nature nanotechnology

Article

https://doi.org/10.1038/s41565-023-01357-8

In-memory factorization of holographic perceptual representations

Received: 17 October 2022

Accepted: 21 February 2023

Published online: 30 March 2023

Check for updates

Jovin Langenegger[1,2], Geethan Karunaratne[1,2], Michael Hersche[1,2], Luca Benini[2], Abu Sebastian[1] ✉ & Abbas Rahimi[1] ✉

Disentangling the attributes of a sensory signal is central to sensory perception and cognition and hence is a critical task for future artificial intelligence systems. Here we present a compute engine capable of efficiently factorizing high-dimensional holographic representations of combinations of such attributes, by exploiting the computation-in-superposition capability of brain-inspired hyperdimensional computing, and the intrinsic stochasticity associated with analogue in-memory computing based on nanoscale memristive devices. Such an iterative in-memory factorizer is shown to solve at least five orders of magnitude larger problems that cannot be solved otherwise, as well as substantially lowering the computational time and space complexity. We present a large-scale experimental demonstration of the factorizer by employing two in-memory compute chips based on phase-change memristive devices. The dominant matrix–vector multiplication operations take a constant time, irrespective of the size of the matrix, thus reducing the computational time complexity to merely the number of iterations. Moreover, we experimentally demonstrate the ability to reliably and efficiently factorize visual perceptual representations.

One of the fundamental problems in sensory perception is unbinding—the separation of causes of a raw sensory signal[1] that contain multiple attributes. For instance, the pixel intensities sensed by photoreceptors result from the combination of different physical attributes[1–5]. For example, the observed luminance at a point on the sensor is a multiplicative combination of reflectance and shading[3]. To be able to estimate these constituent factors, visual perception must begin with the observed luminance and solve an inverse problem that involves undoing the multiplication by which the attributes were combined[1,4]. This factorization problem is also at the core of other levels of conceptual hierarchy, such as factoring time-varying pixel data of dynamic scenes into persistent and dynamic components[6–9], factoring a sentence structure into roles and fillers[10,11], and finally cognitive analogical reasoning[12–16]. How these factorization problems could be efficiently solved by biological neural circuits remains unclear. Moreover, given their ubiquitous presence in perception and cognition, it is essential that future artificial intelligence systems are equipped with compute units that can efficiently perform these factorization operations across very large problem sizes.

An elegant mathematical approach to represent the combination of attributes is via high-dimensional holographic vectors in the context of brain-inspired vector symbolic architectures[17–20] (Supplementary Note 1). They are holographic because the encoded information is equally distributed over all the components of the vector. Moreover, any two randomly drawn vectors, by virtue of their higher dimensionality, are almost orthogonal to each other, that is, their expected similarity is close to zero with a higher probability[20]. These vectors can also be manipulated by a rich set of dimensionality-preserving algebraic operations. In one approach, an object with F attributes can be described by the element-wise multiplication of an associated D-dimensional holographic bipolar ($\{-1, +1\}^D$) vector corresponding

[1]IBM Research—Zurich, Rüschlikon, Switzerland. [2]Department of Information Technology and Electrical Engineering, ETH Zürich, Zürich, Switzerland. ✉e-mail: ase@zurich.ibm.com; abr@zurich.ibm.com

https://doi.org/10.1038/s41565-023-01357-8

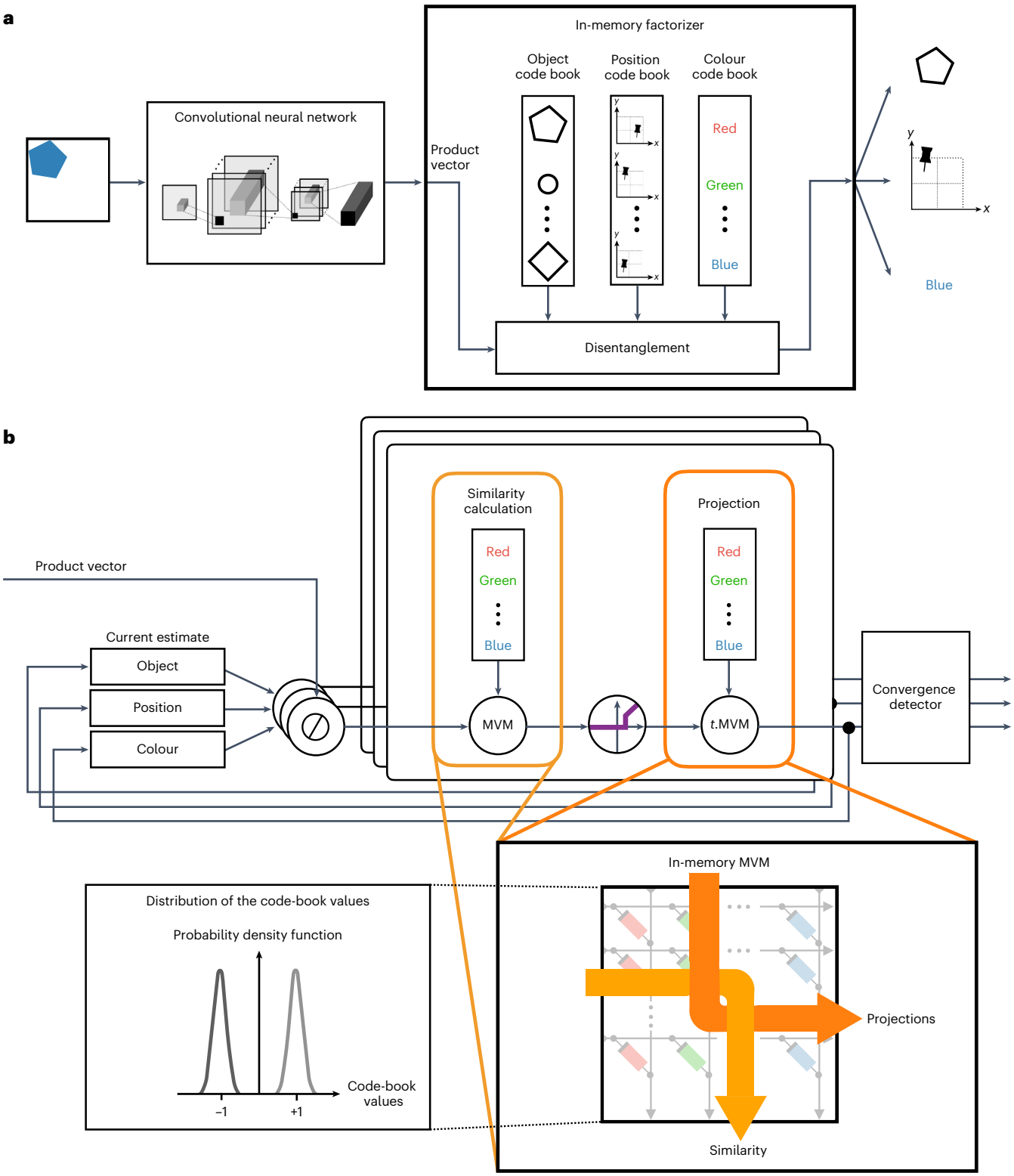


Fig. 1 | Factorization of perceptual representations using the in-memory factorizer. a, A visual input is first fed through a convolutional neural network to approximately map it to a D-dimensional product vector. The generated product vector is applied as an input to the in-memory factorizer, which contains a unique code book of code vectors for each possible attribute. The factorizer disentangles the product vector and predicts the correct attribute factors. **b**, The in-memory factorizer iteratively searches in superposition. For each attribute, an updated estimate is computed during every iteration by unbinding the contribution of the other factors from the product vector. The unbound estimate is fed through the similarity calculation with sparse activation as nonlinearity and the projection to obtain a novel estimate, which is then fed back to be used in the subsequent iteration. The similarity calculation is based on MVM operations and projection is based on transposed MVM operations that can be executed in the in-memory fashion in a crossbar array of memristive devices by exploiting Ohm's law and Kirchhoff's current summation law with a computational time complexity of $\mathcal{O}(1)$. Moreover, the intrinsic stochasticity associated with storing the bipolar code vectors in the array and the resulting imprecise MVM operations serve as a key enabler for the in-memory factorizer.

https://doi.org/10.1038/s41565-023-01357-8

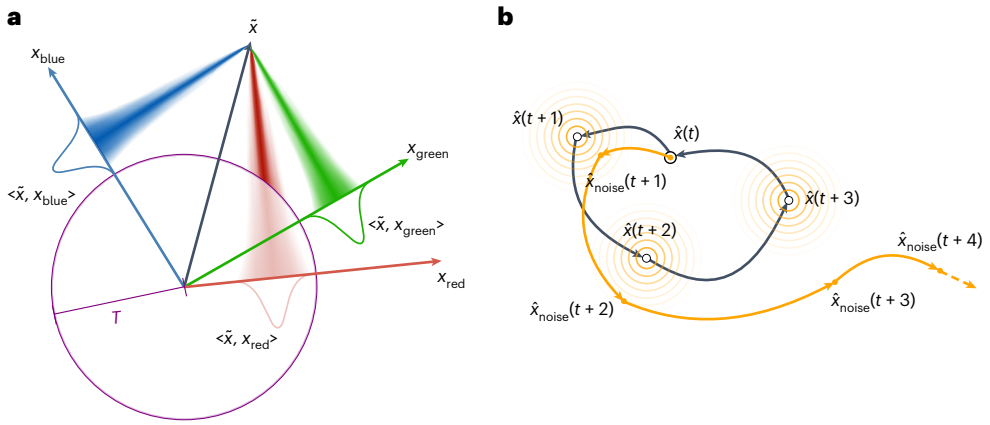


Fig. 2 | Stochastic similarity computation, sparse activations and limit cycles. **a**, The similarity calculation computes the similarities between the unbound estimate vector $\tilde{\mathbf{x}}$ and all the code vectors (for example, $\{\mathbf{x}_{red}, \mathbf{x}_{green}, \mathbf{x}_{blue}\}$). The in-memory similarity calculation, denoted by <.,.>, is stochastic with additive noise. The distributions of noisy similarity results projected onto a two-dimensional space are shown in green, red and blue, respectively. Furthermore, a winner-takes-all approach activates similarity values only above a certain threshold (T) level, which results in a similarity vector with sparse non-zero elements. Similarity values smaller than T are zeroed out; for example, $< \tilde{\mathbf{x}}, \mathbf{x}_{red} >$ is not activated (that is, zeroed), but the other two similarity values ($< \tilde{\mathbf{x}}, \mathbf{x}_{blue} >$ and $< \tilde{\mathbf{x}}, \mathbf{x}_{green} >$), larger than T, remain activated. The activation threshold (T) is depicted in purple. **b**, Visualized by the black arrows, this figure shows a limit cycle of length $l = 4$. When stuck in a limit cycle of length l, we constantly end up checking the same l solutions. In contrast, the orange arrows show an example trajectory of the factor's estimates ($\hat{\mathbf{x}}$) of the in-memory factorizer exploiting stochasticity in both similarity and projection operations, which yields noisy estimates. For a subsequent time step, there is some uncertainty, as visualized by the orange circles. As the search for factorization is an iterative process, the uncertainty for the subsequent time steps increases. Eventually, the uncertainty is high enough for the in-memory factorizer to diverge from a limit cycle and to converge to the correct factorization in time.

to each attribute, which results in a unique product vector of the same fixed dimensionality[21]. The element-wise multiplication operation can be viewed as the binding operation that binds the attribute vectors and generates the product vector. Moreover, it has recently been shown that given the raw image of an object, a deep convolutional neural network can be trained to approximately generate the product vector[22]. The factorization problem can now be posed as the decomposition of an exact product vector or, as in the latter case, an inexact product vector, into its constituent attribute vectors.

In this Article, we propose a non-deterministic, non-von Neumann compute engine that efficiently factorizes such product vectors to obtain estimates of the constituent attributes. The compute engine combines the emerging compute paradigm of in-memory computing (IMC)[23–25] with an enhanced variant of a resonator network[21,26]. We introduce nonlinear sparse activation between two key operations of the resonator network, and exploit the intrinsic stochasticity associated with the memristive devices employed for IMC to enhance the maximally solvable problem size. Finally, we present a large-scale experimental demonstration using IMC cores based on phase-change memory (PCM) technology and applications in visual perception.

In-memory stochastic factorizer with sparse activations

The resonator network is a nonlinear dynamical system capable of factorizing holographic vectors and is indeed a viable neural solution to the factorization problem[21,26]. If there are F attributes, the resonator network iteratively searches across a set of possible estimates referred to as the code book associated with each attribute. For example, in Fig. 1a, $F = 3$, with the attributes being object, position and colour. The vectors associated with each code book are referred to as code vectors. All the code vectors are randomly drawn, which makes them quasi-orthogonal to each other in high-dimensional space, as mentioned previously[20]. When each code book contains a finite set of M code vectors, there are M^F possible combinations to be searched to factorize the D-dimensional product vector into its constituent factors, where $D \ll M^F$. The product vectors could be constructed by binding the randomly drawn code vectors or could be approximate variants that are generated by a convolutional neural network (Fig. 1a). Factorizing these product vectors that exhibit no correlational structure forms a hard combinatorial search problem. By exploiting the quasi-orthogonality of the code vectors, the resonator network is able to rapidly search through the various combinations in superposition by iteratively unbinding all but one of the factors from the product vector, and then projecting it into the space of possible solutions of the considered factor. Note that in bipolar space, unbinding is also performed via element-wise multiplication. Both similarity search and projection operations associated with the resonator network involve matrix–vector multiplication (MVM) operations where the matrix transpires to be a fixed code book. This is highly amenable to IMC using memristive devices[27,28]. As shown in Fig. 1b, the proposed in-memory factorizer stores the code vectors on crossbar arrays of memristive devices performing analogue in-memory MVM operations. The similarity calculation and projection are based on MVM and transposed MVM operations, respectively. These operations can be executed in the in-memory fashion in a crossbar array of memristive devices by exploiting Ohm's law and Kirchhoff's current summation law.

The unsupervised nature of the conventional resonator network's deterministic search could result in checking the same sequence of solutions multiple times across iterations, resulting in limit cycles that prevent convergence to the optimal solution. One of the key insights from the in-memory factorizer is that the intrinsic stochasticity associated with memristive devices can substantially reduce the occurrence of such limit cycles. As shown in Fig. 2a, during the similarity calculation, the analogue in-memory MVM results in a stochastic similarity vector. The stochasticity enables the factorizer to break free of limit cycles and thus explore a substantially larger solution space (Fig. 2b). Note that unlike earlier energy-based combinatorial optimization problems such as simulated annealing[29], Boltzmann machines[30–33] and Hopfield networks[34–36], the in-memory factorizer is able to leverage device-level stochasticity—as a valuable computational tool—for breaking limit cycles.

https://doi.org/10.1038/s41565-023-01357-8

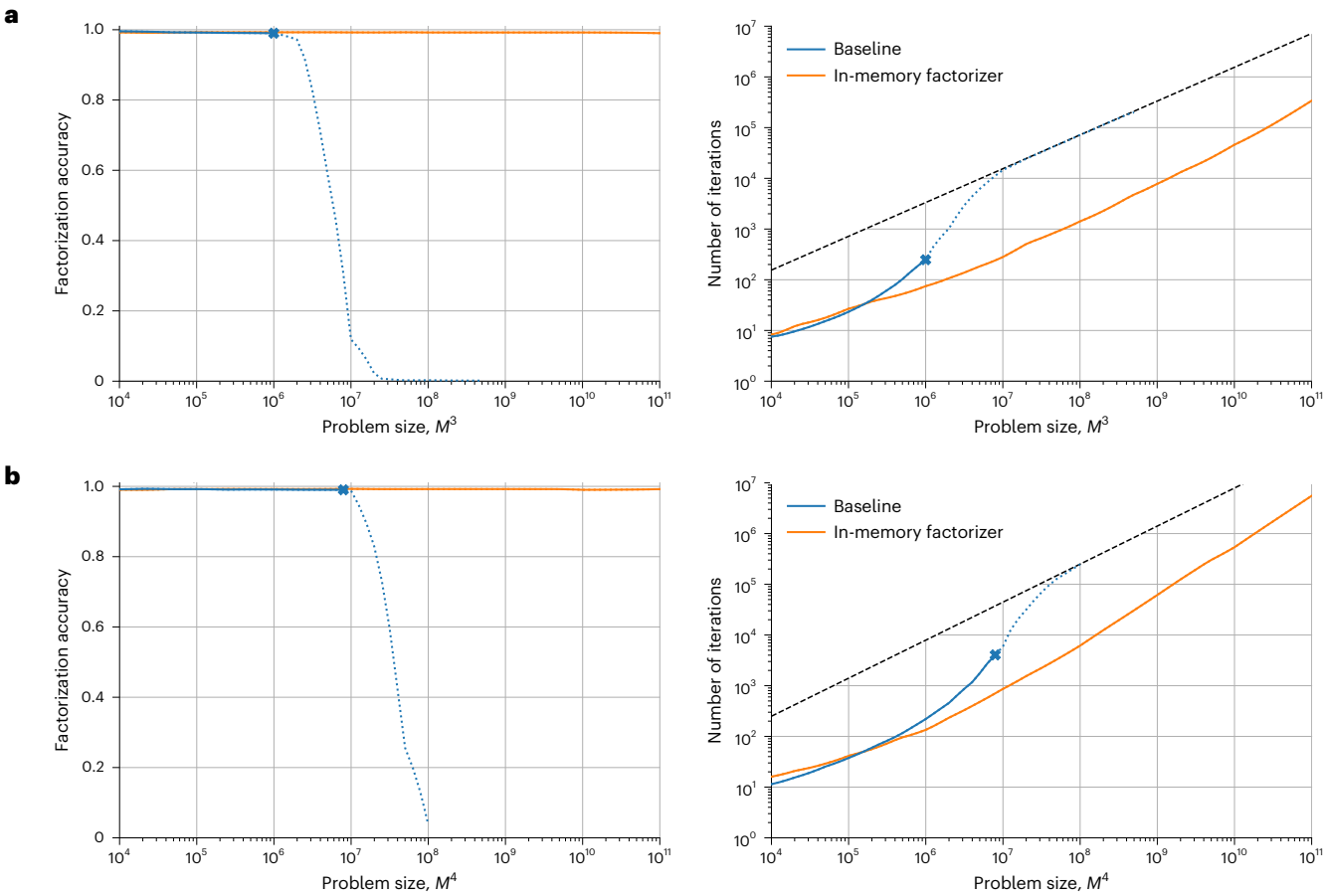


Fig. 3 | Operational capacity of the stochastic in-memory factorizer with sparse activations. a,**b**, Operational capacity for $F = 3$ (**a**) and $F = 4$ (**b**). The problem size M^F is shown on the x axis, and the y axes show the accuracy (left) and number of iterations required to solve a given problem size (right). The black dashed lines indicate the equivalent number of dot-product operations required for a brute-force approach to search among M^F precomputed product vectors. These lines indicate the upper limit for the operation count. For each F, we use the smallest dimension reported by the baseline resonator network[21,26], namely, $D = 1{,}500$ for $F = 3$ and $D = 2{,}000$ for $F = 4$. The blue cross indicates the largest problem size that is within the operational capacity of the baseline resonator network, meaning that problems larger than that size cannot be factorized by the baseline network at 99% accuracy, and the in-memory factorizer can solve problem sizes at least five orders of magnitude larger at 99% accuracy, or higher.

Another limitation of the conventional resonator network is the use of the identity function, as a linear activation, between the similarity search and projection operations. We find that by adopting a nonlinear winner-takes-all approach and by zeroing out the weaker similarity values, both convergence rate and maximally solvable problem size of the in-memory factorizer are enhanced (Supplementary Note 2). The winner-takes-all approach uses an activation threshold T to sparsify the similarity vector (Fig. 2a), which was chosen on the basis of Bayesian optimization (Methods).

The resulting non-deterministic in-memory factorizer with nonlinear sparse activations can be analysed and compared with the state of the art using three figures of merit: dimensionality, computational complexity and operational capacity. The dimensionality refers to the number of elements in a code vector. The computational complexity defines the average number of operations required by a factorizer to factorize a given product vector, where each operation refers to a D-dimensional dot-product calculation. This depends on the problem size and number of factors. Given an upper bound for the first two figures of merit, the operational capacity defines the maximally solvable problem size with an accuracy of at least 99%. To compare the operational capacity of the in-memory factorizer with the baseline resonator network[21,26], we performed extensive simulations of the in-memory factorizer in software (Methods). As shown in Fig. 3, the in-memory factorizer substantially enhances the operational capacity by up to five orders of magnitude, even when the vector dimensionality is reduced by a factor of over four (Supplementary Fig. 1).

Large-scale experimental demonstration

Next we present an experimental realization of the in-memory factorizer using IMC cores based on PCM devices fabricated in 14 nm complementary metal–oxide–semiconductor technology (Methods). We employed two IMC cores, one to calculate the similarities and one for the projections (Fig. 4a). Each IMC core features a crossbar array of 256 × 256 unit cells capable of performing stochastic and analogue MVM operations[37]. Each unit cell comprises four PCM devices organized in a differential configuration and can be programmed to a certain conductance value (the bipolar code books are stored in two out of four devices). However, due to the intrinsic stochasticity associated with crystal nucleation and growth[38], there will be a distribution of conduction values across multiple devices in the crossbar (Fig. 4b). This distribution will become slightly broader with time as a result of the variability associated with the structural relaxation of the atomic configuration in each device[39]. In addition to this, there is read noise that exhibits a $1/f$ spectral characteristic and random telegraph noise (Supplementary Note 3 provides more details)[40]. The input to the IMC core is applied using a constant-pulse-width-modulated voltage applied to all the rows of the

https://doi.org/10.1038/s41565-023-01357-8

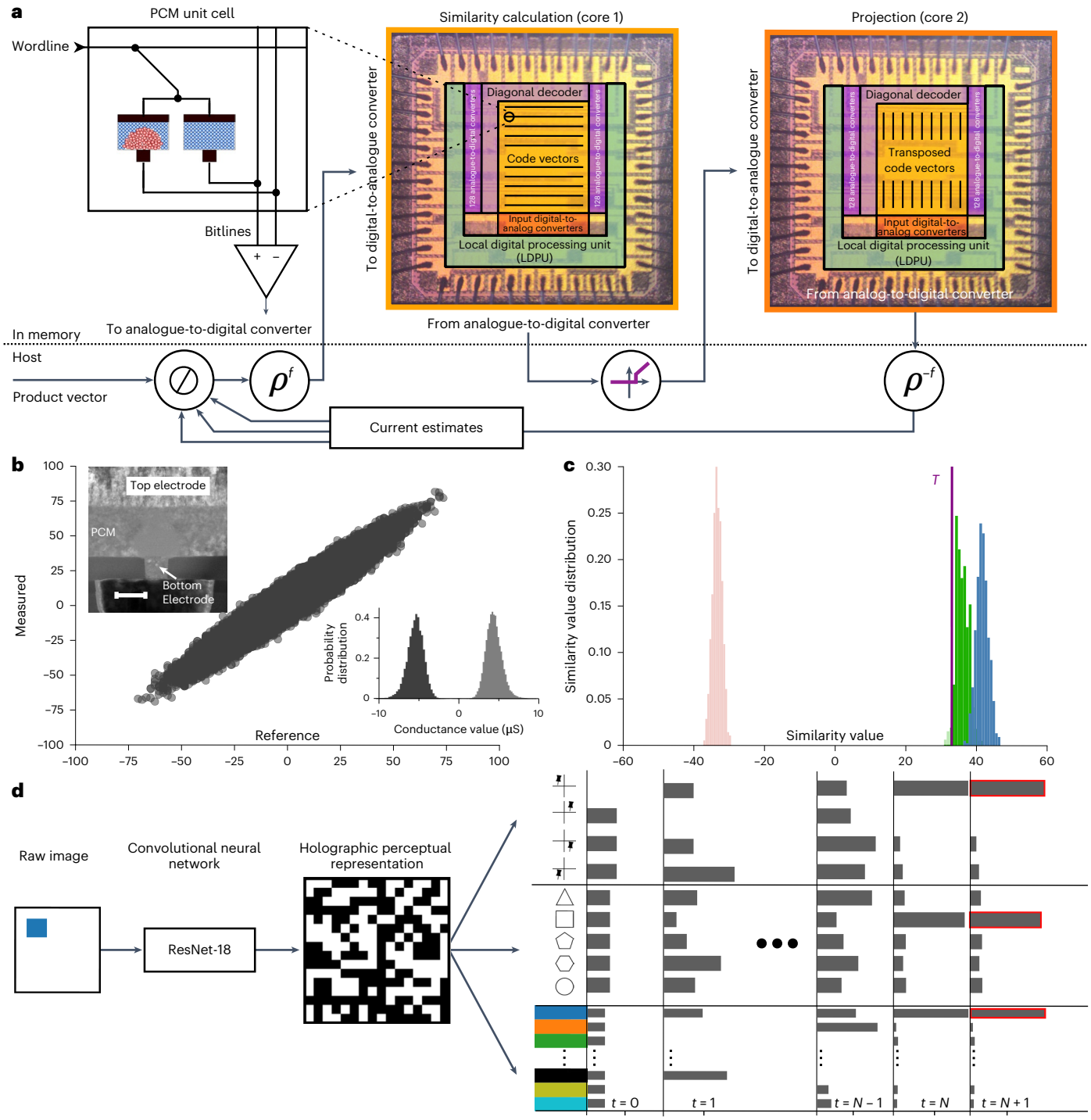


Fig. 4 | Experimental realization of the in-memory factorizer. a, Experimental setup includes two IMC cores for similarity calculation and projection. The associated MVM operations are executed using the crossbars and the fully integrated peripherals. The unbinding, permutation, activation and bipolarization operations are executed on the host computer connected to the cores. On the crossbar, one unit cell stores one bipolar (±1) weight value associated with the code vectors. **b**, Measured output of the in-memory MVM plotted against the reference output obtained at high precision, indicating the intrinsic stochasticity associated with the operation. The grey histogram corresponds to the positive conductance values and the black histogram, to the negative ones. The bottom-right inset shows the distribution of programmed conductance values on the first core executing the similarity calculation. The top-left inset shows a low-angle annular dark-field scanning transmission electron microscope image[44] of a nanoscale PCM device in its RESET state, where a large amorphous region blocks the bottom electrode, resulting in a conductance value close to zero. To encode a 1 or −1 in a unit cell, the amorphous region corresponding to the appropriate device is partially crystallized to achieve a higher conductance value. The target conductance value was set at 5 μS. Scale bar, 50 nm. **c**, The red, green and blue histograms correspond to the distribution of similarity between the same unbound estimate vector ($\hat{\mathbf{x}}$) and three colour code vectors ($\mathbf{x}_{red}$, $\mathbf{x}_{green}$, $\mathbf{x}_{blue}$). The purple vertical line corresponds to the activation threshold at $T = 33$, which avoids activating any similarity values related to $\mathbf{x}_{red}$. **d**, Raw image is fed through a ResNet-18 network to generate a holographic product vector visualized as a binary heat map. The in-memory factorizer iteratively disentangles the holographic vector. In the first time step, all the code vectors show an equal similarity to the current estimate. Over time, the in-memory factorizer converges to the correct factorization, indicated by a high similarity value for a single code vector.

crossbar array in parallel. Ohm's law defines the current flowing through each unit cell, and the current is summed up on the corresponding bitlines in accordance with Kirchhoff's current summation law. This current is digitized and accumulated by 256 analogue-to-digital converters operating in parallel.

Each IMC core performs MVM operations in a constant amount of time, $\mathcal{O}(1)$, which leads to merely reducing the time complexity of factorization to the average number of iterations. Moreover, the intrinsic randomness associated with the PCM devices is exploited to calculate the stochastic similarity and projection vectors that minimize the occurrence of limit cycles (Fig. 4b,c). A permute logic is employed to temporally multiplex one crossbar array for multiple factors, and the hyperparameters such as the activation and convergence thresholds were obtained through Bayesian learning (Methods). The experimentally realized in-memory factorizer is compared with the baseline resonator network[21,26], where both methods used $D = 256$, $M = 256$ and $F = 3$. These parameters set the total problem size to $M^F = 16{,}777{,}216$ and the maximum number of iterations to $N = 21{,}845$, which ensures that the computational complexity does not exceed that of the brute-force search (Methods). When using the same code books and 5,000 randomly selected product vectors as queries, the baseline resonator network is found to be incapable of factorizing any of the product vectors. In contrast, the in-memory factorizer is capable of achieving an outstanding accuracy of 99.71% with an average number of 3,312 iterations.

Finally, we demonstrate the role of the in-memory factorizer in visual perception to disentangle the attributes of raw images. The perception system consists of two main components (Fig. 4d). A convolutional neural network is trained to map the input image to a holographic perceptual product vector, based on a known set of image attributes. Hence, during inference, the output of the neural network is an approximation of the product vector that describes the image. The in-memory factorizer is used to disentangle the approximate product vector using a known set of image attributes. For experiments, we used the input images from the relational and analogical visual reasoning (RAVEN) dataset[41]. The images contain objects with attributes such as type, size, colour and position. Each set of attributes is mapped to a unique code book, leveraging its symbolic nature (Methods). The disentanglement of 1,000 images from the RAVEN dataset to the correct estimate of attributes achieved an accuracy of 99.42% (Supplementary Video 1).

Conclusion

We have compared the in-memory factorizer with a dedicated reference digital design that benefits from the proposed sparse activations and all the other features, except the intrinsic stochasticity that is inherent in PCM devices. Using the same configurations as the experiments ($D = 256$, $M = 256$, $F = 3$ and 5,000 trials), the deterministic digital design was able to reach an accuracy of 95.76% with 3,802 iterations on average. Compared with the in-memory factorizer, the digital design demanded 14.8% more iterations on average, and still exhibited 4.0% lower accuracy. Even if we allow the maximum number of iterations to be arbitrarily large, the digital design is still not able to match the accuracy of the in-memory factorizer due to the limit cycles (Supplementary Note 4), thus highlighting the crucial role of the intrinsic stochasticity associated with PCM devices.

The in-memory factorizer could also result in a notable gain in energy and areal efficiency. By combining the advantages of in-place computation and the reduced number of iterations, it is estimated that a custom-designed in-memory factorizer based on a 512 × 512 crossbar array is capable of factorizing a single query within an energy budget of 33.1 μJ on average, resulting in energy savings of 12.2 times compared with the reference digital design. The total area saving is estimated to be 4.85 times (Supplementary Note 4). The non-volatile storage of the code books is an added advantage compared with the digital design.

Note that the application of in-memory factorizers can go beyond visual perception, as factorization problems arise everywhere in perception and cognition, for instance, in analogical reasoning[12–16,42]. Other applications include tree search[21] and the prime factorization of integers[43]. Recently, it has been shown how the classical integer factorization problem can be solved by casting it as a problem of factorizing holographic vectors[43]. These pave the way for solving non-trivial combinatorial search problems.

To summarize, we have presented a non-von Neumann compute engine to factorize high-dimensional, holographic vectors by searching in superposition. We have experimentally verified it using two state-of-the-art IMC cores based on PCM technology, which provides in-place computation and non-volatility. The experimental results show the reliable and efficient factorization of holographic vectors spanning a search space of 256 × 256 × 256. The intrinsic stochasticity associated with PCM devices is coupled with nonlinear sparse activations to enhance the operational capacity by five orders of magnitude as well as to reduce the spatial and time complexity associated with the computational task. Furthermore, we have demonstrated the efficacy of the in-memory factorizer in disentangling the constituent attributes of raw images. This work highlights the role of emerging non-von Neumann compute paradigms in realizing critical building blocks of future artificial intelligence systems.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41565-023-01357-8.

References

1. Feldman, J. The neural binding problem(s). *Cogn. Neurodyn.* **7**, 1–11 (2013).
2. Land, E. H. & McCann, J. J. Lightness and retinex theory. *J. Opt. Soc. Am.* **61**, 1–11 (1971).
3. Barrow, H. G. & Tenenbaum, J. M. in *Computer Vision Systems* 3–26 (Academic Press, 1978).
4. Adelson, E. & Pentland, A. in *The Perception of Shading and Reflectance* 409–424 (Cambridge Univ. Press, 1996).
5. Barron, J. T. & Malik, J. Shape, illumination and reflectance from shading. *IEEE Trans. Pattern Anal. Mach. Intell.* **37**, 1670–1687 (2015).
6. Memisevic, R. & Hinton, G. E. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Comput.* **22**, 1473–1492 (2010).
7. Burak, Y., Rokni, U., Meister, M. & Sompolinsky, H. Bayesian model of dynamic image stabilization in the visual system. *Proc. Natl Acad. Sci. USA* **107**, 19525–19530 (2010).
8. Cadieu, C. F. & Olshausen, B. A. Learning intermediate-level representations of form and motion from natural movies. *Neural Comput.* **24**, 827–866 (2012).
9. Anderson, A. G., Ratnam, K., Roorda, A. & Olshausen, B. A. High-acuity vision from retinal image motion. *J. Vision* **20**, 34 (2020).
10. Smolensky, P. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif. Intell.* **46**, 159–216 (1990).
11. Jackendoff, R. *Foundations of Language: Brain, Meaning, Grammar, Evolution* (Oxford Univ. Press, 2002).
12. Hummel, J. E. & Holyoak, K. J. Distributed representations of structure: a theory of analogical access and mapping. *Psychol. Rev.* **104**, 427–466 (1997).
13. Kanerva, P. in *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational and Neural Sciences* 164–170 (New Bulgarian Univ., 1998).

14. Kanerva, P. Pattern completion with distributed representation. In *International Joint Conference on Neural Networks* 1416–1421 (IEEE, 1998).
15. Plate, T. A. Analogy retrieval and processing with distributed vector representations. *Expert Syst. Int. J. Knowledge Eng. Neural Netw.* **17**, 29–40 (2000).
16. Gayler, R. W. & Levy, S. D. A distributed basis for analogical mapping: new frontiers in analogy research. In *New Frontiers in Analogy Research, Second International Conference on the Analogy* 165–174 (New Bulgarian University Press, 2009).
17. Gayler, R. W. Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. In *Joint International Conference on Cognitive Science* 133–138 (Springer, 2003).
18. Plate, T. A. Holographic reduced representations. *IEEE Trans. Neural Netw.* **6**, 623–641 (1995).
19. Plate, T. A. *Holographic Reduced Representations: Distributed Representation for Cognitive Structures* (Stanford Univ., 2003).
20. Kanerva, P. Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors. *Cogn. Comput.* **1**, 139–159 (2009).
21. Frady, E. P., Kent, S. J., Olshausen, B. A. & Sommer, F. T. Resonator networks, 1: an efficient solution for factoring high-dimensional, distributed representations of data structures. *Neural Comput.* **32**, 2311–2331 (2020).
22. Hersche, M., Zeqiri, M., Benini, L., Sebastian, A. & Rahimi, A. A neuro-vector-symbolic architecture for solving Raven's progressive matrices. *Nat. Mach. Intell.* https://doi.org/10.1038/s42256-023-00630-8 (2023).
23. Lanza, M. et al. Memristive technologies for data storage, computation, encryption and radio-frequency communication. *Science* **376**, eabj9979 (2022).
24. Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **15**, 529–544 (2020).
25. Wang, Z. et al. Resistive switching materials for information processing. *Nat. Rev. Mater.* **5**, 173–195 (2020).
26. Kent, S. J., Frady, E. P., Sommer, F. T. & Olshausen, B. A. Resonator networks, 2: factorization performance and capacity compared to optimization-based methods. *Neural Comput.* **32**, 2332–2388 (2020).
27. Wong, H.-S. P. & Salahuddin, S. Memory leads the way to better computing. *Nat. Nanotechnol.* **10**, 191–194 (2015).
28. Chua, L. Resistance switching memories are memristors. *Appl. Phys. A* **102**, 765–783 (2011).
29. Shin, J. H., Jeong, Y. J., Zidan, M. A., Wang, Q. & Lu, W. D. Hardware acceleration of simulated annealing of spin glass by RRAM crossbar array. In *Proc. IEEE International Electron Devices Meeting* 3.3.1–3.3.4 (IEEE, 2018).
30. Bojnordi, M. N. & Ipek, E. Memristive Boltzmann machine: a hardware accelerator for combinatorial optimization and deep learning. In *Proc. IEEE International Symposium on High Performance Computer Architecture* 1–13 (IEEE, 2016).
31. Mahmoodi, M. R., Prezioso, M. & Strukov, D. B. Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization. *Nat. Commun.* **10**, 5113 (2019).
32. Borders, W. A. et al. Integer factorization using stochastic magnetic tunnel junctions. *Nature* **573**, 390–393 (2019).
33. Wan, W. et al. 33.1 A 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models. In *Proc. IEEE International Solid-State Circuits Conference* 498–500 (IEEE, 2020).
34. Kumar, S., Strachan, J. P. & Williams, R. S. Chaotic dynamics in nanoscale NbO_2 Mott memristors for analogue computing. *Nature* **548**, 318–321 (2017).
35. Cai, F. et al. Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks. *Nat. Electron.* **3**, 409–418 (2020).
36. Yang, K. et al. Transiently chaotic simulated annealing based on intrinsic nonlinearity of memristors for efficient solution of optimization problems. *Sci. Adv.* **6**, eaba9901 (2020).
37. Khaddam-Aljameh, R. et al. Hermes core—a 14nm CMOS and PCM-based in-memory compute core using an array of 300ps/LSB linearized CCO-based ADCs and local digital processing. In *2021 Symposium on VLSI Circuits* 1–2 (IEEE, 2021).
38. Tuma, T., Pantazi, A., Le Gallo, M., Sebastian, A. & Eleftheriou, E. Stochastic phase-change neurons. *Nat. Nanotechnol.* **11**, 693–699 (2016).
39. Le Gallo, M., Krebs, D., Zipoli, F., Salinga, M. & Sebastian, A. Collective structural relaxation in phase-change memory devices. *Adv. Electron. Mater.* **4**, 1700627 (2018).
40. Le Gallo, M. & Sebastian, A. An overview of phase-change memory device physics. *J. Phys. D Appl. Phys.* **53**, 213002 (2020).
41. Zhang, C., Gao, F., Jia, B., Zhu, Y. & Zhu, S.-C. RAVEN: a dataset for relational and analogical visual reasoning. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 5312–5322 (IEEE, 2019).
42. Kent, S. *Multiplicative Coding and Factorization in Vector Symbolic Models of Cognition*. PhD thesis, Univ. California (2020).
43. Kleyko, D. et al. Integer factorization with compositional distributed representations. In *Proc. 9th Annual Neuro-Inspired Computational Elements Conference* 73–80 (ACM, 2022).
44. Li, J. et al. Low angle annular dark field scanning transmission electron microscopy analysis of phase change material. In *Proc. International Symposium for Testing and Failure Analysis 2021* 206–210 (ASM, 2021).

Methods

Detection of convergence in the in-memory factorizer

The iterative factorization problem is said to be converged if, for two consecutive time steps, all the estimates are constant, that is, $\hat{\mathbf{x}}^f(t+1) = \hat{\mathbf{x}}^f(t)$ for $f \in [1, F]$. To detect this convergence, we define a novel early convergence detection algorithm. The in-memory factorizer is said to be converged if a single similarity value across all the factors surpasses a convergence detection threshold:

$$\text{converged} = \begin{cases} \text{true}, & \text{if } \alpha_i^f > T_{\text{convergence}} \\ \text{false}, & \text{otherwise}, \end{cases} \tag{1}$$

where $i \in [1, |X^f|]$ for $f \in [1, F]$. On convergence, the predicted factorization is given by the code vector associated with the largest similarity value per code book. The novel convergence detection counteracts parasitic code vectors with a high enough similarity value to prevent the in-memory factorizer from converging to a stable solution. Furthermore, there is no need to store the history of prior estimates. Otherwise, the last estimate for each factor needs to be stored to be able to compare it with the latest estimate and detect convergence, resulting in a total of $F \times D$ stored bits. We used Bayesian optimization to find the optimal convergence detection threshold. It stays at a fixed ratio of D for any given set of hyperparameters and problem sizes.

Maximal number of iterations

To use strictly fewer search operations than the brute-force approach, the maximal number of iterations (N) must be constrained to

$$N \times M \times F < M^F. \tag{2}$$

In all our experiments, we constrain the maximum number of iterations to $N < \frac{M^{F-1}}{F}$ to ensure a lower computational complexity compared with the brute-force approach.

Hyperparameter optimization via Bayesian optimization

We find the optimal hyperparameters of the in-memory factorizer $h^* = [T^*, T^*_{\text{convergence}}]$ by solving a Bayesian optimization problem, where T^* is the activation threshold and $T^*_{\text{convergence}}$ is the convergence threshold. We define the error rate l to be the ratio of the wrongly factorized product vectors. More formally, we try to find the optimal set of hyperparameters h^* by minimizing the error rate l, which is a function of the hyperparameters h:

$$h^* = \arg\min_h l(h). \tag{3}$$

We model the error rate as a Gaussian process with a radial basis function kernel and minimize it using Bayesian optimization. To sample possible hyperparameters during optimization, we use the expected improvement acquisition function.

We reduce the computational complexity of the error rate evaluation for a given set of hyperparameters h by limiting the maximum number of iterations to $N' = N/10$ and the number of trials to 256. A low number of iterations N' yields higher error rates, yet they still provide a good indication of the in-memory factorizer's performance given a set of hyperparameters. Additionally, the reduced number of trials gives noisy evaluations of l, which is modelled as additive Gaussian measurement noise.

We derive the final parameter estimates by averaging results over the five best experiments. For the hardware experiments, the stochasticity is inherently provided by the PCM crossbar arrays. However, for simulating the in-memory factorizer in software, we need to model stochasticity as a noise level. Hence, the noise level (n^*) is treated as an extra hyperparameter for optimization. Accordingly, to simulate our method in the software, we optimize for three hyperparameters: the activation threshold, convergence threshold and noise level given by $h^* = [T^*, T^*_{\text{convergence}}, n^*]$.

In-memory experiments

For the experimental demonstration, we employed an IMC core fabricated by IBM Research in a 14 nm complementary metal–oxide–semiconductor technology node[37]. This features 256 × 256 unit cells, each comprising four PCM devices arranged in a differential configuration where one pair of devices is connected in parallel to represent positive conductance and another pair to represent negative conductance on the unit cell. For these experiments, however, we program only one device, either on the positive or negative side, as it provides a sufficient dynamic range in conductance to achieve more than 99% accuracy. This effectively allows us to optimize the unit cell to consist of just two PCM devices (Fig. 4a).

A custom printed circuit board houses two such HERMES cores, and a field-programmable gate array board is used to control the communication protocol and loading data to and from the cores. The field-programmable gate array, in turn, is controlled by a Linux machine running a Python version 3.6 environment on top. The host machine performs the unbinding and applies the activation function, and the two cores perform the dominant MVM similarity search and projection operations. An iterative programming scheme of the PCM devices is employed to store the code vectors in the crossbars. The output of the in-memory MVM is measured in terms of the analogue-to-digital converter count units. Subsequently, a linear correction is applied to correct circuit-level mismatches. The linear correction parameters are calculated before MVM operations.

The single crossbar core with dimensions of 256 × 256 limits the total number of supported code vectors across all the code books to 256. To overcome this limitation, we propose a permute logic to temporally multiplex one single crossbar array for all the F factors. This enables us to exploit the complete crossbar for one single code book with up to 256 code vectors, and reuse it across an arbitrary number of factors. As shown in Fig. 4a, before the similarity calculation, we apply the permute logic as a factor-wise exclusive circular shift on the estimated unbinding. This results in a quasi-orthogonal time multiplexing of the crossbar. Before updating the estimates, we reverse the circular shift to obtain unaltered estimates.

Software simulations of the in-memory factorizer

Stochasticity is the key enabler of the in-memory factorizer. Adding some stochasticity helps to diverge from the limit cycles as each solution becomes unique. For the experiments reported in Fig. 3, this important aspect is modelled in software by simulating the noisy behaviour of the MVMs on the crossbar as an additive Gaussian noise with zero mean:

$$\alpha_i' = f(\alpha_i) = \alpha_i + n, \tag{4}$$

where α_i is a single entry of the output vector and n is normally distributed with $n \sim \mathcal{N}(0, \sigma^2)$.

In total, we simulated $F \times (M + D)$ additive Gaussian noise sources: M on the similarity vector and D on the projection vector. The noisy similarity vector is required to break free of the limit cycles. Due to the random distribution of similarity values, there is always a chance of activating none of the similarity values if they do not cross the activation threshold. Adding noise on top of this projection prevents such an all-zero estimation by randomly initializing the vector before bipolarization.

The PCM devices on the crossbar array exhibit a similar noisy behaviour, which can be modelled as a combination of noise components such as programming noise, read noise and drift variability. In Supplementary Note 3, we model the extent to which each of these noise components is present in the experimental crossbar arrays and analyse the sensitivity to change in the noise components as reflected

in the performance figures, such as factorization accuracy and the number of iterations required for convergence.

To concretely quantify the effect of the aggregated PCM device noise on the performance of the factorizer, we conducted simulations where the aggregated noise standard deviation gradually increases from zero noise, and maintaining the ratio of the standard deviation between read noise and programming noise, as observed on the experimental platform (σ_r/σ_p = 0.3951/1.1636). These results are shown in Extended Data Fig. 1.

We observe that with zero noise, the factorizer performs poorly, with an accuracy of 25.4% and requiring on average 16,000 iterations to converge. This expected behaviour is due to the deterministic nature of the search and the resulting inability to break free of the limit cycles. The factorizer, however, operates at its peak performance when the standard deviation of aggregated noise is maintained within the range of [0.293 μS, 1.277 μS]. Note that the standard deviation of aggregated noise observed on the experimental platform (0.98 μS) falls in the middle of this tolerated noise range (Extended Data Fig. 1).

Visual disentanglement

We use the RAVEN[41] dataset, which provides a rich set of progressive matrices tests, for visual abstract reasoning tasks. We only focus on the visual perception part of the task to disentangle a sensory input image from its underlying attribute factors. The RAVEN dataset provides a total of 70,000 tests, each consisting of 16 panels of images. In our experiment, we considered the 2 × 2 image constellation with a single object. Each object can have one of four possible positions, ten colours, six sizes and five types. There are, thus, 1,200 possible combinations. We mapped each set of attributes to a single code book of the in-memory factorizer. We reused the same code book as for the synthetic experiment. The first code book represents the position attribute and the second, the colour attribute. For the third code book, we fused the size and type attributes into a single code book, ending up with a total of 30 possible size–type combinations.

Each image can be described by a product vector formed by the binding of the corresponding code vectors. To map the input image to the product vector, we used a convolutional neural network. We used the ResNet-18 convolutional neural network and mapped its output to our 256-dimensional vector space using the training schema proposed elsewhere[22]. After training, we ran all the test images through the trained ResNet-18 network to obtain an estimate of the product vector.

Next, we passed all the estimated product vectors through the same experimental test setup as the synthetic experiments. The main difference lies in the product vector: for the synthetic one, we used the exact product vectors, but for the visual perception task, the product vectors are the output of the convolutional neural network, which will be an approximate product vector as opposed to being the exact one.

Data availability

The data that support the findings of this study are available via Zenodo at https://zenodo.org/record/7599430. Source data are provided with this paper.

Code availability

Our code is available via GitHub at https://github.com/IBM/in-memory-factorizer.

Acknowledgements

This work is supported by the IBM Research AI Hardware Center and by the Swiss National Science Foundation (SNF) (grant no. 200800). We thank M. Le Gallo for the technical help; K. Brew and J. Li for assistance with TEM imaging of PCM devices; and V. Narayanan, C. Apte and R. Haas for managerial support.

Author contributions

J.L., G.K., M.H., A.S. and A.R. conceived the idea and designed the experiments. J.L. performed the experiments and characterization. J.L., A.S. and A.R. wrote the paper, with input from all the authors. All the authors provided critical comments and analyses.

Competing interests

The authors declare no competing interests.

Additional information

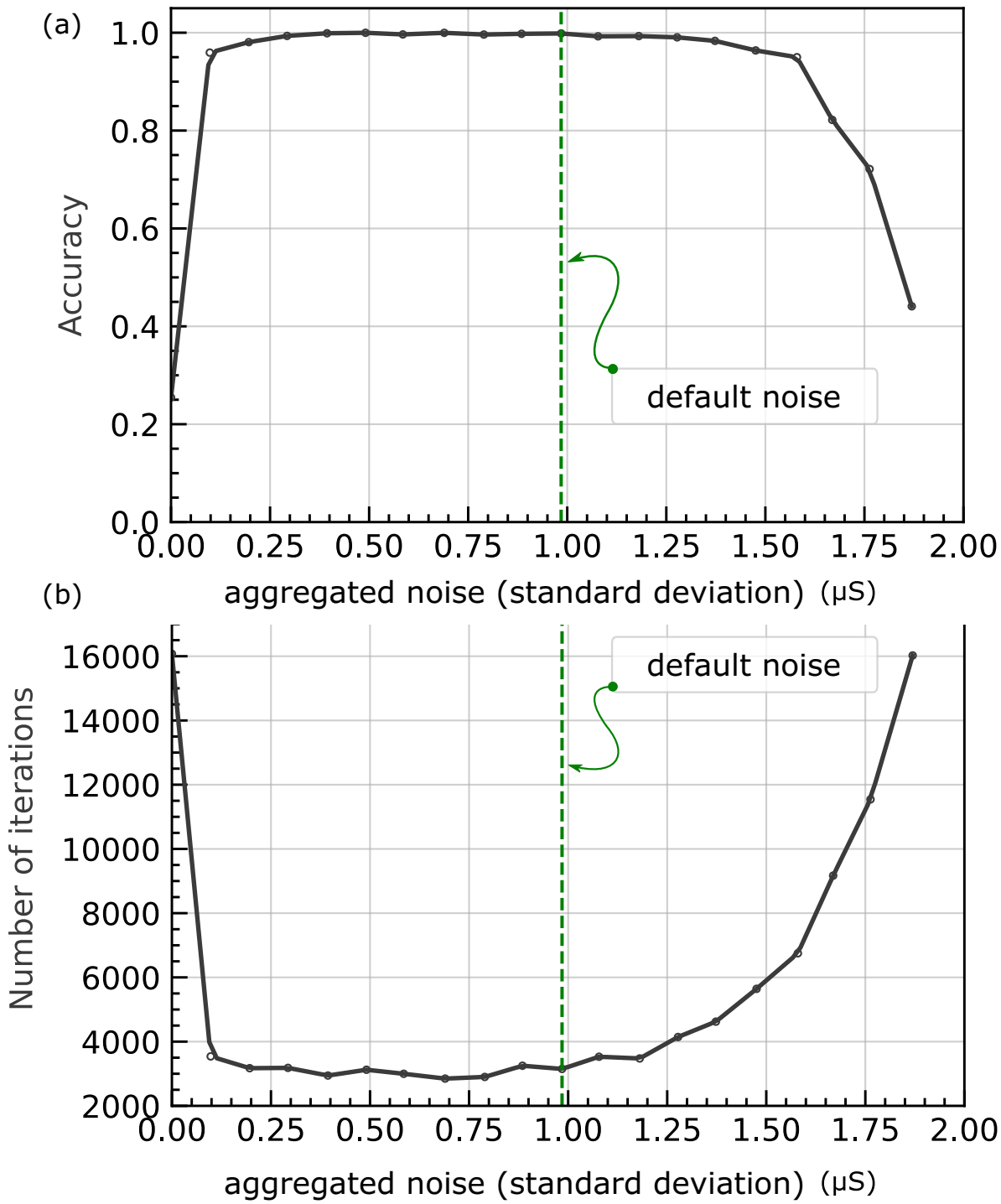
Extended data is available for this paper at https://doi.org/10.1038/s41565-023-01357-8.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41565-023-01357-8.

Correspondence and requests for materials should be addressed to Abu Sebastian or Abbas Rahimi.

Peer review information *Nature Nanotechnology* thanks Mario Lanza and Yuchao Yang for their contribution to the peer review of this work.

https://doi.org/10.1038/s41565-023-01357-8



Extended Data Fig. 1 | Desirable range of noise. The aggregated noise corresponding to the programming noise, drift variability, and read noise in the PCM devices affects (a) the accuracy of factorization, and (b) the number of iterations to converge. The optimal range for the standard deviation of the noise lies between $0.293\mu S$ and $1.277\mu S$. As indicated by the green vertical line, the level of noise observed in the experimental crossbar array lies within the desirable range of noise.