Proceedings Track

Storing overlapping associative memories on latent manifolds in low-rank spiking networks

Editors: List of editors' names

Abstract

Associative memory architectures such as the Hopfield network have long been important conceptual and theoretical models for neuroscience and artificial intelligence. However, translating these abstract models into spiking neural networks has been surprisingly difficult. Indeed, much previous work has been restricted to storing a small number of primarily non-overlapping memories in large networks, thereby limiting their scalability. Here, we revisit the associative memory problem in light of recent advances in understanding spike-based computation. Using a recently-established geometric framework, we show that the spiking activity for a large class of all-inhibitory networks is situated on a low-dimensional, convex, and piecewise-linear manifold, with dynamics that move along the manifold. We then map the associative memory problem onto these dynamics, and demonstrate how the vertices of a hypercubic manifold can be used to store stable, overlapping activity patterns with a direct correspondence to the original Hopfield model. We propose several learning rules, and demonstrate a linear scaling of the storage capacity with the number of neurons, as well as robust pattern completion abilities. Overall, this work serves as a case study to demonstrate the effectiveness of using a geometrical perspective to design dynamics on neural manifolds, with implications for neuroscience and machine learning.

Keywords: spiking neural networks (SNNs); associative memory; convex polytopes

1. Introduction

Associative and content-addressable memory is a widespread computational motif with broad applications that include neuroscience and machine learning (Hertz et al., 1991). In neuroscience, it represents a plausible model for long-term memory storage (Treves and Rolls, 1994), and the Hopfield and related models (Hopfield, 1982; Hertz et al., 1991) have had a considerable influence on how learning and memory are understood in the brain (Chaudhuri and Fiete, 2016). In machine learning and artificial intelligence, associative memory inspired early connectionist models (Ackley et al., 1985), and recent advances have linked it with state-of-the-art architectures such as transformers (Ramsauer et al., 2020; Krotov, 2023), demonstrating its continuing relevance.

Despite its neuroscientific origins, the mechanistic details of biological associative memory have yet to be worked out, as evidenced by the long-standing search for the "engram", i.e., the physical memory trace in the brain (Lashley, 1950; Josselyn et al., 2015). From the perspective of computational neuroscience, perhaps one issue is that the abstract and conceptual ideas of Hopfield and others have been difficult to replicate at scale in more biologically-plausible circuits with spiking dynamics (Gerstner et al., 2014). Despite much work on the topic of spiking associative memory (Amit and Treves, 1989; Gerstner and van Hemmen, 1992; Maass and Natschläger, 1997; Mueller and Herz, 1999; Sommer and Wennekers, 2001), there is still no established means of storing a large number of overlapping

memories into such networks, and recent studies have been limited to storing relatively few patterns in large networks with non-overlapping groups (e.g., Litwin-Kumar and Doiron (2014); Zenke et al. (2015)). In addition to its neuroscientific implications, this also raises questions about the use of associative memory for spiking network applications in low-power and neuromorphic computing (Davies, 2019; Zenke et al., 2021).

In this work, we propose a new recipe for this long-standing problem using a geometrical perspective on spiking networks (Calaim et al., 2022; Mancoo et al., 2020; Podlaski and Machens, 2024). In Section 2, we describe this framework in detail, and then explain how a particular choice of the manifold results in a correspondence to the original Hopfield model (Hopfield, 1982), with analogous learning rules. Then, in Section 3, we demonstrate recall, memory capacity, and pattern completion abilities. Finally, in Section 4 we briefly touch upon the implications of this work for neuroscience and machine learning.

2. Methods

2.1. Building spiking dynamics on convex, piecewise-linear manifolds

In Section 2.1.1, we introduce a network of leaky integrate-and-fire (LIF) neurons with rank-constrained, all-inhibitory recurrent connectivity, and describe how spiking activity is confined to move along a low-dimensional manifold or *boundary*. Then, in Section 2.1.2, we consider how to design and control the dynamics along this manifold.

2.1.1. Rank-constrained spiking networks form convex latent boundaries

We consider a network of N LIF neurons (Gerstner et al., 2014), with voltage dynamics

$$\dot{\mathbf{V}}(t) = -\mathbf{V}(t) + \mathbf{W}\mathbf{s}(t) + \mathbf{c}(t), \tag{1}$$

where $\mathbf{W}$ is the recurrent connectivity, $\mathbf{s}(t)$ is the vector of delta-pulse synaptic inputs or spikes, and $\mathbf{c}(t)$ is a time-dependent external input (see Appendix A for additional details). Voltages are compared with a set of thresholds $\mathbf{T}$, and when one neuron exceeds its threshold ($V_i \geq T_i$), it emits a spike and causes each neuron j's voltage to jump by W_{ji}. One convenient method for dealing with these discontinuous spiking dynamics is to model the membrane potential as filtering its input with a one-sided exponential filter $h(t) = H(t)\exp(-t)$, with $H(t)$ being the Heaviside function (Gerstner et al., 2014). In doing so, we can integrate Equation (1) (Appendix A) and rewrite it in terms of filtered spikes $\mathbf{r}(t) = (h * \mathbf{s})(t)$ and filtered input $\mathbf{I}^{\text{ext}}(t) = (h * \mathbf{c})(t)$ to obtain

$$\mathbf{V}(t) = \mathbf{W}\mathbf{r}(t) + \mathbf{I}^{\text{ext}}(t). \tag{2}$$

Next, we assume a rank-K constraint on the recurrent weights and decompose it as

$$\mathbf{W} = \mathbf{E}\mathbf{D}, \tag{3}$$

using the two matrices $\mathbf{E} \in \mathbb{R}^{N\times K}$ and $\mathbf{D} \in \mathbb{R}^{K\times N}$, which we refer to as the *encoder* and *decoder* matrices, respectively. We then introduce a K-dimensional latent variable $\mathbf{y} \in \mathbb{R}^K$ which is defined as

$$\mathbf{y}(t) = \mathbf{D}\mathbf{r}(t), \tag{4}$$

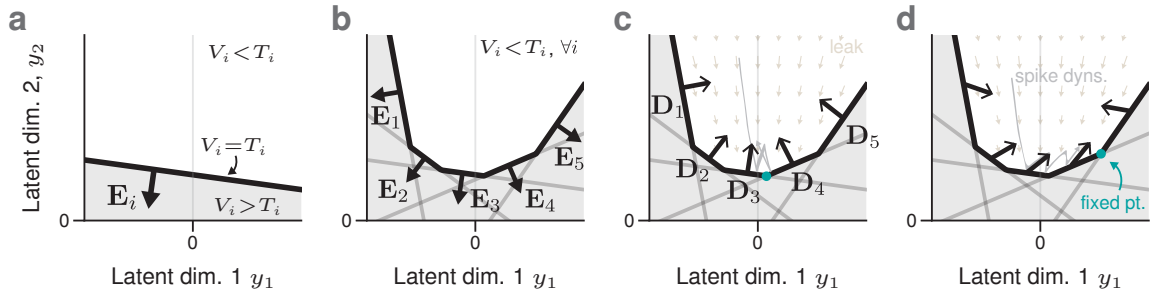

Figure 1: The boundaries and dynamics of rank-constrained spiking networks. **a**: Each neuron's spike threshold forms a linear boundary in latent space that separates subthreshold ($V_i < T_i$) from suprathreshold ($V_i > T_i$) areas. **b**: A population of neurons forms a convex, piecewise-linear boundary between all neurons being subthreshold and suprathreshold areas. **c**: When decoders point orthogonally towards the subthreshold area, spiking dynamics (gray, schematized) oscillate around a "fixed point" at the vertex closest to the origin (green). **d**: Decoders can also point in arbitrary other directions, modifying or forming other fixed points.

making it a linear readout of filtered spikes. We use the definitions from Equations (3) and (4) to rewrite each neuron's voltage in Equation (2) as

$$\mathbf{V}(t) = \mathbf{E}\mathbf{y}(t) + \mathbf{I}^{\text{ext}}(t). \tag{5}$$

We can now visualize each neuron's voltage in the latent space using Equation (5), which, for a constant input, will be a linear equation of each of the latent variables (Figure 1a). We then delineate two half spaces of the latent space — a subthreshold area for which the neuron is below threshold ($V_i < T_i$), and a suprathreshold area for which it is above threshold ($V_i > T_i$), with the line $V_i = T_i$ acting as a boundary between them. We can then arrange several neurons in this space to form a convex, piecewise-linear boundary between an area where all neurons are subthreshold, and an area where at least one neuron is above threshold (Figure 1b). This boundary generalizes in arbitrary K-dimensional latent space to a convex, piecewise-linear polyhedron, with each neuron forming a face (i.e., a $(K-1)$-dimensional linear subspace). The parameters $\mathbf{E}$ and $\mathbf{T}$, along with the input $\mathbf{I}^{\text{ext}}(t)$, will determine the shape and position of this polyhedron, which, as we will show, forms the *manifold* upon which neural trajectories are situated.

2.1.2. Arbitrary dynamics can be specified on the boundary manifold

Due to the relationship between the latent variables and the filtered spikes in Equation (4), we can write the latent dynamics as (see Appendix A)

$$\dot{\mathbf{y}}(t) = -\mathbf{y}(t) + \mathbf{D} \cdot \mathbb{I}_{\leq \mathbf{0}} \left(\mathbf{E}\mathbf{y}(t) - \mathbf{T} + \mathbf{I}^{ex}(t) \right), \tag{6}$$

where $\mathbb{I}_{\leq \mathbf{0}}(\cdot)$ is the element-wise characteristic function of convex analysis (i.e., $\mathbb{I}_{\leq \mathbf{0}}(\mathbf{x})_i = 0$ if $x_i \leq 0$ and $+\infty$ otherwise; Boyd and Vandenberghe (2004)). We thus see that the latent

dynamics have two regimes. In the absence of spiking, the dynamics leak to zero. If a neuron i's voltage exceeds its threshold, indicated by satisfying the characteristic function Equation (6), then the latents instantaneously "jump" in a direction $\mathbf{D}_i$. In order for this to yield stable dynamics, we require that the leak dynamics move into the boundary, and that each spike pushes the dynamics back into the subthreshold (non-spiking) area (Figure 1c, schematized spiking dynamics in gray). This corresponds to a network of spontaneously-active inhibitory neurons (Podlaski and Machens, 2024), which we will utilize here.

Previous work using this perspective has primarily focused on cases in which the latent manifold has a single "fixed point"[1] determined by the input and leak dynamics (Figure 1c; Calaim et al. (2022)), which maps onto a convex optimization problem (Mancoo et al., 2020). In principle, however, additional dynamics may be implemented, such as creating one or more alternative fixed points by orienting decoders in particular directions (Figure 1d), which we will exploit here. In sum, rank-constrained spiking networks can be understood through a decomposition of the recurrent weights into two interpretable components, which determine the shape of the manifold ($\mathbf{E}$), and the dynamics along this manifold ($\mathbf{D}$).

2.2. Mapping the associative memory problem onto the convex manifold

Now that we have established a geometrical picture of the latent manifold and dynamics, we can focus on the main aim of this paper, which is to store a set of p memory patterns as stable attractors of the network dynamics.

2.2.1. Binary patterns can be arranged on the vertices of a hypercube

The first step in designing associative memory dynamics will be to define a specific shape of the convex boundary, and then to choose a set of fixed-point locations along this boundary. For reasons that will become clear in a moment, we choose the latent manifold to be a $(K+1)$-dimensional pyramidal boundary, composed of a K-dimensional hypercube centered at the origin, plus an additional dimension that opens the hypercube and makes it into a cone (Figure 2a). This additional dimension is needed to ensure stable, all-inhibitory dynamics (as mentioned above; also see Section 2.2.4), but for the moment we will ignore it and focus on dynamics in the other K dimensions. From the perspective of the K-dimensional hypercube, this choice constrains us to have $N = 2K$ neurons forming the faces of the hypercube, and enforces a simple and practical structure for the encoding matrix of the network, which we can simply write as

$$\mathbf{E} = \begin{pmatrix} \mathbf{I}_K \\ -\mathbf{I}_K \end{pmatrix}, \tag{7}$$

where $\mathbf{I}_K$ is the identity matrix of length K. We thus see that each neuron's encoder points along one axis in latent space, and makes each neural boundary orthogonal to all others except the one that shares its axis (Figure 2b). We set the length of each side of the hypercube as a constant $2c$, where c depends on the thresholds $\mathbf{T}$ and input $\mathbf{I}^{\text{ext}}(t)$.

Next, we choose the p fixed point locations. An obvious choice is to select the vertices of the hypercube, where the different neural faces meet (Figure 2b, green points), which

1. While the spiking dynamics are oscillatory, the average dynamics can be approximated as a fixed point.

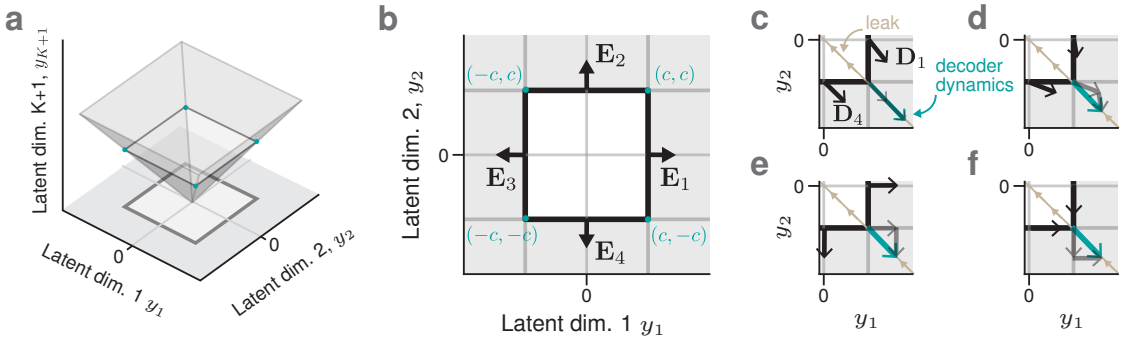

Figure 2: Stabilizing vertices on a hypercubic boundary. **a**: A $(K+1) = 3$-dimensional boundary composed of a hypercube (square), plus an additional open dimension that enables stable dynamics. **b**: 2d slice through panel (**a**) reveals a $(K = 2)$-d square boundary made of $N = 2K = 4$ neurons, and with $2^K = 4$ vertices as possible memory patterns. **c-e**: Close-up of the vertex at $(c, -c)$, demonstrating a symmetric learning rule (**c**), asymmetric learning rule (**d**), the degenerate case of each boundary stabilizing itself (**e**), and a solution with the removal of each neuron's self-connection (**f**).

constrains the patterns to take on binary values in latent space ($\pm c$, corresponding to the 2^K vertices), thereby establishing a clear analogy to the original Hopfield model (Hopfield, 1982). Furthermore, because each vertex is composed of the intersection of K neural boundaries, we expect half of the total $2K$ neurons to be active for each pattern, generating substantial overlap across patterns. We denote the μth pattern in latent ($\mathbf{y}$) space as $\boldsymbol{\xi}^\mu \in \{\pm c\}^K$ and the matrix of p patterns as $\boldsymbol{\xi} \in \{\pm c\}^{(K \times p)}$. Analogously, we denote the μth pattern in neural ($\mathbf{r}$) space[2] as $\boldsymbol{\eta}^\mu \in \mathbb{R}^N_{\geq 0}$ and the matrix of p patterns as $\boldsymbol{\eta} \in \mathbb{R}^{(N \times p)}_{\geq 0}$.

2.2.2. Vertex fixed-point stability requires precise anti-leak dynamics

Having established the convex boundary and memory patterns, we can now consider how to set the decoding weights such that these patterns are stable attractors. To gain some intuition, we choose one vertex of the simplified square boundary (Figure 2b) to be a fixed point, say $\boldsymbol{\xi}^\mu = (c, -c)$, and we see that the leak dynamics are pointing away from this vertex towards the origin, in the direction $(-c, c)$ (Figure 2c). Accordingly, to stabilize this vertex, the decoder dynamics should point away from the leak, in the direction of $\boldsymbol{\xi}^\mu$ itself, which is equivalent to writing

$$\mathbf{D}\boldsymbol{\eta}^\mu = \boldsymbol{\xi}^\mu, \quad \forall \mu, \tag{8}$$

where $\boldsymbol{\eta}^\mu$ is the neural ($\mathbf{r}$-space) activity in the memory state. We note that this simply restates the relationship between the latent and neural spaces (Equation (4)) for the μ-th memory pattern state. If we knew the neural activity $\boldsymbol{\eta}^\mu$ for each pattern state, then we could simply solve Equation (8) for $\mathbf{D}$. However, as already mentioned above, this is

2. Due to the difference in dimensionality, Equation (4) is an underconstrained problem, and thus given a latent pattern $\boldsymbol{\xi}^\mu$, the neural activity pattern $\boldsymbol{\eta}^\mu$ is not well defined without further assumptions.

not possible without additional assumptions on $\boldsymbol{\eta}^\mu$. To make this problem tractable, we thus make the assumption that the active neurons should all have the same average firing rate, denoted by the positive constant κ, and thus $\boldsymbol{\eta}^\mu \in \{0, \kappa\}^N$. This is a reasonable constraint due to the symmetry of each neural boundary with respect to the vertex. With this additional constraint, we can define the memory patterns in neural space as

$$\boldsymbol{\eta}^\mu = \text{ReLU}(\kappa \mathbf{E} \boldsymbol{\xi}^\mu), \tag{9}$$

with ReLU$(\cdot)$ being the element-wise rectified linear function. With this definition, Equation (8) now specifies a linear system of equations for the decoding matrix $\mathbf{D}$, with one equation for each of the K latent dimensions of each of the p patterns. Given that $\mathbf{D}$ has $KN = 2K^2$ elements, this suggests that the linear system can only have a solution provided that $p \leq 2K$, analogous to previous results from Hopfield networks and perceptrons (see Discussion; Cover (1965); Gardner (1988); Hertz et al. (1991)).

2.2.3. Three candidate learning rules

We can now consider what type of learning rules may satisfy Equation (8). Following Hopfield (1982), we being with a simple Hebbian rule. Since we wish to optimize the decoders only, we can keep the encoder as defined in Equation (7) and set the decoder to

$$\mathbf{D}_{\text{hebb}} = \boldsymbol{\xi} \boldsymbol{\xi}^\top \mathbf{E}^\top, \tag{10}$$

which yields a symmetric weight matrix analogous to that of Hopfield (1982). For a single pattern, this solution points each decoder precisely in the direction of the vertex (Figure 2c), thereby satisfying Equation (8). For multiple patterns, however, these equality constraints will typically not be satisfied, as they are much more strict compared to the inequality constraints of the standard Hopfield model (Hertz et al., 1991).

If we can guarantee that each pattern adds an orthogonal component to the decoder matrix, then it should be possible to store more than one pattern without affecting each memory's average decoding vector. In fact, this is precisely what the pseudoinverse learning rule does (Personnaz et al., 1985; Kanter and Sompolinsky, 1987), which neutralizes correlations between the patterns and yields a learning rule

$$\mathbf{D}_{\text{pinv}} = \boldsymbol{\xi} \mathbf{Q}^{-1} \boldsymbol{\xi}^\top \mathbf{E}^\top, \tag{11}$$

where $\mathbf{Q} = \boldsymbol{\xi} \boldsymbol{\xi}^\top$, which will again result in a symmetric weight matrix (Figure 2c).

Finally, we can consider optimizing Equation (8) directly. For our purposes here, we formulate the linear system in Equation (8) as a least squares optimization problem in order to obtain a setting of the matrix $\mathbf{D}$ that stabilizes all memory patterns, which we write as

$$\begin{aligned} \mathbf{D}_{\text{opt}} = \underset{\mathbf{D}}{\text{argmin}} \quad & \|\mathbf{D}\|_2^2 \\ \text{subject to} \quad & \mathbf{D}\boldsymbol{\eta} = \boldsymbol{\xi}, \end{aligned} \tag{12}$$

where we are assuming $\boldsymbol{\eta}$ as defined in Equation (9). Not only is this optimized formulation more general in that it allows for asymmetric weights (Figure 2d), but the presence of a solution can be used to assess feasibility before any spiking simulations have to be run.

2.2.4. Additional considerations

Vertex stability is improved when each neuron is not self-stabilizing. In the degenerate case in which each neuron *only* stabilizes its own boundary, the vertex does not become a stable fixed point, despite the fact that the constraint in Equation (8) is satisfied (Figure 2e). Thus, in practice, each neuron is constrained to point parallel to its own boundary (Figure 2f), thereby improving stability (see Appendix B.1).

Enforcing stable, all-inhibitory dynamics. As mentioned above, unconstrained decoding directions may result in an inherently unstable boundary with runaway positive feedback (Podlaski and Machens, 2024). The additional dimension of the latent space (Figure 2a) enforces stable, all-inhibitory connectivity, and allows dynamics in the other dimensions to be unconstrained (see Appendix B.2).

3. Results

We tested out the theoretical framework outlined above by simulating various spiking networks following Equations (1), (3) and (7), and learning rules as in Equations (10), (11) and (12) (see Appendix C for additional details).

3.1. Comparing the learning rules in small networks

We first constructed small networks with $K = 10$ dimensions and $N = 20$ neurons, storing $p = 4$ random attractor patterns (consisting of 10 active and 10 inactive neurons), and compared the three proposed learning rules. In order to assess memory stability, we simulated each network initialized at each pattern state and measured recall dynamics through the overlap[3] between the latent activity $\mathbf{y}(t)$ and the stored pattern $\boldsymbol{\xi}^\mu$ using cosine similarity, which we refer to as $m^\mu_\xi(t)$ (Hertz et al., 1991).

As expected, we found that the Hebbian learning rule was unable to successfully recall any of the four stored memory patterns (example shown in Figure 3a). As can be seen from the overlap traces, the network does typically find an attractor state, but it tends to be a spurious mixture of patterns, as indicated by several non-zero overlap values. We then simulated networks with both pseudoinverse and optimized weights, and found that they could successfully store all four patterns (shown for optimized solution in Figure 3b). When we initialize the network in a noisy version of the pattern state with 10% (1 of 10) of the latent variables flipped, it was successfully able to pattern complete and recall the correct pattern. We thus see that, as expected, the pseudoinverse and optimized weights stabilize the patterns, whereas the Hebbian weights cannot. Comparing the decoding weights, we observed that all three rules impose a similar, Hebbian-like structure (Figure 3c,d), but that only the pseudoinverse and optimized rules were successfully able to satisfy the stability constraint in Equation (8) (Figure 3e; horizontal line indicates precise stability).

3. While patterns are uncorrelated in latent space, the neural activity will have substantial correlation, with an average of 50% of active neurons shared for any pair of patterns.

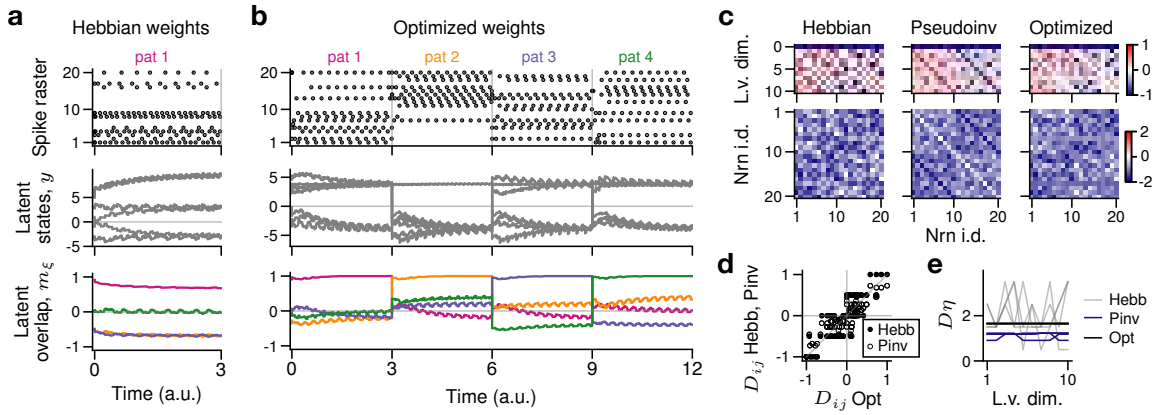

Figure 3: Spiking associative memory in small networks. **a**: Example unsuccessful recall dynamics of the Hebbian network. **b**: Example recall dynamics of the optimized network, with successful recall (see Figure 5 for results for the pseudoinverse rule). **c**: Comparison of the decoder matrices (top), and full recurrent weight matrices (bottom) for the three learning rules. **d**: Comparing elements of the decoder matrix shows strong correlation between all three, suggesting a Hebbian-like association between patterns. **e**: Quantification of Equation (8) for each of the 4 patterns, demonstrating how the pseudoinverse and optimized rules, but not the Hebbian rule, more precisely satisfy the constraints.

3.2. Assessing storage capacity and pattern completion in larger networks

We then scaled up the pseudoinverse and optimized learning rules (example shown for $N = 100$ neurons in Figure 4a). We tested memory capacity in networks of size $N = 400$, 800, and 1600 neurons by measuring the average overlap across all patterns for increasing storage load, and found that both rules exhibited a linear storage capacity (Figure 4b; $p/N \approx 0.3$ for pseudoinverse, $p/N \approx 0.5$ for optimized). Lastly, we found that the self-connection cost had to be optimized as a function of the network load (Figure 4c).

We then tested the pattern completion ability of these networks by initializing them in a noisy pattern state and measuring the overlap following recall dynamics. We found that this also scaled linearly with the storage load, varying from noise in 30% of the pattern bits for lower storage load, and going to zero at capacity (Figure 4d). Lastly, we tested a further scaled up example by training a network of size $N = 1568$ (i.e., $2(28^2)$) on some example MNIST digits, and measuring pattern completion (Figure 4e). This not only demonstrates the scalability of our approach, but also shows the power of the learning rules even to store correlated, non-random patterns with successful recall and pattern completion.

4. Discussion

This work can be placed as part of a recent trend in computational neuroscience of using low-rank matrices to model the low-dimensional dynamics seen in neural data (Mastrogiuseppe and Ostojic, 2018; Chung and Abbott, 2021; Jazayeri and Ostojic, 2021), recently extended

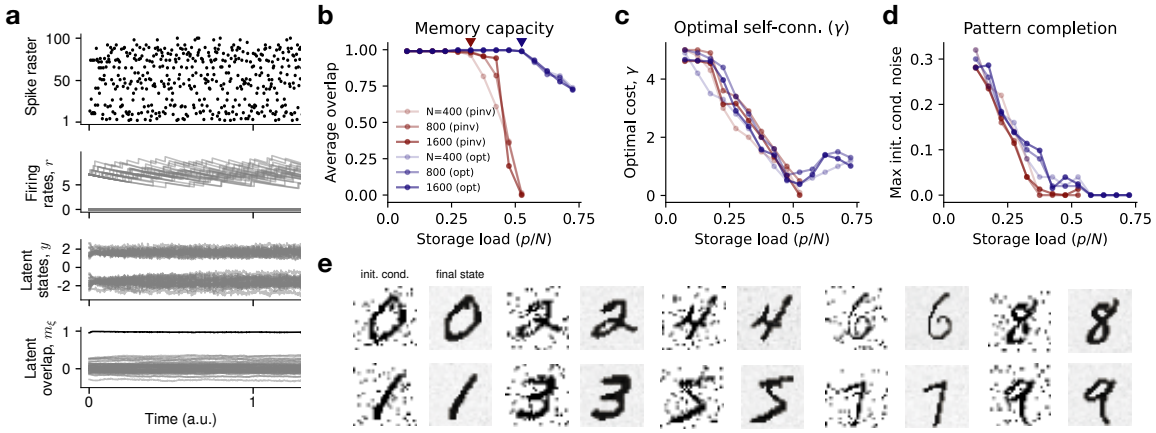

Figure 4: Assessing storage capacity and pattern completion in larger networks. **a**: Demonstration of the attractor dynamics for a network of $N = 100$ neurons storing $p = 40$ patterns with the optimized learning rule. **b,c**: Average overlap across all patterns (**b**) and optimal self-connection strength (**c**) as a function of network load (p/N). **d**: Pattern completion ability via the maximum amount of noise in initial conditions (ratio of flipped bits). **e**: Examples of pattern completion in a network of $N = 2(28^2)$ neurons storing one example of each of the ten MNIST classes (LeCun et al., 1998), binarized to take on ± 1 values; networks were initialized at a noisy version of the pattern with 118 (15%) of the bits flipped.

to spiking networks (Mancoo et al., 2020; Podlaski and Machens, 2024). This insight allowed us to derive a spiking network model with a direct correspondence to the original Hopfield model, and then to take several established ideas and learning rules from the Hopfield literature and to translate them to the spiking domain. The differences that we noted, such as the importance of equality constraints rather than the inequality constraints of classic models (Gardner, 1988), are intriguing, and may lead to further connections. Of particular interest will be to determine if this framework can be extended to more recent models with exponential capacity (Krotov and Hopfield, 2016; Ramsauer et al., 2020; Sharma et al., 2022). We also note that while temporal models of spike-based learning (e.g., Zenke and Vogels (2021)) and pattern storage (e.g., Frady and Sommer (2019)) have been proposed, our framework's geometric perspective may offer more interpretability.

More generally, the work presented here can be seen as a case study of a more widely-applicable framework, whose main insight is the interpretable decomposition of recurrent connectivity into a component that specifies a manifold geometry, and another that specifies dynamics on the manifold. Interestingly, other work has made links between this spiking boundary perspective and analogous rank-constrained rate networks, as well as excitatory-inhibitory networks (Podlaski and Machens, 2024). This suggests that the geometric perspective employed here may be more general, and could be used to analyze more traditional models with continuous dynamics (Vyas et al., 2020), as well as in the analysis of low-dimensional manifolds in neural data (Gallego et al., 2017).

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Daniel J Amit and Alessandro Treves. Associative memory neural network with low temporal spiking rates. *Proceedings of the National Academy of Sciences*, 86(20):7871–7875, 1989.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Nicolas Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8:183–208, 2000.

Nuno Calaim, Florian A Dehmelt, Pedro J Gonçalves, and Christian K Machens. The geometry of robustness in spiking neural networks. *Elife*, 11:e73276, 2022.

Rishidev Chaudhuri and Ila Fiete. Computational principles of memory. *Nature neuroscience*, 19(3):394–403, 2016.

SueYeon Chung and Larry F Abbott. Neural population geometry: An approach for understanding biological and artificial neural networks. *Current opinion in neurobiology*, 70: 137–144, 2021.

Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, 1 (3):326–334, 1965.

Mike Davies. Benchmarks for progress in neuromorphic computing. *Nature Machine Intelligence*, 1(9):386–388, 2019.

E Paxon Frady and Friedrich T Sommer. Robust computation with rhythmic spike patterns. *Proceedings of the National Academy of Sciences*, 116(36):18050–18059, 2019.

Juan A Gallego, Matthew G Perich, Lee E Miller, and Sara A Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017.

Elizabeth Gardner. The space of interactions in neural network models. *Journal of physics A: Mathematical and general*, 21(1):257, 1988.

Wulfram Gerstner and J Leo van Hemmen. Associative memory in a network of 'spiking' neurons. *Network: Computation in Neural Systems*, 3(2):139–164, 1992.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Gabrielle J Gutierrez and Sophie Denève. Population adaptation in efficient balanced networks. *ELife*, 8:e46926, 2019.

John Hertz, Anders Krogh, and Richard G Palmer. Introduction to the theory of neural computation, 1991.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Mehrdad Jazayeri and Srdjan Ostojic. Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Current opinion in neurobiology*, 70:113–120, 2021.

Sheena A Josselyn, Stefan Köhler, and Paul W Frankland. Finding the engram. *Nature Reviews Neuroscience*, 16(9):521–534, 2015.

I Kanter and Haim Sompolinsky. Associative recall of memory without errors. *Physical Review A*, 35(1):380, 1987.

Dmitry Krotov. A new frontier for hopfield networks. *Nature Reviews Physics*, 5(7):366–367, 2023.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Karl S Lashley. In search of the engram. *Society for Experimental Biology, Physiological mechanisms in animal behavior*, pages 454–482, 1950.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Ashok Litwin-Kumar and Brent Doiron. Formation and maintenance of neuronal assemblies through synaptic plasticity. *Nature communications*, 5(1):5319, 2014.

Wolfgang Maass and Thomas Natschläger. Networks of spiking neurons can emulate arbitrary hopfield nets in temporal coding. *Network: Computation in Neural Systems*, 8(4): 355, 1997.

Allan Mancoo, Sander Keemink, and Christian K Machens. Understanding spiking networks through convex optimization. *Advances in Neural Information Processing Systems*, 33: 8824–8835, 2020.

Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018.

R Mueller and Andreas VM Herz. Content-addressable memory with spiking neurons. *Physical Review E*, 59(3):3330, 1999.

L Personnaz, I Guyon, and G Dreyfus. Information storage and retrieval in spin-glass like neural networks. *Journal de Physique Lettres*, 46(8):359–365, 1985.

William F Podlaski and Christian K Machens. Approximating nonlinear functions with latent boundaries in low-rank excitatory-inhibitory spiking networks. *Neural Computation*, 36(5):803–857, 2024.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2020.

Sugandha Sharma, Sarthak Chandra, and Ila Fiete. Content addressable memory without catastrophic forgetting by heteroassociation with a fixed scaffold. In *International Conference on Machine Learning*, pages 19658–19682. PMLR, 2022.

Friedrich T Sommer and Thomas Wennekers. Associative memory in networks of spiking neurons. *Neural networks*, 14(6-7):825–834, 2001.

Alessandro Treves and Edmund T Rolls. Computational analysis of the role of the hippocampus in memory. *Hippocampus*, 4(3):374–391, 1994.

Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43(1):249–275, 2020.

Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.

Friedemann Zenke, Everton J Agnes, and Wulfram Gerstner. Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks. *Nature communications*, 6(1):6922, 2015.

Friedemann Zenke, Sander M Bohté, Claudia Clopath, Iulia M Comşa, Julian Göltz, Wolfgang Maass, Timothée Masquelier, Richard Naud, Emre O Neftci, Mihai A Petrovici, et al. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4):571–575, 2021.

Appendix A. Additional details about the LIF network derivation

We consider LIF networks with instantaneous delta-pulse synaptic inputs (e.g., Brunel (2000)), and thus for a neuron i emitting a series of spikes at times $(t_i^{(1)}, t_i^{(2)}, ...)$, we can write the ith element of $\mathbf{s}(t)$ as $s_i(t) = \sum_f \delta(t - t_i^{(f)})$, where $\delta(\cdot)$ is the Dirac delta function. We note that the neurons do not have an explicit voltage reset, but this is modeled through a self-connection in the recurrent connectivity, $\mathbf{W}_{ii}$ for each neuron i (see also Appendix B.1). The voltage dynamics in Equation (1) have an implicit time constant — for simplicity we assumed that it was equal to 1, which is equivalent to time being in the units of this time constant.

We defined $\mathbf{r}(t)$ and $\mathbf{I}^{\text{ext}}(t)$ through convolutions, but it is convenient to write them in a differential form here as

$$\dot{\mathbf{r}}(t) = -\mathbf{r}(t) + \mathbf{s}(t), \tag{13}$$

$$\dot{\mathbf{I}}^{\text{ext}}(t) = -\mathbf{I}^{\text{ext}}(t) + \mathbf{c}(t). \tag{14}$$

The easiest way to see the relation between Equations (1) and (2) in the main text is to take the derivative of Equation (2), which introduces the terms $\dot{\mathbf{r}}(t)$ and $\dot{\mathbf{I}}^{\text{ext}}(t)$, for which we can use the definitions above from Equations (13) and (14).

Moreover, the differential equation forms in Equations (13) and (14) allow us to define a similar form for the latent variables of the network, as

$$\dot{\mathbf{y}}(t) = -\mathbf{y}(t) + \mathbf{D}\mathbf{s}(t). \tag{15}$$

We thus see that the latent dynamics have two regimes: leaking and instantaneous spiking. We can equivalently separate these two regimes and write the dynamics as

$$\dot{\mathbf{y}}(t) = -\mathbf{y}(t) \tag{16}$$

$$\mathbf{y}(t) \leftarrow \mathbf{y}(t) + \mathbf{D}_i \quad \text{if} \quad \mathbf{E}_i\mathbf{y} + I_i^{\text{ext}}(t) \geq T_i, \tag{17}$$

where $\mathbf{D}_i \in \mathbb{R}^K$ and $\mathbf{E}_i \in \mathbb{R}^{1\times K}$ are the ith column and row, respectively, of the decoder and encoder matrices. We use this formulation to derive Equation (6) in the main text.

Appendix B. Additional considerations for the Hopfield network

B.1. Vertex stability is improved when each neuron is not self-stabilizing

It is common practice in building Hopfield networks to remove the self connections from the recurrent weight matrix, i.e., $W_{ii} = 0, \forall i$, as they have been found to negatively affect dynamics and spurious memory states (Kanter and Sompolinsky, 1987; Hertz et al., 1991). Similar intuitions apply here as well, which we can see through the simple $2D$ example (Figure 2e). In the degenerate case in which each neuron *only* stabilizes its own boundary, the constraint in Equation (8) is satisfied for all vertices, but in fact none of the vertices are truly stable — instead, each neuron becomes an independent fixed-point attractor, making a total of $p = N = 2K$ single-neuron patterns (analogously to the standard Hopfield model, Hertz et al. (1991)). Instead, setting each neuron's self connection to zero prevents each neuron from stabilizing its own boundary (Figure 2f), which promotes a more distributed

code at the vertex. In the spiking network, the self connection W_{ii} has a direct correspondence to the after-spike voltage reset, as well as potential adaptation currents, and has similarly been shown to promote a distributed code (Gutierrez and Denève, 2019; Calaim et al., 2022; Podlaski and Machens, 2024). Considering both of these points together, we thus impose a negative self connection

$$W_{ii} = -\gamma, \quad \forall i. \tag{18}$$

where γ is a positive constant. Empirically, we found that γ had to be optimized to promote stable recall ability, which was done numerically (see Figure 4c).

Due to the simplified form of the encoding matrix $\mathbf{E}$, Equation (18) can be incorporated into the optimization problem in Equation (12) by simply omitting the elements of $\mathbf{D}$ that correspond to self connections in the optimization. However, Equation (18) can only be applied post-hoc for the Hebbian and pseudoinverse rules. In those cases, Equation (18) is simply imposed following the setting of $\mathbf{W} = \mathbf{ED}$.

B.2. Enforcing stable, all-inhibitory dynamics

As mentioned in the main text, unconstrained decoding directions may result in dynamics at the boundary that are inherently destabilizing. Related work has shown that the boundary dynamics will be stable if the connectivity is all-inhibitory, which implies that all decoders point in a subthreshold direction. We can enforce this by considering that the network encodes an extra dimension, which we will refer to as the 0th dimension. This adds an inhibitory component to each element of the weight matrix. To see how this works, we assume a rank-K encoder matrix as in Equation (7) and a rank-K decoder matrix defined with any of the three learning rules. We then define a new rank-$(K+1)$ weight matrix $\tilde{W}_{ij}$ defined as

$$\tilde{W}_{ij} = E_{i0}D_{0j} + W_{ij}, \tag{19}$$

$$= E_{i0}D_{0j} + \sum_{k=1}^{K} E_{ik}D_{kj}, \tag{20}$$

where we have isolated the extra dimension in the first term. We then set these extra terms to

$$E_{i0} = 1, \quad \forall i, \tag{21}$$

$$D_{0j} = -\alpha 1, \quad \forall j, \tag{22}$$

where α is a constant that we set as $\max(W_{ij}), \forall i, j$. It is then straightforward to show that the synaptic connectivity becomes all-inhibitory, thereby leading to stable dynamics. We refer the reader to Podlaski and Machens (2024), where such sign constraints are discussed in more detail.

Appendix C. Simulation details and code

We simulated networks of spiking neurons using the Euler method on Eq. 1 with a time step of $dt = 1e-4$. As the networks simulated were relatively small, all simulations were

done on a personal laptop and simulation time was less than 1 minute for all simulations. For the estimation of storage capacity, many networks were run, with a simulation time of approximately a few hours on a MacBook computer. Code to simulate the networks and to generate all plots will be made available upon the eventual publication of this work.

Appendix D. Supplementary figures

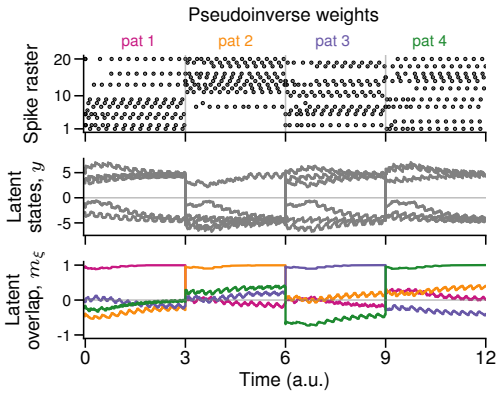

Figure 5: Demonstration of the small network simulation of $N = 20$ neurons for the pseudoinverse learning rule, showing successful pattern completion and recall (compare with Figure 2).