore reflect real device-induced variance under realistic operating conditions.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised **Figs. 4f, 5d** and **Supplementary Figure 14** with added annotations of standard error.

Comment 12: Ablation study generality

Ablation confirms component importance but under limited conditions. The relative benefit of DPM and VPR likely depends on matrix condition number, singular value spectrum, and noise level.

Actionable fix: Extend ablation to at least one ill-conditioned matrix and one high-dimensional synthetic dataset. Report how performance gains vary with condition number and noise amplitude. Add a plot of error vs iteration with/without each component across these conditions.

Response:

Thank you for this suggestion. We respectfully note that the concerns raised are already addressed by the results presented in the manuscript and supplementary materials.

Regarding ill-conditioned matrices, it is standard practice in numerical linear algebra to handle ill-conditioning through preconditioning or spectral shifting, which reformulates an ill-conditioned problem into a well-conditioned one before iterative solution [1]. This is a general and well-established technique that is orthogonal to the MSVD architecture itself and fully compatible with our framework. Any preconditioning step can be applied at the algorithmic level prior to invoking MSVD, so the hardware does not need to solve ill-conditioned systems directly. We therefore do not view ill-conditioning as a distinct architectural concern that requires specialized hardware support.

Regarding high-dimensional inputs, the ablation study in the **Supplementary Information** for the LoRA application already operates on high-dimensional weight matrices (up to 8192×3072) drawn from real large-model layers, directly covering the high-dimensional regime requested.

Regarding the dependence on singular value spectrum and noise level, **Fig. 6a** of the main text reports the number of iterations required for convergence as a function of the singular gap (a spectral quantity closely related to the condition number), both with and without SREA. SREA effectively controls the noise of the iterative process. The results show the expected trend: larger singular gaps lead to faster convergence, and using SREA further reduces the iteration count by suppressing the effective noise level. **Fig. 6a** therefore provides the iteration-count characterization across spectral and noise conditions, with and without key components explicitly contrasted.

[1] Golub, G. H. & Van Loan, C. F. Matrix computations. (JHU press, 2013).

Comment 13: Incremental MSVD scaling behavior

Expanded explanation but still lacks system-level analysis. Incremental updates can incur rank growth, re-orthogonalization costs, and memory overhead. Stability may degrade with many updates.

Actionable fix: Provide complexity analysis: time and memory as a function of number of users/data blocks. Report rank growth behavior and whether truncation is applied. Include a stability experiment showing error vs number of incremental updates (e.g., 2 → 32 users, even if partially simulated).

Response:

Thank you for this comment. We respectfully note that the system-level analysis requested is already provided in the manuscript, and we summarize the relevant locations below.

Quantitative comparisons of speed and energy as a function of user scale are presented in **Supplementary Figure 15**, **Supplementary Note 6**, and **Supplementary Note 8** in the previous-round, where the time and energy costs of incremental MSVD updates are reported and contrasted with digital CMOS baselines.

Regarding rank growth and truncation, our experiments retain fixed ranks throughout all incremental stages, as reflected by the RSV dimensions reported in the manuscript: rank 6 (RSVs of size 500×6) for the EMG-based gesture recognition task (**Fig. 5c-e**) across 2→8 users, and rank 5 (RSVs of size 500×5) for the EEG-based sleep stage detection task (**Fig. 5g**) across 15→75 subjects over four incremental stages. As detailed in these experiments, at each update, the power iteration–deflation computes the low-rank decomposition with the fixed target rank, structurally preventing unbounded rank growth and the associated re-orthogonalization overhead.

Regarding stability over multiple updates, the experimental results demonstrate numerical stability throughout this incremental process: the EMG classification accuracy (**Fig. 5e**) and EEG detection accuracy (**Fig. 5g**) are maintained across successive update stages. While the reviewer suggests testing up to 32 users, our experiments already demonstrate stability across substantial data scaling ($4\times$ for EMG, $5\times$ for EEG), and the fixed-rank formulation ensures that the observed stability properties extend to larger systems.

Comment 14: Temporal dataset difficulty
Justification provided.

Response:

Thank you for this acknowledgment.

Comment 15: Convergence criteria and energy fairness

Iteration counts clarified, but stopping criteria still ambiguous. Energy comparisons are only meaningful if final error tolerances are matched across platforms.

Actionable fix: Explicitly state stopping conditions (e.g., $\|A - USV^T\|/\|A\| < \epsilon$ or singular vector change $< \delta$). Provide a table showing final error vs energy for each platform at matched ϵ . If tolerances differ, include a Pareto curve (energy vs error) to enable fair comparison.

Response:

Thank you for this comment. We have now explicitly stated the convergence criterion and clarified the basis for our energy comparisons.

On the memristor platform, convergence at each iteration is determined by the relative change of the square of temporary singular value estimate between consecutive iterations. Specifically, iteration stops when the relative change of $s_{i,\text{tmp}}^2$ falls below $\delta = 2\%$, with a preset maximum-iteration safeguard. We have updated the manuscript to state this condition explicitly in the **Results** section.

Regarding your suggestion for a matched-tolerance table or Pareto curve, we note that our experimental design intentionally separates accuracy validation from efficiency benchmarking. Solution accuracy comparable to software implementations is demonstrated across all applications (**Figs. 3-5**), while energy and speed advantages are quantified in **Fig. 6**. Critically, even under the most conservative comparison scenario, allowing the GPU to perform only a single iteration (its minimum possible energy of 26.7 μJ), this still exceeds the memristor system's complete execution (13.4 μJ). This fundamental gap indicates non-overlapping operating regimes rather than a continuous tradeoff space, making traditional Pareto analysis uninformative. The reported energy advantage reflects architectural efficiency of compute-in-memory, not an artifact of mismatched stopping criteria.

Revisions:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the “**Performance analysis and benchmarking**” section on page 10 as:
“... The convergence criterion is set to a relative difference of the temporary squared singular value estimates of less than 2% between consecutive iterations ...”
2. We revised the caption of **Supplementary Figure 14** as: “... Convergence is reached when the relative difference of the temporary squared singular value estimates between consecutive iterations falls below 2%; a maximum limit of up to 30 iterations is imposed if convergence is not achieved ...”

Comment 16: Device variability and robustness

Device mismatch and drift can degrade orthogonality and convergence.

Actionable fix: Include either (i) measured variability statistics across devices or (ii) Monte Carlo simulations using realistic device distributions. Show impact on final singular values/vectors and convergence speed. If mitigation is used (e.g., redundancy, recalibration), quantify its benefit and overhead.

Response:

Thank you for this comment. We note that comprehensive device variability characterization and mitigation analysis were already included in our original submission, and we have now emphasized these aspects more prominently in this response.

Measured device variability statistics are reported in **Figs. 2f,g** and **Supplementary Figures 4,5**, where we directly characterize the write-variation and read-fluctuation distributions of our fabricated devices. These measurements establish the realistic operating conditions of our system.

To mitigate these non-idealities, we propose SREA, a technique specifically designed to address device variability and ensure robustness of the iterative solver. Comparisons between MSVD with and without SREA directly demonstrate recovery of convergence and accuracy from degradation induced by device non-idealities. The impact on convergence speed is illustrated in **Fig. 6a** and **Supplementary Figure 14**, which show iteration counts under different noise configurations.

The overhead introduced by SREA in terms of additional area and energy is explicitly quantified in **Supplementary Notes 3,4**. Together, these results demonstrate that MSVD achieves robust performance under realistic device variations, with well-characterized trade-offs between accuracy, speed, and hardware overhead.

Comment 17: Long-term convergence stability

Lacks experimental evidence but if possible add one. Iterative methods may accumulate error over long runs; drift can cause divergence or slow convergence. Actionable fix: Provide a long-horizon experiment (e.g., 500–1000 iterations or repeated SVD runs). Plot error vs iteration over extended runs. Discuss drift compensation strategies (e.g., periodic re-normalization, recalibration).

Response:

Thank you for this comment. We note that long-horizon convergence stability was already characterized in our original submission, and we clarify the relevant evidence here.

On the concern about drift over extended operation: we have explicitly demonstrated device read stability over 1,000 consecutive cycles (**Fig. 2f**), which far exceeds the typical iteration count (~ 30 iterations, **Fig. 6a**, **Supplementary Figure 14**) required for convergence in our experiments. This establishes that the analog substrate remains stable well beyond the iteration horizons encountered in any of our tasks.

Regarding cumulative error accumulation across multiple SVD operations: our multi-stage incremental learning task provides a stress test for this concern. Across successive decompositions in this task, the system performs a total of approximately 400 power iterations while maintaining software-comparable accuracy (**Fig. 5**), demonstrating that iterative errors do not compound over repeated use.

To further ensure robustness in deployment, we impose hard upper bounds on the iteration count (20 or 30 iterations depending on the task, as summarized in **Supplementary Table 3**). These measures, combined with the device stability and convergence characteristics described above, have proven sufficient in our experiments without requiring additional drift compensation mechanisms such as periodic re-normalization or recalibration.

Comment 18: Simulation vs experimental validation (precision scaling)

Precision scaling is central to claimed efficiency; simulation may not capture real quantization and nonlinearity effects.

Actionable fix: Clearly label all plots as “Measured” vs “Simulated” in captions. If possible include a discrepancy analysis between simulation and measurement for a shared configuration.

Response:

Thank you for this comment. We note that the distinction between simulated and measured results was already documented in our original submission, and we have further clarified the labeling in the revision.

The **Methods** section specifies which parts of each experiment are simulated and which are measured from hardware. In the **Supplementary Information**, all but one plots involving simulation are labeled as “Simulated” or “Simulation” in their captions. We have now also updated the captions of **Figs. 5g,7** and **Supplementary Figure 25** in the revised manuscript to explicitly indicate the simulated portions, ensuring consistent labeling throughout the manuscript.

Our simulation framework is parameterized using device-level statistics extracted from real hardware measurements (**Figs. 2f,g**) and the behaviors of peripheral circuits. These statistics characterize the programming error and read fluctuation distributions of the fabricated devices, which directly inform the noise models used in simulation. Simulation is employed primarily for large scale problems (**Figs. 5g,7**) where hardware experiments would be costly, while the core experimental results (**Figs. 3,4,5d,5e,6**) are based on hardware measurements. Because the simulator is calibrated against the same devices used in the hardware experiments, the projections reflect realistic device behavior under the explored precision regimes.

Revision:

To illustrate these points, we have made the following revisions in the manuscript:

1. We revised the captions **Figs. 5g,7** and **Supplementary Figure 25** with explicit “simulation” annotations.

Comment 19: Framing and scope of claims
Limitations acknowledged.

Response:

Thank you for this feedback.

Comment 20: Strength of baseline comparisons

Improved with ASIC comparisons, but still incomplete. Lack of comparison to optimized digital SVD (e.g., FPGA/HPC libraries) limits competitiveness assessment.

Actionable fix: Add at least one strong digital baseline (e.g., FPGA implementation or optimized CPU/GPU library with matched precision/tolerance). If not feasible, include a careful discussion of expected performance relative to such baselines and why direct comparison is challenging.

Response:

Thank you for this comment. We note that digital baseline comparisons were already included in our original submission, and we clarify our benchmarking approach here.

Our comparisons include normalized benchmarks against prior ASIC implementations of decomposition algorithms (SVD and EVD, **Supplementary Table 7**), establishing MSVD's position relative to state-of-the-art custom digital designs. We also benchmarked MSVD against the peak energy efficiency of modern GPUs and TPUs (**Supplementary Notes 6,8, Figures 28,29**).

We chose to benchmark against peak efficiency because it represents the theoretical performance ceiling of these platforms, independent of specific software implementations. This approach provides a hardware-level reference that is both rigorous and conservative, since any practical library implementation will necessarily achieve lower efficiency due to software overhead and algorithmic constraints. Given that MSVD demonstrates advantages relative to this theoretical upper bound, we believe this comparison establishes a strong baseline for assessing competitiveness against digital platforms.

Together with the ASIC comparisons, these benchmarks situate MSVD against both custom digital designs and mainstream HPC platforms.

Comment 21: Reproducibility and accessibility

Evaluation: Code release commitment is positive. Custom hardware limits reproducibility; without models, results cannot be independently verified.

Actionable fix: Release a behavioral simulator or calibrated model (e.g., Python/Matlab) that reproduces key results. Provide example scripts and datasets to replicate figures. Document hardware abstraction (API or interface) so others can map algorithms onto the architecture. If possible, a video of the setup and real-time demo of result would be interesting to contain.

Response:

Thank you for this suggestion. We are committed to supporting reproducibility.

We will release a calibrated behavioral simulator built upon device-level statistics measured from our fabricated memristor hardware (**Figs. 2f,g**). This simulator supports the findings in this study will be deposited in a publicly accessible repository (https://github.com/Dchenc12/memristive_svd) upon publication of the manuscript.

Reviewer #2 (Remarks to the Author):

The authors have substantially overhauled the paper, and I find the revised manuscript much more convincing.

The Figures are greatly improved. Most of the necessary information is now present. However, I still do not think the clarity is ideal, particularly with regard to my concerns about the mapping. The relevant details can be recovered, but the presentation still feels somewhat like a puzzle for the reader.

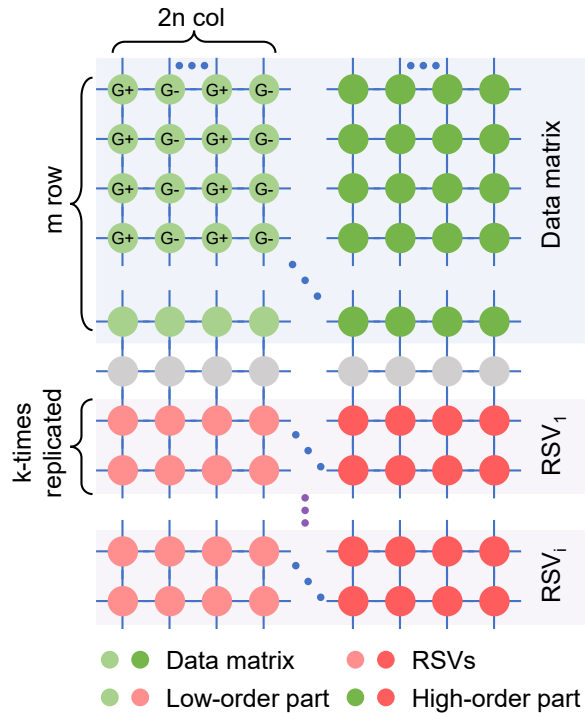
Regarding the energy benchmarks, some aspects could still be debated, but the methodology is now clearly described, and I consider it appropriate for an early-stage study of this type.

Overall, I think the manuscript can be published, although I would recommend further improvements in readability.

Response:

Thank you for the positive overall assessment and for the constructive feedback throughout the review process, which has substantially improved the manuscript. We are pleased that you find the revised manuscript acceptable for publication.

To further enhance the readability of the mapping-related content, we have revised **Supplementary Figure 26** to more clearly illustrate the mapping method. In particular, the i solved RSVs are now depicted separately rather than grouped together.



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.

In addition, we have revised the sentences in **Results** and reorganized “**Data mapping for MSVD and experiment configuration**” and revised its second paragraph to make the mapping procedure easier to follow. The revised text reads: “**Supplementary Figure 26** illustrates how the data matrix and the computed RSVs are mapped onto a single macro for implementing MSVD with SREA. The input data matrix is programmed onto the array at the start of computation, and the solved RSVs are subsequently written onto the array as the decomposition proceeds. The DPM scheme partitions the data matrix and RSVs into two segments, whereas VPR replicates the RSVs several times on the array...”.

We hope these refinements address your concerns about readability and facilitate readers’ understanding of the mapping methodology.

Revisions:

1. We revised **Results** on page 5 as: “DPM partitions the matrices into high-order and low-order components (e.g., mapping an 8-bit matrix onto a low-resolution array storing the 2 high-order bits and a high-resolution array storing the 6 low-order bits).”; “To counter memristor variations, each solved RSV is replicated several times within the crossbar array, with the replicas’ outputs summed at the output line and then averaged. Together, VPR effectively reduces the residual components.”
2. We revised **Supplementary Figure 26** in **Supplementary Information**.
3. We revised the paragraph in section “**Data mapping for MSVD and experiment configuration**” on page 17.

Reviewer #3 (Remarks to the Author):

(Please see my original review for general comments on the significance and merits of this paper.)

Comment #1

This second draft neatly addresses the concerns raised in our original review, particularly the additions to the supplementary material, which brought in some great clarity. I also enjoyed reading the responses to the other reviewers. Thank you.

The supplementary notes 1 and 2 are particularly helpful in answering many of my previous technical comments, as are the flowcharts, and the SREA mapping diagram in Supplementary Figure 25.

Response:

Thank you for the positive feedback and for the thoughtful and constructive comments provided throughout the review process. We are delighted that you find our revisions effectively addressed the previous concerns. We have also carefully addressed the remaining comments point by point in the following responses.

Comment #2

Regarding the error analysis in supplementary note 2, it is fairly unsatisfying that the bounds for the limits of spectral resolution are expressed in terms of $O(\text{constant})$. Without knowing the constant of proportionality for this big-O expression, the inequalities are only useful as they provide roughly guidelines or heuristics. For instance one might ask whether a hypothetical user should believe a particular singular value, and as such the answer is unclear. If there is anything the authors can do put in actual numerical limits (perhaps within some probability?) the extra numerical rigor would make this result more practical. However, I do not feel this is a pre-requisite publication.

Response:

Thank you for this valuable suggestion. We agree that providing more concrete numerical bounds would enhance the practical utility of the analysis.

To address this, we establish quantitative thresholds through simulation-based analysis of the two representative tasks discussed in the revised **Supplementary Information**. One task favors relatively accurate low-rank approximation (COVID-19 task) while the other is more error-tolerant for feature extraction (EMG task).

Following the derivation in **Supplementary Note 2**, a practical guideline for considering a resolved singular value to be reliably distinguishable from noise is based on constraining the perturbation angle. To ensure that the angle θ between the computed and ground-truth singular vectors satisfies $\sin(\theta) < \sin(\theta')$ for a target accuracy $\theta' \in (0^\circ, 90^\circ)$, the normalized spectral gap must satisfy

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

where s_i denotes the i -th singular value of the data matrix X , T denotes the error matrix, and $\|\cdot\|_2$ represents the spectral norm of the matrix. The magnitude of $\|T\|_2/\|X\|_2$ depends on both the hardware noise characteristics and the specific properties of the data matrix being analyzed. Although the spectral norm $\|X\|_2$ of the data matrix can be readily obtained for a given problem, the spectral norm of the error matrix $\|T\|_2$ varies throughout the actual computation and therefore cannot be assigned a fixed value. We thus estimate the ratio $\|T\|_2/\|X\|_2$ by averaging over multiple simulations to provide a representative reference. We perform ten independent simulations under the configuration reported in **Supplementary Table 3**, with equivalent noise of $\sigma_r = 0.3 \mu\text{S}$, and average the results to obtain $\|T\|_2/\|X\|_2$.

For the COVID-19 low-rank approximation task requiring relatively high accuracy, the simulations yield $\|T\|_2/\|X\|_2 \approx 0.0065 \pm 0.0002$. At the truncation point adopted in our experiments, the normalized spectral gap is 0.096, approximately 14 times larger than this relative error, providing a sufficient margin for reliable resolution. If constraining the deflection angle between the last resolved singular vector and its ground truth to within 6° (corresponding to a cosine similarity of 0.995), the recommended threshold is $0.0065/\sin(6^\circ) \approx 0.062$. In practical use, singular values with normalized gaps exceeding approximately 0.07 can be considered reliably distinguishable from noise.

For the EMG feature extraction task with less stringent accuracy requirements, the simulations yield $\|T\|_2/\|X\|_2 \approx 0.00215 \pm 0.00005$. The normalized spectral gap at the truncation point (0.0085) is approximately 4 times larger than this relative error. If allowing the deflection angle to be within 15° (corresponding to a cosine similarity of 0.966), the threshold is $0.00215/\sin(15^\circ) \approx 0.0083$, suggesting that singular values with normalized gaps exceeding this value can serve as a rough indicator of reliable resolution.

In summary, for a more quantitative criterion, the ratio $\|T\|_2/\|X\|_2$ 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 θ' , the corresponding normalized spectral gap threshold is given by $(\|T\|_2/\|X\|_2)/\sin(\theta')$, and singular values whose normalized spectral gaps comfortably exceed this threshold can be taken as a rough criterion for reliable resolution.

Revision:

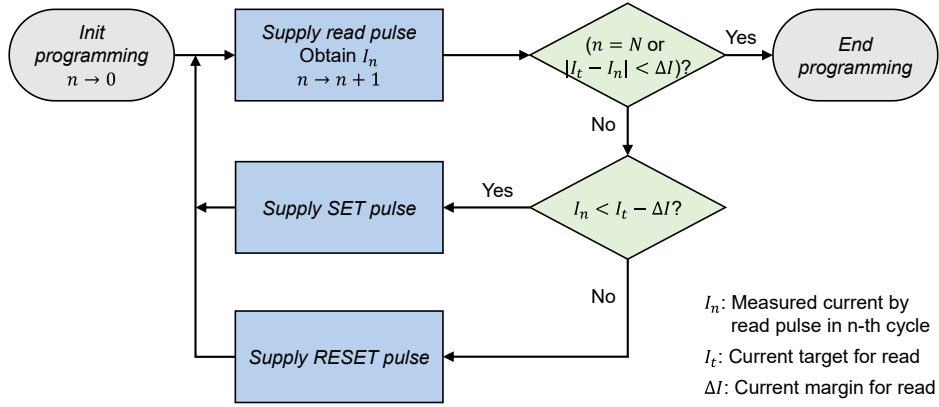
1. In **Supplementary Note 2**, we added: “For a sharper quantitative bound, if the perturbation angle is required to be within a target value θ' ... We thus estimate the ratio $\|T\|_2/\|X\|_2$ by averaging over multiple simulations to provide a representative reference.”; “Based on ten repeated simulations, we estimate that under the experimental configuration... that the normalized spectral gap to be resolved may need to exceed $(\|T\|_2/\|X\|_2)/\sin(6^\circ) = 0.062$.”, “... for EMG feature-extractor, simulation results $\|T\|_2/\|X\|_2$ under a memristor noise of $0.3 \mu\text{S}$ yield approximately 0.0085 vs. $0.00215 \pm 2 \times 10^{-5}$... which may serve as a rough reference for identifying reliably resolved components.”; “In summary, for a more quantitative criterion, ... can be taken as a rough criterion for reliable resolution.”.

Comment #3

As an editorial note (to both the authors and editors), there is at least one typo in a figure (in supplementary Figure 22 the top blue box in this flowchart reads “Suppling read pulse”, which is clearly a typo.) These kinds of errors are easy to overlook when relying on spell check.

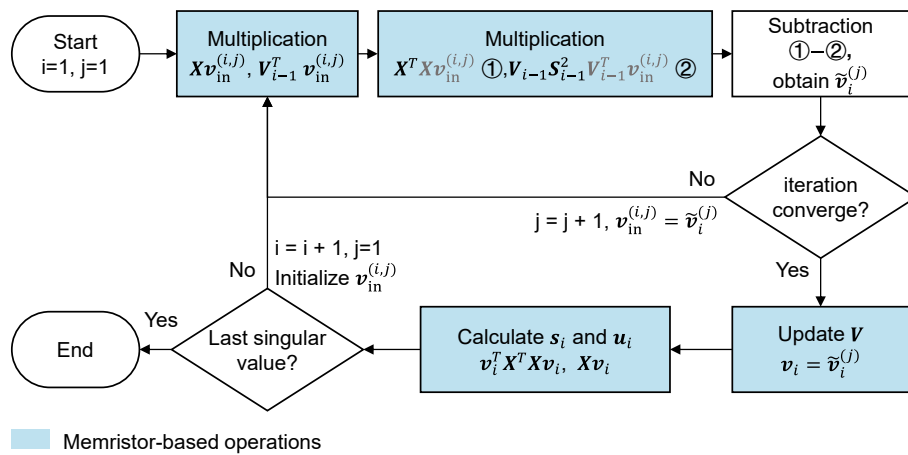
Response:

Thank you for pointing out this typo. We have corrected it in **Supplementary Figure 23** (original **Supplementary Figure 22**), and the updated figure is shown below:



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

In addition, we conducted a proofreading of figures and text, and identified another typo in **Supplementary Figure 19** (“Substruction” → “Subtraction”), which has also been corrected, as shown below:



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.

Additionally, we have also corrected several typos identified in the text during this proofreading process.

Revisions:

1. We revised the **Supplementary Figure 23** in **Supplementary Information**.
2. We revised the **Supplementary Figure 19** in **Supplementary Information**.
3. We revised several typos in the text.

General Comments:

This paper aims to perform Singular-Value Decomposition (SVD) using memristive arrays, hence their proposed method: Memristive SVD, or MSVD. The authors make use of the iterative power method for extracting the largest singular value of the matrix (along with the corresponding left and right eigenvectors). These modes can be projected (or “deflated”) out of the original matrix and the process can then be repeated to extract successive singular value modes. This algorithm is well-established, used for many applications, and does appear well-suited for application in this setting. The authors also introduce a pair of innovations to deal with errors which they refer to collectively as selective representation enhancement architecture, or SREA.

This application is important, yet the paper itself needs further revision before the reader can truly assess the value of the techniques described here. As a reviewer I was tremendously enthusiastic about this paper, as SVD is a tremendously important tool for machine learning and data analytics in general. So, I was intrigued to learn about the author's methods and learn more about the possibilities they present. I was even more interested to assess the limitations of the approach here. How broadly applicable is it? This reviewer is truly curious! Unfortunately, there are many places where I and my co-reviewer were left unable to deeply explore our questions about the authors' methods, because the descriptions of the methods lack sufficient detail to really quantify the strengths or limitations of their techniques. In general, the paper could benefit greatly from some reorganization for readability. It also lacks sufficient detail as to how the techniques are employed. Finally, the paper shares the limitations of many in-memory compute papers in that the comparison to the state of the art is presented in a very one-sided fashion, yet the paper does not provide enough detail to allow the astute reader to check their claims.

The review here is therefore by necessity incomplete. An updated revision would be required to clarify the methods here, and then a full review could be made to dig into the finer points of the paper. We have made an effort however to point out as much as we can see.

Minor Comments:

1. In Supplementary Material, the equation on line 132 is incomplete, could just be left as S^2 .
2. The insets in Figure 4d,e are not very clear – at least not without zooming in a lot. Could the figures be adjusted so that they (along with the text) are bigger?
3. It would be quite helpful for the community interested in this work if the authors could release code for the main MSVD algorithm with SREA for one or few of the datasets presented in this paper. The hardware platform could be replaced with arbitrary devices following some normal distribution, perhaps one calibrated to Supplementary Figure 4c.
4. In Figure 2f, 2g, how were the 100 devices chosen? For 2f, could the marker size be decreased? In its current state, it is difficult to see minor fluctuations clearly.

Major Comments:

1. *General Readability:* This paper satisfies the detailed guidelines for the structure of Nature Communications papers with Results and Methods sections. However, in the case of this paper these guidelines have led to a paper that is very hard to process. One simple solution would be to define terms clearly as soon as they are introduced. For instance, the terms “dual-precision mapping” and “vector processing refinement” are used without definition in the Results section or reference to the Methods section. Also results for PSNR and SSIM are unfortunately introduced before the author's definition is given in methods.
 - a. *Recommendation 1:* Introduce the full terms and acronyms in the Results section “Dual-Precision Mapping (DPM)” “Vector Processing Refinement (VPM)”, and refer the reader to the Methods section the first time these are introduced.
 - b. *Recommendation 2:* Write out the full names of the PSNR and SSIM metrics when they are first introduced and refer the reader to the methods section (as is commonly done in many of the references for this paper).

More broadly, the authors are strongly encouraged to look closely at the manuscript for linear presentation of their material. As it stands the manuscript requires several reads of the Results, Methods, and Supplemental sections before a coherent picture of the work becomes clear. Again, this is largely an artifact of the format for this journal, but it is particularly difficult with this paper, which has several unique definitions, and no fewer than three example applications.

2. *Presentation of the Iterative Power Method.* There are two sections to describe the iterative power method, one in the main Methods section (starting in line 290) and another in Supplementary Note 1. It is the opinion of this reviewer that the interpretation of the Methods treatment does not stand well on its own, as the text does not give the reader a good sense of the method. These paragraphs in the supplemental section could be potentially replaced with Supplemental note 1 for more clarity (though Methods section Equation (3) would need to be retained to complete the discussion). These two sections could be combined for a net saving of space and clearer presentation, if the word- and line-limits allow.
3. *Presentation on the use of the arrays in the iterative power method.* The use of the array for computing the singular vectors needs to be fleshed out more as this is given only light treatment in both the main paper and the supplementals, and understanding these details is of critical importance for the careful reading of this paper. However, almost all of these operational details are left uncertain. Here is what the reader can gather (but is not made explicit)
 - a. The Array is used to calculate iterative powers of $A=X^T X$ (from supplemental note 1), and
 - b. The Array has 32 fixed conductance states (from supplemental figure 4)

This implies that there are peripheral circuits that operate on a voltage drive, and that the matrix X is mapped to conductance from its original units, and the random input vectors are mapped to units of voltage at the start of the iterative power algorithm. Then a scaling factor, call it " R_{scale} " (as it has units of resistance), is applied to the output currents before it is applied back to the inputs of X (presumably in the other direction to realize the necessary transpose). How is this scaling factor chosen? Is it fixed throughout the algorithm, or is it adaptively chosen to maximize signal to noise? If it is adaptively chosen, is the power for the of this additional processing included in the power budget? The pipeline for actually performing these operations should be added as a Supplemental note. Several of the following comments will return to this issue of definition several times.

4. *General comments on SREA (DPM and VPR):* DPM and VPR could both be referred to as varieties of bit-slicing as defined by Le Gallo et al. "Precision of bit slicing with in-memory computing based on analog phase-change memory crossbars" *Neuromorphic Computing and Engineering*, Volume 2, No. 1 (2022). In light of this paper, DPM could be referred to as Positional Bit-Slicing, while VPR can be referred to as Equal-Fill Bit-Slicing. If the authors agree that DPM and VPR are indeed reflective of this analysis, they should make use of this reference and the common terms used therein. Doing so will not only connect the author's work to the common nomenclature for these techniques it could also potentially save critical word-count and give the authors the opportunity to leverage the error propagation analyses in these works, which actually give this reviewer some concern about the broader benefit of these techniques, as I point out in some of the following comments. If there are distinctions to be made between the bit-slicing approaches used here and one's used in the Le Gallo (2022) analysis, then a note of how the author's new bit-slicing approaches are unique would be a useful enhancement to the present paper.
5. *Limits of DPM?:* This method is described in line 313, and illustrated in Figure 2d. The concept is to use two arrays to enhance the precision of values of X . It is a form of Positional Bit slicing (to use the nomenclature from Le Gallo et al (2022), see Comment 4). However, it is unclear how this improves precision in practice if the arrays used to store the high-order and low-order arrays have similar noise. If the two arrays have similar noise magnitude of noise, then the noise in the high-order array will get amplified in equation 7 along with the high-order signal when the high-order and low-order signals are merged. This could easily result in noise that drowns out the extra precision from the low-order parts (an issue that the authors do not address). The resolution to this problem is not clear at all from the text, and frankly it puts some doubt as to the utility of this technique here. However, the author's data suggests that it is still useful, but how then does it actually

work? Please include a formal error propagation analysis as a supplemental to explain to the reader why noise in the high-order array is not detrimental to DPM.

6. *Implementation details for VPR*: The definition of VPR is not very clear, the use of equal-fill bit-slicing (Comment 4) is appropriate if these vectors are stored as analog quantities. However the author then makes the statement that for VPR, the RSV's are "typically replicated 4-times with 6-additional input bits" (line 348). What is the significance of "6-additional input bits"? Additional with respect to what? A close reading of this sentence indicates that the *inputs* to the V array are digitally encoded, even though the V-array itself is an analog vector, but the phrasing here is nonetheless still confusing to the reader.
7. *Deflation*: How does the deflation operation work in MSVD? Where are the LSVs (U-vectors) stored? Fundamentally: How does storing analog values for V save energy? This step is not clear from the current text. In line 146 of the supplemental, we see that X_{defl} is calculated from previously calculated LSV's and RSV's. However, it is not clear from the text what role the V-array plays in the calculation of X_{defl} . What vector is passed through the V array? Where does it come from? Are the U vectors explicitly stored elsewhere, or recomputed from the V vector somehow?
8. *Minimum detectable singular vectors*: The authors give a lot of examples in the text of applications where the methods here have shown reasonable fidelity, over a wide range of singular values. However, it seems that there is something being left implicit about the power iteration scheme in Supplemental Note 1, Notably the choice of R_{scale} (Mentioned in Comment 3, left implicit in the text). For instance, looking at equation 3 if after deflation the value of the largest remaining eigenvector is less than 1. Note that we can consider X to be dimensionless if we take into account the factor R_{scale} . Given that the details behind the choice of R_{scale} are omitted from the current manuscript, it's hard to say how noise and errors propagate from iteration to iteration. Once the selection criteria for R_{scale} is defined, one can do another error-propagation analysis to potentially put a lower bound on the minimum detectable singular values, given typical amplitudes for the cell noise, relative to the maximum weight value in DPM. This error and noise analysis, which is strongly related to the error analysis requested in Comment 5, along with the noise figures for the chip at hand, would really enhance the theoretical foundations for this technique.
9. *Use of the terms Accuracy vs. Precision*: Figure 3a indicates the use standard errors (standard deviation of the experiments run), and 3b and 3c then compares distribution of

these errors. The tightening of the standard error (which reflects repeatability of successive experiments) should be referred to as increases in the “precision” of the system. However, in line 145, these are listed as improvements in “accuracy”. This should either be listed in the text as improvements to precision, or what would be more interesting: Figures 3b and 3c should be redone to reflect *actual* improvements in *accuracy* (i.e. difference from ground truth). The distinction between accuracy and precision is important and the authors should review the document for any repeats of such typos.

10. *Benchmarks of energy efficiency and normalized speed:* A major promise of neuromorphic compute is an increase in the power performance relative to the conventional state of the art. This paper claims a 50x power advantage for MSVD w/SREA over an A100 GPA and a 37x advantage in computing speed. Though these numbers are both potentially underselling power of the GPUs. Looking at supplemental reference 17, it would appear that the authors are taking the 78 Peak TFLOPs for non-tensor FP16 operations and dividing this by the 400W consumed by an A100. This seems reasonable, were it not for the fact that vector operations such as SVD are prime candidates for tensor memory processing. If SVD uses mostly tensor cores then the energy efficiency goes up 4x to 312 TFLOPs, which is a significant decrease bringing the energy efficiency boost down to approximately 13x, which is still high. If the authors strongly believe that the Tensor operations are not employed here, then some argument is in order. For instance, the strongest argument would come from profiling of the actual SVD tasks as performed on an A100. Meanwhile, in terms of area efficiency, the authors are considering the entire area of the GPU die rather than the active CMOS area occupied by the FP16 cores used in this. Is the MSVD area quoted also a whole chip estimate? Please confirm that the area figure of merit for “required area” for MSVD and the A100 is the same.
11. *Micrographs:* Please add dimensions to the chip micrographs in Supplementary Figure 3. The "area required for MSVD implementation" of this chip is cited in the text as 0.199 to 0.243 mm² which seems much smaller than this micrograph. Looking at the number of bond pads and assuming a 70um pad size the chip appears to be about approximately 2.5mm on a side(?)
12. *Completeness of noise models and actual chip power in Supplementary Note 3:* It is clear that comparing the power of an A100 GPU to a chip fabricated in a 130nm process is not a fair comparison, and so some scaling is in order. Though why do the authors choose 32nm to model the chip power, are there any tools available to you which can give estimates of power in 7nm, the same process as the A100? Also, it would be reassuring to have a complete power model at both 130nm and 32nm, and to also cite experimental power

measurements for the 130nm device as operating in the lab. Completeness of power modelling is sometimes challenging to get right, and so having this comparison between the actual chip and the expected power 130nm consumption would be a strong confirmation of the completeness of the power model.

13. *Definition of “Global” vs. “Individual” landscape views:* The convergence plots in Figure 6 are fascinating though they could use some more explanation. Firstly, the role of stochastic error in these figures is hard to assess. In panel 6a, the impact of DPM and VPR are made clear by presenting differences in the actual trajectory tracked through the loss what appears to be loss trajectories for the MSVD landscape. However in 6b, the reconstruction errors themselves have different shapes to reflect the different loss trajectories (if it helps to explain these, there are a number of works on loss trajectories such as <https://arxiv.org/pdf/1712.09913>), however it is not clear to this reviewer how the presence of stochastic noise (and the noise management approaches in SREA) lead to different shapes for the loss surfaces in Figure 6b. Could you please explain?
14. *Cell switching characteristics:* In Supplementary Figure 4, cell switching characteristics are shown.
 - a. *IV curves:* How are these measured? Why has a single device been chosen? Ideally, the authors should report IV curves across multiple devices to show the overall distribution of the array (keeping the measurement methodology fixed).
 - b. *Conductance histograms:* These should be further explained. How are these extracted? What is the write and read methodology? What portion of devices in the array are being used for these results (for example, are different portions of the array being tuned to different target conductance states and then read back)? Is there any filtering being done (for example, were there any shorts or stuck devices that had to be avoided?). I understand that this is an algorithmic demonstration, but explaining device properties across the array would paint a better picture of what would be needed at a device-level for recreating the algorithm’s performance on the same benchmarks in some other prototyping system.
15. *Hard Faults:* In Figure 2 and surrounding text, the authors discuss how component residuals are a problem caused by device-level conductance mapping deviations. However, the deviations being reported/observed do not include hard faults (such as stuck or broken devices), which are typically a significant problem with any computation involving memristors. It would be great if the authors could add additional commentary on this. Think about questions along these lines:
 - a. Were there no such devices in the fabricated array?
 - i. If yes, could some fabrication-level details be added?
 - ii. If no, was there explicit consideration or enforcement done to avoid such devices?

- b. How does MSVD's performance change if there were hard faults participating? Is SREA more tolerant to them compared to w/o SREA?
16. *Correlations*: In Figure 4f related text in the results section, the overall methodology for computing correlations is unclear. It would be helpful to know how these are being computed. Precisely which vectors are being compared to obtain these correlation values? If the analysis here is specific to this data set, please add a footnote here to the appropriate citations. Additionally, it would also help if the authors could comment on why SREA produces more correlated decompositions against the software SVD baseline compared to the non-SREA version (just to bring everything together).
17. *Incremental MSVD*: For incremental MSVD (specifically for results in Figure 5), I have a few questions. The manuscript would likely benefit from an added discussion on this aspect/clarification of Supplementary Figure 7 in the main text.
 - a. Going from the 0-update model with 2 users to the 1-update model with 4 users, I assume the 0-update model was used as a baseline and allowed to update based on Equations 11 and 12. If this is the case, then how is information from 2 additional users incorporated when only a single update is allowed? If this is not the case, then what is?
 - b. I am unclear on whether the linear regression classifier must also change with increasing # of allowed updates.
 - c. What are the sources of errors for Figure 5e experiments? Have the authors looked at source-wise contributions i.e. how much of it is noise from devices or the system vs. the algorithms? Do the 15 repeats use the same sets of devices?
18. *Dimensions of singular values & gaps*: In figure 6a, the authors present comparisons of the singular values across the three applications specified. However, it is important to note that in most cases the datasets under study have dimensioned data, and comparisons of magnitudes of the singular values (and hence a comparison of the gaps) are meaningless unless the datasets are first normalized to dimensionless quantities. For example, the famous Iris data set consists of linear spatial dimensions (petal and sepal lengths and widths, etc.). If the original Iris data set is expressed in units of mm, the singular values, s^2 , could change by a factor of 100 or even 1,000,000 if the units of this data set are converted to cm or m. Such a change of units would not change the spectral structure of the data, though it does make comparisons between disparate datasets problematic. This uncertainty of normalization when doing comparisons should be eliminated by expressing all gaps normalized to the first singular value for the same dataset. (i.e. each gap should be divided by the corresponding value of s_1^2). This also gives a more honest assessment of the capability of the MSVD, because in the end each dataset has to be renormalized to units of conductivity before they can be mapped into the X array, and so a normalization process is already *implicitly* done as part of the conductivity mapping.