nature portfolio

Peer Review File

In-memory analog computing for non-negative matrix factorization

Corresponding Author: Professor Zhong Sun

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

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The paper "In-memory analog computing for non-negative matrix factorization" by Shiqing Wang and collaborators presents a novel hardware solution that is able to perform a complicated linear algebra decomposition technique that has multiple industrial applications. Their implementation relies on a new circuit that enables the calculation of the generalized inverse of a linear system, which is implemented in a way that requires significantly less energy, and which enables a way to perform non-negative matrix factorization (NMF) using memristor-based analog in-memory computing hardware. The authors showcase impressive results on image compression and recommender systems that show several orders of magnitude improvement on runtime and energy efficiency over GPU and FPGA-based implementations.

While the paper is of high quality and the results shown on the manuscript are very impressive, there are a number of aspects that we think could be improved:
- Line 36: The authors point that digital hardware struggles to solve NMF with large matrices. It would be useful to clarify what they mean by large matrices and justify how and when RRAM-based hardware will be able to tackle matrices of the size used in industry-relevant applications. It wasn't clear to me whether decomposition techniques will be enough to be enable practical work on industrial applications.
- Line 69: One of the advantages of RRAM-based hardware is that the logic is very dense, resulting in small circuits, which potentially could enable massive parallelization. Some discussion on how this could affect NMF applications would likely be useful
- Line 106: Analog hardware has significant promise, but also some drawbacks when compared with digital hardware, such as limited precision. The manuscript lacks sufficient details on the precision limitations on the hardware, and its impact on the two applications being evaluated.
- Lines 106-107 claim that "the ANLS algorithm’s alternating optimization framework and non-negativity constraints inherently suppresses error propagation". This is cited without references and should be justified in more detail.
- Line 166: the adaptive accuracy mapping method that was used on the write-verify method should have a link to the supplementary material. Details on its contribution to latency or energy costs would be welcome.
- Line 204: the ANLS algorithm is shown to converge extremely quickly for the problem sizes evaluated on the hardware. This is very impressive, but it would be useful to show whether this is consistent across multiple problem/matrix sizes.
- Line 188: Identical should be identity instead
- In the abstract and introduction the authors mention that the NMF algorithm has multiple industrial applications "...recommender systems, bioinformatics, and image processing". They then proceed to compare the AMC hardware on two applications with implementations on GPU and FPGA, showing very significant improvements. However, those two chosen applications (image compression and recommender systems) have multiple implementations, and it is unclear from the paper whether the GPU and FPGA points of comparison represent the state of the art. It would be useful to establish what is the state of the art on these applications, and compare the AMC hardware with those. In the case of recommender systems, given that machine learning techniques are most often used to train recommender systems, it would be useful to also compare the quality of the recommender system.
- I found the comparison on latency and energy with the GPU implementation a bit strange. The GPU used in the comparison was relatively old (Titan RTX from 2018), a more modern system would have been preferable. More importantly, for speed purposes the GPU was run on a single core (out of 4608), but for energy purposes the full energy consumption of the GPU was taken into account. It would be very useful to see a comparison on the implementation using the full GPU resources, and comparing it with a simulation of AMC hardware being run on a parallel fabric.

Reviewer #2

(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.

Major comment:

Line 151: The sentence, “convenient to introduce various constraints in the target cost function,” highlights what I believe is the most novel contribution of the paper—extending prior work on linear regression in hardware (Refs. 19/20). However, the manuscript lacks sufficient detail to understand how this constraint integration is actually realized in hardware. Additional explanation would strengthen this part significantly.

Minor comments:

Line 52: I suggest using "a system of linear equations" rather than "matrix equations".

Line 53: I suggest adding the phrase "without having to compute the inverse of the data matrix" to clarify the computational advantage being described.

Line 95: The technical term L2 constraint appears without a formal definition.

Line 99: The symbols Θ and Ψ are not constraints themselves. It would be more accurate to refer to them as penalty functions, which is the commonly used term in such contexts. Correspondingly, α and β are more appropriately described as penalty parameters or regularization parameters.

Line 103: The phrase regularity of matrix computing is unclear. Please consider rephrasing or elaborating on what is meant here.

Line 109: While the notation $||X||_F$ unambiguously refers to the Frobenius norm, $||X||_2$ is often interpreted as the spectral norm (i.e., the largest singular value of X). To avoid confusion, I recommend consistently using $||X||_F$ unless the spectral norm is specifically intended.

Line 117: The use of the superscript star in Ω^* is unclear. Please clarify the meaning of this notation.

Line 124: Consider removing the word inner from the text.

Line 127: This section focuses on a vector (p-by-1 variable) rather than a matrix variable. While this is understandable due to the problem’s separability, it would be helpful to explicitly mention this and briefly explain why it suffices to consider a vector formulation.

Reviewer #3

(Remarks to the Author)
The paper presents an analog computing framework for NMF, based on a compact and reconfigurable GINV circuit. The proposed design aims to reduce the number of OPAs and supports various constrained regression formulations such as NRR and NCRR. Experimental validations on image compression and recommender systems highlight the potential of the approach in terms of energy efficiency and convergence speed.
However, several points require further clarification or quantitative analysis to substantiate the claimed advantages and practical trade-offs of the proposed design:
1.The main novelty of the paper appears to be the reduction in the number of OPAs and the ability to support both NRR and NCRR using a compact GINV circuit. However, since the performance benchmarking does not include prior analog NMF solvers, the specific advantages of the proposed circuit are not clearly demonstrated. How does this design compare—specifically in terms of power, latency, and performance—to a conventional GINV circuit implementation?
2.According to the energy breakdown in Fig. S12, the majority of energy consumption arises from DACs and ADCs, not OPAs. In that case, is the reduction in OPA count from p+q to p truly impactful at the system level?
3.The significantly higher conductance range of g_c compared to G_left and G_right, as shown in Fig. 5c and Fig. S7, raises concerns regarding potential increases in programming latency and energy consumption. It would be helpful if the authors could quantitatively evaluate this overhead in comparison with traditional GINV implementations.
4.How do retention and temperature variation of RRAM devices affect the stability or performance of the NMF process? Can their impact be considered negligible in practical scenarios?
5.As shown in Fig. S3, some devices fail to reach the target conductance even after the large number of programming cycles. Does this result in any performance degradation in the NMF process, and if so, how is this issue addressed or mitigated at the system level?

Reviewer #4

(Remarks to the Author)
This paper proposes a memory-efficient computational scheme for accelerating non-negative matrix factorization (NMF). The authors designed a novel, compact, and reconfigurable generalized inverse (GINV) circuit and combined it with the alternating non-negative least squares (ANLS) algorithm, successfully demonstrating its application in image compression and recommendation systems on hardware. The experimental results, particularly the orders-of-magnitude improvements in speed and energy efficiency compared to FPGA and GPU shows its potential in real life applications. However, in order to further enhance the rigor and persuasiveness of the manuscript, there are several key issues that need to be clarified and discussed in greater depth by the authors. Detailed comments are listed below.

1. The paper mentions that the ANLS algorithm has error tolerance, which is cited as a key factor supporting the feasibility of the analog computing scheme. However, the discussion in the paper is rather general, and this claim requires more quantitative analysis to support it. For example, how do device-to-device variations in resistive memory arrays, limited programming precision, and the non-ideal characteristics of operational amplifiers (such as limited gain, bandwidth, and noise) specifically affect the convergence process and final accuracy of NMF? It is recommended that the authors supplement the relevant data, such as injecting different levels of noise into the analog calculations and observing the changes in the NMSE or PSNR of the final decomposition results, to clarify the robust boundaries of this scheme.
2. On the rigor and cost of the block matrix-based method: To process large-scale datasets (such as MovieLens 100k) on limited hardware, the authors propose a “block matrix- based method.” This might be a practical engineering solution, but its theoretical rigor requires further clarification. The description in the paper, particularly the steps involving “maximization and scaling” to merge intermediate results, appears to have a heuristic nature.
Question 1: To what extent does this approximate processing method alter the convergence properties of the original ANLS algorithm?
Question 2: How sensitive is the accuracy of the final results to the choice of block size d?
Question 3: The scaling factor is described as “empirically chosen to be 1.15,” which undermines the method's universality. The authors should provide a more robust justification for selecting this value or discuss an adaptive method for determining it.
3. The Gap Between Experimental Scale and Performance Extrapolation: The hardware experiments in this paper were conducted on a very small scale (e.g., a 4x5 RRAM array). The remarkable performance data presented in the paper (e.g., hundreds of times faster than GPUs) is based on theoretical extrapolation and estimation from these small-scale experimental results. However, there is typically a significant gap between performance predictions from small-scale physical demonstrations and large-scale applications. The authors should discuss the new challenges that may arise when scaling up, such as long-line delays, power consumption distribution in larger arrays, and the complexity of controlling thousands of ADCs/DACs. While an estimation model is provided in Supplementary Note S4, discussing these potential challenges would make the performance predictions more persuasive.
4. The paper mentions that the compensation conductance is “pre-computed with minimal computational overhead.” However, in each iteration of ANLS, the matrices U and V are constantly updated, which means that CC also needs to be frequently re-computed and re-programmed. Is this overhead still “minimal” when dealing with large-scale, rapidly changing matrices? It is recommended that the authors quantify the computational and time overhead of this part and include it in the overall performance evaluation.
5. Supplementary Figure S12 shows that the energy consumption of the ADC/DAC accounts for 98% of the total energy consumption, which is a significant bottleneck. The authors propose in the discussion section that using a low-precision DAC is feasible. This is a good point, but it lacks data support. It is recommended to supplement the relevant data to demonstrate the impact of using ADC/DACs with different bit widths (e.g., 3-bit, 4-bit) on the final NMF decomposition accuracy, which would strongly support their argument regarding energy efficiency optimization.
6. In the performance comparison with GPUs, the authors normalized GPU performance to a “single core.” Although the authors provided justification for this, it may be controversial because matrix decomposition algorithms running on modern GPUs are highly parallelized and optimized to leverage the collaborative work of thousands of cores. If possible, the performance of the entire simulation system should be compared with the most advanced NMF algorithm implementation running on the entire GPU, or the limitations of this comparison method should be more clearly articulated in the discussion. Similar issues arise with FPGA comparisons, such as whether the comparison is against the best available FPGA, which can significantly impact the comparison results.

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The authors addressed most of my questions and comments.
However, there are some points that would benefit from further clarification.

1)
I am still unable to follow how the nonnegativity constraints are involved in solving

min_{v>=0} ||r-U*v||_2^2 (S5).
From the supplementary material, it is stated that "The solution v for the GINV is identical to the solution of Equation (S5), which represents the circuit equation of the compact-GINV circuit in this configuration. Notably, the non-negative constraint is enforced through the use of a single supply source, eliminating the possibility of negative outputs".
However, the vector v obtained from (S4) is the standard linear least-squares solution without any nonnegativity constraint. As such, the vector v from the formula (S4) cannot engage the nonnegativity constraint from (S5) on its own. Hence the claim "The solution v for the GINV is identical to the solution of Equation (S5)" appears to be misleading. Are there any structural properties that guarantee the nonnegativity of v?

2)
In Fig2a, the unknowns U, V are given as fixed matrices at each step of obtaining U_i, V_i. At each step, either U_i or V_i need to be updated and `reprogrammed' in the hardware.
Has the overhead for performing this every iteration been analyzed relative to the total runtime (either in wall-clock time or in programming cycles, similar to the pie chart as shown in Fig. S20) with respect to different instances presented in Fig. S7? If such an analysis is included in the manuscript, could you please point indicate where it is presented?
Additionally, is the programming cycle axis in Fig S4 correspond to the same unit in Fig S7?

3)
How many cycles were required in Fig. S20? In other words, when converted to cycles, how many were needed for each portion of the runtime?
Additionally, how many alternating directions did the ANLS algorithm require to solve the large instance?

Minor comments.

1) Related to (S4), the inverse of (U^TU) may not exist, if U is rank deficient.

2) For the Frobenius norms, only using ‘F’ is sufficient; the usage of 'F=2' seems unnecessary.

3) While I agree that the use of “matrix equation” aligns with conventions in this field and covers a broader spectrum, the term itself does not explicitly convey linearity, nor does it indicate that multiple equations are involved, as targeted by this paper. Its usage remains somewhat ambiguous and should be clearly defined upon first introduction.

4) It would be helpful to specify the instances used to generate the histogram presented in Fig. S7.

5) In Fig. S20, it is unclear what all legend items refer to, particularly what “Computing” represents. Clarification would be helpful.

Reviewer #3

(Remarks to the Author)
The revised manuscript has addressed the concerns appropriately, and I recommend it for acceptance in Nature Communications.

Reviewer #4

(Remarks to the Author)
Thank the authors for their work on the revisions and for the detailed response letter. This revised version has successfully addressed most of the major concerns I raised previously. But I still and one request for clarification on a potential system-level bottleneck and one minor suggestion on presentation.

Potential Bottleneck in the Block-Matrix Method's Global Aggregation Step: Validation of the block matrix-based method raised in the response introduces a "maximization and scaling" step, which appears to require global data aggregation from all active cores to the central "Functional Module". My concern is whether the system-level performance model (specifically, the latency breakdown in Fig. S20) fully accounts for this potentially costly step. For large-scale problems using hundreds or thousands of cores, this implies a massive data movement and synchronization event. If this latency analysis models the potential bus contention when all cores attempt to write their intermediate results back simultaneously? This step seems like a potential system-level bottleneck that is distinct from the in-core analog computation, and a more detailed analysis would be good to the overall performance claims.

Figures: While the content of the figures in the main text is informative, their presentation should be polished for better professional quality. There are some inconsistencies in font types and sizes. Also, some figures are quite text-heavy, which can make readers difficult to get the main idea of the paper and being confused by too many technical details. Authors should consider moving some of them to SI.

Version 2:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
I am satisfied that the authors have addressed the concerns that were raised. I recommend the manuscript for publication in Nature Communications.

Reviewer #4

(Remarks to the Author)
The authors have satisfactorily addressed the issues raised in the prior rounds of reviews, and the manuscript should be ready for publishing.

Reviewer #1:

The paper "In-memory analog computing for non-negative matrix factorization" by Shiqing Wang and collaborators presents a novel hardware solution that is able to perform a complicated linear algebra decomposition technique that has multiple industrial applications. Their implementation relies on a new circuit that enables the calculation of the generalized inverse of a linear system, which is implemented in a way that requires significantly less energy, and which enables a way to perform non-negative matrix factorization (NMF) using memristor-based analog in-memory computing hardware. The authors showcase impressive results on image compression and recommender systems that show several orders of magnitude improvement on runtime and energy efficiency over GPU and FPGA-based implementations.

While the paper is of high quality and the results shown on the manuscript are very impressive, there are a number of aspects that we think could be improved:

Reply:
We thank the Reviewer for the positive feedback and for the insightful suggestions to improve our work. We have carefully addressed all the concerns outlined below and made the corresponding revisions to the manuscript.

Q1:
Line 36: The authors point that digital hardware struggles to solve NMF with large matrices. It would be useful to clarify what they mean by large matrices and justify how and when RRAM-based hardware will be able to tackle matrices of the size used in industry-relevant applications. It wasn't clear to me whether decomposition techniques will be enough to be enable practical work on industrial applications.

Reply:
We thank the Reviewer for raising the important point regarding the definition of "large matrices" and the capability of RRAM-based solvers to handle real-world applications. For example, the Netflix dataset contains approximately 480,000 users and 17,000 items, requiring NMF to factorize a $480{,}000\times17{,}000$ matrix to extract latent features of both users and movies. Similarly, the Yahoo Music dataset contains about 1,200,000 users and 137,000 items, corresponding to an even larger matrix for NMF tasks.

To address such large-scale problems, AMC circuits with larger RRAM arrays and the proposed block-matrix method can be employed. While the present work demonstrates a limited experimental scope, closed-loop AMC circuit architectures have already been shown to support matrix inversion up to 64×64 [P. Mannocci *et al*., in *2023 International Electron Devices Meeting (IEDM)*, 1-4.]. Scaling up the single-core solver would therefore enable larger matrix computations. Moreover, by leveraging block-matrix algorithms and multi-core architectures, the proposed RRAM-based solver can be extended to tackle industry-scale NMF tasks.

In the revised manuscript (line 37), we have modified the sentence as follows:

"However, for modern large-scale matrices, such as the Yahoo Music dataset (1,200,000×137,000 rating matrix), traditional digital hardware struggles to meet the demands of real-time NMF tasks[13–14]. This limitation arises from intrinsic factors, including high algorithmic complexity, the slowdown of Moore's Law[15–16], and the von Neumann bottleneck[17–18]. Analog computing leverages massive parallelism to accelerate matrix operations. In particular, by combining block-matrix algorithms to scale up the addressable problem size[19] [L. Pan *et al.*, *DATE 2024*, pp. 1–6], analog computing can provide a viable path to solving large-scale NMF challenges."

Q2:

Line 69: One of the advantages of RRAM-based hardware is that the logic is very dense, resulting in small circuits, which potentially could enable massive parallelization. Some discussion on how this could affect NMF applications would likely be useful.

Reply:

We appreciate the Reviewer's suggestion regarding the advantage of RRAM-based hardware in enabling massive parallelization for NMF applications, and we have added the corresponding discussion in the revised manuscript (line 73):

"AMC represents a highly parallel computing paradigm, where matrix operations are executed in a massively parallel fashion. During computation, all cells in the activated rows and columns participate simultaneously. This stands in sharp contrast to digital computing, where basic operations (*e.g.*, multiplication and addition) are performed sequentially, leading to polynomial complexity for solving NMF problems. By leveraging massive parallelism, AMC is particularly well suited for accelerating NMF with low computational complexity."

Q3:

Line 106: Analog hardware has significant promise, but also some drawbacks when compared with digital hardware, such as limited precision. The manuscript lacks sufficient details on the precision limitations on the hardware, and its impact on the two applications being evaluated.

Reply:

We agree with the Reviewer that analog computing is subject to limited precision. In the revised manuscript, we have expanded the discussion on this constraint and demonstrated that the impact of analog computing errors on final results is small, owing to the strong error tolerance of the ANLS algorithm.

In our experiments, precision loss may arise from multiple sources, including mapping errors and variability in resistor arrays, quantization errors from DACs and ADCs, and noise from analog components such as OPAs, power supplies, and PCBs. We performed extensive experiments on NLR, NRR, and NCRR. Compared with ideal solutions obtained from an FP64 digital solver, all experimental results yielded NMSE values below 0.01, with an average NMSE of 0.0072 across ~430,000 experimental trials.

Although the circuits exhibit an average NMSE of 0.0072 relative to FP64 results, the effect on final application-level outcomes is significantly mitigated by the substantial basin of attraction around

the global optimum in NMF problems [Y. Chi *et al.*, *IEEE Trans. Signal Process.* **67**, 5239–5269 (2019)]. For instance, in the image compression experiments, the recovered image shows only a 1.39 dB PSNR reduction (32.88 dB vs. 34.27 dB for FP64) and a minor NMSE increase of 0.0009 (0.0033 vs. 0.0024). In the recommender system experiment (MovieLens 100k), the NMSE on the test dataset increases by just 0.0001 after the third cycle (0.0800 vs. 0.0799).

Moreover, to achieve precision comparable to digital computing, iterative refinement algorithms can be employed within an analog–digital hybrid system [M. Le Gallo *et al.*, *Nat. Electron.* **1**, 246–253 (2018)].

In the revised manuscript (line 337), we have added a detailed analysis of AMC precision and the error tolerance of the ANLS algorithm in the new "Analysis of error tolerance" section.

Q4:

Lines 106-107 claim that "the ANLS algorithm's alternating optimization framework and non-negativity constraints inherently suppresses error propagation". This is cited without references and should be justified in more detail.

Reply:

We thank the Reviewer for the helpful suggestion to provide more details on the error suppression properties of the ANLS algorithm. In the revised manuscript, we have added the corresponding references and expanded our explanation, supported by both theoretical justification and experimental verification.

The ANLS algorithm reformulates the non-convex NMF problem into a sequence of convex subproblems (*i.e.*, NLR, NRR, NCRR), enabling efficient convergence. A substantial basin of attraction often exists around the global optimal solution [Y. Chi *et al*., *IEEE Trans. Signal Process* **67**, 5239–5269, 2019], meaning that a wide range of initial points can converge to the optimum. As a result, even when large computation errors occur in one cycle, subsequent iterations are still able to converge to the global solution with high probability.

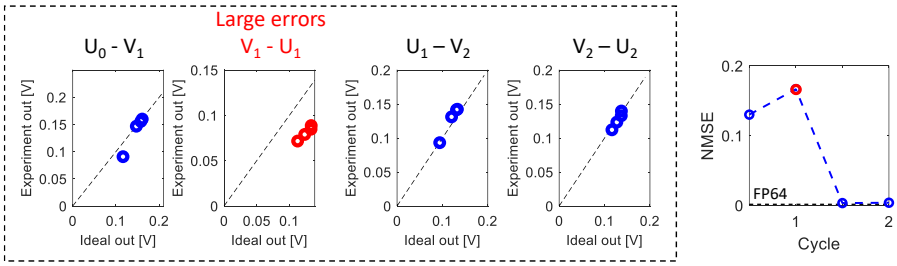

Fig. R1. Experimental validation of ANLS error tolerance

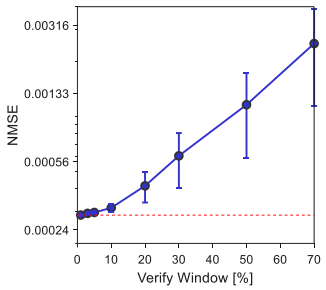

Fig. R2. Simulation validation of ANLS error tolerance

To support this claim, we provide an experiment in Fig. R1, where large computing errors in the transition from $\boldsymbol{V}_1$ to $\boldsymbol{U}_1$ initially cause a high NMSE. Nevertheless, subsequent analog computations drive the NMSE to converge to values close to the FP64 benchmark (red line).

We also systematically investigated the relationship between convergence behavior and error magnitude. Since mapping error is the dominant error source in our analog NMF solver, we evaluated different error levels [±1%, ±3%, ±5%, ±10%, ±20%, ±30%, ±50%, ±70%]. In these simulations, a 4×4 matrix was factorized into two 4×2 matrices, with 100 trials performed for each error level. As shown in Fig. R2, the ANLS algorithm exhibits strong tolerance: under ±10% write errors, the reconstructed NMSE is 3.14×10^{-4}, only slightly higher than the FP64 benchmark (2.86×10^{-4}). Even under ±70% write errors, the algorithm maintains an average reconstructed NMSE of 2.5×10^{-3}.

In the revised manuscript, we have added the following clarifications:
(Line 70): "In low-rank matrix factorization, a substantial basin of attraction often exists around the global optimal solution[34]. The alternating non-negative least squares (ANLS) algorithm reformulates the non-convex problem into multiple convex subproblems, efficiently converging to this basin[35], and is therefore widely adopted for NMF."
(Line 115): "Thanks to the substantial basin of attractions around the global optimal solution[33], a large set of initial points can converge to the global optimum. Therefore, even when large computing errors occur in one cycle, subsequent iterations are still able to converge to the optimal solution with high probability."

We have also added a detailed discussion in the new "Analysis of error tolerance" section (line 337).

Q5:
Line 166: the adaptive accuracy mapping method that was used on the write-verify method should have a link to the supplementary material. Details on its contribution to latency or energy costs would be welcome.

Reply:
We thank the Reviewer for suggesting that we provide a link to the supplementary material and quantify the contribution of the adaptive accuracy mapping method. In the conventional programming scheme, all conductance states share a single verify tolerance, which often requires more write cycles when mapping to higher conductance states. By contrast, our proposed scheme

adaptively adjusts the verify tolerance based on the magnitude of the target conductance, thereby accelerating the programming process. Compared with the conventional scheme (13.3 cycles on average per device) [Z. Jiang *et al.*, *IEDM 2023*, pp. 1–4], the adaptive accuracy programming approach reduces the average cycles to 9, achieving a speed improvement of approximately 32.3%. This reduction in latency also contributes to lower energy consumption.

In the revised manuscript, we have linked the adaptive accuracy mapping method to Fig. S3 and included a discussion of its contribution to latency in the main text (line 186).

Q6:

Line 204: the ANLS algorithm is shown to converge extremely quickly for the problem sizes evaluated on the hardware. This is very impressive, but it would be useful to show whether this is consistent across multiple problem/matrix sizes.

Reply:

We thank the Reviewer for the suggestion to examine the convergence speed across different problem and matrix sizes. We have supplemented the analysis, and the results indicate that problem/matrix size has only a negligible impact on convergence speed.

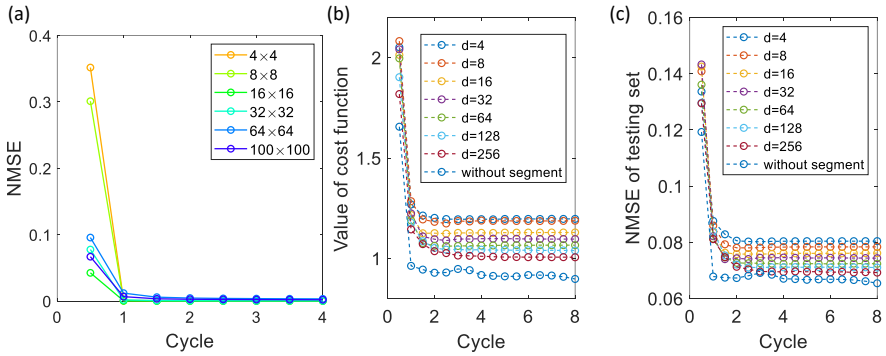

Fig. R3 (a) Convergence speed in image compression application with different matrix size. (b)-(c) Convergence speed in recommender system application with various block matrix size.

Specifically, we performed small-scale (4×4) decomposition experiments, which showed rapid convergence within two cycles (Fig. 3e). For larger-scale tasks, we evaluated both image compression and recommender systems. In the image compression application, we used the same nebula image but varied the block sizes for NMF. As shown in Fig. R3(a), all block sizes converged within two cycles. In the recommender system case, we studied the MovieLens 100k dataset with different block sizes (d = 4 to 256). The results demonstrate that convergence behavior is consistent and independent of block size (Fig. R3b–c).

In the revised manuscript, we have added these results to the image compression section (line 245, Fig. S7) and the recommender system section (line 321, Fig. S13).

Q7:

Line 188: Identical should be identity instead.

Reply:
We thank the Reviewer for pointing out this error, and we have corrected “identical” to “identity” in the revised manuscript.

Q8:
In the abstract and introduction the authors mention that the NMF algorithm has multiple industrial applications "...recommender systems, bioinformatics, and image processing". They then proceed to compare the AMC hardware on two applications with implementations on GPU and FPGA, showing very significant improvements. However, those two chosen applications (image compression and recommender systems) have multiple implementations, and it is unclear from the paper whether the GPU and FPGA points of comparison represent the state of the art. It would be useful to establish what is the state of the art on these applications, and compare the AMC hardware with those. In the case of recommender systems, given that machine learning techniques are most often used to train recommender systems, it would be useful to also compare the quality of the recommender system.

I found the comparison on latency and energy with the GPU implementation a bit strange. The GPU used in the comparison was relatively old (Titan RTX from 2018), a more modern system would have been preferable. More importantly, for speed purposes the GPU was run on a single core (out of 4608), but for energy purposes the full energy consumption of the GPU was taken into account. It would be very useful to see a comparison on the implementation using the full GPU resources, and comparing it with a simulation of AMC hardware being run on a parallel fabric.

Reply:
We thank the Reviewer for the valuable feedback on our benchmarking methodology and for the helpful suggestion regarding parallel fabric. In response, we have designed a multi-core architecture and performed comparisons with state-of-the-art (SOTA) GPU and FPGA implementations.

For GPUs, the NVIDIA TITAN RTX (130 TOPs, INT 8) was the most advanced platform we initially identified for NMF. To ensure fairness, we have now extended the benchmarking to the latest SOTA GPU, NVIDIA H100 (4000 TOPS, INT8), by scaling performance estimates according to peak compute capabilities ($T_{H100} = 130/4000 \cdot T_{TITAN}$). For FPGAs, the AMD Virtex UltraScale+ VU9 was previously considered, but we now also include the AMD Versal™ AI Core (VC2802), the most advanced FPGA we identified for NMF, with performance again estimated from peak computing capabilities.

Following the Reviewer’s suggestion, we designed a multi-core architecture for the analog NMF solver, described in detail in Reply Note S1 (page 9 of this reply). Considering the dominant sources of latency and power consumption, we conducted a comprehensive performance evaluation. The design assumes 1108 cores, consistent with the PUMA architecture [A. Ankit *et al*., *ASPLOS* 2019, pp. 715–731.]. For the MovieLens 100k dataset, only 7 cores are required, while large-scale datasets such as Netflix and Yahoo Music utilize the full 1108 cores. For the Movielens 100K dataset, our solver achieves a 212× speedup and 4.6×10^4 improvement of energy efficiency compared to AMD

Versal™ AI Core (VC2802). For the Netflix dataset, our solution achieves a 12.4× speedup and a 280.9× improvement in energy efficiency over the H100 (14,592 CUDA cores), and a 9.3×10^3 speedup and 2.1×10^4 improvement over the AMD Versal™ AI Core (VC2802). For the Yahoo Music dataset, the proposed solver achieves a 3.3× speedup and a 34.2× improvement in energy efficiency compared to the H100, and a 7.0×10^3 speedup and 6.9×10^3 improvement compared to the VC2802. Updated latency and energy breakdowns are shown in Fig. R5.

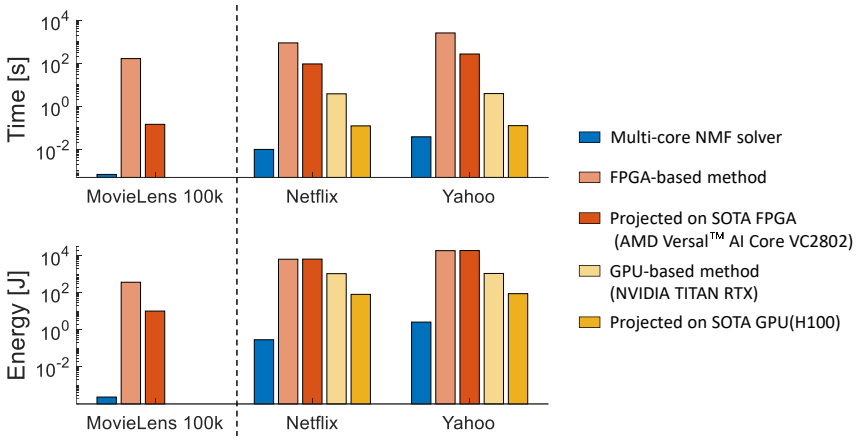

Fig. R4. Updated Benchmark results

We also evaluated solution quality against advanced machine learning algorithms using the MovieLens 100k dataset. For consistency, we adopted the RMSE (Root Mean Square Error) metric, defined as $(\|\boldsymbol{x}\text{-}\hat{\boldsymbol{x}}\|_2^2/|\boldsymbol{T}|)^{1/2}$, where $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ denote the predicted and ground-truth values, respectively, and $|\boldsymbol{T}|$ is the size of the test set. The most advanced algorithm we identified [Z. Z. Darban *et al*., *Expert Syst. Appl.* **200**, 116850, 2022] combines graph models with deep learning and achieves an RMSE of 0.887. As presented in Table. R1, our unsegmented simulation produces slightly worse results than advanced deep learning methods but remains comparable to standard machine learning approaches, such as recurrent neural networks [F. Monti *et al*., *NeurIPS 2017*]. With the block matrix-based method, however, performance is somewhat degraded due to information loss at very small segment sizes (d=4). As shown in Fig. R6, increasing d significantly improves accuracy, suggesting that larger RRAM arrays will yield better quality results.

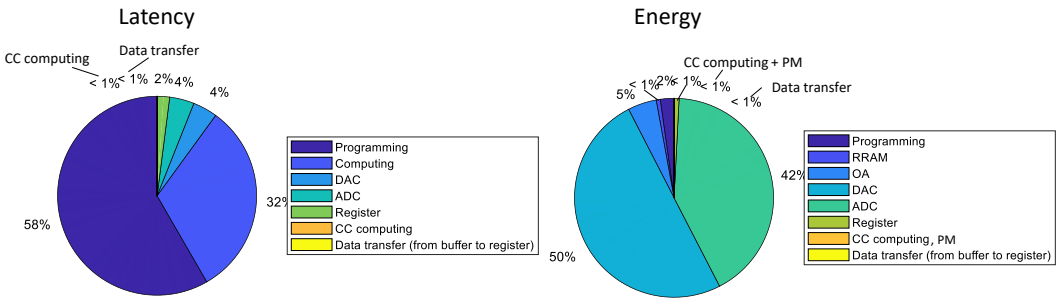

Fig. R5. Updated latency and energy breakdown image.

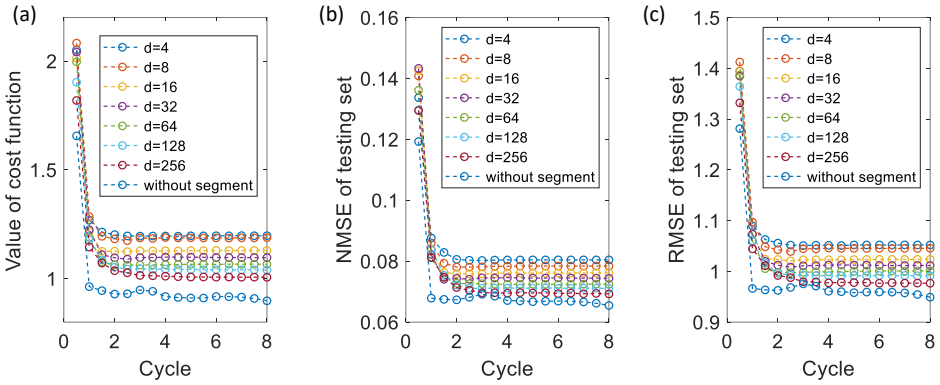

Fig. R6. The scaling behavior of block matrix method. (a) Value of cost function in training set (b) NMSE of testing set (c) RMSE of testing set.

Method	**RMSE**
Graph model + deep learning	0.887
Recurrent neural network	0.929
NMF without segment (This work)	0.958
NMF (d=4, experiment) (This work)	1.048
NMF (d=256, simulation) (This work)	0.975

Table. R1. Solving quality against deep learning method.

In the revised manuscript, we have added the multi-core architecture as Supplementary Note S4 and updated the corresponding benchmark results (line 375, Fig. S19 and Fig. S20). We have referenced the benchmark results in the main text (line 404) as following:

"Task of Movilens 100k occupies 7 cores and achieves a 212× speedup and 4.6×10^4 improvement of energy efficiency compared to SOTA FPGA. In tasks of Netflix and Yahoo, the proposed architecture operates with full 1108 cores and it achieves a speedup of thousands of times compared to SOTA FPGA and several times compared to H100 GPU on average (Fig. S19). Additionally, it demonstrates thousands of times improvement in energy efficiency over SOTA FPGA and hundreds of times over H100 GPU on average (Fig. S19). The assessment reveals that latency is primarily dominated by device programming (58%, Fig. S20), while power consumption is largely attributed to DACs and ADCs (92%, Fig. S20) in the analog NMF solver."

For recommender systems, we have included comparisons of solution quality with deep learning methods (line 326, Table S2).

Reply Note S1:

As shown in Fig. RS1, we illustrate a multi-core architecture designed to solve the NMF problem using the ANLS method. This architecture consists of 1108 cores, consistent with the PUMA architecture [A. Ankit *et al*., *ASPLOS 2019*, pp. 715–731]. For large-scale recommender system datasets such as Netflix and Yahoo Music, this architecture can operate at full capacity with all 1108 cores.

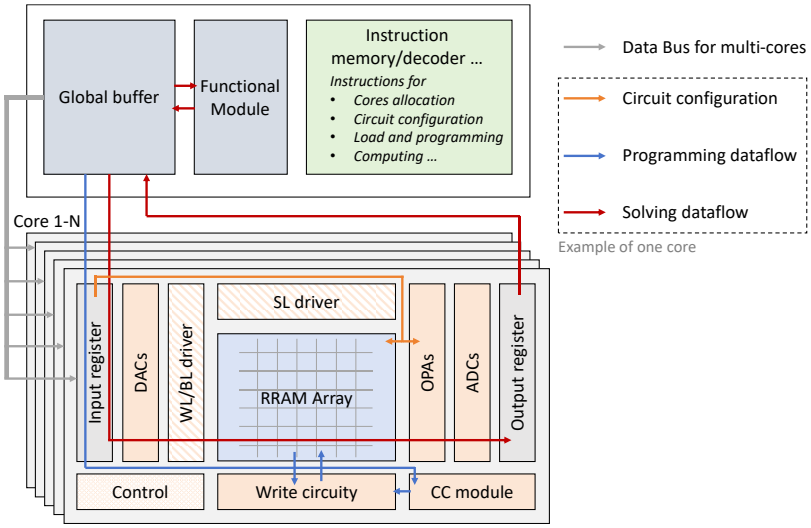

Fig. RS1. Multicore architecture and data flow.

Overall Structure:

Global Buffer: The global buffer stores the matrix to be decomposed and the latent feature matrices during ANLS cycles, such as $\boldsymbol{U}_1$, $\boldsymbol{V}_1$. It is also connected to the cores via a shared data bus.

Functional Module: The functional module contains logic units for nonlinear functions, such as the comparators used in MS operations. It includes 1107 8-bit digital comparators for selecting the maximum value from the outputs of the 1108 cores.

Instruction Module: The instruction module is used to store and translate system instructions.

Core Structure:

Input and Output Registers: Each core has input and an output registers. The input register receives data from the global buffer, including circuit configuration instructions, target conductance for RRAM array, and input voltages. The output register is connected to the ADC for capturing outputs generated by the compact-GINV circuit.

RRAM Array: Each core contains a 1T1R RRAM array of size 256×66. Thus, a single core can implement up to two 256×32 matrices, with the remaining 256×2 section used for compensation conductance.

OPAs: Each core integrates 32 OPAs. Together with the RRAM array, they form the configurable compact-GINV circuit. Circuit configuration instructions generate control signals that specify whether the circuit performs NLR, NRR, or NCRR operations.

DAC and ADC: The DAC is used to generate input voltages for compact-GINV circuit, and the ADC is used for data readout.

SL Driver and WL/BL Driver: These drivers manage the access and operations of the RRAM

array.

Control module: The control module provides logic to manage the RRAM array and other core components.

Write circuitry: It executes the write-verify algorithm for RRAM programming.

CC Module: It is used to calculate values of compensation conductance, implemented as an adder tree (255 adders) [H. Naseri *et al*., *IEEE Trans. VLSI Syst.* **26**, 1481–1493, 2018].

This analog NMF solver is assumed operated at a 200 MHz system clock to ensure efficient timing granularity for mixed-signal data conversion [L. Pan *et al*., *DATE 2025*, pp. 1-2.]. The bandwidth between the SRAM-based global buffer and cores is set to 4 Kb/clock, consistent with [W. -H. Huang *et al*., *ISSCC 2023*, pp. 15-17.]. Module design parameters largely reference the PUMA architecture [A. Ankit *et al*., *ASPLOS 2019*, pp. 715–731.], based on 32 nm CMOS node. OPA and ADC specifications are drawn from [S. Wang *et al*., *Sci. Adv.* **9**, eadj2908, 2023] and [Kull, L. *et al*., *IEEE J. Solid-State Circuits* **48**, 3049–3058, 2013], respectively. A 3-bit R-2R DAC was designed for evaluation (Fig. RS2), with measured power consumption of 301 uW. Module parameters are summarized in Table RS1.

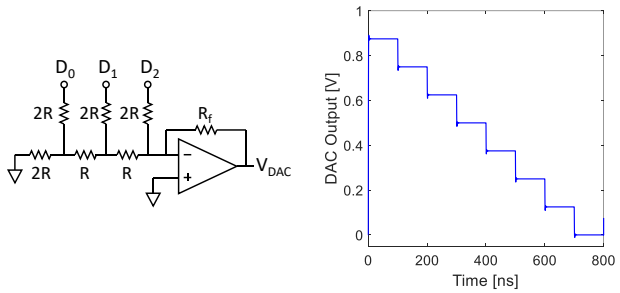

Fig. RS2. 3-bit DAC design and its simulation results.

Component	Power (mW)	Parameter	Specification
FM	3.73	# comparators	1107
Global buffer	9.14	Bandwidth	4 Kb/clock
Input register	0.12	Capacity	2 Kb
Output register	0.015	Capacity	0.25 Kb
DAC	0.3	Latency	10 ns
ADC	2.2	Latency	10 ns
OPA	0.224	GBW	1 GHz
RRAM array	4.2	Capacity	256x66
CC module	0.011	# adders	63
Write circuity	3.32	Parallel degree	8

Table. RS1. The applied parameters in architecture.

The execution sequence for a single core (Fig. RS1) begins with GINV-circuit configuration (yellow lines), which sets the operation mode (NLR, NRR or NCRR) of compact-GINV circuit. In the programming dataflow (blue lines), $\boldsymbol{U}_i/\boldsymbol{V}_i$ data is transferred from the global buffer to the input registers. The write circuitry then performs the write-verify process for array programming. In

parallel, the CC module calculates the compensation conductance, which is programmed via the write circuitry. In the solving dataflow (red lines), input data (*e.g.*, known ratings, 3 bits per value) is supplied from the global buffer to the input registers, delivered through DACs to the compact-GINV circuit. ADCs digitize the outputs, which are temporarily stored in output registers and then returned to the global buffer. The functional module subsequently performs nonlinear operations (MS operation) in a pipelined manner.

As illustrated in Fig. RS3, externally provided parameters-including matrix dimensions for decomposition, number of latent features, and block size are compiled into instructions that govern chip operation. The number of cores is first determined based on matrix dimensions and block size (via core allocation instructions). Next, hardware configurations are applied to each core's compact-GINV circuit (via configuration instructions), specifying activated row/column counts and computing modes (NLR, NRR, or NCRR). Finally, dataflow control instructions (for load/programming and computing) manage the overall ANLS process.

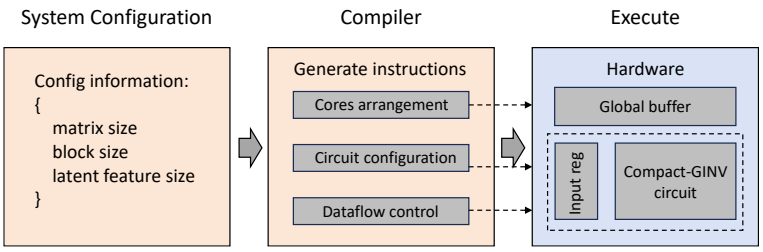

Fig. RS3. The execution process: from source code to hardware execution.

Reviewer #2:

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.

Reply:

We thank the Reviewers for their insightful suggestions for improving our work. We have addressed all the concerns below and made corresponding revisions to the manuscript.

Major comment:

Q1:

Line 151: The sentence, "convenient to introduce various constraints in the target cost function," highlights what I believe is the most novel contribution of the paper—extending prior work on linear regression in hardware (Refs. 19/20). However, the manuscript lacks sufficient detail to understand how this constraint integration is actually realized in hardware. Additional explanation would strengthen this part significantly.

Reply:

We thank the Reviewer for the suggestion to provide more details on the compact-GINV circuit. In the revised manuscript, we have added specific connection details of the circuit under different operating modes (NLR, NRR, and NCRR), as illustrated in Fig. R7.

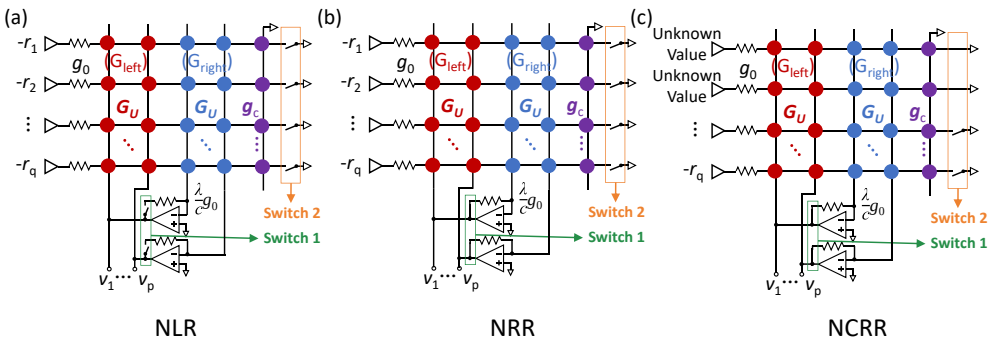

Fig. R7. (a) configuration for solve NLR problem. (b) configuration for solve NRR problem. (c) configuration for solve NCRR problem.

When both Switch 1 and Switch 2 are OFF, the circuit performs the NLR function (Fig. R7(a)). If Switch 1 is ON, an L_2 penalty term is introduced into the loss function, and the circuit executes the NRR function (Fig. R7(b)). Specifically, as described in Supplementary Note S1, the circuit equations become $\boldsymbol{U}\boldsymbol{v}\text{-}\boldsymbol{r}=c\mathbf{V}_{\mathrm{BL}}$ and $\boldsymbol{U}^{\mathrm{T}}\mathbf{V}_{\mathrm{BL}}+\lambda/c\boldsymbol{v}=0$, $\boldsymbol{v}\geq 0$, which yields the circuit output $\boldsymbol{v}=\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{r}$, $\boldsymbol{v}\geq 0$. When a subset of Switch 2 is ON, all RRAM devices in the corresponding row are grounded, meaning the input voltage of that row does not affect the output voltages. In this mode, the circuit performs NCRR computing (Fig. R7c).

In the revised manuscript, the circuit connectivity for each operational mode is now provided in Fig. S1, and the main text has been updated with additional explanation (line 175).

Minor comments:

Q2:

Line 52: I suggest using "a system of linear equations" rather than "matrix equations".

Reply:

We thank the Reviewer for this kind suggestion. However, since AMC circuits are capable of solving not only linear systems of equations [Z. Sun *et al*., *Proc. Natl. Acad. Sci. U.S.A.* **116**, 4123–4128, 2019; Z. Sun *et al*, Sci. Adv. **6**, eaay2378, 202.] but also eigenvalue problems [Z. Sun *et al*., *IEEE Trans. Electron Devices* **67**, 1466-1470, 2020] and nonlinear matrix equations such as sparse approximation [S. Wang *et al*., *Sci. Adv.* **9**, eadj2908, 2023], we feel it is more accurate to retain the broader term "matrix equations" in the revised manuscript.

Q3:

Line 53: I suggest adding the phrase "without having to compute the inverse of the data matrix" to clarify the computational advantage being described.

Reply:

We thank the Reviewer for this helpful suggestion, and we have added the phrase "without having to compute the inverse of the data matrix" in the revised manuscript (line 57).

Q4:

Line 95: The technical term L2 constraint appears without a formal definition.

Reply:

We thank the Reviewer for highlighting the need for a formal definition of the L_2 constraint. We have added this definition in the revised manuscript (line 104): "L_2 constraint (L_2 regularization term $\|x\|_2^2$)".

Q5:

Line 99: The symbols Θ and Ψ are not constraints themselves. It would be more accurate to refer to them as penalty functions, which is the commonly used term in such contexts. Correspondingly, α and β are more appropriately described as penalty parameters or regularization parameters.

Reply:

We thank the Reviewer for this helpful suggestion, and we have revised the manuscript accordingly, replacing "constraints" with "penalty functions" for $\Theta(\cdot)$ and $\Psi(\cdot)$, and referring to α and β as "penalty parameters" or "regularization parameters" (line 108).

Q6:

Line 103: The phrase regularity of matrix computing is unclear. Please consider rephrasing or

elaborating on what is meant here.

Reply:
We thank the Reviewer for this suggestion, and we have added the following explanation in the revised manuscript (line 112): "Matrix computing consists of regular, massive multiply-and-accumulate operations. This regularity makes the ANLS algorithm well suited for acceleration using analog parallel architectures."

Q7:
Line 109: While the notation $||X||_F$ unambiguously refers to the Frobenius norm, $||X||_2$ is often interpreted as the spectral norm (i.e., the largest singular value of X). To avoid confusion, I recommend consistently using $||X||_F$ unless the spectral norm is specifically intended.

Reply:
We thank the Reviewer for this kind suggestion. To avoid confusion between the Frobenius norm and the spectral norm in matrix notation, we now consistently use $\|\boldsymbol{X}\|_{\mathrm{F}=2}^2$ for matrix terms in the revised manuscript.

Q8:
Line 117: The use of the superscript star in Ω^* is unclear. Please clarify the meaning of this notation.

Reply:
We thank the Reviewer for this suggestion. The superscript star has no special meaning in this context, and to avoid any misunderstanding, we have removed it in the revised manuscript.

Q9:
Line 124: Consider removing the word inner from the text.

Reply:
We thank the Reviewer for this suggestion, and we have done so in the revised manuscript.

Q10:
Line 127: This section focuses on a vector (p-by-1 variable) rather than a matrix variable. While this is understandable due to the problem's separability, it would be helpful to explicitly mention this and briefly explain why it suffices to consider a vector formulation.

Reply:
We thank the Reviewer for this suggestion, and we have added the following explanation in the revised manuscript (line 152): "Since the cost function is defined as the squared Frobenius norm (*i.e.*, the sum of squared errors), the matrix-form cost function is equivalent to the sum of vector-form cost functions. For example: $\|(\boldsymbol{R} - \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})\|_{\mathrm{F}=2}^2 = \sum_{i=1}^{n} \|(\boldsymbol{r}_i - \boldsymbol{U}\boldsymbol{v}_i)\|_2^2$, where $\boldsymbol{r}_{\mathrm{i}}$ and $\boldsymbol{v}_{\mathrm{i}}$ denote the i^{th} column vectors in $\boldsymbol{R}$ and $\boldsymbol{V}^{\mathrm{T}}$, respectively. Minimizing each vector-form cost functions is therefore equivalent to minimizing the matrix-form cost function. For convenience, we adopt the

vector formulation in the following circuit analysis.”

Reviewer #3:

The paper presents an analog computing framework for NMF, based on a compact and reconfigurable GINV circuit. The proposed design aims to reduce the number of OPAs and supports various constrained regression formulations such as NRR and NCRR. Experimental validations on image compression and recommender systems highlight the potential of the approach in terms of energy efficiency and convergence speed.

However, several points require further clarification or quantitative analysis to substantiate the claimed advantages and practical trade-offs of the proposed design:

Reply:
We thank the Reviewer for the positive feedback and for the insightful suggestions to improve our work. We have carefully addressed all the concerns outlined below and made the corresponding revisions to the manuscript.

Q1:
The main novelty of the paper appears to be the reduction in the number of OPAs and the ability to support both NRR and NCRR using a compact GINV circuit. However, since the performance benchmarking does not include prior analog NMF solvers, the specific advantages of the proposed circuit are not clearly demonstrated. How does this design compare—specifically in terms of power, latency, and performance—to a conventional GINV circuit implementation?

Reply:
We appreciate the Reviewer's valuable feedback regarding the performance benchmarking of our compact-GINV circuit. To directly address how our design compares with a conventional (vanilla) GINV implementation in terms of OPAs, power, latency, and accuracy, we conducted circuit-level analyses on a 256×32 ridge regression task, and system-level evaluations within the multi-core architecture (Reply Note S1, page 23 of this reply) applied to a recommender system.

Number of OPAs: For NLR computing, the vanilla GINV circuit needs $q+p$ OPAs. To perform NRR computing, an additional p OPAs are needed, bringing the total to $q+2p$ [P. Mannocci *et al*., *IEEE J. Explor. Solid-state Comput. Devices Circuits* **9**, 47–55, 2023]. This results in a requirement of 256 + 32 + 32 = 320 OPAs (Fig. R8). In contrast, the compact-GINV circuit solves NRR problem using only 32 OPAs, representing a 90% reduction in core analog components.

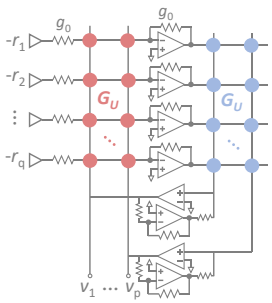

Fig. R8. Valina GINV circuit to solve NRR problem.

Computing Latency: At the circuit level, we performed SPICE simulations using 1 GHz gain-bandwidth-product OPA [S. Wang *et al*., *Sci. Adv.* **9**, eadj2908, 2023], including long-wire delay. For the vanilla circuit (two 256×32 arrays), we added 30 fF capacitance to column lines and 3.75 fF (Fig. R9) to row lines [Y. Zhang *et al*., *IEEE Electron Device Lett*. **43**, 1203-1206, 2022]. For the compact-GINV circuit (one 256×66 array, with 2 columns for compensation conductance), we added 30 fF to column lines and 7.5 fF to row lines. Fig. R10 shows that the compact-GINV circuit achieves slightly lower latency (79.2 ns) compared to the vanilla circuit (82.6 ns). At the system level, we accounted for latency from both computing and programming compensation conductance. To minimize compensation conductance values, two parallel columns were used for compensation conductance mapping. Based on the multi-core architecture (Reply Note S1, page 23 of this reply), decomposition of the Yahoo Music dataset required 0.0384 s with the vanilla GINV circuit versus 0.0382 s with the compact-GINV circuit, showing only a 0.52% increase due to CC-related latency.

Power Efficiency: At the circuit level, which includes the RRAM array and OPAs, the vanilla GINV circuit consumes approximately 9× more power than our compact-GINV design (Fig. R11a). At the system level, substituting the compact-GINV with the vanilla circuit increases overall power consumption by 1.46×, highlighting the scalability of our power advantage (Fig. R11b).

Computing Accuracy: SPICE simulations with ±5% device programming tolerance show that both circuits achieve nearly identical solving precision, with the vanilla circuit yielding 5.53% relative error versus 5.52% for the compact-GINV circuit (Fig. R12).

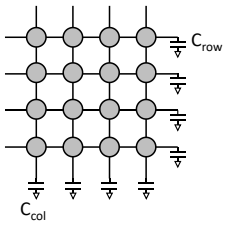

Fig. R9. Model of long-wire delay.

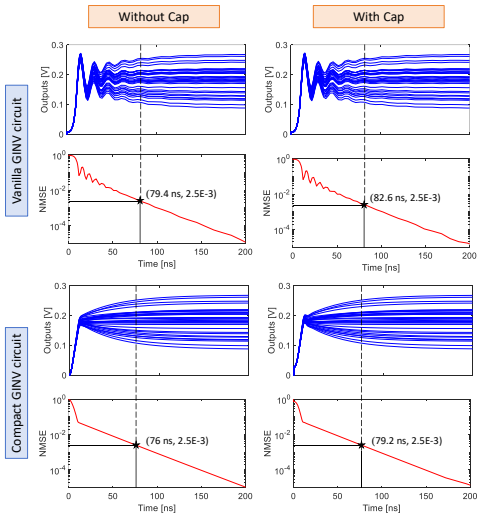

Fig. R10. Transient simulation results of vanilla and compact GINV circuit and the impact of parasitic capacitance.

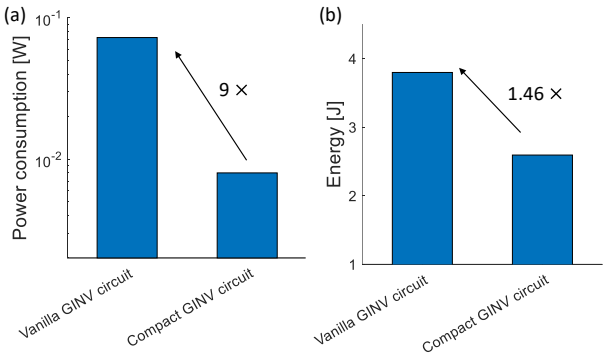

Fig. R11. Power consumption of two circuits in (a) circuit level and (b) system level.

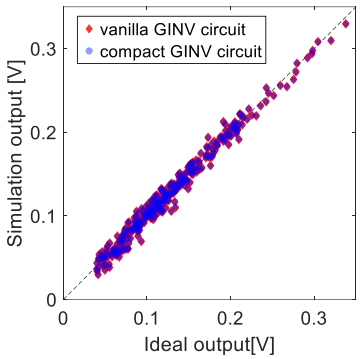

Fig. R12. Computing accuracy of two circuits.

In the revised manuscript, we have incorporated this comparative analysis into Supplementary Note S5 and referenced the results in the main text (line 413).

Q2:

According to the energy breakdown in Fig. S12, the majority of energy consumption arises from DACs and ADCs, not OPAs. In that case, is the reduction in OPA count from p+q to p truly impactful at the system level?

Reply:

We thank the Reviewer for the insightful observation regarding the energy distribution in Fig. S12.

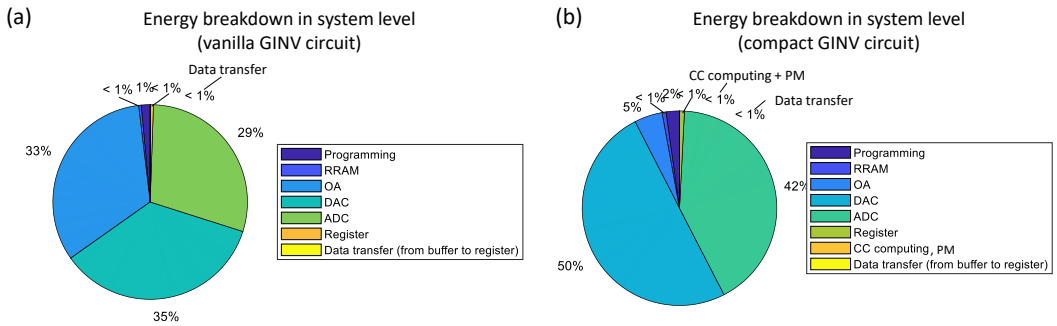

Fig. R13. (a) Energy breakdown of vanilla GINV circuit (b) Energy breakdown of compact GINV circuit.

The reduction in OPAs has a meaningful impact on total power consumption when evaluated at the system level with the compact-GINV design. Using the multi-core analog-NMF solver model (described in Reply Note S1, page 23 of this reply), we re-evaluated the energy breakdown. As shown in Fig. R13, in the vanilla circuit OPAs account for 33% of total system power, whereas in the compact-GINV circuit this share drops to 5%. Overall, system-level power consumption is reduced to 1/1.46 of that in the vanilla implementation (Fig. R11).

In the revised manuscript, we have incorporated this comparative analysis into Supplementary Note S5.

Q3:

The significantly higher conductance range of g_c compared to G_left and G_right, as shown in Fig. 5c and Fig. S7, raises concerns regarding potential increases in programming latency and energy consumption. It would be helpful if the authors could quantitatively evaluate this overhead in comparison with traditional GINV implementations.

Reply:

We thank the Reviewer for raising the concern about the conductance range of $\boldsymbol{g}_c$ and its potential impact on programming latency and energy consumption. During computation, the additional power consumption caused by high-conductance states is negligible, as the RRAM array accounts for less than 1% of total power consumption (Fig. R13). Therefore, our analysis focuses on the additional latency and power overhead from programming operations.

To reduce mapped conductance values, we employ a configuration in which a single CC column ($\boldsymbol{g}_c$) is mapped onto two parallel columns. This effectively halves each individual compensation conductance value. Based on the multi-core architecture (Reply Note S1, page 23 of this reply), decomposition of the Yahoo Music dataset requires 0.0384 s with the vanilla GINV circuit and 0.0382 s with the compact-GINV circuit, indicating that computing and programming CC introduce only a 0.52% latency increase at the system level. In contrast, replacing the compact-GINV circuit with the vanilla design increases overall system power consumption by 1.46× (Fig. R11).

In the revised manuscript, we have added this analysis to Supplementary Note S5.

Q4:

How do retention and temperature variation of RRAM devices affect the stability or performance of the NMF process? Can their impact be considered negligible in practical scenarios?

Reply:

We thank the Reviewer for raising the important point of device retention and temperature variation.

In our analog NMF solver, computing is performed immediately after programming the entire array. Based on the circuit outputs, the RRAM array is then reprogrammed, and this process iterates. Using the designed multi-core architecture (Reply Note S1, page 23 of this reply), the NMF solver completes all tasks within 0.5 seconds. This is well within the retention capabilities of RRAM devices. For example, in our previous work [S. Wang *et al.*, *Sci. Adv.* **9**, eadj2908, 2023], retention of eight distinct device states was tested over 10^4 seconds and showed high consistency across time (Fig. R14).

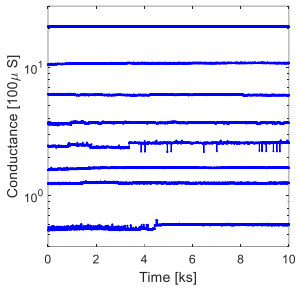

Fig. R14. Retention test of RRAM.

Temperature variations do influence device conductance states. We referenced the temperature model of analog RRAM from [S. Yang *et al.*, *ICICDT 2023*, pp. 141–144]. At high conductance, the device behaves metallically, so increasing temperature decreases conductance. At low conductance, the device behaves semiconductingly, so increasing temperature increases conductance. Similar behavior has been verified in 1-bit RRAM [W. Christian *et al.*, *IEEE Trans. Electron Devices* **58**, 3124–3131, 2011]. The temperature model is expressed as:

$$R = R^0(1 + \rho(T - T_0)) \qquad \text{(R1)}$$

where R is the resistance at temperature T, R^0 is the resistance at room temperature T_0, ρ is the temperature coefficient. The coefficient ρ varies with the initial state (R^0), and we used data reported in the original reference. Fig. R15 shows the values of ρ under different R^0 conditions.

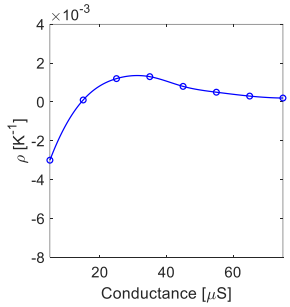

Fig. R15. Relationship between ρ and initial state.

We evaluated the impact of temperature using an image compression task. In SPICE simulations, a 200×66 array was used with ±5% mapping errors relative to the original conductance. A practical AD8572 OPA model (Analog Devices Inc.) with 1.5 MHz GBW was applied. The 200×100×3 nebula image was decomposed into 3 pairs of matrices (200×32 and 100×32) over 2 iteration cycles. Assuming programming at room temperature (20°C), simulations were conducted at 20°C, 40°C, 60°C, 80°C, 100°C, and 120°C. As shown in Fig. R16a, the image was successfully reconstructed

within two cycles across all tested temperatures, demonstrating robust stability. The corresponding NMSE and PSNR values (Fig. R16b) further confirm that temperature variations have only a limited effect on final accuracy, owing to the strong error tolerance of the ANLS algorithm to mapping variations.

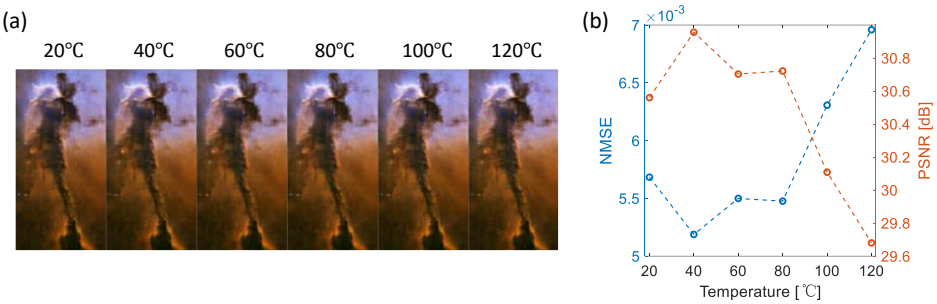

Fig. R16. (a) The reconstructed image under various operation temperatures. (b) NMSE and PSNR under different temperatures.

In the revised manuscript, we have supplemented this analysis to the "Analysis of error tolerance" section (line 357) and updated Fig. S16.

Q5:

As shown in Fig. S3, some devices fail to reach the target conductance even after the large number of programming cycles. Does this result in any performance degradation in the NMF process, and if so, how is this issue addressed or mitigated at the system level?

Reply:

We thank the Reviewer for considering the impact of devices that may fail to the targets. In advanced, factory-ready RRAM technology, it demonstrates near 100% yield and the reliable programming results [W. Wan *et al*. *Nature* **608**, 2022; Wang, Q. *et al. IEDM 2023*, pp. 1–4], which can efficiently avoid this issue.

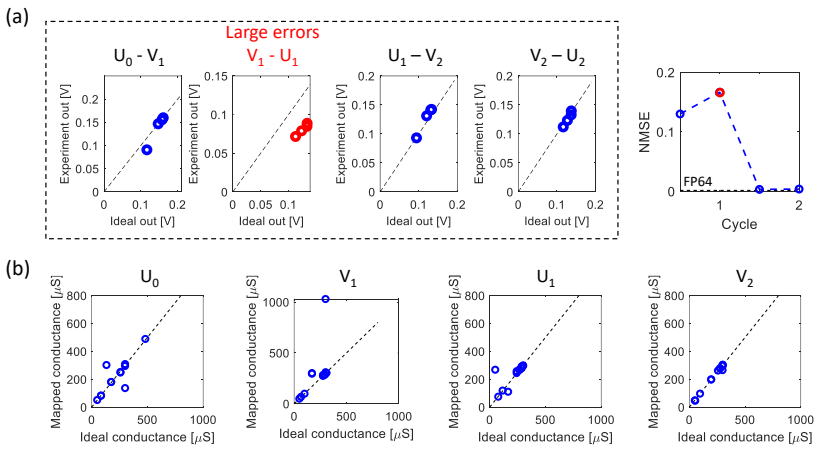

Fig. R17. (a) Experimental validation of ANLS error tolerance (b) Programming results in experiment.

Even if significant programming errors occur, the impact on final results can be reduced owing to the robust error tolerance. The ANLS algorithm reformulates the non-convex NMF problem into multiple convex subproblems (*i.e.*, NLR, NRR, NCRR), enabling efficient convergence. A substantial basin of attraction often exists around the global optimal solution [Y. Chi *et al*., *IEEE Trans. Signal Process* **67**, 5239–5269, 2019], meaning that a large set of initial points can still converge to the optimum. As a result, even when significant computing errors occur in one iteration, subsequent iterations can still drive the solution toward the optimal region with high probability.
In our analog NMF solver, this robustness ensures that occasional programming failures do not critically degrade performance. For example, Fig. R17a shows a case where large programming errors (Fig. R17b) in the transition from $\boldsymbol{V}_1$ to $\boldsymbol{U}_1$ initially result in a high NMSE. Nevertheless, subsequent iterations reduce the NMSE, converging toward the FP64 reference solution. This algorithmic tolerance allows occasional programming failures to be effectively mitigated by only moderately increasing the number of iterations (*e.g.*, by half a cycle or one additional cycle).

In the revised manuscript, we have added this study of error tolerance in the "Analysis of error tolerance" section (line 337) and included Fig. S14.

Reply Note S1:

As shown in Fig. RS1, we illustrate a multi-core architecture designed to solve the NMF problem using the ANLS method. This architecture consists of 1108 cores, consistent with the PUMA architecture [A. Ankit *et al.*, *ASPLOS 2019*, pp. 715–731]. For large-scale recommender system datasets such as Netflix and Yahoo Music, this architecture can operate at full capacity with all 1108 cores.

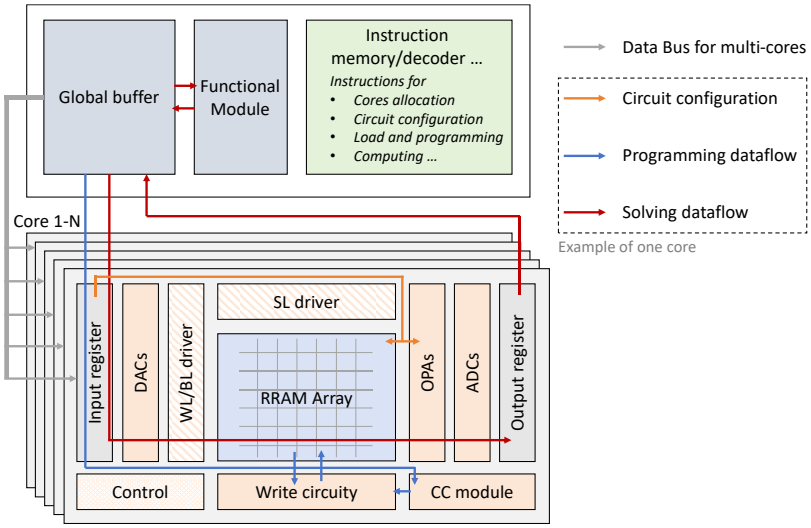

Fig. RS1. Multicore architecture and data flow.

Overall Structure:

Global Buffer: The global buffer stores the matrix to be decomposed and the latent feature matrices during ANLS cycles, such as $\boldsymbol{U}_1$, $\boldsymbol{V}_1$. It is also connected to the cores via a shared data bus.

Functional Module: The functional module contains logic units for nonlinear functions, such as the comparators used in MS operations. It includes 1107 8-bit digital comparators for selecting the maximum value from the outputs of the 1108 cores.

Instruction Module: The instruction module is used to store and translate system instructions.

Core Structure:

Input and Output Registers: Each core has input and an output registers. The input register receives data from the global buffer, including circuit configuration instructions, target conductance for RRAM array, and input voltages. The output register is connected to the ADC for capturing outputs generated by the compact-GINV circuit.

RRAM Array: Each core contains a 1T1R RRAM array of size 256×66. Thus, a single core can implement up to two 256×32 matrices, with the remaining 256×2 section used for compensation conductance.

OPAs: Each core integrates 32 OPAs. Together with the RRAM array, they form the configurable compact-GINV circuit. Circuit configuration instructions generate control signals that specify whether the circuit performs NLR, NRR, or NCRR operations.

DAC and ADC: The DAC is used to generate input voltages for compact-GINV circuit, and the ADC is used for data readout.

SL Driver and WL/BL Driver: These drivers manage the access and operations of the RRAM

array.

Control module: The control module provides logic to manage the RRAM array and other core components.

Write circuitry: It executes the write-verify algorithm for RRAM programming.

CC Module: It is used to calculate values of compensation conductance, implemented as an adder tree (255 adders) [H. Naseri *et al*., *IEEE Trans. VLSI Syst.* **26**, 1481–1493, 2018].

This analog NMF solver is assumed operated at a 200 MHz system clock to ensure efficient timing granularity for mixed-signal data conversion [L. Pan *et al*., *DATE 2025*, pp. 1-2.]. The bandwidth between the SRAM-based global buffer and cores is set to 4 Kb/clock, consistent with [W. -H. Huang *et al*., *ISSCC 2023*, pp. 15-17.]. Module design parameters largely reference the PUMA architecture [A. Ankit *et al*., *ASPLOS 2019*, pp. 715–731.], based on 32 nm CMOS node. OPA and ADC specifications are drawn from [S. Wang *et al*., *Sci. Adv.* **9**, eadj2908, 2023] and [Kull, L. *et al*., *IEEE J. Solid-State Circuits* **48**, 3049–3058, 2013], respectively. A 3-bit R-2R DAC was designed for evaluation (Fig. RS2), with measured power consumption of 301 uW. Module parameters are summarized in Table RS1.

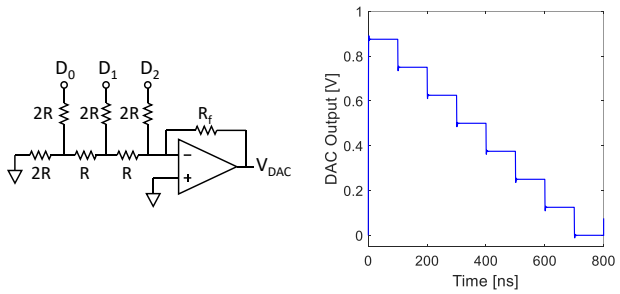

Fig. RS2. 3-bit DAC design and its simulation results.

Component	Power (mW)	Parameter	Specification
FM	3.73	# comparators	1107
Global buffer	9.14	Bandwidth	4 Kb/clock
Input register	0.12	Capacity	2 Kb
Output register	0.015	Capacity	0.25 Kb
DAC	0.3	Latency	10 ns
ADC	2.2	Latency	10 ns
OPA	0.224	GBW	1 GHz
RRAM array	4.2	Capacity	256x66
CC module	0.011	# adders	63
Write circuity	3.32	Parallel degree	8

Table. RS1. The applied parameters in architecture.

The execution sequence for a single core (Fig. RS1) begins with GINV-circuit configuration (yellow lines), which sets the operation mode (NLR, NRR or NCRR) of compact-GINV circuit. In the programming dataflow (blue lines), $\boldsymbol{U}_i/\boldsymbol{V}_i$ data is transferred from the global buffer to the input registers. The write circuitry then performs the write-verify process for array programming. In

parallel, the CC module calculates the compensation conductance, which is programmed via the write circuitry. In the solving dataflow (red lines), input data (*e.g.*, known ratings, 3 bits per value) is supplied from the global buffer to the input registers, delivered through DACs to the compact-GINV circuit. ADCs digitize the outputs, which are temporarily stored in output registers and then returned to the global buffer. The functional module subsequently performs nonlinear operations (MS operation) in a pipelined manner.

As illustrated in Fig. RS3, externally provided parameters-including matrix dimensions for decomposition, number of latent features, and block size are compiled into instructions that govern chip operation. The number of cores is first determined based on matrix dimensions and block size (via core allocation instructions). Next, hardware configurations are applied to each core's compact-GINV circuit (via configuration instructions), specifying activated row/column counts and computing modes (NLR, NRR, or NCRR). Finally, dataflow control instructions (for load/programming and computing) manage the overall ANLS process.

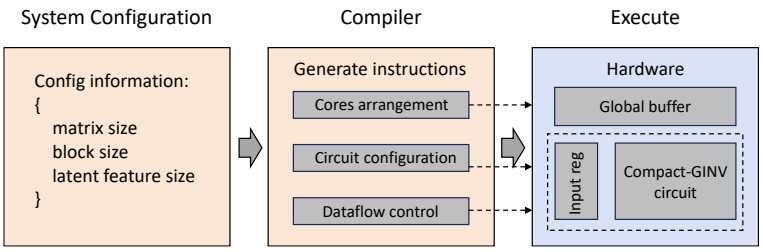

Fig. RS3. The execution process: from source code to hardware execution.

Reviewer #4:

This paper proposes a memory-efficient computational scheme for accelerating non-negative matrix factorization (NMF). The authors designed a novel, compact, and reconfigurable generalized inverse (GINV) circuit and combined it with the alternating non-negative least squares (ANLS) algorithm, successfully demonstrating its application in image compression and recommendation systems on hardware. The experimental results, particularly the orders-of-magnitude improvements in speed and energy efficiency compared to FPGA and GPU shows its potential in real life applications.

However, in order to further enhance the rigor and persuasiveness of the manuscript, there are several key issues that need to be clarified and discussed in greater depth by the authors. Detailed comments are listed below.

Reply:

We thank the Reviewer for the positive comments, and for his/her insightful suggestions for improving our work. We have addressed all the concerns below and made corresponding revisions to the manuscript.

Q1:

The paper mentions that the ANLS algorithm has error tolerance, which is cited as a key factor supporting the feasibility of the analog computing scheme. However, the discussion in the paper is rather general, and this claim requires more quantitative analysis to support it. For example, how do device-to-device variations in resistive memory arrays, limited programming precision, and the non-ideal characteristics of operational amplifiers (such as limited gain, bandwidth, and noise) specifically affect the convergence process and final accuracy of NMF? It is recommended that the authors supplement the relevant data, such as injecting different levels of noise into the analog calculations and observing the changes in the NMSE or PSNR of the final decomposition results, to clarify the robust boundaries of this scheme.

Reply:

We thank the Reviewer for the helpful suggestion to provide additional simulations and data on the error tolerance of the ANLS algorithm. Using image compression as an example, we evaluated the performance of the NMF solver under varying noise and error conditions.

In SPICE simulations, we considered different device programming errors ranging from ±1% to ±60%. To account for the non-ideal characteristics of OPAs, we used the commercial OPA model AD8572 from ADI (1.5 MHz bandwidth, 145 dB DC open-loop gain, 51 nV/$\sqrt{\text{Hz}}$ voltage noise). A 200×66 RRAM array was employed to decompose the 200×100×3 nebula image into 3 pairs of matrices (200×32 and 100×32), executed over 2 cycles.

The simulation results are shown in Fig. R18a. Across all tested programming error levels, the image was accurately reconstructed, indicating minimal impact on convergence. Even under extreme programming errors (±50% and ±60%), convergence was achieved within 2–3 cycles (Fig. R18b). The final NMSE and PSNR values for different error levels are also presented in Fig. R18a,

demonstrating the strong error tolerance of the ANLS algorithm. Within ±20% programming error, the effect on NMSE and PSNR is negligible. Remarkably, even at ±60% programming error, the image could still be successfully recovered.

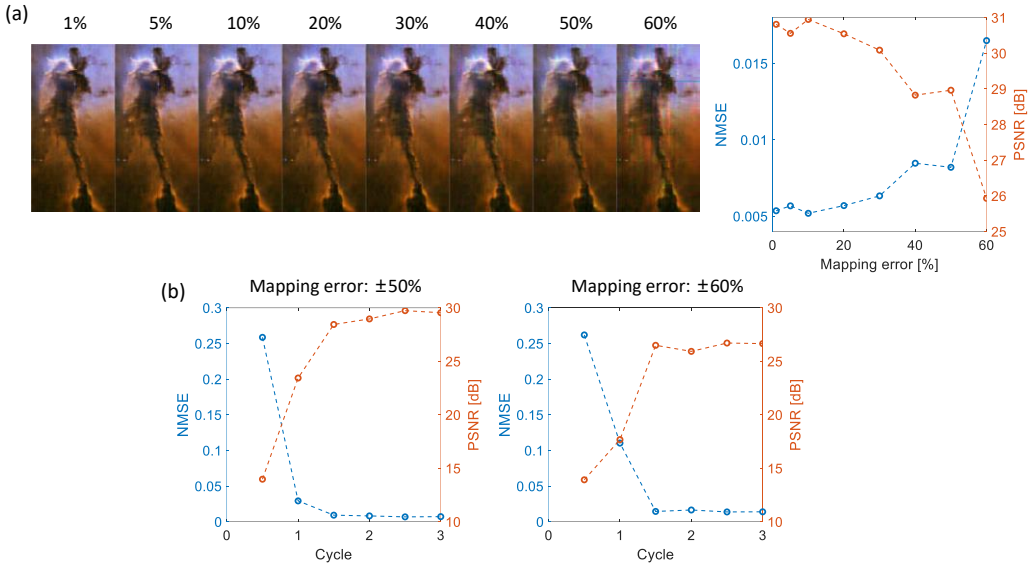

Fig. R18. (a) Recovered results under different programming error. (b) The convergence process under ±50% and ±60% programming error.

In the revised manuscript, we have supplemented these data into the "Analysis of error tolerance" section (line 337) and added Fig. S15.

Q2:

On the rigor and cost of the block matrix-based method: To process large-scale datasets (such as MovieLens 100k) on limited hardware, the authors propose a "block matrix- based method." This might be a practical engineering solution, but its theoretical rigor requires further clarification. The description in the paper, particularly the steps involving "maximization and scaling" to merge intermediate results, appears to have a heuristic nature.

***Question 1:** To what extent does this approximate processing method alter the convergence properties of the original ANLS algorithm?*

***Question 2:** How sensitive is the accuracy of the final results to the choice of block size d?*

***Question 3:** The scaling factor is described as "empirically chosen to be 1.15," which undermines the method's universality. The authors should provide a more robust justification for selecting this value or discuss an adaptive method for determining it.*

Reply:

We sincerely thank the Reviewer for raising this important concern regarding the theoretical foundation of our block matrix-based method. To address this, we conducted additional numerical analyses to validate the approach.

In recommender systems, each computing corresponds to a ridge regression based on observed data (NCRR). Therefore, we first validated the maximization and scaling operations of the block method

using simple ridge regression problems. As shown in Fig. R19, a dataset of N=20 points (x_i, y_i) was randomly partitioned into subsets with size $\leq d$ points (where d=4). For each subset, ridge regression with λ=0.2 was performed, and the maximum slope is recorded as w_{max}. The slope obtained from the full dataset is denoted as w_{all}. The ratios w_{max}/w_{all} for the four different partitions are approximately the same, at around 1.03. This value is used as the scaling factor in the MS operation.

Then, we designed a more comprehensive numerical experiment to reflect the statistical characteristics:

1. Three datasets of $N = 5000$ points were generated (Fig. R20a): (1) $y_i = 2x_i + 2 + \varepsilon_i$ with $\varepsilon_i \in [-1,1]$; (2) $y_i = \text{ceil}(5x_i)$; (3) $y_i = \text{ceil}(5x_i) + \varepsilon_i$ with $\varepsilon_i \in [-0.5,0.5]$. Dataset (ii) and (iii) mimic discrete user ratings in recommender systems.
2. Each dataset was randomly partitioned into small blocks ($\leq d$).
3. Ridge regression (λ=0.2) was applied to each block to obtain a slope coefficient w. The maximum value w_{max} was recorded.
4. Steps 2–3 were repeated 1000 times to examine the relationship between w_{max} and w_{all}.

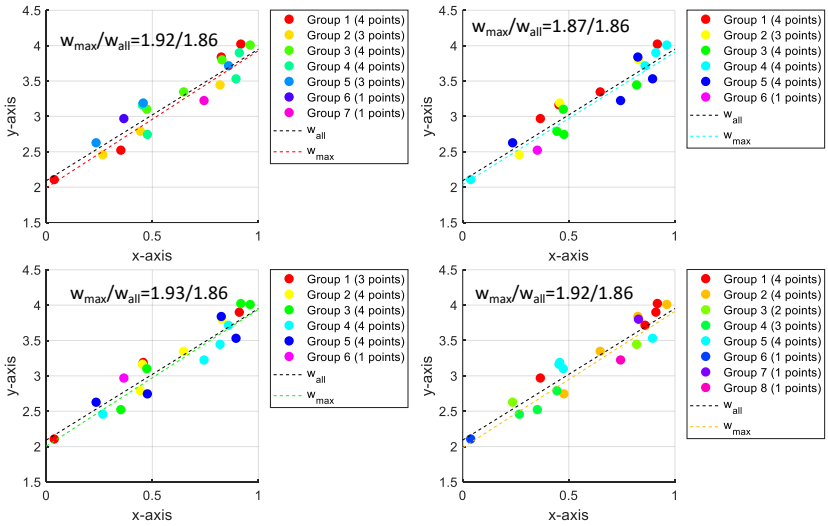

Fig. R19. Illustration of randomly partitions in 20 points of (x, y).

The simulations (Fig. R20b) show that w_{max}/w_{all} (the scaling factor) maintains a low variance across a range of database sizes (N=300–30,000) and block sizes (d=4–128), suggesting that the values obtained from various partitions are predictable. This variance, measured as the coefficient of variation (standard deviation/mean) is consistently below 6% (Fig. R20c). Increasing dataset size narrows the distribution of w_{max}/w_{all}, which further improves decomposition accuracy in large-scale settings. Furthermore, a logarithmic relationship exists between the scaling factor and the dataset size N. This enables a practical estimation method: the factor can be computed on small, randomly selected subsets and then extrapolated to full datasets using this log-linear relationship.

We fully acknowledge that the method retains heuristic elements, particularly in deriving the scaling factor and its dependence on data distribution. A rigorous theoretical analysis will be the subject of

future work.

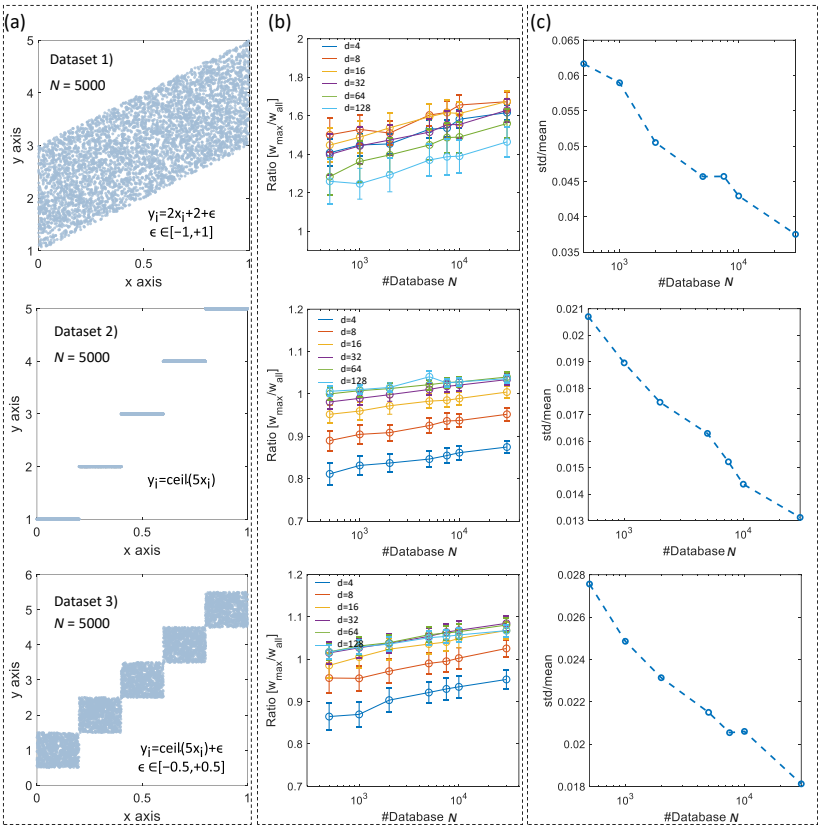

Fig. R20. (a) The three databases with $N = 5000$. (b) The relationship between scaling factor (w_{max}/w_{all}) and the size of database (N). The error bar is also shown. (c) Over 1000 randomly cycles of partitioning, the ratio between standard deviation and average of the scaling factor (w_{max}/w_{all}) under various sizes of database (N). The data are from Fig. 20b and the average value of all block size d is shown here.

Question 1: Convergence Properties

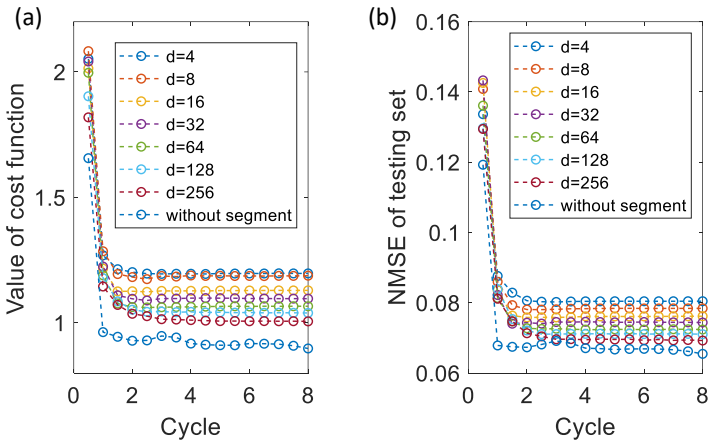

Fig. R21. The scaling behavior of block matrix method. (a) Value of cost function in training set (b) NMSE of testing set.

On the MovieLens 100k dataset, we compared convergence curves of the block method (with varying *d*) against the original unsegmented ANLS algorithm. All methods converged within 3 cycles (Fig. R21), demonstrating that the block method preserves the convergence behavior of ANLS. Larger block sizes also improved solution quality, likely because they capture more global information.

Question 2: Sensitivity to Block Size *d*

We evaluated the block method with *d*=4–256 on MovieLens 100k. For each case, the empirically optimal scaling factor was applied (derived via the log(*N*)-based strategy). The results confirm that larger blocks yield better solution quality (Fig. R21).

Question 3: Scaling Factor Justification

The scaling factor of 1.15 in the MovieLens 100k experiment is dataset-specific, not a universal preset. For *d*=4, we estimated this factor by systematic subsampling and extrapolation. Submatrices of sizes 10×20, 32×64, and 100×200 were randomly sampled from the full dataset, and their optimal scaling factors were calculated. Extrapolation via the log(*N*) relationship yielded a factor of ~1.15 for the full dataset (943 × 1682) (Fig. R22).

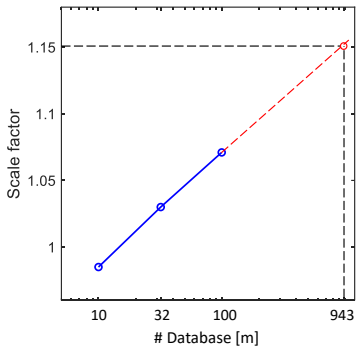

Fig. R22. The extrapolated results of MovieLens 100K datasets (*d*=4).

In the revised manuscript, we have added these validations of the block matrix-based method and the scaling factor strategy in Supplementary Note S3, and referenced them in the main text (line 285). The impact of block size *d* on convergence speed and solution quality has also been added (line 321).

Q3:

The Gap Between Experimental Scale and Performance Extrapolation: The hardware experiments in this paper were conducted on a very small scale (e.g., a 4x5 RRAM array). The remarkable performance data presented in the paper (e.g., hundreds of times faster than GPUs) is based on theoretical extrapolation and estimation from these small-scale experimental results. However, there is typically a significant gap between performance predictions from small-scale physical demonstrations and large-scale applications. The authors should discuss the new challenges that may arise when scaling up, such as long-line delays, power consumption distribution in larger arrays, and the complexity of controlling thousands of ADCs/DACs. While an estimation model is provided in Supplementary Note S4, discussing these potential challenges would make the

performance predictions more persuasive.

Reply:

We appreciate the Reviewer for reminding us about the gap between small-scale experiments and large-scale performance evaluations. To strengthen the persuasiveness of our assessment, we designed a multi-core architecture for the analog NMF solver (detailed in Reply Note R1, page 37 of this reply). Based on this architecture, we benchmarked our system against state-of-the-art FPGA (AMD Versal™ AI Core, VC2802) and GPU (NVIDIA H100) platforms (Fig. R23). For the Movielens 100K dataset (using 7 cores), our solver achieves a 212× speedup and 4.6×10^4 improvement of energy efficiency compared to SOTA FPGA. Using the same number of cores as the PUMA architecture (1108 cores), the analog NMF solver achieved the following: for the Netflix dataset, a 9.3×10^3 speedup and 2.1×10^4 improvement in energy efficiency compared to FPGA, and a 12.4× speedup with 280.9× higher energy efficiency compared to GPU. For the Yahoo dataset, it delivered a 7.0×10^3 speedup and 6.9×10^3 higher energy efficiency over FPGA, and a 3.3× speedup with 34.2× higher energy efficiency versus GPU.

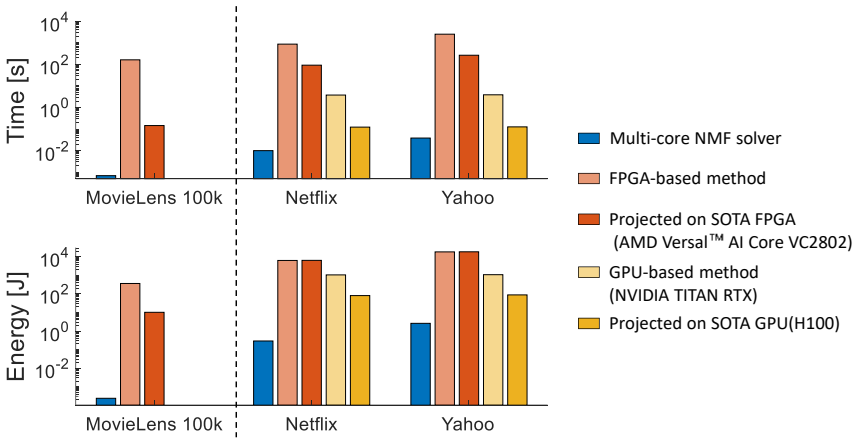

Fig. R23. Updated benchmark based on multi-core NMF architecture.

Each core incorporates a large-scale RRAM array (256×66). When evaluating the solving time of the compact-GINV circuit, we explicitly accounted for long-wire delays in the RRAM array. As shown in Fig. R24a, parasitic capacitances (7.5 fF per row, and 30 fF per column) were included in SPICE simulations [Y. Zhang *et al*., *IEEE Electron Device Lett*. **43**, 1203-1206, 2022]. For the same

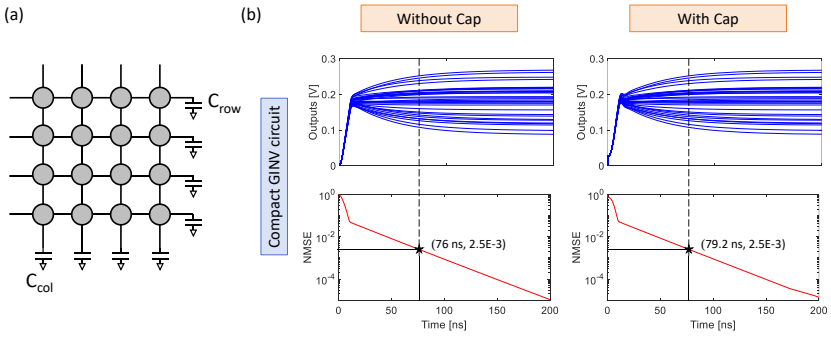

Fig. R24. (a) Model of long-wire delay. (b) Impact of long-wire delay.

NRR/NCRR problem, the solving time increased from 76 ns to 79.2 ns (Fig. R24b), and this effect was incorporated into our system-level performance evaluation.

Regarding power consumption in large arrays, our comprehensive analysis based on the multi-core architecture (detailed in Reply Note R1, page 36 of this reply) confirms that ADC/DAC components still dominate total power consumption (Fig. R25), while the RRAM array itself consumes less than 1%.

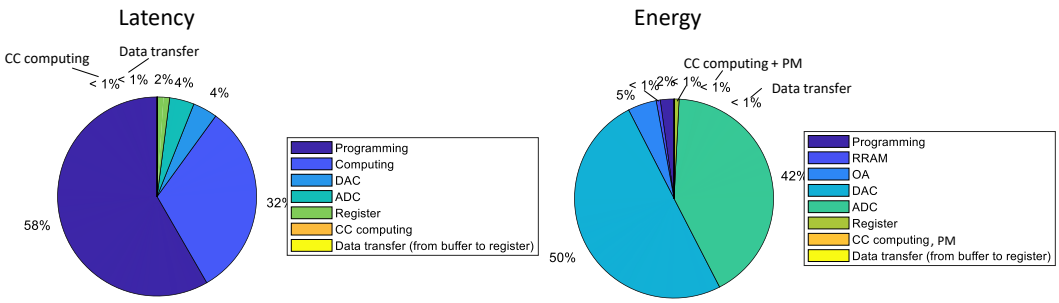

Fig. R25. Updated breakdown of latency and energy.

In our design, each core integrates 256 DACs and 32 ADCs. To mitigate the complexity of scaling to thousands of converters, a common strategy is to reduce the number of ADCs through multiplexing (MUX) and resource sharing, at the cost of some computing throughput. Such approaches have been successfully demonstrated in recent works [W. Wan *et al*., *Nature* **608**, 504-512, 2022; W.-S. Khwa *et al*., *Nature* **639**, 617-623, 2025].

In the revised manuscript, we have added the multi-core architecture in Supplementary Note S4 and referred it in line 375 of the main text, and updated benchmark results in Fig. S19 and Fig. S20. We have referenced the benchmark results in the main text (line 404) as following:
"Task of Movilens 100k occupies 7 cores and achieves a 212× speedup and 4.6×10^4 improvement of energy efficiency compared to SOTA FPGA. In tasks of Netflix and Yahoo, the proposed architecture operates with full 1108 cores and it achieves a speedup of thousands of times compared to SOTA FPGA and several times compared to H100 GPU on average (Fig. S19). Additionally, it demonstrates thousands of times improvement in energy efficiency over SOTA FPGA and hundreds of times over H100 GPU on average (Fig. S19). The assessment reveals that latency is primarily dominated by device programming (58%, Fig. S20), while power consumption is largely attributed to DACs and ADCs (92%, Fig. S20) in the analog NMF solver."

The impact of long-wire delays and the limitations of ADC scaling have been discussed in the Discussion section (line 385) and illustrated in Fig. S18.

Q4:

The paper mentions that the compensation conductance is "pre-computed with minimal computational overhead." However, in each iteration of ANLS, the matrices U and V are constantly updated, which means that CC also needs to be frequently re-computed and re-programmed. Is this overhead still "minimal" when dealing with large-scale, rapidly changing matrices? It is recommended that the authors quantify the computational and time overhead of this part and include

it in the overall performance evaluation.

Reply:

We thank the Reviewer for reminding us to evaluate the overhead of re-computing and re-programming the compensation conductance (CC). In our performance evaluation, we designed the multi-core architecture to explicitly include this overhead. The CC computation module is implemented as an adder tree with 63 adders [H. Naseri *et al*., *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*. **26**, 1481–1493, 2018], which sums 64 conductance values within 6 clock cycles and consumes 0.19 mW of power. To reduce the mapped conductance values, the compensation conductance column is split into two parallel columns. Accordingly, the array size is set to 256×66, with two additional columns allocated for compensation conductance.

Based on the designed multi-core architecture (detailed in Reply Note S1, page 36 of this reply), the system-level time and energy breakdown are illustrated in Fig. R25. he computation of CC itself occupies only ~0.05% of the total time and energy. Even when including the overhead for programming the two CC columns, it accounts for only 3.47% of the total time and 0.12% of the total energy. These results confirm that the additional overhead remains minimal even for large-scale, iterative ANLS processes.

In the revised manuscript, this analysis has been added to the Discussion section (line 391).

Q5:

Supplementary Figure S12 shows that the energy consumption of the ADC/DAC accounts for 98% of the total energy consumption, which is a significant bottleneck. The authors propose in the discussion section that using a low-precision DAC is feasible. This is a good point, but it lacks data support. It is recommended to supplement the relevant data to demonstrate the impact of using ADC/DACs with different bit widths (e.g., 3-bit, 4-bit) on the final NMF decomposition accuracy, which would strongly support their argument regarding energy efficiency optimization.

Reply:

We thank the Reviewer for the suggestion to evaluate the use of low-precision DACs/ADCs. In applications such as recommendation systems (*e.g.*, MovieLens and Yahoo Music datasets), the rating matrices consist of discrete values ranging from 1 to 5. These ratings are directly used as circuit input voltages, making it feasible to employ low-precision DACs at the input stage. The

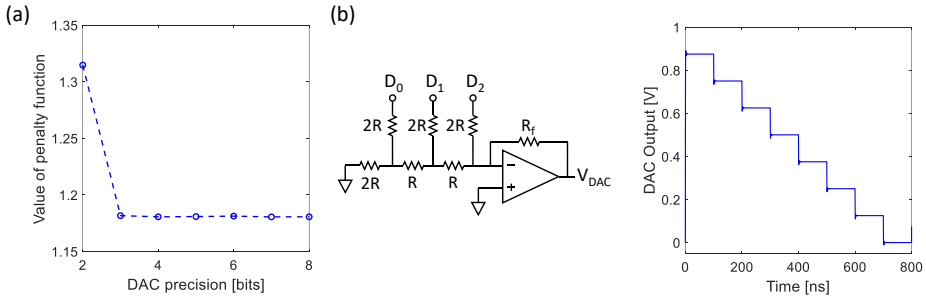

Fig. R26. (a) Impact of DAC precision on recommender system. (b) The 3-bit DAC and its simulation results.

circuit outputs, however, represent latent user/item features that are not discretized; thus, we retain the original 8-bit ADC for the output stage. Since output voltages determine the target conductance in the next cycle and device programming inherently has limited precision, a higher-precision ADC would not provide additional benefit.

To quantify the impact, we simulated the MovieLens 100K dataset using DACs of different bit-widths (2–8 bits) and evaluated the variation in the final cost function. As shown in Fig. R26a, the 3-bit DAC offers the best trade-off between accuracy and hardware efficiency. We then designed a 3-bit DAC based on an R-2R ladder structure and incorporated it into the performance evaluation of our multi-core architecture (detailed in Reply Note S1, page 36 of this reply). The design and corresponding energy analysis are illustrated in Fig. R26b.

In the revised manuscript, these results have been added in Fig. S17 and referenced in the Discussion section (line 379).

Q6:

In the performance comparison with GPUs, the authors normalized GPU performance to a "single core." Although the authors provided justification for this, it may be controversial because matrix decomposition algorithms running on modern GPUs are highly parallelized and optimized to leverage the collaborative work of thousands of cores. If possible, the performance of the entire simulation system should be compared with the most advanced NMF algorithm implementation running on the entire GPU, or the limitations of this comparison method should be more clearly articulated in the discussion. Similar issues arise with FPGA comparisons, such as whether the comparison is against the best available FPGA, which can significantly impact the comparison results.

Reply:

We thank the Reviewer for the helpful suggestion regarding benchmarking against state-of-the-art platforms. For GPUs, the NVIDIA TITAN RTX (130 TOPs, INT 8) was the most advanced GPU we could access for solving NMF problems. To ensure fairness, we extrapolated to the current state-of-the-art GPU platform (NVIDIA H100, 4000TOPs, INT8) by normalizing performance according to peak computational capability ($T_{H100} = 130/4000 \cdot T_{TITAN}$). For FPGAs, we initially considered the AMD Virtex UltraScale+ VU9, but in line with the Reviewer's comment, we also included the state-of-the-art AMD Versal™ AI Core (VC2802), with performance estimated by scaling from its reported peak computing capability.

Following the Reviewer's suggestion, we further designed a multi-core architecture for our analog NMF solver (described in Reply Note S1, page 36 of this reply). Taking into account the primary sources of latency and power consumption in such architectures, we performed a more comprehensive system-level performance evaluation. We considered 1108 cores, consistent with the PUMA architecture [A. Ankit *et al*., *ASPLOS* 2019].

The evaluation results are shown in Fig. R23. For the Movielens 100K dataset, our solver achieves a 212× speedup and 4.6×10^4 improvement of energy efficiency compared to AMD Versal™ AI Core (VC2802). For the Netflix dataset, our solution achieves a 12.4× speedup and 280.9×

improvement in energy efficiency compared with the H100 (14,592 CUDA cores), and a 9.3×10^3 speedup with a 2.1×10^4 improvement in energy efficiency compared with the AMD Versal™ AI Core (VC2802). For the Yahoo Music dataset, our solution demonstrates a 3.3× speedup and 34.2× energy efficiency gain over the H100, and a 7.0×10^3 speedup with a 6.9×10^3 improvement in energy efficiency relative to the VC2802. The updated time and energy breakdowns are shown in Fig. R25.

In the revised manuscript, the multi-core architecture has been added as Supplementary Note S4 and referred in line 375 of the main text, and the updated benchmarking results are included in Fig. S19, Fig. S20. We have referenced the benchmark results in the main text (line 404) as following:
"Task of Movilens 100k occupies 7 cores and achieves a 212× speedup and 4.6×10^4 improvement of energy efficiency compared to SOTA FPGA. In tasks of Netflix and Yahoo, the proposed architecture operates with full 1108 cores and it achieves a speedup of thousands of times compared to SOTA FPGA and several times compared to H100 GPU on average (Fig. S19). Additionally, it demonstrates thousands of times improvement in energy efficiency over SOTA FPGA and hundreds of times over H100 GPU on average (Fig. S19). The assessment reveals that latency is primarily dominated by device programming (58%, Fig. S20), while power consumption is largely attributed to DACs and ADCs (92%, Fig. S20) in the analog NMF solver."

Reply Note S1:

As shown in Fig. RS1, we illustrate a multi-core architecture designed to solve the NMF problem using the ANLS method. This architecture consists of 1108 cores, consistent with the PUMA architecture [A. Ankit *et al*., *ASPLOS 2019*, pp. 715–731]. For large-scale recommender system datasets such as Netflix and Yahoo Music, this architecture can operate at full capacity with all 1108 cores.

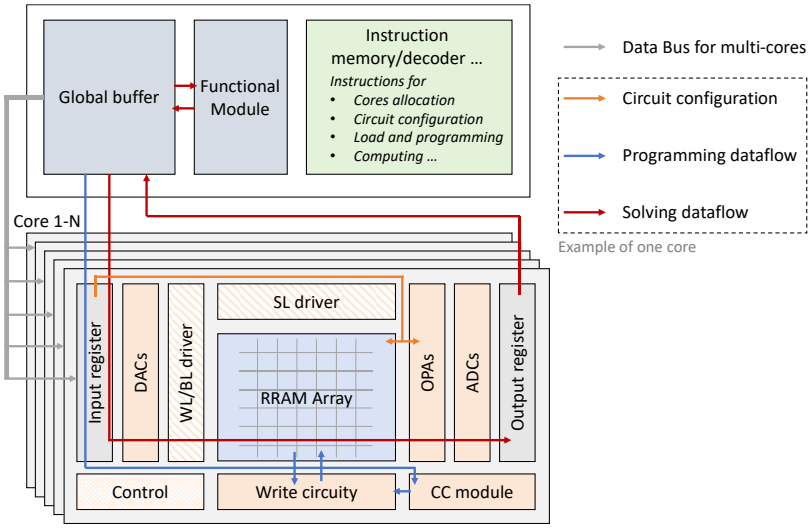

Fig. RS1. Multicore architecture and data flow.

Overall Structure:

Global Buffer: The global buffer stores the matrix to be decomposed and the latent feature matrices during ANLS cycles, such as $\boldsymbol{U}_1$, $\boldsymbol{V}_1$. It is also connected to the cores via a shared data bus.

Functional Module: The functional module contains logic units for nonlinear functions, such as the comparators used in MS operations. It includes 1107 8-bit digital comparators for selecting the maximum value from the outputs of the 1108 cores.

Instruction Module: The instruction module is used to store and translate system instructions.

Core Structure:

Input and Output Registers: Each core has input and an output registers. The input register receives data from the global buffer, including circuit configuration instructions, target conductance for RRAM array, and input voltages. The output register is connected to the ADC for capturing outputs generated by the compact-GINV circuit.

RRAM Array: Each core contains a 1T1R RRAM array of size 256×66. Thus, a single core can implement up to two 256×32 matrices, with the remaining 256×2 section used for compensation conductance.

OPAs: Each core integrates 32 OPAs. Together with the RRAM array, they form the configurable compact-GINV circuit. Circuit configuration instructions generate control signals that specify whether the circuit performs NLR, NRR, or NCRR operations.

DAC and ADC: The DAC is used to generate input voltages for compact-GINV circuit, and the ADC is used for data readout.

SL Driver and WL/BL Driver: These drivers manage the access and operations of the RRAM

array.

Control module: The control module provides logic to manage the RRAM array and other core components.

Write circuitry: It executes the write-verify algorithm for RRAM programming.

CC Module: It is used to calculate values of compensation conductance, implemented as an adder tree (255 adders) [H. Naseri *et al*., *IEEE Trans. VLSI Syst.* **26**, 1481–1493, 2018].

This analog NMF solver is assumed operated at a 200 MHz system clock to ensure efficient timing granularity for mixed-signal data conversion [L. Pan *et al*., *DATE 2025*, pp. 1-2.]. The bandwidth between the SRAM-based global buffer and cores is set to 4 Kb/clock, consistent with [W. -H. Huang *et al*., *ISSCC 2023*, pp. 15-17.]. Module design parameters largely reference the PUMA architecture [A. Ankit *et al*., *ASPLOS 2019*, pp. 715–731.], based on 32 nm CMOS node. OPA and ADC specifications are drawn from [S. Wang *et al*., *Sci. Adv.* **9**, eadj2908, 2023] and [Kull, L. *et al*., *IEEE J. Solid-State Circuits* **48**, 3049–3058, 2013], respectively. A 3-bit R-2R DAC was designed for evaluation (Fig. RS2), with measured power consumption of 301 uW. Module parameters are summarized in Table RS1.

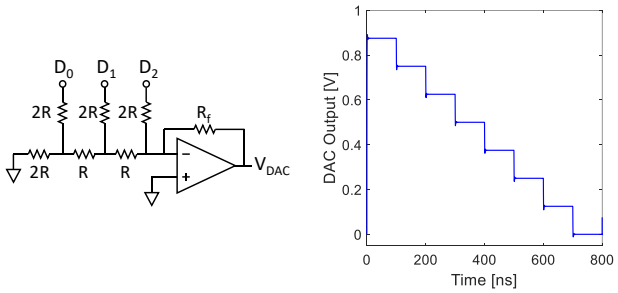

Fig. RS2. 3-bit DAC design and its simulation results.

Component	Power (mW)	Parameter	Specification
FM	3.73	# comparators	1107
Global buffer	9.14	Bandwidth	4 Kb/clock
Input register	0.12	Capacity	2 Kb
Output register	0.015	Capacity	0.25 Kb
DAC	0.3	Latency	10 ns
ADC	2.2	Latency	10 ns
OPA	0.224	GBW	1 GHz
RRAM array	4.2	Capacity	256x66
CC module	0.011	# adders	63
Write circuity	3.32	Parallel degree	8

Table. RS1. The applied parameters in architecture.

The execution sequence for a single core (Fig. RS1) begins with GINV-circuit configuration (yellow lines), which sets the operation mode (NLR, NRR or NCRR) of compact-GINV circuit. In the programming dataflow (blue lines), $\boldsymbol{U}_i/\boldsymbol{V}_i$ data is transferred from the global buffer to the input registers. The write circuitry then performs the write-verify process for array programming. In parallel, the CC module calculates the compensation conductance, which is programmed via the

write circuitry. In the solving dataflow (red lines), input data (*e.g.*, known ratings, 3 bits per value) is supplied from the global buffer to the input registers, delivered through DACs to the compact-GINV circuit. ADCs digitize the outputs, which are temporarily stored in output registers and then returned to the global buffer. The functional module subsequently performs nonlinear operations (MS operation) in a pipelined manner.

As illustrated in Fig. RS3, externally provided parameters-including matrix dimensions for decomposition, number of latent features, and block size are compiled into instructions that govern chip operation. The number of cores is first determined based on matrix dimensions and block size (via core allocation instructions). Next, hardware configurations are applied to each core's compact-GINV circuit (via configuration instructions), specifying activated row/column counts and computing modes (NLR, NRR, or NCRR). Finally, dataflow control instructions (for load/programming and computing) manage the overall ANLS process.

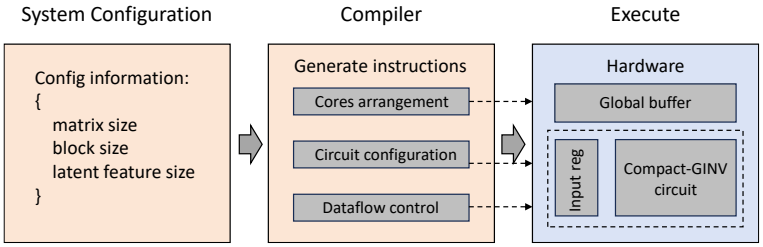

Fig. RS3. The execution process: from source code to hardware execution.

Reviewer #1:

The authors addressed most of my questions and comments. However, there are some points that would benefit from further clarification.

Reply:

We thank the Reviewer for their time and effort in evaluating our manuscript. We appreciate the constructive feedback, which has helped us further improve the quality and clarity of our work. We have carefully addressed all the concerns outlined below and made the corresponding revisions in the manuscript.

Q1:

*I am still unable to follow how the nonnegativity constraints are involved in solving min_{v>=0} ||r-U*v||_2^2 (S5).*

From the supplementary material, it is stated that "The solution v for the GINV is identical to the solution of Equation (S5), which represents the circuit equation of the compact-GINV circuit in this configuration. Notably, the non-negative constraint is enforced through the use of a single supply source, eliminating the possibility of negative outputs".

However, the vector v obtained from (S4) is the standard linear least-squares solution without any nonnegativity constraint. As such, the vector v from the formula (S4) cannot engage the nonnegativity constraint from (S5) on its own. Hence the claim "The solution v for the GINV is identical to the solution of Equation (S5)" appears to be misleading. Are there any structural properties that guarantee the nonnegativity of v?

Reply:

We thank the Reviewer for this careful and constructive comment. Upon re-examining the text regarding Equation (S4), we realized that the sentence "The solution $\boldsymbol{v}$ for the GINV is identical to the solution of Equation (S5)" was inaccurately phrased. In the revised supplementary material, we have removed this sentence and rephrased the surrounding text to better clarify the underlying principles of the compact-GINV circuit.

In Supplementary Note S1, combining Equations (S2) and (S3), the actual circuit equation is $\boldsymbol{U}^{\mathrm{T}}(\boldsymbol{Uv}-\boldsymbol{r})=0$, which corresponds to the derivative of the cost function $\min \|\boldsymbol{r}-\boldsymbol{Uv}\|_2^2$ with respect to $\boldsymbol{v}$. If $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ is invertible, this leads to Equation (S4). Therefore, the compact GINV circuit effectively minimizes the cost function $\min \|\boldsymbol{r}-\boldsymbol{Uv}\|_2^2$ by setting its derivative to zero, which is a convex optimization problem. By powering the operational amplifier with a single supply voltage, negative output values are inherently avoided. This hardware constraint is equivalent to imposing a non-negativity constraint on $\boldsymbol{v}$, transforming the original minimization problem into $\min_{\boldsymbol{v}\geq 0} \|\boldsymbol{r}-\boldsymbol{Uv}\|_2^2$, which corresponds to standard non-negative linear regression (NLR). The functionality of this circuit has been validated experimentally. In this work, the ideal solutions to NLR (computed in FP64 precision using Lagrange multipliers) are compared with extensive circuit experiments, comprising approximately 430,000 compact-GINV operations. The average NMSE between the circuit outputs and the ideal solutions was 0.0072, confirming the circuit's correct functionality.

To prevent any misunderstanding, we have removed the original statement claiming that "The solution v for the GINV is identical to the solution of Equation (S5)" and replaced it with the explanations above. Additionally, Equation (S4) has been modified to $\boldsymbol{U}^{\mathrm{T}}(\boldsymbol{Uv}\text{-}\boldsymbol{r})=0$ in the revised Supplementary Note S1.

Q2:

In Fig2a, the unknowns U, V are given as fixed matrices at each step of obtaining U_i, V_i. At each step, either U_i or V_i need to be updated and `reprogrammed' in the hardware.
Has the overhead for performing this every iteration been analyzed relative to the total runtime (either in wall-clock time or in programming cycles, similar to the pie chart as shown in Fig. S20) with respect to different instances presented in Fig. S7?
If such an analysis is included in the manuscript, could you please point indicate where it is presented?
Additionally, is the programming cycle axis in Fig S4 correspond to the same unit in Fig S7?

Reply:

We thank the Reviewer for the careful attention to programming-related issues. The "programming cycle" in Fig. S4 refers to the number of cycles used to program a single device. A single programming cycle is defined as follows: using the write-verify method (Fig. S3), one voltage pulse is applied, and the device conductance is subsequently read and evaluated once. In contrast, the "iteration cycle" in Fig. S7 refers to the number of cycles of the ANLS algorithm, in which both the $\boldsymbol{U}$ and the $\boldsymbol{V}$ matrices are updated once. Each iteration cycle includes programming the updated $\boldsymbol{U}_{\mathrm{i}}/\boldsymbol{V}_{\mathrm{i+1}}$ matrix into the RRAM array and obtaining the corresponding output voltage (i.e., $\boldsymbol{V}_{\mathrm{i+1}}/\boldsymbol{U}_{\mathrm{i+1}}$) via the GINV circuit. Therefore, the units of the programming cycle in Fig. S4 and the iteration cycle in Fig. S7 are not directly the same.

Fig. S7 presents simulation results, aimed at investigating the convergence behavior of the ANLS algorithm for the image compression task. Specifically, across matrices of different scales, the number of iteration cycles required is roughly similar; no performance evaluation is involved in this figure.

Regarding the overhead of device programming, this has already been accounted for in our benchmark. As shown in Fig. S20, for large-scale recommendation system tasks solved by our analog NMF solver, device programming accounts for 54% of the total runtime but only 2% of the total energy consumption.

To avoid any potential misunderstanding, we have added explicit definitions of programming cycles and iteration cycles in the revised manuscript (lines 119 and 190 in the main text). We have also updated the relevant figures and all associated descriptions of "cycles" throughout the manuscript.

Q3:

How many cycles were required in Fig. S20? In other words, when converted to cycles, how many were needed for each portion of the runtime?

Additionally, how many alternating directions did the ANLS algorithm require to solve the large instance?

Reply:
We thank the Reviewer for the interest in the evaluation details of Fig. S20. The evaluation in Fig. S20 was conducted using the architecture described in Supplementary Note S4, considering the same number of iteration cycles as in the experiment, *i.e.*, 3 cycles. At the system level, a complete iteration cycle includes data transfer, crossbar computation, device programming, GINV circuit convergence, and DA/AD conversion, among other operations.

Specifically, for device programming, we considered 9 programming cycles, consistent with the experimental results shown in Fig. S7. Each programming cycle uses a 50 ns pulse, and the programming parallelism is defined as 8 [Chen, J. *et al*., *IEEE Trans. Electron Devices* **67**, 2020], meaning that 8 devices are programmed simultaneously.

Thanks to the ultra-high parallelism of matrix computations, large-scale recommendation system decomposition tasks generally do not require additional iteration cycles in the ANLS algorithm. For example, in [Chen, J. et al., *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* 2017, 409–418], four large-scale recommendation systems (MovieLens 10M, Netflix, Yahoo Music R1, and Yahoo Music R4) achieved convergence with only 2–3 iteration cycles. Accordingly, in our performance benchmarking, we applied 3 iteration cycles.

In the revised manuscript, we have added more evaluation details in Supplementary Note S4, including the number of iteration cycles for large instances. We have also added explicit definitions of programming cycles (line 190) and iteration cycles (line 119) in the main text.

Minor comments:
Q4:
Related to (S4), the inverse of (U^TU) may not exist, if U is rank deficient.

Reply:
We thank the reviewer for raising this concern. We agree that if the rank of the $\boldsymbol{U}$ ($m\times k$, $m>k$) matrix is less than k, then $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ becomes non-invertible, which could pose a problem in conventional numerical solutions. However, in practical applications such as image compression or recommendation systems, m is typically much larger than k, ensuring that $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ is generally invertible. In Supplementary Note S1, combining Equation (S2) and (S3), the actual circuit equation is $\boldsymbol{U}^{\mathrm{T}}(\boldsymbol{Uv}\text{-}\boldsymbol{r})=0$, which corresponds to the derivative of the cost function $\min \|\boldsymbol{r}\text{-}\boldsymbol{Uv}\|_2^2$ with respect to $\boldsymbol{v}$. If $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ is invertible, this leads to equation (S4). Importantly, the compact-GINV circuit always operates by minimizing the cost function through the derivative, regardless of the invertibility of $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$.

If $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}$ is rank-deficient, there are theoretically infinite solutions to $\boldsymbol{U}^{\mathrm{T}}(\boldsymbol{Uv}\text{-}\boldsymbol{r})=0$. In practice, the compact-GINV circuit naturally selects the solution that minimizes its own power dissipation. Because the bit-line voltage $\mathbf{V}_{\mathrm{BL}}$ ($\mathbf{V}_{\mathrm{BL}}=(\boldsymbol{Uv}\text{-}\boldsymbol{r})/c$) is near zero, the power dissipation is proportional

to $\sum_i(\sum_j v_i^2 U_{ji})$, where v_i is the i^{th} output voltage and the U_{ji} is the conductance of the device at row j row and column i. Thus, when $\boldsymbol{U}^T\boldsymbol{U}$ is non-invertible, the circuit output $\boldsymbol{v}$ satisfies $\boldsymbol{U}^T(\boldsymbol{Uv}-\boldsymbol{r})=0$ and $\min(\sum_i(\sum_j v_i^2 U_{ji}))$.

In the revised manuscript, we have updated Equation (S4) to $\boldsymbol{U}^T(\boldsymbol{Uv}-\boldsymbol{r})=0$ added a statement clarifying the case when $\boldsymbol{U}^T\boldsymbol{U}$ is invertible (Supplementary Note S1).

Q5:

For the Frobenius norms, only using 'F' is sufficient; the usage of 'F=2' seems unnecessary.

Reply:

We thank the Reviewer for this helpful suggestion. We have made the corresponding modification in the revised manuscript.

Q6:

While I agree that the use of "matrix equation" aligns with conventions in this field and covers a broader spectrum, the term itself does not explicitly convey linearity, nor does it indicate that multiple equations are involved, as targeted by this paper. Its usage remains somewhat ambiguous and should be clearly defined upon first introduction.

Reply:

We thank the Reviewer for this helpful suggestion. In the revised manuscript, we clarify that a "matrix equation" is defined as an equation in which the coefficients and unknowns are matrices or vectors (line 46 in the main text).

Q7:

It would be helpful to specify the instances used to generate the histogram presented in Fig. S7.

Reply:

We thank the Reviewer for this helpful suggestion. In the revised manuscript, we have added the data source in the caption of Fig. S7. All instances are derived from the original nebular images. Specifically, matrices of different sizes are extracted from the "Red" channel of the original image. Starting from the top-leftmost element of this channel, we extend downward and rightward to select matrices of sizes 4×4, 8×8, 16×16, 32×32, 64×64, and 100×100, respectively.

Q8:

In Fig. S20, it is unclear what all legend items refer to, particularly what "Computing" represents. Clarification would be helpful.

Reply:

We thank the Reviewer for this helpful suggestion. In the revised manuscript, we have clarified the legend in Fig. S20. The item previously labeled "Computing" refers to the convergence time of the GINV circuit, and it has now been replaced with "GINV circuit" for clarity.

Reviewer #3:

The revised manuscript has addressed the concerns appropriately, and I recommend it for acceptance in Nature Communications.

Reply:

We are delighted that the Reviewer recommends our manuscript for publication and sincerely thank the Reviewer for the insightful and constructive comments.

Reviewer #4:

Thank the authors for their work on the revisions and for the detailed response letter. This revised version has successfully addressed most of the major concerns I raised previously. But I still and one request for clarification on a potential system-level bottleneck and one minor suggestion on presentation.

Reply:

We thank the Reviewer for taking the time and effort to review our manuscript. We greatly appreciate the constructive feedback, which has helped improve the clarity and quality of our work. We have carefully addressed all the concerns outlined below and made the corresponding revisions in the manuscript.

Q1:

Potential Bottleneck in the Block-Matrix Method's Global Aggregation Step: Validation of the block matrix-based method raised in the response introduces a "maximization and scaling" step, which appears to require global data aggregation from all active cores to the central "Functional Module". My concern is whether the system-level performance model (specifically, the latency breakdown in Fig. S20) fully accounts for this potentially costly step. For large-scale problems using hundreds or thousands of cores, this implies a massive data movement and synchronization event. If this latency analysis models the potential bus contention when all cores attempt to write their intermediate results back simultaneously? This step seems like a potential system-level bottleneck that is distinct from the in-core analog computation, and a more detailed analysis would be good to the overall performance claims.

Reply:

We thank the Reviewer for raising this important question. In the architecture of the proposed recommendation system (Supplementary Note S4), the bandwidth has been defined as 4 Kb/clock [W. -H. Huang *et al*., *ISSCC* 2023, pp. 15-17]. We have comprehensively re-evaluated the latency and energy consumption associated with data movement. This evaluation includes not only the output data of the compact-GINV circuit (for M/S operations) but also the programming data ($\boldsymbol{U}_i/\boldsymbol{V}_i$ matrices) and the input data for the compact-GINV circuit ($\boldsymbol{R}$ matrix).

Without considering any pipeline design, we accounted for the sequential movement of data from the registers of all cores to the global buffer and the functional module (FM). We have re-assessed the performance of the analog NMF solver and updated the relevant figures and data (Figs. S19 and S20). In the updated Fig. S20, data movement contributes 8% of the total latency and less than 1% of the total energy consumption. Therefore, data movement does not constitute a bottleneck in this architecture.

In the revised manuscript, we have updated the benchmarking results (Figs. S19 and S20, lines 395 and 413 in the main text) and added detailed descriptions of the data transfer in Supplementary Note S4.

Q2:

Figures: While the content of the figures in the main text is informative, their presentation should be polished for better professional quality. There are some inconsistencies in font types and sizes. Also, some figures are quite text-heavy, which can make readers difficult to get the main idea of the paper and being confused by too many technical details. Authors should consider moving some of them to SI.

Reply:

We thank the Reviewer for pointing out the issues regarding figure presentation, which has helped us improve their quality. In the revised manuscript, we have standardized the font type and size across all figures in the main text and moved most of the text content from Fig. 2b to Fig. S1 in the Supplementary Information.